MPE III 2028

7/80

HEWLETT PACKARD

COMMUNICATOR

ISSUE NUMBER 25

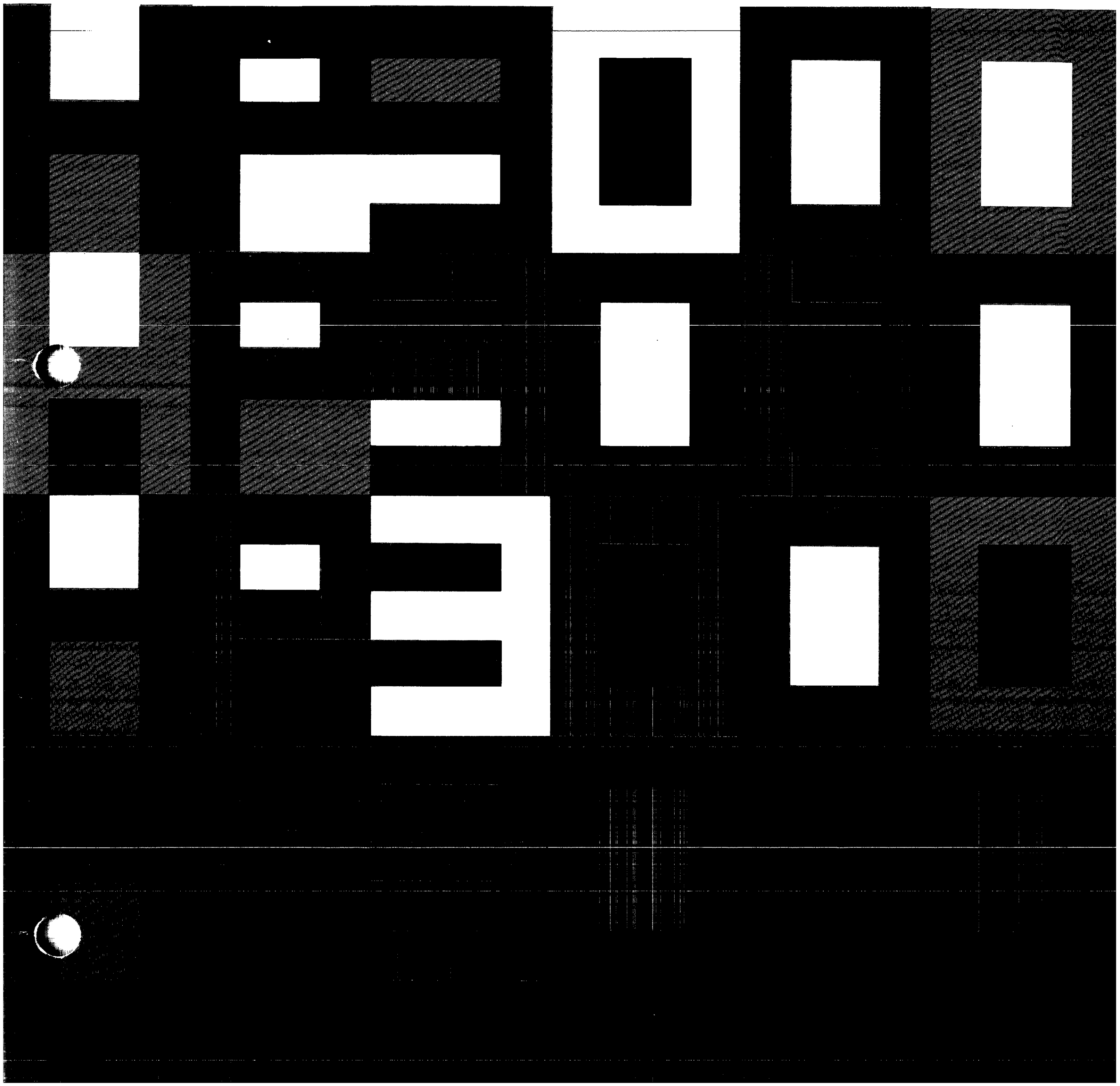


Table of Contents

Editor's Note

The articles in this issue of Communicator 3000 should give you a good idea of what is included in Installation Tape 2028. The opening article summarizes MPE III enhancements. One change to note is the size of the HP 7925 disk free space table, explained on page 9.

FLEXIBLE DISCCOPY/3000, an important new product for the Series 30 and 33, which copies IBM 3741-formatted data sets, is discussed in depth beginning on page 10. The new 2631B Remote Spooled Printer is described in detail starting on page 14.

Enhancements to FCOPY/3000, EDIT/3000, IMAGE/3000, V/3000, COBOL II/3000 and RPG/3000 are discussed in articles beginning on page 19 and running through page 36.

An introduction to IML/3000, a new data communications product, starts on page 37. Two other significant articles include an overview of Network File Transfer (NFT) on page 41, and an announcememnt of new Data Capture Procedures beginning on page 43. A method for recreating an MRJE/3000 configuration file for the 2028 tape installation starts on page 39.

A discussion of RPG/3000 programming optimization techniques begins on page 47. Starting on page 52 is a discussion of how to use the Editor to modify BASIC programs. An article entitled "Tips on :ALLOCATE" can be found on page 55.

Two significant articles relating to COBOL II/300 begin on page 57. The first is a comparison of stack layouts for COBOL II/3000 and COBOL/3000. The second is a COBOL II User Survey, which provides you a valuable opportunity for customer feedback.

The Documentation Section, beginning on page 73, reports recent documentation activity. The latest Catalog of Customer Publications begins on page 82.

:MPECONTROL, a new MPE command, is documented in a handy tear-out section starting on page 93.

Editor
COMMUNICATOR 3000
HP General Systems Division
19447 Pruneridge Avenue
Cupertino, CA 95014

Many New MPE Enhancements

By Adrienne Bresso, Terry Ishida, Stephanie Littell, Robin Rakusin and Bob Stamps, General Systems Division

We are continuing our commitment to improve MPE. Among the most significant enhancements coming your way on Installation tape (IT) 2028 are:

- User Logging enhancements
- System logging enhancements
- FCOPY enhancements
- Remote Spooled Printer
- HP 7925 Disc Free Space Table Modifications
- Series 30/33 driver name changes

Each of these enhancements is described in detail below. We suggest that you keep these descriptions for future reference. These exciting new enhancements should prove useful in many of your applications.

USER LOGGING ENHANCEMENTS

There are several enhancements to MPE USER LOGGING on the 2028 release of MPE. These include changes to the WRITELOG intrinsic and the addition of four new intrinsics.

Note: IMAGE has not yet been enhanced to take advantage of this added capability.

The WRITELOG intrinsic has been modified to allow the writer to write data to the logging buffer and flush the contents of the buffer to the disc (or disc buffer for a tape file) simultaneously. This avoids leaving data in the logging buffer which is susceptible to loss in case of a system failure. Null records in the buffer will not be flushed. In order for the writer to write and flush, a value of two must be specified for the "mode" parameter of the writelog intrinsic.

The four new intrinsics that have been added to MPE USER LOGGING include BEGINLOG, ENDLOG, LOGSTATUS and FLUSHLOG. These intrinsics provide capabilities to mark the beginning and end of logical transactions in the log file, obtain information about an opened logging file, and write the contents of the user logging memory buffer to the disc destination file. These new intrinsics and their parameters are described below:

```
PROCEDURE BEGINLOG(INDEX,DATA,LEN,MODE,STATUS);
DOUBLE INDEX;
INTEGER LEN,MODE,STATUS;
ARRAY DATA;
OPTION EXTERNAL;
```

The BEGINLOG intrinsic is used to mark the beginning of a logical transaction in the log file. It post a special record in the log file and flushes the logging memory buffer to ensure that the record gets to the logging file.

User data can also be posted to the logging file with this intrinsic by using the DATA parameter. This function of the intrinsic is identical to that of the WRITELOG intrinsic.

PARAMETERS:

DATA-	An array in which is passed the actual information to be logged.
LEN-	The length of the data in DATA. A positive count indicates words, and negative count indicates bytes.
INDEX-	The parameter returned from OPENLOG that identifies the user's access to the logging system.
STATUS-	An integer that the logging system uses to return error information to the user. OK status is identified by zero.
MODE-	An integer which specifies whether the user wants his process impeded by the logging process in the event that the logging buffer becomes full. If set, the WRITELOG intrinsic will, in the event that it is not possible to complete the request without impeding the process, return an indication in the status word that the request was not completed. Mode zero implies wait, mode one implies nowait.

BEGIN TRANSACTION MARKER, CODE=11

0	2	3	4	6	7	8	127
rec#	cksum	code	time	DATE	log#	len	user area

```
PROCEDURE ENDLOG(INDEX,DATA,LEN,MODE,STATUS);
DOUBLE INDEX:
INTEGER LEN,MODE,STATUS;
ARRAY DATA;
OPTION EXTERNAL;
```

The ENDLOG intrinsic is used to mark the end of a logical transaction in the log file. A special record is posted and the user logging memory buffer is flushed to ensure that the record gets to the logging file.

The DATA parameter of the intrinsic can be used to post user data to the log file. This function of the procedure is identical to the WRITELOG intrinsic.

PARAMETERS:

DATA-	An array in which is passed the actual information to be logged.
LEN-	The length of the data in DATA. A positive count indicates words, and negative count indicates bytes.
INDEX-	The parameter returned from OPENLOG that identifies the user's access to the logging system.
STATUS-	An integer that the logging system uses to return error information to the user. OK status is identified by zero.
MODE-	An integer which specifies whether the user wants his process impeded by the logging process in the event that the logging buffer becomes full. If set, the WRITELOG intrinsic will, in the event that it is not possible to complete the request without impeding the process, return an indication in the status word that the request was not completed. Mode zero implies wait, mode one implies nowait.

END TRANSACTION MARKER, CODE=10

0	2	3	4	6	7	8	127
rec#	cksum	code	time	DATE	log#	len	user area

```
PROCEDURE LOGSTATUS(INDEX,LOGINFO,STATUS);
INTEGER STATUS;
DOUBLE INDEX;
LOGICAL ARRAY LOGINFO;
OPTION EXTERNAL;
```

The LOGSTATUS intrinsic is used to obtain information about the opened logging file. It's primary use will be to determine the amount of space used and remaining in a disc logging file.

PARAMETERS:

INDEX- The parameter returned from OPENLOG that identifies the user's access to the logging system.

LOGINFO- A formatted array in which is returned the following information.

words 0 and 1 - Total records written to log file.
words 2 and 3 - The size of the logging file.
words 4 and 5 - The space remaining in log file.
word 6 - The number of users using the log system.

STATUS- An integer in which is returned error information to the caller. OK status is identified by zero.

```
PROCEDURE FLUSHLOG(INDEX,STATUS);
DOUBLE INDEX;
INTEGER STATUS;
OPTION EXTERNAL;
```

The FLUSHLOG intrinsic is used to write the contents of the User Logging memory buffer to the disc destination file. This helps to preserve the contents of the memory buffer in the event of a system failure. Null records will not be flushed.

PARAMETERS:

INDEX- The parameter returned from OPENLOG that identifies the user's access to the logging system.

STATUS- An integer in which is returned error information to the caller. OK status is identified by zero.

SYSTEM LOGGING

The system logging facility will now allow an I/O driver to log the appropriate number of error status words into a log file. In the past, MPE drivers have logged only the first word of information into the file. This data was then accessed through the system utility LISTLOG2.PUB.SYS by requesting output with an event code of 11, signifying I/O errors. The resulting format appeared as follows:

```
                                 DRT/UNIT-*-LDEV-*-STATUS-*-TYPE/-*-DVR/-*-XMISSION/-*-DRIVER/
                                                         SUBTYPE  FUNCT   COUNT       DATA    A
4:46:45:2  I/O   SYS             6  0       7       100552  24 0     1     4096        100001
                                 ASDYWBCFMPLRT  G-*-TARGET/-*-TARGET/-*-PCB/-*-DVR/-*-DVR/  UENC
                                                     DST#       ADDRESS    STAT    PAR1   PAR2-DVR C
                                 0000100100000000     000133    010116    010021 000000    000014
```

As a result of the system logging internal changes on the 2028 release, the output format for LISTLOG2.PUB.SYS has been changed. The appropriate number of status words logged by a driver into the logging file (i.e. for the disc driver two words are logged into the file) will be formatted by the LISTLOG2 utility in octal rather than the existing binary format. Other format changes, including two words, IOQ(QMISC) and IOQ(QFLAG), are shown in this example of the new LISTLOG2 I/O error format.

```
  TIME      TYPE  JOB#        B01.0                  DATE: MON, JUN, 16,1980     LOGFILE: 77

                            * - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
                              DRT UNIT LOEV          TYPE SUBTYPE FUNCT XFER COUNT IOQ(QMISC)
 9 :22:9 :3  I/O   SYS        6  0     7             24    0        1      1020       100001
                              IOQ(QFLAG)  DST#    ACDRESS  PCB/STAT  PAR1      PAR2
                              006400      100136  001620   013021    000000    000014
  1   DEVICE STATUS WORDS -
 100552
```

Please note that this information is generally used by the CE in resolving technical system problems.

Remote Spooled Printer

This version of MPE will allow the new 2631B printer to be attached to a Series III/30/33 terminal port as a remote spooled printer. Please refer to the in-depth article on the Remote Spooled Printer on page 14 of this issue.

HP 7925 Disc Free Space Table

Every system/private volume disc on the HP 3000 keeps track of free areas through a table called the Disc Free Space Table. This table resides on the low sector area of every pack.

In this version of MPE, this table will be enlarged on a RELOAD for 7925's ONLY. This will allow MPE to keep track of more free areas. However, once a RELOAD has been done, earlier versions of MPE will no longer be able to correctly read the 7925.

In the event that you must convert back to an earlier version of MPE, the following steps should be taken:

Note: Step #2 applies to Series 30/33 only, and should be omitted for Series II/III.

1. Full SYSDUMP on 2028.
2. Series 30/33 only — :RUN CONF33. CREATOR.SYS;PARM=1.
3. RELOAD,ACCOUNTS option from 2028 (or earlier) backup tape.
4. :RESTORE @.@.@ from 2028 SYSDUMP tape.
5. :RESTORE @.PUB.SYS without KEEP option from old backup tape.

For this reason, it is recommended that users keep a dated SYSDUMP tape of their previous system (before 2028) for one month. Similarly for private volumes, the new Free Space Table will be created in the VINIT > INIT command and must be reinitialized if converting back to an older version of MPE.

Series 30/33 Driver Names

The I/O drivers on the Series 30/33 will be changing their names. System managers/supervisors and console operators should be notified of this change because this will affect the configuring of new peripherals. We also recommend that users list their new I/O configurations offline for archival/reference purposes. For your reference the new names are listed below:

OLD	NEW
IOMDISC1	HIOMDSC1
IOTAPE0	HIOTAPE0
IOLPRT0	HIOLPRT0
IOLPRT1	HIOLPRT1
IOFLOP0	HIOFLOP0
IOTERM0	HIOTERM0

Announcing FLEXIBLE DISCCOPY/3000

For the Series 30 and Series 33, a new product that copies IBM 3741-formated data sets.

by Joseph Vollmer, General Systems Division

FLEXIBLE DISCCOPY/3000 (DISCCOPY) is a new product which reads data on IBM 3741-formatted flexible discs and converts this data into the standard file format of the HP 3000. DISCCOPY operates within the operating system (MPE) environment of the HP 3000 Series 30 and Series 33.

DISCCOPY is easy to use. You load the IBM 3741 diskette into the 7902 flexible disc drive of a Series 30 or Series 33 and then run DISCCOPY.PUB.SYS. DISCCOPY reads the volume label and lists the data sets on the volume. (IBM data sets are the equivalent of HP 3000 files.) With simple commands, you select the data sets you want to copy. DISCCOPY creates an MPE file for each data set selected, copies the data set into the new file, and converts the copied data into the standard file format processible on the HP 3000.

DISCCOPY is equipped with numerous, helpful messages. These messages indicate errors, serve as warnings, or give status information. Additionally, DISCCOPY is able to re-prompt you if you input a syntactical error. (This ability to re-prompt is available only in sessions, as batch mode is non-interactive.)

Because of DISCCOPY's extensive message catalog, and because DISCCOPY re-prompts you in interactive mode if you make an incorrect entry, DISCCOPY is best suited to be run in sessions. However, DISCCOPY also can be run in batch mode. DISCCOPY's interaction with the System Job Control Word lends added flexibility to batch mode use.

DISCCOPY provides for both single volume and multiple volume conversion. Each IBM 3741 flexible disc is called a volume. There may be up to 19 data sets on a volume. Some data sets may span several volumes (99 maximum).

Besides format conversion, DISCCOPY also provides for character code translation. Most IBM 3741 discs are written in EBCDIC character code. DISCCOPY automatically translates EBCDIC characters into ASCII unless you use the NOTRANSLATE option when specifying the data sets to be copied, or unless DISCCOPY sees in the data set label that character translation may not be desirable. You can make individual translation decisions for each data set to be copied.

DISCCOPY saves you time by immediate extent allocation. DISCCOPY allocates all required extents for the new file before copying the data set. If all the required extents cannot be obtained, DISCCOPY will not begin to copy the data set. Thus, you're saved the time of proceeding with a conversion that is bound to fail due to insufficient space. (There is one exception: in multiple volume conversions when the number of volumes is unknown, extents are allocated as needed.)

When you specify a number of data sets to be copied, DISCCOPY sorts these data sets according to size. DISCCOPY then copies the data sets in ascending order, starting with the smallest data set first. In this way, DISCCOPY maximizes the number of data sets that can be copied into accounts of limited file space.

When DISCCOPY creates a new file, it uses MPE disc space efficiently by compacting copied records. For each data set copied, DISCCOPY tests block lengths of one to eight sectors and picks the most efficient blocking factor.

DISCCOPY defaults to give the new MPE file the same name as the data set that is being copied into it. For example, if data set SET1 is to be copied, DISCCOPY names the corresponding new file SET1. However, there may be times when the data set name matches an an existing filename in the target group. (The MPE File System does not allow duplicate filenames in the same group.) DISCCOPY handles the problem of duplicate names differently in sessions and jobs. In a session, DISCCOPY asks you to either rename the new file or purge the existing file. In a job, DISCCOPY gives the new file a unique alphanumeric name (similar to the Kfile names used in Edit/3000). You can also use file equations to equate a data set name to another filename.

You can use any of the MPE Command Intrinsic commands (for instance, :LISTF). These commands can be entered whenever DISCCOPY requests user input.

No special capabilities are needed to run DISCCOPY-- only SF (Save Files), ND (Non-sharable Device), IA (Interactive Access) for sessions, and BA (Local Batch Access) for jobs. It is possible, however, to copy a data set into a file which resides on a private volume. To do this, you need UV (Use Volumes) and/or CV (Create Volumes) capabilities.

Another requirement for running DISCCOPY is that the flexible disc drive unit must be configured in a Foreign Disc Class. The default Foreign Disc Classname used by DISCCOPY is FDISC; however, this can be overridden by a :FILE equation. For more information about the Foreign Disc Facility, see the article on this subject in COMMUNICATOR issue #24 and the discussion of the Foreign Disc Facility in the System Manager/ System Supervisor Reference Manual.

The FLEXIBLE DISCCOPY/3000 User's Guide is now available. This new manual (part number 32199-90001) explains how to use DISCCOPY in sessions and jobs, provides a basic understanding of how DISC-COPY operates, and attempts to predict and answer any questions the user may have regarding DISCCOPY.

DISCCOPY's standard product number is 32199A; the RIGHT-TO-COPY product number is 32199R.

****

The following example of DISCCOPY illustrates a single volume conversion in a session. User input is underlined.

```
:RUN DISCCOPY.PUB.SYS

HP32199.XX.XX  DISCCOPY  (time and date)
(C) HEWLETT-PACKARD CO. 1980

PLEASE SELECT A FUNCTION:
  1  3741 DATA SET TO MPE DISC FILE CONVERSION.
  E  EXIT.
>1
 -

3741 DATA SET TO MPE DISC FILE CONVERSION.
NAME OF VOLUME TO BE COPIED (<return> IF UNKNOWN): VOLX
                                                   ----

LIST OF DATA SETS ON VOLUME VOLX:
   1.  NUTS
   2.  BOLTS
   3.  WASHERS
   4.  BUTTONS  <<Data sets are listed here in the order of their
   5.  BOWS        appearance on the volume.>>
   6.  RIBBONS
   7.  THREAD
   8.  NEEDLES
WHICH DATA SETS DO YOU WANT TO COPY?
>1,3,4/6,THREAD;NOTR=2
 ---------------------

BOWS
5 RECORDS COPIED
THREAD
18 RECORDS COPIED <<Data sets are copied in ascending order of file size.
NUTS                 File size is determined by record length and
47 RECORDS COPIED    number of records.>>
WASHERS
129 RECORDS COPIED
RIBBONS
155 RECORDS COPIED
BOLTS
187 RECORDS COPIED
ALL SELECTED DATA SETS PROCESSED.

PLEASE SELECT A FUNCTION:
  1.  3741 DATA SET TO MPE DISC FILE CONVERSION.
  E.  EXIT.
>E
 -

END OF PROGRAM
:
```

Introducing the 2631B Remote Spooled Printer

By Terry Ishida, General Systems Division

The HP2631B is a serial character printer intended for low speed, low volume applications. Version B.01.02 of MPE enables the 2631B to be attached to the ATC/ADCC on the Series II/III/30/33 as an RS-232C terminal port. Under RS-232C, the printer may be located up to 100 feet away hardwired or connected via full duplex modem to the system. In additon, MPE will allow the 2631B to be spooled for output and monitored for error conditions, such as 'paper out' or 'offline'. The result is a remote spooled printer on the HP3000 family.

Special Printer Features

Users will be able to access most advanced printing features (alternate character sets, compressed mode, expanded mode, auto-underline) by embedding escape sequences in data passed to the printer as output. It will be the user's responsibility, when finished, to manage the printer environment via escape sequences. In the event of a powerfail, the printer will be reset and the application may have to be rerun. Accordingly, advanced features are not officially supported on the HP3000. Alternate character sets may be invoked through "switch in/switch out" escape sequences only. The use of 8-bit data to select alternate characters is not available due to parity checking for data verification. Additionally, running output containing advanced features will not produce the same results on a local printer (2631A, 2608A, 2613, etc.).

Users will not be allowed programmatic access to printer reset, page length, on/offline, identify request, return status, self-test or downloading of VFC's. In the latter case, all output formatting must be done through carriage control values within the user's program.

Printer Speed

It is recommended that the 2631B be operated at 1200 baud to best match the print speed of 180 cps. However, the printer may be operated at 2400 baud to maximize printer throughput at the expense of an increase in resulting CPU overhead.

Maximum Configuration

A maximum of 4 2631B printers may be attached to a Series II/III/30/33 . Each printer counts as one terminal toward the maximum number of terminals allowed on the system (32 ports for Series 30/33, 64 ports for Series II/III).

Hardware Configuration

There are two sets of external DIP switch panels that must be correctly set in order to operate the 2631B on an HP3000. The first set is the I/O Adapter Panel next to the RS-232 port connector at the rear of the machine. The second set is the Printer Control Panel on the top of the machine. The I/O Adapter switches at the rear should be set as follows:

S1-1	Open	enable XON/XOFF operation.
S1-2	Closed	disable ENQ/ACK operation.
S1-3	Closed	
S1-4	Closed	
S1-5	Closed	
S1-6	Closed	
S1-7	Open	
S1-8	Closed	

The Print Control Panel on the front of the 2631B contains two groups of switches, the left set designated as S2, the right set designated as S3. The switches should be set as follows:

S2-1	Open	only full duplex is supported
S2-2	Closed	Odd parity
S2-3	Open	Odd parity
S2-4	Closed	

S2-5	Baud Rate	Open	Open	Closed	Closed	Closed
S2-6		Closed	Closed	Open	Open	Closed
S2-7		Closed	Open	Open	Closed	Open
S2-8		Open	Open	Closed	Open	Open
	Baud	1200	2400	300	150	110

S3-1 lines per inch; OPEN = 6 LPI
(Note: driver will set the device to 10 characters/inch and 6 LPI when device is "opened." The switches, therefore, have no effect in normal operation. A user may programmatically change the values.)

S3-2	char./inch	Open	= 10 characters per inch
S3-3		Open	= 10 (standard)

S3-4 CLOSED; skip page perforations in full restricted mode

S3-5	Page Length	Open	11 inch page length (standard)
S3-6		Open	
S3-7		Open	
S3-8		Open	

See the configuration sheet accompanying the printer for additional page length and characters per inch options.

(Note: The next two sets of DIP switches are internal and can only be set by removing the entire cover. This should only be done by qualified persons as there is a potential shock hazard).

There is an 8-bit DIP switch on the Serial I/O board, which should be set as follows:

S4-1	Ext baud rate	Closed	= 1200 baud	Closed	= 2400 baud
S4-2	Ext baud rate	Open	= 1200 baud	Open	= 2400 baud
S4-3	Ext baud rate	Open	= 1200 baud	Closed	= 2400 baud
S4-4	Ext baud rate	Closed	= 1200 baud	Closed	= 2400 baud

S4-5 Open; strip delete characters from data stream
S4-6 Open; in 'no parity', 8th bit = 1 when sent
S4-7 Closed; in 'no parity', the received 8th bit is ignored
S4-8 Closed

Finally, there is a 4 bit DIP switch on the PCA board which should be set as follows:

S5-1 Open; power on page length = 11 inches
S5-2 Open; power on self test speed test disabled
S5-3 Closed; partial restricted mode
S5-4 Open

MPE I/O Configurations

The 2631B should be configured in SYSDUMP/INITIAL dialogue as follows:

```
LDEV? a number in the terminal range is suggested
DRT? DRT number
UNIT? ATC port on Series II/III; 0 on Series 30/33
SOFTWARE CHANNEL? 0
TYPE? 32 (printer)
SUBTYPE? 14 - hardwired
         15 - full duplex modem
TERMTYPE? 19 (2631B)
SPEED? any valid speed, 120 suggested
REC WIDTH? any valid record width, 66 words (132 chars) suggested
OUTPUT DEVICE? 0
ACCEPT JOBS/SESSIONS? NO
ACCEPT DATA? NO
INTERACTIVE? NO
DULICATIVE? NO
INITIALLY SPOOLED? YES/NO
DRIVER NAME? HIOTERM0 on Series 30/33
             IOTERM0 on Series III
DRIVER CLASS? any valid class name,
              MODEMLP,REMOTELP,RLP,RP suggested
```

For each spooled 2631B, an additional 16 IOQ's and 5 terminal buffers should be configured in the SYSDUMP 'System Table Changes' section.

Console Messages

'LDEV nn PAPER OUT' is sent to the console when the device is out of paper. 'LDEV nn NOT READY' is sent to the console when any of the following conditions is detected:
parity error, device offline, paper jam, or device not responding.

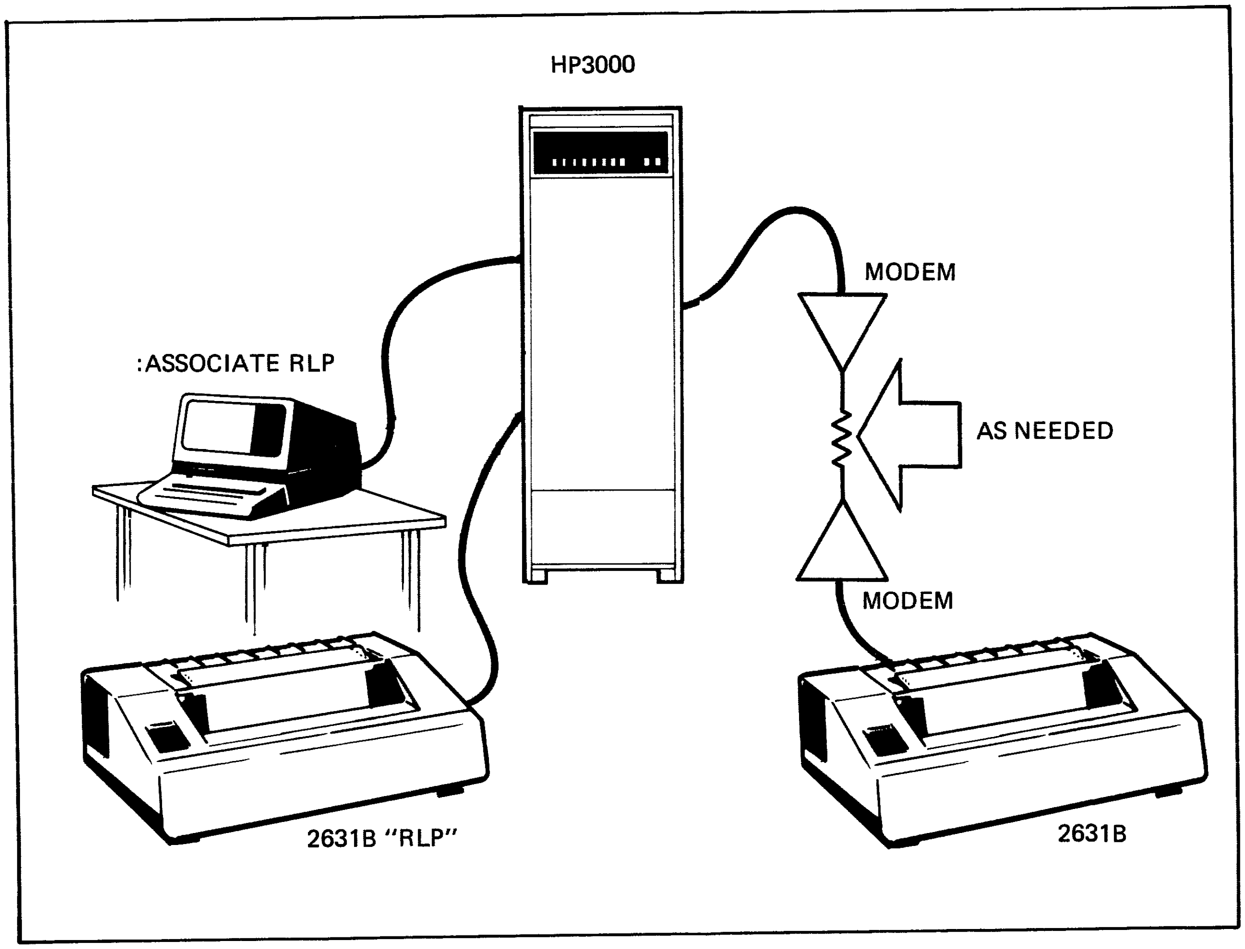
In addition, a parity error, printer powerfail or device not responding condition will cause the last operation to complete in error with the console message of 'LDEV nn GENERAL I/O STATUS % 53' (if the device is spooled). This in turn will cause the spooler to stop with a 'SPOOLER I/O ERROR', and, if via modem, will also cause a switched modem to disconnect.

The RESET button may be used while the printer is idle or offline to reset top-of-form. However, using the reset button while printing will have the same effect as a printer powerfail.

Possible Applications

In situations where the printer need not be connected for the entire day, a modem linkup provides a good solution. Output spoolfiles may be gathered for the printer during normal working hours; at the end of the day, the telephone link may be connected and a spoolfile printed. The modems will be automatically disconnected after each spoolfile is printed if the modem is switched. Care must be taken to provide enough disc space for accumulating spoolfiles.

A remote station can be created by using the MPE: ASSOCIATE command to allow a nearby terminal to act as the console for the 2631B. All console messages (PAPER OUT, NOT READY) and console commands (:STARTSPOOL, :STOPSPOOL) for the printer will then be redirected to the associated terminal.



FCOPY/3000 Enhancements

Several important enhancements have been added to FCOPY:

1. The end-of-file pointer in the tofile is no longer set to zero when the file is opened. Therefore, data in the tofile is retained when a user responds "NO" to warnings *200* or *201*.

2. FCOPY can now have a command passed to it with the INFO parameter of the :RUN command. For example:

   ```
   :RUN FCOPY.PUB.SYS;INFO="FROM=FROMFILE;TO=TOFILE"
   ```

   When INFO is used, FCOPY does an automatic EXIT after executing the command passed. Also, an MPE command has been added to run FCOPY by simply typing "FCOPY" instead of "RUN FCOPY.PUB.SYS". The INFO parameter can be used with the new MPE command by simply entering the FCOPY command after the word "FCOPY". For example:

   ```
   :FCOPY  FROM=FROMFILE;TO=TOFILE;NEW;SUBSET=10,20
   ```

3. FCOPY will now print CIERR 975, "UNKNOWN COMMAND NAME" when an invalid MPE command is entered. Previously it would only print "**** COMMAND ERROR 0, 0" for that particular error.

A Look at EDIT/3000

With the 2028 release, several enhancements have been added to the EDIT/3000 subsystem:

1. EDIT/3000 now checks for invalid characters in the file names for the JOIN, KEEP, TEXT, and USE commands. If a file name contains an invalid character, it will print an EDIT/3000 message to indicate that the file could not be opened and FSERR 54, "INVALID FILE REFERENCE."

2. A COPY command has been added to EDIT/3000. It can be used to duplicate a range of lines from one location to another. The syntax of this command is:

```
CO[PY][Q] range TO linenumber [BY increment]
```

where range specifies the lines to be copied and linenumber is the location at which to put them. It is optional to indicate the increment by which the line numbers will be added. If not included, the increment will be the current DELTA value. The increment will be decreased if necessary to fit all line numbers into the location specified. The words "TO" and "BY" may be replaced with a comma.

3. EDIT/3000 will no longer allow garbage characters to follow the "Y" when it is entered in response to the question "PURGE OLD?".

4. The access type has been changed for the text, keep, and work files in the KEEP and TEXT commands in order to speed up those operations. However, more disc space will be required for the data stack during TEXTing and KEEPing. If it is undesirable for you to use that extra disc space you can specify MAXDATA=5588 in the :RUN command to keep the data stack from increasing in size. (The default MAXDATA is 8000.) The MAXDATA parameter could also be used to increase the possible stack size to as much as 31232 words. By increasing it you may be able to speed up the TEXT and KEEP operations even more for large files.

A Word About Image/3000

The 2028 installation tape includes these enhancements to IMAGE/3000:

DBUNLOAD allows up to 20 successful tape write retries before rejecting the tape and forcing the user to restart at the beginning of the reel. You can force DBUNLOAD to allow up to N<=32767 retries by using PARM=N:

```
:RUN DBUNLOAD.PUB.SYS;PARM=N
```

IMAGE/3000 does NOT support labelled tapes. In particular, DBSTORE does NOT write labelled tapes and DBRESTOR does NOT read labelled tapes.

This version of IMAGE replaces release 32215B.02.05. Systems running versions of MPE II should continue to use the latest version of IMAGE A.

****

More Enhancements for V/3000

by Carla Klein, General Systems Division

Version A.01.02 of V/3000 contains two enhancements which significantly improve the product's capabilities. The code record size has been expanded and new features have been added to support labeled functions keys on the 2626A terminal.

At compilation time, much of the information pertaining to a form must be stored in one code record. This record includes form information, the screen design, various field information, and all processing specifications. On version A.01.01, this record also includes custom error messages and has a maximum length of 8K bytes. When this maximum was exceeded, the message "Too many statements, code is too big" appeared at compilation time.

Now the maximum has been increased to 12K bytes (for non-KSAM forms files only) and custom error messages have been removed from the form code record (for KSAM and non-KSAM forms files). These changes should allow the forms designer to specify approximately 50% more processing specifications or additional fields (up to a maximum of 128) and more screen design. If the new limit is exceeded during compilation, FORMSPEC will now point to the statement within the form that caused the overflow. This should give you a good idea of how much should be deleted from the form to successfully compile it.

The second enhancement to V/3000 is the addition of features to specify and display function key labels on the 2626A terminal. When a form is displayed on a 2626A by ENTRY or an application program, an associated set of function key labels will be shown in the function key labels window. The user will be able to define global labels which will apply to all forms in a forms file except when form local labels have been defined for a specific form. Local labels will only be displayed when that form is the current form on the screen. If no labels are defined in a forms file, the default labels (f1-f8) will be displayed. The function key window itself cannot be disabled because this would also disable the function keys.

In FORMSPEC, the globals menu and forms menu will both contain a field which the user will fill in if they want to define function key labels. Then either the global or form function key labels menu will allow the user to enter 8 labels to correspond to the 8 terminal function keys.

Both FORMSPEC and ENTRY will display function key labels when they are run on a 2626A. The labels will contain the information that is found on the V/3000 templates.

Programmers will be able to retrieve and define function key labels using three new intrinsics: VGETKEYLABELS, VSETKEYLABELS, and VSETKEYLABEL. To use the intrinsics or display function key labels which have been defined in FORMSPEC, you must enable the function key display by setting COMAREA (10) (relative to 1) to 1 BEFORE your program calls VOPENFORMF. The function key labels will require about 260 bytes of stack space, which must be acquired when VOPENFORMF is called. Setting this word to 0 will indicate that the default labels (f1-f8) will be displayed. Once COMAREA (10) has been set and VOPENFORMF has been called, changing its value will have no effect unless the forms file is closed and re-opened. The calling sequence and functions of the intrinsics are as follows:

VGETKEYLABELS (COMAREA, FORM'OR'GLOB, NUM'OF'LABELS, LABELS)

VGETKEYLABELS will allow the programmer to retrieve the global function key labels or the labels for the current form.

Parameters:

COMAREA (logical array) The V/3000 communication area.

FORM'OR'GLOB (integer) An integer value indicating which type (form or global) of labels to retrieve.
0 - retrieve global labels
1 - retrieve labels for current form

NUM'OF'LABELS (integer) An integer value indicating how many labels are to be retrieved. This value should be from 1 to 8, inclusive.

LABELS (byte array) An array in which the labels will be passed back to the user program. The length of the array must be at least NUM'OF'LABELS * 16. Each 16 byte string represents a label of 2 lines of 8 bytes each.

VSETKEYLABELS (COMAREA, FORM'OR'GLOB, NUM'OF'LABELS, LABELS)

VSETKEYLABELS will allow the programmer to temporarily set new global or local function key labels. The label definitions set in FORMSPEC will not be changed. The temporary labels will be displayed the next time that VSHOWFORM is called.

Parameters:

COMAREA (logical array) The V/3000 communication area.

FORM'OR'GLOB (integer) An integer value indicating which type (form or global) of labels to set.
- 0 - Replace global lables
- 1 - Replace labels for current form

NUM'OF'LABELS (integer) An integer value indicating how many labels are to be set. This value may be 0-8, where 0 indicates that the labels defined in FORMSPEC should be set.

LABELS (byte array) An array in which the labels will be defined. The length of the array must be at least NUM'OF'LABELS * 16. Each 16 byte string will be displayed as a label of two lines of 8 bytes each.

New global labels will remain in effect until the forms file is closed or new global labels are defined with another call to VSETKEYLABELS. If new form labels are set, they will remain active until the next form is retrieved or new form labels are specified.

VSETKEYLABEL (COMAREA, FORM'OR'GLOB, KEY'NUM, LABEL)

VSETKEYLABEL allows the user to set an individual function key label. The new label will be displayed the next time that VSHOWFORM is called.

Parameters:

COMAREA (logical array) The V/3000 communication area.

FORM'OR'GLOB (integer) An integer value specifying which type of label is to be set.
- 0 - Set global label
- 1 - Set form label

KEY'NUM (integer) An integer value between 1 and 8 indicating which function key label is to be set.

LABEL (byte array) A 16-byte array to hold the text for the label.

Labels set via VSETKEYLABEL will remain active as described for VSETKEYLABELS.

Any application will be able to call the intrinsics, even if not run on a 2626A. This would enable an application to retrieve the labels and place them in display-only fields on a terminal which does not have the function key labels window.

COBOL II/3000 Enhancements

The 2028 installation tape includes the following COBOLII/3000 enhancements:

1. Changes were made in the following compiler error messages:

<u>New messages:</u>

- -Warning #009 "Files in multiple file tape clause must be sequential".
- -Warning #030 "Non-88 level item in FILE SECTION has value clause".
- -Warning #031 "Value clause on non-88 level item in LINKAGE SECTION ignored."
- -Warning #032 "Occurs clause used on 01 level item".
- -Serious error #407 "File name undefined or is not unique".
- -Serious error #411 "Compiler error: illegal internal (IDS) format".
- -Serious error #412 "Parameter must be 01 or 77 level item in LINKAGE SECTION".

<u>Modified messages:</u>

- -Message #351 "Reference not unique" now has the name of the non-unique data item inserted in the text of the message.
- -The message formerly numbered #372 "Too many value clauses is now numbered #459 and has a severity of disastrous instead of questionable.
- -The warning #017 "File name is undefined or is not unique" is now classified as a serious error and has been renumbered to #407.

2. Additional information is now printed with the symbol table map. Addresses of index names are now printed, as well as information on runtime storage layout, indicating location of compiler tables, buffers, and scratch areas. This information can be useful in debugging. For a more complete discussion, refer to the article by Tony Lemberger entitled "COBOL II/3000 - COBOL/3000 Stack Layout" on page 57.

3. In the ASSIGN clause of the SELECT statement, an 8 character COBOL word is now permitted for file-info-n. Formerly, this was required to be a nonnumeric literal. A nonnumeric literal is still legal and is required for more complex file information if specified in the ASSIGN.

4. If both a textfile and a masterfile are specified, a "+" will appear on the listing for all lines coming from the textfile.

5. KSAM is now required on the system for COBOLII compilations only if the programs being compiled contain COPY statement(s).

Enhancements to RPG/3000

The following ten enhancements have been added to RPG/3000:

1. A Header Record option has been added to request RPG to perform repeated *PLACE specifications just as RPGII/300 and IBM do. To request this, you put a "1" in column 51 of the Header Record.

Previously, RPG/3000 *PLACE repeated all those fields which preceeded it, including *PLACE entries. For example, two consecutive *PLACE entries repeat the initial field four times.

```
1st *PLACE repeats initial field -> fieldfield
2nd *PLACE repeats result of 1st -> fieldfieldfieldfield
```

Whereas, RPGII/300 and IBM *PLACE repeat only those previously specified fields that are not produced by a *PLACE. Two consecutive *PLACE entries will repeat the initial field twice.

2. When file locking (KLOCK/KNOLOCK) is specified on dsname'd files, the first File Description Specification for the dsname'd file must specify KLOCK or KNOLOCK so that the dsname'd file may be opened at runtime with locking enabled. This allows the other dsname'd file accesses to perform locking. If this condition is violated, a runtime abort will now occur with the added error message:

```
FATAL FILE ERROR, FILENAME=(name of file)
KLOCK/KNOLOCK NOT SPECIFIED FOR THIS FILE TO
ENABLE LOCKING FOR OTHER DSNAME'D FILE ACCESS
```

3. An enhancement has been implemented which allows you to programmatically put and find JCW's. The new Calculation operations are PUTJW and FNDJW which perform like the Intrinsics PUTJCW and FINDJCW (see the Intrinsics Manual for a discussion on these intrinsics and JCW's).

a. PUTJW

To put a JCW into the system JCW table, you specify "PUTJW" as the Calculation operation (col. 28-32), the value to be PUT in Factor 1, the name of the JCW to obtain the value in Factor 2, and Result Indicators which return status information on the PUTJW.

Factor 1 may be a value in the range 0 to 65535 or it may be a variable declared from 1 to 8 digits size with zero decimals. If you incorrectly try to put a value not in the range given above, RPG will give the runtime error message "Invalid Numerical Data" and not accomplish the PUT.

Factor 2 may contain an alphanumeric literal in quotes or it may contain an alpha variable. In any case, the JCW name must begin with a letter and TERMINATE WITH A BLANK.

At least one Result Indicator must be specified for this operation. After execution of PUTJW, the High Indicator will be on if a system table overflow occurred, the Low Indicator will be on if you try to put a value in a JCW name that doesn't begin with a letter, and the Equal Indicator will be on if the put was successful.

If the named JCW already exists in the JCW table, then its value is replaced by the Factor 1 value. Otherwise, a new entry is made in the JCW table with initial value equal to Factor 1.

b. FNDJW

To find a JCW in the system JCW table, you specify "FNDJW" in the Calculation operation field, the name of the JCW you are finding in Factor 2, the variable to obtain the JCW's value found in the Result Field, and Result Indicators which return status information on the FNDJW.

Factor 2 may be an alphanumeric literal or a variable, and the JCW name specified here must begin with a letter and TERMINATE WITH A BLANK.

The Result Field must be defined as numeric with zero decimal positions.

At least one Result Indicator must be specified for this operation. After execution of FNDJW, the High Indicator will be on if the JCW was not found, the Low Indicator will be on if the JCW name does not begin with a letter, and the Equal Indicator will be on if the JCW was found and the value was returned to the variable in the Result Field.

c. Summary Table for PUTJW and FNDJW

Factor 1	Calc OP	Factor 2	Res Fld	Result Ind
value	PUTJW	JCW name	blank	> table overflow < illegal name = successful
blank	FNDJW	JCW name	variable	> JCW not found < illegal name = successful

4. An enhancement has been implemented which allows you to programmatically obtain the MPE file number for any file used in your program. With this new capability, you can pass a file number to an external subprogram which may then call any of the system intrinsics neccessary for powerful file processing.

To accomplish this, you specify "FNUM" in the Calculation operation field (col. 28-32), a file name in Factor 2, and a numeric variable in the Result Field to which the MPE file number is returned.

Factor 2 must be a file name that is specified in your file description specifications. If it is not, error #623T will be issued.

The Result Field must be numeric with 0 decimal positions. The MPE file number of the file specified in Factor 2 is returned to this variable.

5. A variety of user-controlled file locking capabilities have been implemented. This enhancement allows you to perform conditional or unconditional locking and unlocking when you want to on Image, KSAM, or MPE files. For an Image file, you may perform locks at the data base, data set, or record level. For a KSAM and MPE file, you may only lock at the file level.

An unconditional lock is a lock request on a data object such that if the data object is already locked by another process, the lock request will wait in a queue of all lock requests for the data object until it is its term to lock. All the while the process which requested the unconditional lock is suspended until the lock request is granted. The time the process is suspended may seriously degrade performance in an interactive processing environment. To overcome this, conditional locks should be performed.

A conditional lock is a lock request on a data object such that if the data object is already locked by another process, the lock will fail and a resulting Indicator will turn on to inform you of the situation. The process is not suspended, so it may now proceed to perhaps process a different data object, returning later when this data object is available.

To perform all locking and unlocking yourself, you use the new calculation operations LOCK and UNLCK.
In the discussion below, every use of LOCK also refers to UNLCK.

a. Locking an Image Data Base

To lock an Image data base, you specify "LOCK" in the calculation operation field, a file name in Factor 2, a data base name in the Result Field, and Result Indicators.

The file name specified in Factor 2 must be a file name given in your File Specifications which describes the Image data base to be locked with K extensions.

The data base name specified in the Result Field must be the data base name given in the KIMAGE specifications for the file in Factor 2. The data base name here in the Result Field is the data base you wish to lock. It must be a literal string, not a variable. THE NUMBER OF CHARACTERS IN THE DATA BASE NAME MUST BE SPECIFIED IN THE RESULT LENGTH FIELD (col. 49-51).

The KIMAGE specifications for the Image data base must also specify "L" (enable locking) as the Image open mode in column 66.

Result Indicators must be specified for this operation. The High Indicator is optional, but one of either the Low or Equal Indicators is required. The presence of the High Indicator specifies conditional locking, whereas its absence specifies unconditional. The Result Indicators return the following status information:

Result Indicator ON		Status Information
High	>	lock failed - already locked by another process (conditional)
LOW	<	lock failed - not opened with locking enabled or need MR CAP
Equal	=	successful lock

b. Locking an Image Data Set

To lock an Image data set, you specify "LOCK" in the calculation operation field, a filename in Factor 2, and Result Indicators. The Result Field must be blank.

The file name specified in Factor 2 is the name of the data set to be locked as described in the File Description Specifications. Furthermore, the KIMAGE specifications for the data base which contains the data set must specify "L" (enable locking) as the Image open mode in column 66.

Result Indicators must be specified, and their individual purposes are the same as explained above in Locking an Image data base.

c. Locking an Image Record

To lock an Image record, you specify "LOCK" in the calculation operation field, a search key in Factor 1, a data set file name in Factor 2, and Result Indicators. The Result Field must be blank.

The key specified in Factor 1 is used as the search key for the record(s) which contain that key. All records with items that equal the search key will be locked.

The file name in Factor 2 is the name of the data set you wish to search for records to lock. This file must be described in the File Specifications with KIMAGE and KITEM extensions. The KIMAGE specifications must specify "L" as the Image open mode in column 66 to enable locking. KITEM specifies the IMAGE item field to be used against the search key.

Result Indicators are required and are the same as explained above in the first section on Locking an Image Data Base.

d. Locking a KSAM file or MPE file.

To lock a KSAM file or MPE file, you specify "LOCK" in the calculation operation field, the file name in Factor 2, and Result Indicators.

The file name in Factor 2 must be a name given in the File Description Specifications. If the file is described there as a KSAM file, then this is a KSAM lock operation, else it is an MPE file lock.

The File Specifications for the file to be locked must specify a KNOLOCK extension to enable locking. Furthermore, Result Indicators are required and are the same as explained above in the first section on Image data base locking.

e. Programming Cautions

1. Multiple RIN Capability

   If you wish to request more than one outstanding lock, you must have MR special capability. See the Image Reference Manual for a discussion on MR CAP.

2. Dsname'd Files

   If you request RPG to perform all locking for you on a cycle by cycle basis, you cannot do your own locking in the Calculation Spec.

f. New Error Messages

```
ERROR 681T - LOW OR EQUAL RESULT INDICATOR MUST BE
              SPECIFIED FOR THIS OPERATION

ERROR 682T - IMAGE FILE LOCKING MODE IS NOT L TO
              ENABLE LOCKING ONLY

ERROR 683T - FILE NOT DESCRIBED WITH NOLOCK TO
              ENABLE LOCKING

ERROR 684T - DB NAME IN RESULT FIELD MUST BE SAME
              NAME SPECIFIED IN IMAGE SPEC FOR FILE
              IN FACTOR 2
```

g. Summary Table of LOCK/UNLCK

Type of Lock	Factor 1	Oper	Factor 2	Result Fld
Image Record	key	LOCK	filename	blank
Image DataSet	blank	LOCK	filename	blank
Image DataBase	blank	LOCK	filename	data base
KSAM file	blank	LOCK	filename	blank
MPE file	blank	LOCK	filename	blank

Result Indicator ON		Status Information
High	>	lock failed - already locked by another process (conditional)
Low	<	lock failed - not opened with locking enabled or need MR CAP
Equal	=	successful lock

h. Examples

```
FILE DESCRIPTION SPECIFICATIONS
=================================================================

FINFILE  IP  F      80          DISC
FKSDATA1 IC  F      72R14AI   51 DISC
F                                           KIMAGE EMP1  L5
F                                           KITEM  CITY

=================================================================

CALCULATION SPECIFICATIONS
=================================================================

**  IMAGE DATA BASE LEVEL LOCKING
**
C                       LOCK KSDATA1   EMP1    4  737475
C   75      CITKEY      CHAINKSDATA1              8058
C   75                  UNLCKKSDATA1   EMP1    4     76

=================================================================
```

6. A Header Record option has been added which allows you to specify if overflow should occur when the overflow line has been passed. (Before this enhancement, overflow always occurred when the overflow line was either reached or passed.)

To specify overflow should occur when the overflow line is passed, you put a "P" in column 50 of the Header Record. This feature is primarily for IBM compatibility. Just leave column 50 blank to specify overflow should occur when the overflow line is either reached or passed.

7. A file described as an Output Chain file must not be a KSAM, IMAGE, or INDEXED file. Error #280T will now be issued if this condition is violated.

8. Warning #764 will now be issued when an Output, Update, or Combined file is not referenced in the Output Specifications.

9. An enhancement has been implemented which allows you to reset UDATE, UMONTH, UYEAR, and UDAY as you wish at run-time. To do this, you put an "F" in column 17 of the Header Record and provide RPG with a formatted date record which specifies the new values for the date. If column 17 is blank, UDATE is loaded from the system date.

The "F" in column 17 directs RPG to obtain the formatted date record of new values from the formal file designated as "RPGUDATE". At program initialization time, RPG will open this file, read the date record into an 80 byte buffer, and then close the file. You may use the MPE FILE command to equate the formal file name RPGUDATE to the actual file that holds the date record before running your program. For example:

```
     :FILE RPGUDATE = DATEFILE.PUB
or   :FILE RPGUDATE = $STDIN
```

Equating RPGUDATE to $STDIN allows you to enter in the date record at your session terminal, or to imbed it in your job stream after the RUN command (after USWITCH records, if any). This usuage in a job stream allows you to simulate the IBM //DATE OCL command.

In the date record, the first numeric character indicates the beginning of the date values (does not have to be in column 1). The date may be entered on the date record in either of the following formats:

mmddyy	- mm is month (01 to 12) - dd is day (01 to 31) - yy is year (00 to 99) All six digits are required.
mm#dd#yy	- # represents a separator consisting of one or more non-numeric characters. In this format, leading zeroes may be omitted.

If mm, dd, yy are all equal to 0 (zero), or the formal file RPGUDATE is equated to $NULL, then RPG will obtain the date from the system date. This feature allows you to default back to the system date without removing the "F" in column 17 and recompiling.

Following are examples of valid date records which RPG obtains from the formal file RPGUDATE. What appears on each separate line is exactly the contents of a single date record. Assuming the system date is 4/10/80, all of these examples will cause UDATE to be initialized to 041080.

```
000000    => DEFAULT TO SYSTEM DATE.    (041080)
041080
// DATE 041080       IBM FORMAT
DATE IS 041080 = APRIL 10, 1980.
FORMAT OF DATE IS MMDDYY = 041080.
4/10/80
MONTH IS 4;  DAY IS 10;  YEAR IS 80;
0 0 0     => DEFAULT TO SYSTEM DATE.    (041080)
```

Following are examples of invalid date records along with explanations.

```
4/10          (year is missing   )
4/10/1980     (year must be 0 to 99)
040080        (day must be 01 to 31)
```

If the date file does not exist or the date file contains no records or the date is not formatted correctly then your RPG program will abort immediately with one of the following error messages:

```
"UNABLE TO OPEN FILE RPGUDATE TO OBTAIN DATE."
"UNABLE TO READ DATE RECORD IN FILE RPGUDATE."
"INVALID DATE RECORD IN FILE RPGUDATE."
"INVALID MONTH ON DATE RECORD IN RPGUDATE."
"INVALID DAY ON DATE RECORD IN FILE RPGUDATE."
"INVALID YEAR ON DATE RECORD IN RPGUDATE."
```

10. This enhancement allows you to enter the USWITCH record in the format used on the IBM System/3. In this format, switch settings are coded in an 8 byte string, where each byte represents one user switch (U1 to U8). The valid entries for each switch position are:

0	- set this user switch off.
1	- set this user switch on.
X	-do not change the setting of this user switch. This is equivalent to our current "JCW" setting.

This 8 byte string would appear after the "USWITCH:" header; blanks may separate the two.

Examples:

```
          USWITCH:     00001111
          USWITCH:     0011XX01
```

NOTE: All Uswitches that are not used in the RPG program always default to OFF, even if you set them on with the Uswitch record. This implies that you may only set Uswitches that are used in the RPG program that is about to execute.

Introducing IML/3000

By Connie Ishida, General Systems Division

IML/3000 Interactive Mainframe Link is a new data communications product. It provides interactive access to remote IBM mainframes and program-to-program communication between HP3000 applications and applications running on the remote IBM host. IML uses IBM's bisync multidrop communications protocol and standards for remote 3270 cluster controllers with attached devices.

There are two modes of IML usage, programmatic access and IDF (Inquiry and Development Facility). The first mode provides programmatic access from user programs to a remote host program through calls to a set of IML Intrinsics. The Intrinsics are callable from COBOL, BASIC, FORTRAN or SPL. There are thirteen Intrinsics for use with standard waited I/O and three other Intrinsics for use only with no-wait I/O. These Intrinsics provide high-level access to the fields of data within the screen. The communication data stream from the host is decoded and placed into a screen image maintained internally by IML. The IML Intrinsics provide access for the user program to the data within the screen image. Most of the Intrinsics are field oriented; examples are READFIELD and WRITEFIELD.

Much of the activity of the 3270 cluster controller, including protocol handling and data stream decoding is offloaded from the HP3000 onto the Intelligent Network Processor (INP). This usage of the INP moves much of the overhead associated with a data communications product out of the 3000. IML requires an INP; it is not supported on the SSLC.

The second mode of IML usage is the Inquiry and Development Facility (IDF), which allows HP264X block mode terminals to be used as though they are 327X type terminals for casual direct access to the remote host. IDF is written in SPL as an application program, which utilizes the IML Intrinsics in no-wait I/O mode. IDF is intended for occasional access to host data bases and host applications. IDF is a tool for casual direct host access and a tool to aid in IML program development. IDF is not a direct replacement for a 3270 system and should not be used for dedicated data entry.

The major feature of IML/3000 is the new set of IML Intrinsics which allow development of a user program on the HP computer to communicate with an existing or new IBM host application. IML provides interactive programmatic access to an existing host data base or application. IML contributes to distributed data processing by allowing users to move much of their teleprocessing from their remote host to their local HP3000. A user program which uses IML Intrinsics may also call IMAGE, QUERY and V/3000 Intrinsics to distribute a data base between the remote host and the local 3000.

An IML/3000 Reference Manual (32229-90001) is available. The above discussion is only a brief introduction to the product. Refer to the IML Reference manual for further information.

Now Introducing — MRJE on the INP

By Jitendra Singh, General Systems Division

MRJE/3000 (Multileaving Remote Job Entry) is a product that allows multiple users batch access to a remote processor in a multiprogramming environment. The HP 3000 appears as a HASP workstation to the host processor. Although MRJE/3000 has been available on the HP 3000 SeriesII/ III for over two years, it works only with the SSLC (Synchronous Single Line Controller).

With the 2028 release, MRJE/3000 will be supported on the INP (Intelligent Network Processor) at communication speeds up to 9600 bps. This will provide the HP 3000 Series 30 and 33 the same MRJE capability currently available on the Series II/III. In addition to the expanded MRJE/3000 support for the HP 3000 product line, the 2028 release provides for more efficient CPU utilization for MRJE activity, due to a new data compression algorithm.

How to Recreate an MRJE/3000 Configuration File for 2028 Tape Installation

By Bob Mayer, General Systems Division

When the 2028 Installation Tape is distributed, it will be necessary to recreate the configuration file because of a structural change. The structural change is to accommodate unsolicited output designators for seven printer streams and for seven punch streams.

First, before installing the 2028 tape, and before recreating the configuration file, establish a host session with an =MRJE START so that all outstanding output may be received with job management in effect.

Second, secure hardcopy of the contents of the configuration file and possibly the job log file for all hosts. Hardcopy may be obtained as follows:

Enter a job or session with a user whose capability includes OP.

:LISTF MRJECON@.PUB.SYS	This optional step will identify all configuration files.
:FILE LISTING;DEV=LP	Use this for sessions.
:MRJE	
	Manager mode must be in force for MRJE/3000.
#HOST HOSTID	Selects a particular host.
#DISPLAY CONFIG	Lists the configuration file in its entirety.
#DISPLAY JOBLOG	Displays information about all jobs that were submitted. This step may be omitted.
#EXIT	

Third, install the 2028 tape, which must contain a compatible set of MRJE/3000 and CS programs and routines.

Fourth, recreate all of the configuration files. Use the following as a guide:

:HELLO MANAGER.SYS,PUB	In order to recreate all configuration files a user must be in this group and account, and have this name. This may be a job or a session. This user must be configured with OP capability, but need not be configured with SM capability.
:FILE LISTING;DEV=LP	This command is needed for sessions only.
:MRJE	Manager mode must be in force.
#NEW HOSTID	This is used to enter a dialogue to create or recreate the configuration files for a given host machine. Use the hardcopy created in the second step, above, for a script to provide identical item values.
#ALTER ITEMLIST	Use this command to specify the items in the configuration file to be changed that were not already requested in the new dialogue.
#DISPLAY CONFIG	Use this command once the configuration file has been recreated . Return control to the operating system.
#EXIT	Use this only is you need immediate hardcopy from a session.

At this time output will be generated of the new configuration file. Compare this with the older version configuration file listing of the same host. If there are any items within the configuration file that are in error, correct them now. The user interface of MRJE/3000 should be re-entered and the ALTER command should be used to do this.

This step should be repeated for all host configurations.

Finally, a test job should be submitted for each configured host, transmitted to the host, and received from the host to assure that recreation of the configuration file has been completed successfully.

Introducing Network File Transfer (NFT)

An overview of Network File Transfer -- a new product facility to transfer files between HP3000 systems.

by Dennis Carelli, General System Division

Network File Transfer (NFT) is an enhancement to Distributed Systems/3000 which provides the ability to transfer disc files between HP 3000 systems more efficiently than the FCOPY utility and remote file access. The NFT program, when initated over a DS/3000 data communication link, can copy a file to or from any other HP 3000 computer which also has the NFT capability.

NFT copy operations can be initiated from sessions, jobs, or programs. The terminal user/program need not be on the same system as the file or the system to which the file is being transferred; you can initiate the transfer from a third HP 3000. When a copy request names a remote source file, the DS line to that computer must be open and a remote session must exist. The same is also true when a remote target file is specified. Intra-system file transfers can also be accomplished using NFT.

Key features of NFT:

- There is only one NFT command to learn - :DSCOPY.

- There are two intrinsics: DSCOPY and DSCOPYMSG. The intrinsics are callable from programs written in SPL, COBOL, FORTRAN and BASIC.

- The files referenced by a DSCOPY command (or intrinsic) may reside on public or private volumes.

EXAMPLES:

To copy a file named SFILE (on the local system) to the computer attached to the dsline SYSB, and name the new file SFILE, enter:

```
            :DSCOPY SFILE TO ,SYSB
```

To copy a file from one remote system to another enter: (In this case, the communications lines to both remote computers must be open and a remote session must exist on each system.)

```
            :DSCOPY SFILE,SYSA TO TFILE, SYSB
```

To make a local copy of SFILE and name the new file TFILE, enter:

```
            :DSCOPY SFILE TO TFILE
    or      :DSCOPY SFILE; TFILE
```

* *

Announcing New Data Capture Procedures

An overview of the HP 3000 Data Capture Procedures which simplify the interfacing of HP 307X terminals to the HP 3000.

by Ron Fountain, General System Division

Data Capture Procedures allow HP 3000 users high level access to HP 307X data capture terminals from programs written in Cobol, Cobol II, Fortran, and SPL. These procedures manage the interface between the user program and the terminal. They handle the reading of data from, and the transferral of data to, the terminal. The procedures provide a convenient method of handling errors should they occur. (Prior to this, complex escape sequences were required to access data capture terminals from an HP 3000 application program).

With these procedures on the 3000, such applications as inventory tracking, shop floor loading, and labor reporting can be enhanced through the use of data capture terminals. Also, because the procedures are integrated into the standard software, there is no need for users to learn new languages or operating systems.

The following 14 procedures are used to control the data capture terminals:

DCACLEARISP	Clear the display.
DCACLOSE	Close the terminal.
DCAOPEN	Open a terminal.
DCACONTROL	Perform control operations on a terminal.
DCADISPLAY	Display a message on terminal display.
DCAERROR	Return an error message.
DCAPRINT	Print data on the terminal printer.
DCARBADGE	Read from type V badge reader.
DCARBAR	Read from bar code reader.
DCARKEYBOARD	Read from keyboard.
DCARMAGBADGE	Read from magnetic badge reader.
DCARMULTI	Read from multifunction reader.
DCASETKEYS	Activate special Function Keys as terminator.
DCASETLIGHTS	Turn prompting lights on/off.

These procedures are designed so that only one terminal peripheral can be accessed at a time. These peripherals include the keyboard, multifunction reader, thermal printer, numeric and alphanumeric display, CRT display, type V badge reader, bar code reader, and magnetic badge reader. Thus, if the keyboard has been activated with the DCARKEYBOARD procedure, then all other input devices, such as the multifunction reader, are disabled.

The user of these procedures should be familiar with the capabilities of the 307X terminals used by the applications. In designing applications the user must know which terminal features are available for each device. The following features of 307X terminals are not supported using the HP 3000 Data Capture Procedures:

Communications test,
Clock-on-read mode for the multifunction reader,
Multifield read mode,
Buffered mode for the time reporting terminal,
Self-test, and
Cursor positioning for the CRT display.

For additional information, please consult the Data Capture Procedures Reference manual (32243-90001). This manual includes instructions for connecting the 307X terminals to the HP 3000, a description of each Procedure and a sample program using them.

****

CIS/3000 Software Update

CIS/3000 HP32902A.00.03
MAT 2004, 1980, N0ON902A.HP32902.SUPPORT

A. CORRECTIVE SOFTWARE CHANGES.

1. SR #4547 --CISSCH-- After changing edit information using Format 08, CISSCH would not allow user to specify a "Show" or "Add" transaction code. This problem was corrected.

2. SR #5263 --CISSCH-- FORMAT08 would not allow a user mutiple "Changes" to edits in a table. Also CISSCH would not accept the first "Change" transaction; it had to be entered twice. These problems have been corrected.

3. SR #5264 --CISCRS-- On Formats 33 and 39 if an "Add" transaction was done after a "Change", for the same ID number, the "Add" acted as a "Change". This problem has been corrected.

4. SR #5265 --CISCRS -- Format 39 of CISCRS produced error messages containing a portion of a previous message. This problem has been corrected and error messages now contain the IMAGE condition code which describes the error.

5. SR #5266 --CISSCH-- FORMAT07 would not allow a new item to be added to the data base with an "add" transaction code. In addition, the enter key had to be depressed twice when a "Change" transaction code was entered. Both problems have been fixed.

6. SR #5267 --CISBAT and CISCRS-- Only four digit numeric phone extensions were allowed to pass the phone extension edits. The numeric edits have been altered to allow one to four digit numeric phone extensions to pass.

7. SR #6009 -- CISRPT2 -- Student schedule printed the number of units for a variable unit course from the COURSE-MSTR data set rather than from the STUDENT-COURSE data set. This problem has been corrected.

8. SR #6010 -- CISRPT2 -- Student Schedules Report printed an unnecessary additional page if the number of classes in which a student was enrolled filled the schedule form exactly. This problem has been corrected.

9. SR #6256 --CISGRD -- CISGRD displayed negative IMAGE Condition Codes as positive in error messages. This problem has been corrected.

10. SR #7022 -- CISGRD -- In CISGRD final grades could not be entered if miterm grades were already posted. This problem has been corrected.

11. SR #7250 -- THe SECTION-FEE field has been extended to allow five rather than three digits to the left of the decimal place.

12. SR #7692 --CISDET-- When the Security File is set up to allow access only to Format 19, Program CISDET displayed the error message "Invalid Format Request for CISDET" when Format 19 was requested. This problem has been corrected.

13. SR #9596 --CISBAT-- When using CHANGE CARD STUDENT, S50C, the "STATUS" field in the STUDENT data set was was not updated, although the STUDENT-MSTR data set and "VALUE-STATUS" and "DEPT-STATUS" fields in the STUDENT data set were. This problem has been fixed.

14. SR #9597 --CISRPT1-- When running CISRPT1 to obtain selected class lists and a class with zero enrollment was encountered, the job aborted and the remaining classes were flushed. This problem has been corrected.

15. SR #10465 --CISBAT-- When Card 95 was run through CISBAT to change a "STUDENT-ID" already on the Data Base, CISBAT looped and the program had to be aborted. No indication of a problem was given. This problem has been corrected.

16. SR #11283 --LINK FILE B02B902A --Program file P02P902A did not include USL U51U902A, and B02B902A had to be updated to solve this problem. The problem was corrected in this release.

B. KNOWN PROBLEMS.

None.

RPG/3000 Programming Optimization

BY David E. Walmsley, General Systems Division

The RPG compiler is a two pass translator. Pass 1 is comprised of eight scanners, each scanner being responsible for syntactically checking one of the card specification types:

H - control specifications
F - file specifications
E - file extension specifications
L - line counter specifications
I - input specifications
C - calculation specifications
O - output specifications
A - table/array filename specifications

The scanners also generate intermediate code into a temporary disc file.

The second pass uses the intermediate code from the first pass to generate both machine code and calls to procedures in the RPG run-time library. This code is placed in the USL file specified by the user, and represents, along with the optional listing, the output from, or the product of, the compiler.

The segmentation of the object code generated by the compiler is chosen by the compiler subject to the SEG=1,2,3,4 compiler directive which directs the compiler to limit the size of each segment to 1K, 2K, 3K, 4K words respectively. The default maximum code size is 4K words.

The number of segments generated for a particular user program is thus controlled by the directive with the segment names being, RPG'00, RPG'01, etc. Each segment contains a variety of procedures/ entry points, with the last generated segment containing the outer block or main program (RPG'OB). There is a set of procedures common to all RPG generated USL files:

R'OUT - main procedure to handle output specs
R'INT - main procedure to handle input specs
R'CALD - main procedure to handle CALC specs

In addition, there is a separate procedure/entry point for each tag declared in the program. If the input specs describe multiple files and/or several record group sequences for a particular file, these entry points are named I'0001, I'0002 etc. The user is typically not concerned with the content of each segment, although the size of each segment can have a dramatic effect on individual program execution time and overall system performance.

Each time, during execution, there is a call to a procedure, there is a given amount of 'non-productive' overhead induced to allow for the context switch. If the call is to a procedure which is within the same segment, this time is significantly shorter than if the procedure is within another segment. In order to execute the called procedure, the segment containing the procedure must be resident in memory. If, at the time of the call, the segment is absent, then the memory manager portion of MPE must be invoked to make the segment present. On a lightly loaded system with a sizable amount of free memory, this is not a problem, because the memory manager's working set algorithm tries to guarantee that the segment will be in memory when you request it.

On the other hand, on a heavily loaded system with virtually no free memory, the memory manager's task is much more complex. It is therefore quite likely that the requisite segment will have to be fetched into real memory from virtual memory on the system disc. It can thus be seen that the number of program segments and the size of each segment conflict with one another with reqards system performance. A general rule of thumb for RPG program segmentation is to determine how the flow through various program specifications will be accomplished. For example, if the program will usually follow the normal RPG cycle, then the 4K work segment size is usually optimal. If, however, the program does not follow the cycle because of demand reads and exception output, then an attempt should be made to keep the procedures required for these operations in the same segment.

The amount of object code generated and the size of the individual segments can also be reduced by paying attention to the content of the source program.

File structures play a great part in determining overall efficiency, both by virtue of the amount of code generated and by the amount of time required for a particular operation.

A file should be described in the file specs such that only the specs necessary for that particular use of the file are included. For example:

- Sequence checking should be performed only if absolutely required.

- The logical record length should be just large enough to include the largest record specified for a print file, not the record length of the device. This is because the compiler emits code to blank out the entire record between cycles, which is time consuming.

- The most efficient record length for disc files is 128 words or multiples of 128 words.

- Files which are to be randomly processed should be unblocked whether they are indexed or not. This is because the file system will always read a physical block of records from disc to a buffer before passing the requested record, which wastes time.

- Files which are processed sequentially should be blocked whether indexed or not in order to reduce the number of physical reads.

- If it is necessary to process a file sequentially several times using limits in order to locate a particular record, (for example to read a KSAM file), a technique involving the use of the file sharing group field 'DSNAME' can be used to tie two separate file spec files to the same physical file. The example involves defining the file first as an input chained file, then as a demand file. In CALC specs, the file can be chained to with a key value, then demand read in a loop until either the required record is found, or until the key field changes value.

The following facts should be kept in mind regarding input specifications:

- The number of record types for a particular file should be kept to a minimum, such that only those record types which will be processed by the program will be included.

- The number of fields for a record type should be minimized by referencing only those which are used in CALC or at output time, or by reusing fields from record type to record type and from file to file.

- Input processing can be further speeded up by specifying the complete character rather than the zone or digit portions on the record id.

- Do not specify a field as a numeric field unless the field will actually be used in CALC arithmetic operations.

- Avoid using, as much as possible, field data types such as unpacked numeric and binary. This is because RPG does all arithmetic as packed decimal. Therefore all non-packed fields must be packed before the arithmetic operation can take place.

The calculation specs offer the most significant area in the program to effect run-time performance improvement. Examples of this include the following:

- Minimize the number of work fields by reusing those fields.
- Use the minimum size possible for a work field.
- Do not define a work field as numeric if it will never be used in an arithmetic operation.
- A GOTO tag operation should be used to branch around a series of CALC operations rather than conditioning each CALC line with an indicator.
- Factor 2 should contain the field with the least number of digits in a MULT CALC operation.
- To zero a field, the field should be subtracted from itself rather than blanked after, using Z-ADD or by moving zeros into the field.
- When searching a table for a matching entry the table should be loaded unordered, with the entries having the highest probability of qualifying being placed first in the table.
- The length of table entries and the number of such entries should be kept to a minimum so as to reduce the size of the run-time stack.
- The compiler emits a warning when an indictor is defined but is not referenced in the program. These indicators should be eliminated.
- If only one record type is to be processed, do not condition CALC specs with the indicator specified on the input specs.
- As in most programming languages, internal subroutines are a valuable tool to the programmer. In RPG they allow both single indicator invocation of many CALC operations and also the opportunity to avoid having to repeat the same section of CALC lines in several places.
- When a CALC operation is multiply conditioned by several indicators, the indicator with the highest probability of conditioning the CALC should be specified first.

Output specifications can be optimized through the use of the following:

- When conditioning output, place the most frequently used indicator first in any sequence of "OR" lines.

Other examples of code reduction and execution time enhancements include:

- If the line number identification is not required for run-time error identification, then column 20 on the control record should contain an "N". This will reduce code space and will have a relatively strong impact on reducing execution time, because the compiler otherwise emits code into the USL for each line of the source program. There are two words of code for each such program line.

****

Using Editor on BASIC Programs

The following procedure will enable one to use EDITOR in making changes to BASIC programs.

By Adrian Thomas, General Systems Division

The HP 3000 user can use the EDITOR subsystem for modifying all types of programs. The following is a list of steps to perform in order to take advantage of this facility for BASIC programs.

1. In MPE, build an EDITOR text file that will contain as ASCII version of your BASIC program.

```
:PURGE ASCIBASP
:BUILD ASCIBASP;REC=-80,16,F,ASCII
```

2. In the BASIC Interpreter, get your BASIC program and list it out to the ASCII file you just built.

```
:BASIC
>GET BPROG
>LIST,OUT=ASCIBASP
>EXIT
```

3. In EDITOR, text in the ASCII file, make your changes and keep it unnumbered to the same file.

```
/TEXT ASCIBASP
<<make changes>>
/KEEP ASCIBASP,UNN
/EXIT
```

Note: This step can be expedited by using an entry point to EDIT/3000 that will automatically text in a file when the first command is typed and keep the file when exit is typed.

```
:FILE EDTTEXT=ASCIBASP
:RUN EDITOR.PUB.SYS,BASICENTRY
<<make changes>>
/EXIT
```

A User Defined Command (UDC) could be created to facilitate initiating the EDITOR with this entry point.

```
          BED
          OPTION LIST
          FILE EDTTEXT=ASCIBASP
          RUN EDITOR.PUB.SYS,BASICENTRY
```

4. In BASIC, use the XEQ command to bring your changed ASCII file into the Interpreter, and save your new BASP file.

```
          >XEQ ASCIBASP,ECHO
          >SAVE BPROG!
```

Note: This step can also be expedited by using EDITOR to create a BASIC XEQ file.

```
          :EDITOR
          /ADD
              1     XEQ ASCIBASP,ECHO
              2     SAVE BPROG!
              3     //
          /KEEP IN2BASP,UNN
          /EXIT
```

The following sequence of commands show a BASIC program being modified with the aid of the files declared in the preceding steps and a couple of UDC's.

```
:EDITOR

HP32201A.7.05 EDIT/3000  WED, SEP 26, 1979,   4:36 PM
(C) HEWLETT-PACKARD CO. 1979
/TEXT UDCFILE
/LIST ALL
    1     BINIT
    2     OPTION LIST
    3     PURGE ASCIBASP
    4     BUILD ASCIBASP;REC=-80,,,ASCII
    5     BASIC
    6     ******************************
    7     BED
    8     OPTION LIST
    9     FILE EDTTEXT=ASCIBASP
    10    RUN EDITOR.PUB.SYS,BASICENTRY
    11    *****************************
/EXIT

:END OF PROGRAM
:SETCATALOG UDCFILE
:BINIT
:PURGE ASCIBASP
:BUILD ASCIBASP;REC=-80,,,ASCII
:BASIC
```

```
HP32101B.00.10(4WD)  BASIC  (C)HEWLETT-PACKARD CO 1978
>10 PRINT "FORE SCORE AND SEVEN YEARS AGO ..."
>SAVE BPROG!
>LIST,OUT=ASCIBASP
>EXIT

:END OF SUBSYSTEM
:BED
:FILE EDTTEXT=ASCIBASP
:RUN EDITOR.PUB.SYS,BASICENTRY

HP32201A.7.05 EDIT/3000  WED, SEP 26, 1979,  4:36 PM
(C) HEWLETT-PACKARD CO. 1979
FILE UNNUMBERED
/LIST ALL
    1     BPROG
    2        10 PRINT "FORE SCORE AND SEVEN YEARS AGO ... "
/MODIFY 2
MODIFY     2
   10 PRINT "FORE SCORE AND SEVEN YEARS AGO ... "
              RUR
   10 PRINT "FOUR SCORE AND SEVEN YEARS AGO ... "

EXIT
ASCIBASP ALREADY EXISTS - RESPOND YES TO PURGE OLD AND KEEP NEW
PURGE OLD? YES

END OF PROGRAM
:BASIC

HP 32101B.00.10(4WD)  BASIC  (C)HEWLETT-PACKARD CO 1978
:XEQ ASCIBASP
BPROG
??? <<Note: The first record of ASCIBASP is the program title
            and is therefore not recognized by BASIC as legal
            input as indicated by the '???'.>>
    10 PRINT "FOUR SCORE AND SEVEN YEARS AGO ... "
>RUN
FOUR SCORE AND SEVEN YEARS AGO ...

>EXIT

:END OF PROGRAM
```

Direct character editing can also be performed on the HP 264X Series Terminal (refer to their Reference Manual for further information).

Tips on :ALLOCATE

BY Mark Cousins, Neely Cupertino

In an earlier issue, we discussed the :QUANTUM command and how it can be used to improve performance of the HP 3000. This issue we will discuss another MPE command that can have a major impact on the performance on the machine, the :ALLOCATE command.

When you issue the :RUN command (or call the CREATE intrinsic), MPE invokes a system process called the LOADER. The loader, among other things, reads the program file and attempts to resolve any remaining external references. (An external reference is a call to a procedure that does not exist in the program file itself; for example, if your program calls FOPEN, this is an external reference to FOPEN since its code does not exist in your program.) This involves locating the segment containing the procedure in one of the segmented libraries, resolving any external references it may have, assigning the segment a code segment table (CST) entry, and placing the external label for the procedure in the segment transfer table (STT) for the program segment which calls the procedure. Since an external procedure may itself have unresolved external references, the process of loading a program can be quite time-consuming; the COBOL compiler requires about 150 accesses to the system disc when it is initially loaded.

Additionally, the Loader is run as a separate process and it assures that only one copy of itself is active; that is, only one program load can be occurring at a time. While the Loader is active, it runs at a very high priority. You can see that between the disc and CPU requirements for a program load, the system might not be able to do anything else while this is occurring (and a complicated program can take anywhere from 30 to 90 seconds to load).

Keep in mind that this is done each time a user runs a program that is not currently active elsewhere in the system. For example, if a user on the DP staff were constantly invoking BASIC to test a program, then FCOPY to check the results, s/he might be using 90 percent of system resources for loading the subsystems alone!

The :ALLOCATE command allows you to tell the system to resolve all externals in a program, and to save this information from run to run. Thus a program that normally takes 30 seconds to load might take 2 seconds if it is already running or :ALLOCATED. What should be :ALLOCATED? That can be answered by describing what :ALLOCATE COSTS. It will always use as many code segment table extension (XCST) entries as there are program segments. If the program would normally use CST entries then :ALLOCATE will also; however, a program (like the SPL compiler) that calls only system routines such as the file system will not use any CST entries since these are pre-loaded with the system when it is started. So, if you have a CST problem, be careful with what you allocate. If you don't have a CST problem, allocate a program when in doubt - the cost is very, very small and the benefits can be enormous.

But what if your program calls something in a group or account SL? How do you allocate it? Easy. Have someone run the program. Then, while the program is active, have someone with OP capability allocate it (by using BREAK, even the original user can do this). You will get a message indicating which SL is in use, but allocate actually occurs. (a side benefit of this is that future users need not type "LIB=X" when they run the program - it will use the correct one automatically)

NOTE: Beware that the SL allocated with the program will be used for all future runs (i.e. if you :ALLOCATE with a group or account SL, you cannot override this by specifying "LIB=X").

Suggest that you have a standard job that is run at system startup to allocate most of the PUB.SYS subsystems. You should notice a significant performance improvement, expecially for systems that have a lot of development work.

COBOL II/3000 — COBOL/3000 Stack Layout Comparisons

By Tony Lemberger, General Systems Division

This article will compare and contrast the stack layouts of the COBOL/3000 and the COBOL II/3000 compilers.

COBOL/3000 is Hewlett-Packard's implementation of the 1968 ANSI standard for COBOL, and COBOL II/3000 is the Hewlett-Packard extended implementation of the 1974 ANSI standard for COBOL.

NOTE: The information contained in this article relates to COBOL/3000 version HP32213C.02.02, and COBOL II/3000 version HP 32233A.00.01. This information is subject to change based on future changes to the COBOL/3000 and COBOL II/3000 compilers.

The stack layouts for COBOL/3000 main programs, non-dynamic subprograms, and dynamic subprograms are shown in Figures I, II, and III. Refer to Figures IV, V, and VI for the stack layouts for COBOL II/3000. A description of both the COBOL/3000 stack and the COBOL II/3000 stack follows. The differences between the two compilers are discussed at the end of this section.

COBOL/3000 STACK LAYOUT

The following descriptions regarding the COBOL/3000 stack layout should be used in conjunction with Figures I, II, and III.

DB-5 contains the base address for the current data area being accessed during execution. This may point to the data area for the main program, non-dynamic subprograms or dynamic subprograms.

The SIZE ERROR FLAG is used to signify whether or not the ON SIZE ERROR clause should be executed.

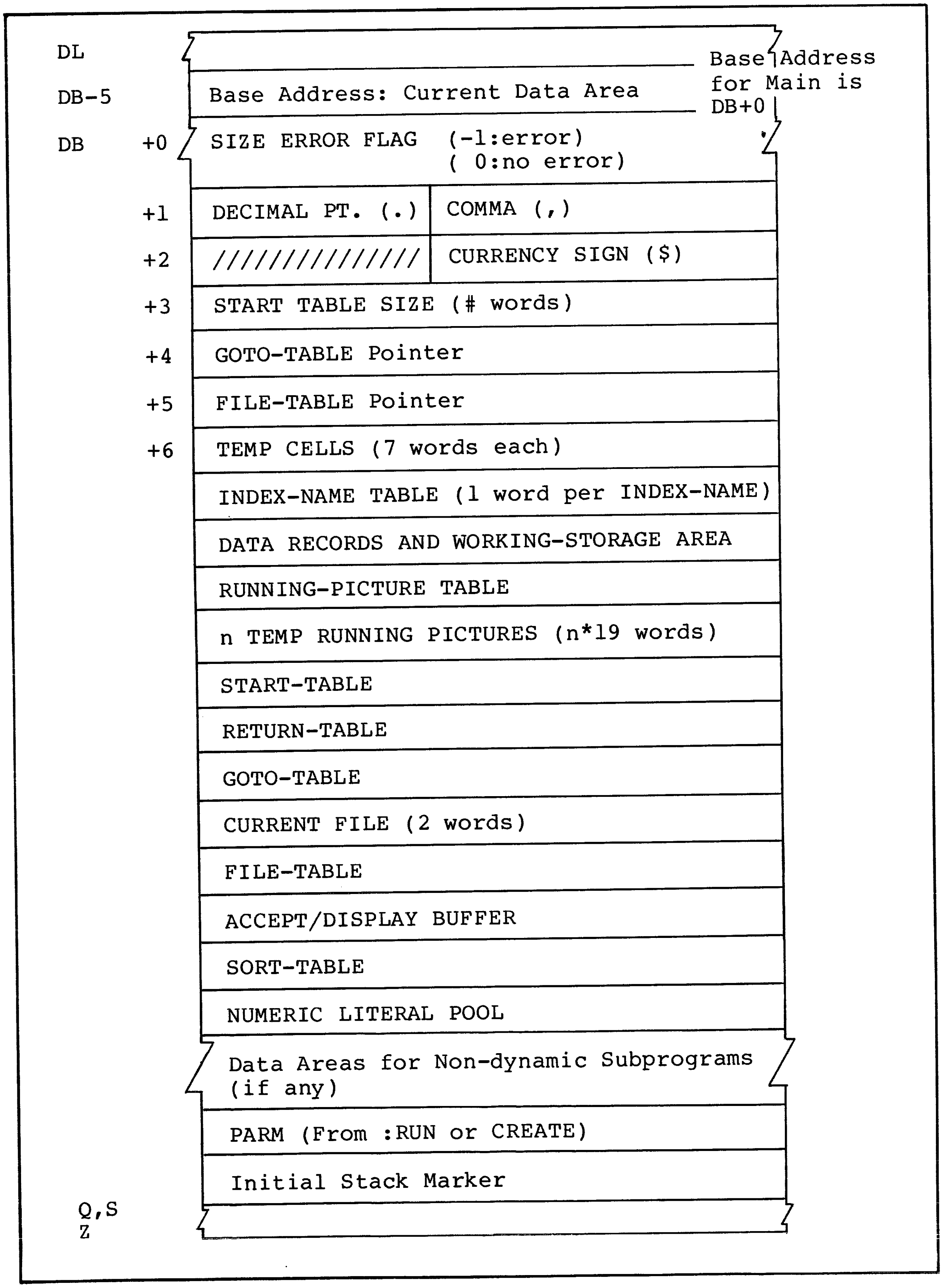

DL
DB-5
DB
Q,S
Z
+0
+1
+2
+3
+4
+5
+6
Base Address: Current Data Area
Base Address for Main is DB+0
SIZE ERROR FLAG (-1:error) (0:no error)
DECIMAL PT. (.)
COMMA (,)
///////////////
CURRENCY SIGN ($)
START TABLE SIZE (# words)
GOTO-TABLE Pointer
FILE-TABLE Pointer
TEMP CELLS (7 words each)
INDEX-NAME TABLE (1 word per INDEX-NAME)
DATA RECORDS AND WORKING-STORAGE AREA
RUNNING-PICTURE TABLE
n TEMP RUNNING PICTURES (n*19 words)
START-TABLE
RETURN-TABLE
GOTO-TABLE
CURRENT FILE (2 words)
FILE-TABLE
ACCEPT/DISPLAY BUFFER
SORT-TABLE
NUMERIC LITERAL POOL
Data Areas for Non-dynamic Subprograms (if any)
PARM (From :RUN or CREATE)
Initial Stack Marker

Figure I COBOL Stack: Main Program Data Area

DECIMAL POINT, COMMA and CURRENCY SIGN are the actual characters to be used for those purposes. This would be different from the default if some SPECIAL NAMES were in effect.

START TABLE SIZE denotes how large the start table is in words.

The GO TO TABLE and FILE TABLE pointers are addresses of the respective tables.

TEMP CELLS are used to hold intermediate results encountered during arithmetic operations.

The INDEX NAME TABLE contains one word entries which hold the value of each index-name encountered in the source.

The DATA RECORDS AND WORKING STORAGE AREA is usually the largest contributor to stack size in a COBOL program. It contains the data storage locations for each data item declared in the FILE and WORKING-STORAGE sections of the Data Division.

The RUNNING PICTURE TABLE contains variable length entries (from 3 to 19 words in size) which describe the picture and location of each data item referenced in the Procedure Division.

The TEMP RUNNING PICTURE TABLE is used to contain the pictures of data items which use the OCCURS DEPENDING ON construct. This is necessary because the size of the item can vary and the current size is needed to pass to the library routines.

The START TABLE has an entry for each paragraph name in the Procedure Division giving the address of where the code for that paragraph starts.

The RETURN TABLE is used to determine the point at which execution must resume following a PERFORM statement.

The GOTO TABLE is used to hold information pertaining to the alterable GO TO construct of COBOL.

CURRENT FILE is a 2 word entry which is used by the library routines to provide for user file label processing.

The FILE TABLE contains a 20 word entry for each file specified in the program's FILE-CONTROL paragraph.

The ACCEPT/DISPLAY BUFFER is a 100 word buffer present for accepting input and formatting output with the ACCEPT and DISPLAY verbs. This buffer is allocated only if the source program has one of these statements present.

The SORT TABLE has an entry for each sort statement in the source code. These entries hold information on the location, type and number of keys used in the sort. The size of the entries will be 6 words + 3 * the number of keys.

The NUMERIC LITERAL POOL is a storage location for numeric literals (constants) in the source. A pool of these is used to allow for the possibility of sharing constants throughout the program.

Non-Dynamic Subprogram Stack Layout

The stack layout for non-dynamic subprograms is identical to that of the main program. The actual place where the data is stored is some displacement away from DB. As the main program data area starts at DB+0, the first non-dynamic subprogram data areas will start at the end of this, followed by any other non-dynamic subprograms.

Dynamic Subprogram Stack Layout

The stack layout for dynamic subprograms is again identical to that of a main program, except that dynamic subprogram storage is Q relative (i.e. it is allocated at run time and will disappear as the subprogram exits).

Parameters are passed to subprograms in the normal way by using up to 60 words before the stack marker to pass the DB relative addresses of the parameters.

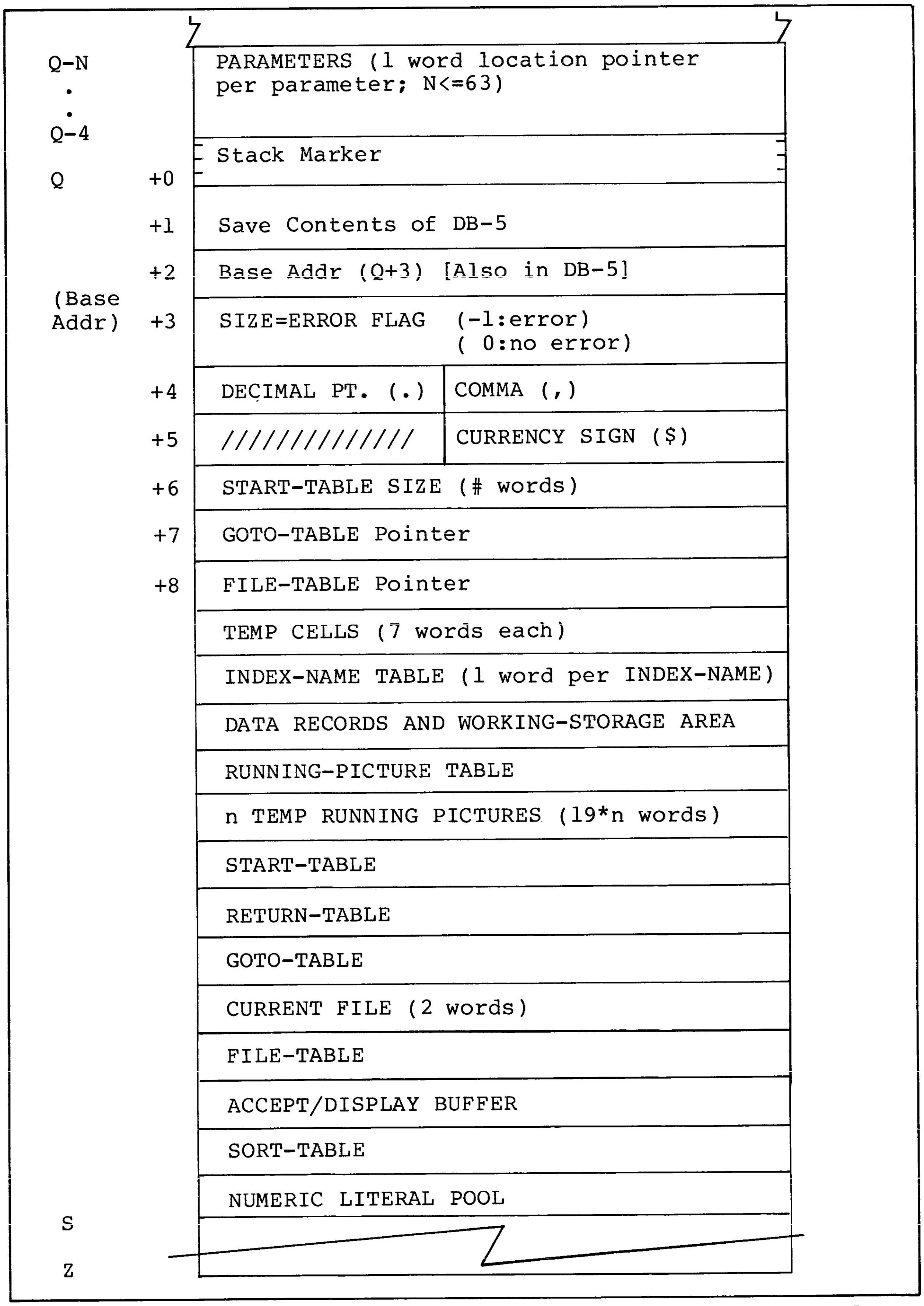


Figure II COBOL/3000 Stack: Dynamic Subprogram Data Area

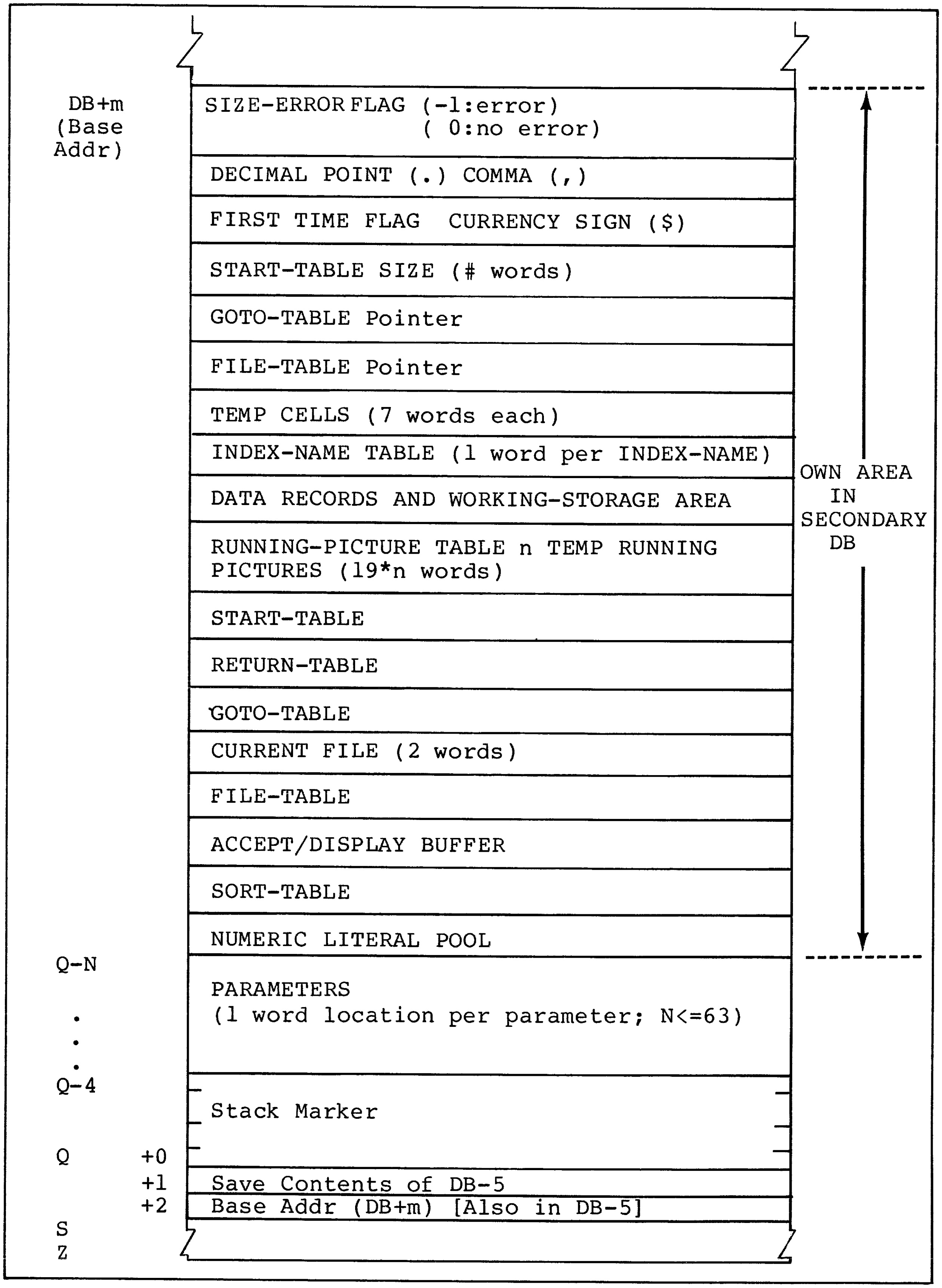

DB+m (Base Addr)
SIZE-ERROR FLAG (-1:error) (0:no error)
DECIMAL POINT (.) COMMA (,)
FIRST TIME FLAG CURRENCY SIGN ($)
START-TABLE SIZE (# words)
GOTO-TABLE Pointer
FILE-TABLE Pointer
TEMP CELLS (7 words each)
INDEX-NAME TABLE (1 word per INDEX-NAME)
DATA RECORDS AND WORKING-STORAGE AREA
RUNNING-PICTURE TABLE n TEMP RUNNING PICTURES (19*n words)
START-TABLE
RETURN-TABLE
GOTO-TABLE
CURRENT FILE (2 words)
FILE-TABLE
ACCEPT/DISPLAY BUFFER
SORT-TABLE
NUMERIC LITERAL POOL
OWN AREA IN SECONDARY DB
Q-N
PARAMETERS (1 word location per parameter; N<=63)
Q-4
Stack Marker
Q +0
+1 Save Contents of DB-5
+2 Base Addr (DB+m) [Also in DB-5]
S
Z

Figure III COBOL/3000 Stack: Non-dynamic Subprogram Data Area

COBOL/3000 SUBPROGRAM CALLS

After a subprogram has been called, several things take place before actual user code begins executing:

The contents of DB-5, which points to the calling procedure's data area, is saved in Q+1.

The base address of the current procedure's data area is placed in Q+2. This would be the DB relative address of Q+3 for dynamic subprograms, since that is where the procedure's data area actually starts. For main programs and nondynamic subprograms, the address stored in Q+2 reflects the DB relative address of the beginning of the module's storage area in secondary DB (commonly called "own storage"). The address that is put in Q+2 is then copied to location DB-5. The reason for this duplication of addresses is that all user code will access data Q-relative by indexing indirectly through Q+2. Library routines, on the other hand, access the procedure's data DB-relative by indexing indirectly through DB-5.

As the subprogram exits, the pointer at Q+1 is put back in DB-5, thus restoring the base address of the calling procedure's data area.

COBOL II/3000 STACK LAYOUT

The following descriptions regarding the COBOL II/3000 stack layout should be used in conjunction with Figures IV, V, and VI to see how COBOL II/3000 manages the accessing of information in the user's stack.

DB-5 contains the base address for the current data area being accessed during execution. This may point to the data area for the main program, non-dynamic subprograms, or dynamic subprograms. (see Figure IV).

DB-4 contains any run-time switches the program may be using. This is needed for use with the ;PARM option of the :RUN command. (see Figure IV).

The area starting at DB is used to hold the DATA AREAs for main programs and non-dynamic subprograms. The data area for a dynamic subprogram is located after Q+13+(2*N), where N is the number of 01 and 77 items in the Linkage Section, and Q is the Q-register value for the dynamic subprograms.

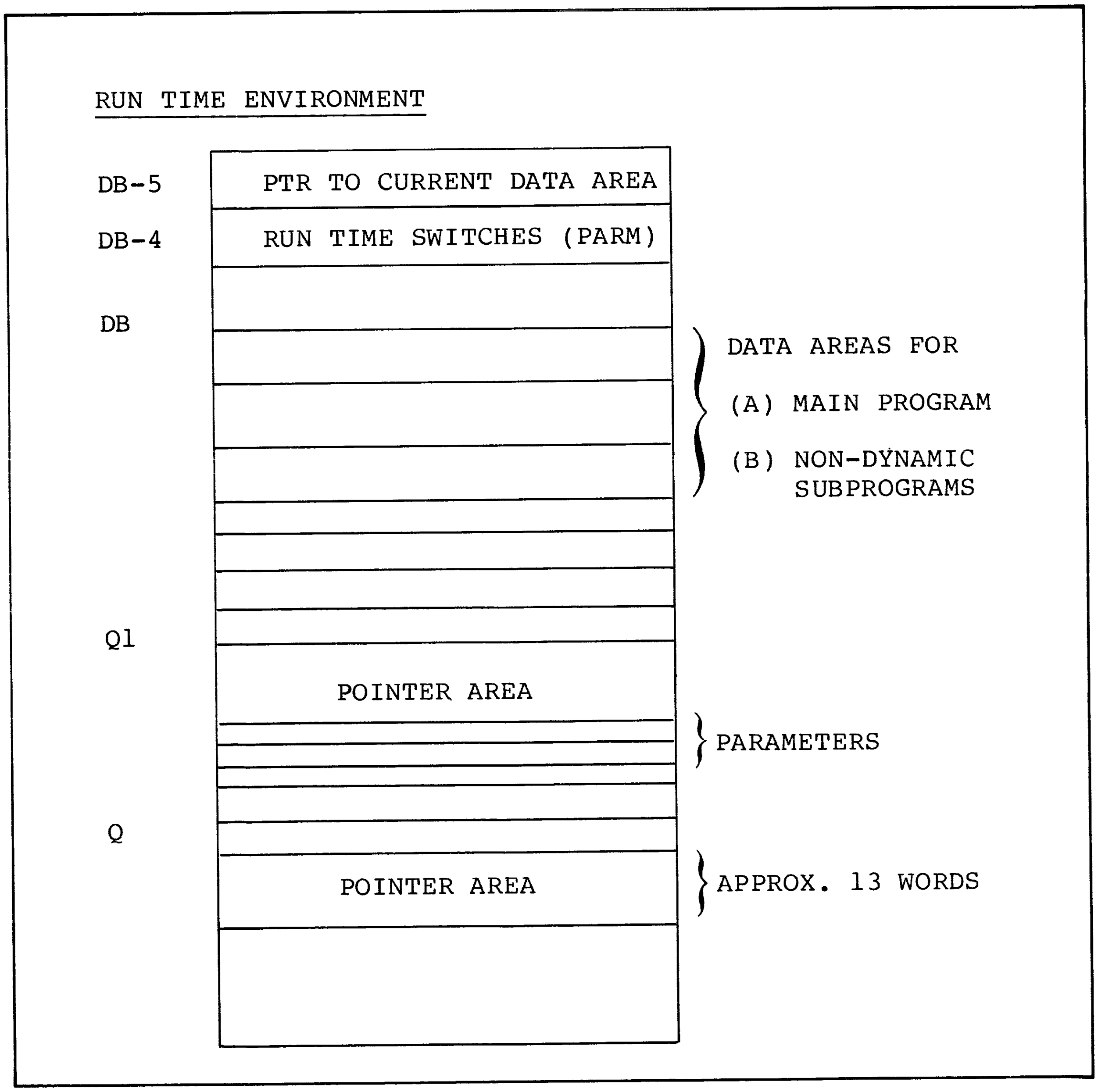


Figure IV COBOL II Run Time Environment

DATA AREA FOR COBOL II/3000

Data areas for main programs and non-dynamic subprograms are located starting at DB+0. Data areas for dynamic subprograms are located after the pointer area (approximately Q+13). A description of the contents of the data area follows (see Figure V).

The INDEX NAME TABLE, the START TABLE, the GO TO TABLE, the ACCEPT/DISPLAY BUFFER, and the FILE TABLE entries all perform the same functions in COBOL II/3000 as they do in COBOL/3000; however the file table is slightly larger in COBOL II.

The PROGRAM COLLATING SEQUENCE TABLE is used when a program specifies a collating sequence other than ASCII.

The DATA RECORDS and WORKING STORAGE area and the RUNNING PICTURE TABLE perform the same functions in COBOL II/3000 as they do in COBOL/3000.

The 9 WORD TEMP CELL area contains intermediate values for decimal computations.

RUN-TIME ENVIRONMENT

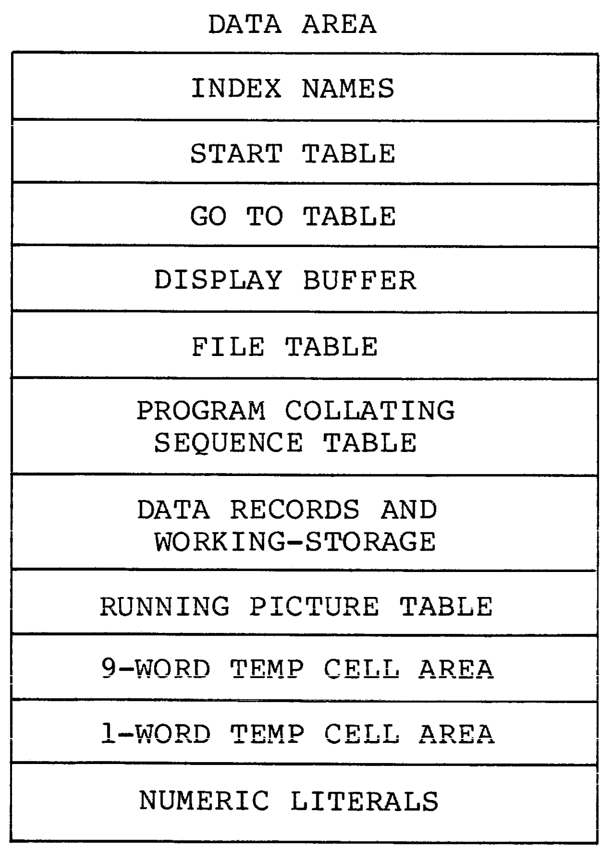


FIGURE V COBOL II DATA AREA

The 1 WORD TEMP CELL area contains intermediate values for binary computations.

The NUMERIC LITERALS area contains packed decimal numeric literals used in the PROCEDURE DIVISION.

POINTER AREA FOR COBOL II/3000

The pointer area is located from Q+1 to approximately Q+13. A description of the pointer area follows (see Figure VI).

The WORD and BYTE ADDRESS OF DATA AREA contains pointers to the DATA AREA for dynamic and non-dynamic subprograms as well as for main programs. This is used for accessing data on word or byte boundaries respectively.

The DECIMAL POINT and COMMA can be interpreted in the same way for both COBOL II/3000 and COBOL/3000.

Q+4 is described as follows:

- The SE field is a size error flag used to signify whether or not the ON SIZE ERROR clause should be executed. (This performs the same function in COBOL/3000)

- The F field is used when PERFORMing a paragraph or section whose segment number is greater than 49. If this is the case, then any alterable GO TO's used in that paragraph or section are set to their initial state the first time the paragraph or section is executed. This occurs for each execution of the PERFORM statement.

- The CURRENCY SIGN is the actual character to be used for that purpose. This would be different than the default if some SPECIAL-NAMES were in effect. (This performs the same function in COBOL/3000).

Q+5 thru Q+7 are self-explanatory.

The SORT-MERGE SWITCH is used to determine if a MERGE OUTPUT PROCEDURE or a SORT OUTPUT PROCEDURE should be called when using a SORT/MERGE output procedure.

The START TABLE ADDRESS contains the address of the START-TABLE in the DATA AREA.

RUN TIME ENVIRONMENT

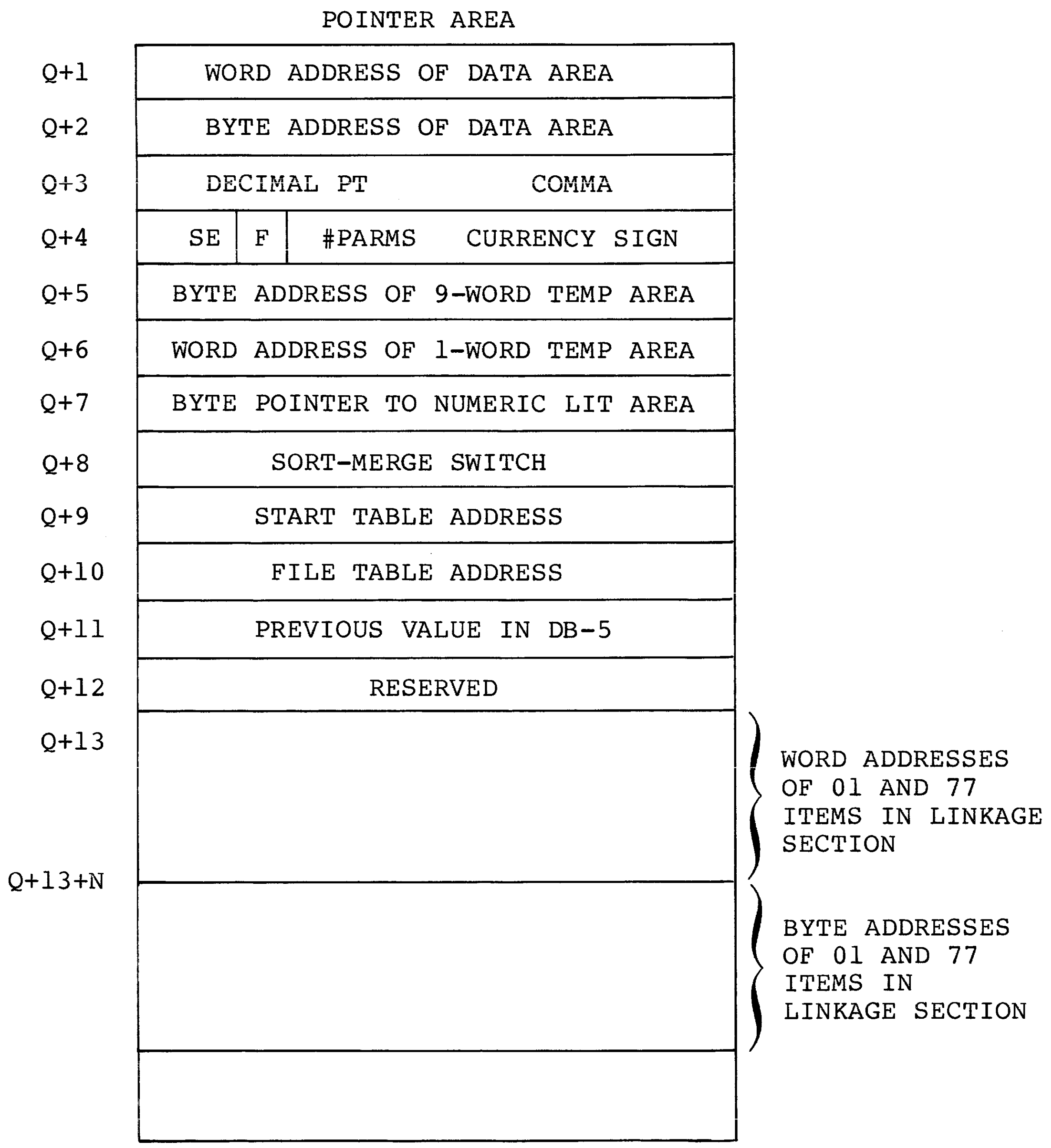

POINTER AREA
Q+1
WORD ADDRESS OF DATA AREA
Q+2
BYTE ADDRESS OF DATA AREA
Q+3
DECIMAL PT
COMMA
Q+4
SE
F
#PARMS
CURRENCY SIGN
Q+5
BYTE ADDRESS OF 9-WORD TEMP AREA
Q+6
WORD ADDRESS OF 1-WORD TEMP AREA
Q+7
BYTE POINTER TO NUMERIC LIT AREA
Q+8
SORT-MERGE SWITCH
Q+9
START TABLE ADDRESS
Q+10
FILE TABLE ADDRESS
Q+11
PREVIOUS VALUE IN DB-5
Q+12
RESERVED
Q+13
Q+13+N
WORD ADDRESSES OF 01 AND 77 ITEMS IN LINKAGE SECTION
BYTE ADDRESSES OF 01 AND 77 ITEMS IN LINKAGE SECTION

Figure VI COBOL II POINTER AREA

The FILE TABLE ADDRESS contains the address of the FILE TABLE in the DATA AREA.

The PREVIOUS VALUE in DB-5 is used to locate the data area for the calling program.

The area from Q+13 to Q+13+(2*N) is used to store the word and byte addresses of 01 and 77 level items used in a Linkage Section. This is used as an indirect reference to each item in the Linkage Section (this construct reduces the entries in the running picture table).

STACK DIFFERENCES

There are several areas of the stack where COBOL II/3000 differs from COBOL/3000:

(1) COBOL II/3000 eliminates the use of a RETURN TABLE. COBOL/3000 used a RETURN TABLE to hold the address of the location to return after the execution of a PERFORM statement. This table required two words for each paragraph in a COBOL program. With COBOL II/3000, the return information is put on the top of the stack and is used by the firmware to return after executing a PERFORM statement.

(2) COBOL II/3000 eliminates most of the RUNNING PICTURE table entries. These tables are used to describe the picture and location of each data item (generally 3 to 19 words for each item). The byte and word addresses of 01 and 77 level items in the Linkage Section POINTER AREA of COBOL II/3000 are indirect references to the data area where these items reside, thus eliminating the need for a running picture for some of these items.

(3) There are no restrictions on where the main program data area begins. In COBOL/3000, the data area was required to begin at DB + 0. The advantage in allowing the main program data area to begin at any location is that when using the Segmenter, the programmer need not worry about where the outer block of the program is located. This will help simplify the use of the Segmenter.

(4) The SORT TABLE has been eliminated from COBOL II/3000. The savings in stack size amount to about ten words for each call to Sort.

****

COBOL II User Survey

Now that many of you have had the opportunity to use COBOL II, the HP lab people would like to find out what you think of it. Please, answer only those questions which apply to your situation and try to provide any additional information which would be helpful. Please use additional paper if necessary.

Your feedback will help us not only in deciding on COBOL II enhancements, but also in the development of future language subsystems.

CONVERSIONS FROM HP COBOL/3000 (COBOL '68) TO COBOL II/3000

1. If you are currently using COBOL '68 and chose not to convert to COBOL II, why?

2. If you converted from COBOL '68 to COBOL II, when did you begin using the new compiler?
   When will you stop using COBOL '68?

3. What is your evaluation of the conversion process?
   [] Easy [] Easier than expected [] As expected
   [] More difficult than expected [] Difficult
   Comments:

4. Considering the added features you are now getting, is the difference in compile times
   [] worth it [] acceptable [] inconvenient
   [] unacceptable
   Comments:

5. How do execution speeds compare to COBOL '68?
   [] COBOL II is faster [] about the same [] slower
   Comments:

6. Were there items needing conversion which the conversion program failed to find? [] yes [] no
   If yes, what were they?

TEAR OUT SECTION

7. The conversion manual is (you may choose more than 1)
   [] very good [] good [] adequate [] poor
   [] accurate [] unclear
   [] well organized [] difficult to find things
   [] good examples [] needs more examples

8. Do you find COBOL II easier to use than COBOL '68?
   [] yes [] no
   Why?

CONVERSIONS FROM OTHER VENDOR'S COBOL TO COBOL II

1. Computer and operating system converting from:

2. Number of programs converted:

3. Size of programs:

4. Type of application:

5. Length of time needed to convert average program:

6. Degree of difficulty: [] high [] medium [] low

7. Major features used:

SYSTEMS	LANGUAGE FEATURES
(ie: Database, screen handling)	(ie: Report writer, indexed I/O)

8. Problem areas:

GENERAL QUESTIONS

1. Do you like COBOL II?
   [] yes [] no
   why?

2. What do you think of the listing format?

	good	acceptable	poor	don't use	comments
source	[]	[]	[]	[]	
error messages	[]	[]	[]	[]	
symbol table	[]	[]	[]	[]	
verb map	[]	[]	[]	[]	
cross reference	[]	[]	[]	[]	

3. What do you think of the compile and run-time diagnostics (error messages)?
[] good [] acceptable [] poor
Comments:

4. The documentation is (you may choose more than 1)
[] very good [] good [] adequate [] poor
[] accurate [] unclear
[] well organized [] difficult to find things
[] has good examples [] needs more examples
Comments:

5. Are you using or do you plan on using any of the extended features offered with COBOL II?
[] yes [] no If so, which ones?

[] MACRO	[] Verbmap
[] ACCEPT-FREE	[] Crossref
[] MPE Intrinsics	[] Parameter Passing options
[] Octal Literals	

[] exclusive and non-exclusive use of files
Comments:

6. Do you plan on using the ability to call subprograms which are written in other languages? [] yes [] no

If so, which ones?
[] SPL [] COBOL '68 [] FORTRAN

7. Are there other languages you would like to have the ability to call? [] yes [] no
If so, which ones?

TEAR OUT SECTION

FOLD FOLD

FIRST CLASS
PERMIT NO. 1020
SANTA CLARA
CALIFORNIA

BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States. Postage will be paid by

Kathleen Weiler
Hewlett-Packard Company
19447 Pruneridge Avenue
Cupertino, California 95014

FOLD FOLD

TEAR OUT SECTION

DOCUMENTATION

The catalog of customer publications at the end of this section lists the currently available customer manuals for HP 3000 Computer Systems products. This list supersedes the catalogs in previous issues of the COMMUNICATOR.

Purchasing

Customers may purchase copies of new manuals, new editions and updates by either Direct Phone Order or by placing orders through their local HP Sales and Service Office.

The Direct Phone Order numbers are (800) 538-8787 (toll free) and, in California, (408) 738-4133 (collect). Calls should be made between 9:00 a.m. and 5:00 p.m. in the caller's time zone. Most orders will be shipped within 24 hours.

The addresses and telephone numbers of local HP Sales and Service Offices are listed in the back of all customer manuals.

Prices of HP documentation are subject to change without notice.

To obtain a manual update, the customer must purchase the manual to which it pertains. The latest edition of the manual, along with the update, will then be sent to the customer.

Terms

A few words about documentation terms and procedures.

NEW	The first printing of the first edition. When first printed, a manual is assigned a part number that is retained for the life of the manual.
UPDATE	A supplement to an existing manual which contains new or changed information. Manual updates, which are issued between editions, contain additional or replacement pages to be merged into the manual by the customer.

Updates are generally issued at the same time Installation Tapes (ITs) are issued. However, THERE IS NO DIRECT CORRELATION BETWEEN SOFTWARE FIXES AND MANUAL UPDATES. Software enhancements that require documentation changes will be accompanied by manual updates, but software fixes and manual corrections may be made independently.

Updates are retroactively inclusive; that is, whenever successive updates are issued, the later update will contain the previous one(s). This means that you need obtain only the latest update to have all the information added or changed since the last printing of the manual.

Manual updates do not have part numbers. They are numbered sequentially from the time the last edition was issued.

NEW EDITION

A complete revision of a manual; obsoletes all previous editions of the manual and its updates.

A new edition is issued when, due to the scope of the changes involved, it is impractical to issue a manual update.

The date on the title page and back cover of every manual is the printing date of the current edition. This date changes only when a new edition is published. A list of the dates of the manual's previous editions and updates (if any) is kept on the Printing History page of every manual. Publication of a new edition does not affect the part number of a manual.

If further updates are required, they are made to the new edition. The update numbers run sequentially, starting from the latest edition.

Materials Management/3000, A User-Oriented Approach to Documentation

By Matt Kuzmich, General Systems Division

Eleven manuals have been designed for use with the new Materials Management/3000 manufacturing application package: nine user manuals and two System Administrator manuals.

Each user manual is documented according to the specific job function that the user normally performs. For example, the information a stock room clerk needs to do stock adjustments, inventory counts and other duties is in a single manual, including sections relating to retrievals and reports.

And in keeping with the customizing feature of Materials Management/3000, each user manual can be tailored to fit the needs of a particular manufacturing facility. Supervisors can determine what should be deleted from a manual or added to it with a minimum of effort.

Nine user manuals relate to specific manufacturing tasks while two manuals describe the System Administrator tasks. The basic format of the nine user manuals is divided into sections defining manufacturing topics, user transactions, reports and retrievals, how to use the system, and appendices for quick reference of terms and messages.

The two System Administrator manuals present an easy-to-use, step by step approach for the customization and daily operation of the the application to suit the particular needs of the manufacturing facility.

The user manuals are written for the non-computer person who doesn't want to be bothered with unfamiliar jargon. The design lets the person work as quickly and as easily as possible, emphasizing the computer as just another tool. In most instances, the transactions have been laid out on facing pages so that the user will not have to change pages while performing a function.

The manuals are in a comfortable 9x9 size. Important areas and words are highlighted in blue, an aid that helps the reader quickly find what is needed. The loose-leaf pages can easily be removed from the binder for changes and insertions of material.

NEW MANUALS

Master Production Scheduling
  and Rough Cut Resource Planning
part number 32260-90001
July, 1980

This new manual describes how to create and maintain the preliminary and final master schedule. Also describes how to use Rough Cut Resources Planning.

Maintaining Parts and Bills
  of Material
part number 32260-90002
July, 1980

This manual explains how to add, maintain, and delete parts, remarks, and bills of material; it also describes how to process engineering and price changes.

Maintaining Routings and
  Workcenters
part number 32260-90003
July, 1980

This new manual details how to add, maintain, and delete routings and work centers.

Material Issues and Receipts
part number 32260-90004
July, 1980

This new manual describes how to issue parts to a work order; also described are allocations,backorders, and extra usage allocations. In addition, the manual explains how to receive work orders, purchase orders, manufacturing orders, and parts from inspection.

NEW MANUALS

Maintaining Work Orders
part number 32260-90005
July, 1980

This new manual describes how to add, maintain, and delete work orders, allocations, backorders, and extra usage allocations.

Managing Inventory Balances
part number 32260-90006
July, 1980

This manual contains explanations of how to maintain inventory location information and adjust inventory counts.

Maintaining Purchase Orders
part number 32260-90007
July, 1980

Contains descriptions of how to add, maintain, and delete purchase orders and vendor information.

Material Requirements
  Planning
part number 32260-90008
July, 1980

This manual explains how to run the MRP program, assign order policies, and interpret MRP reports.

Standard Product Costing
part number 32260-90009
July, 1980

The subject of this new manual is how to determine current costs of products and their components (Cost Roll-Up) and set standard costs (Cost Roll-Over).

NEW MANUALS

Two System Administrator Manuals

System Customization
part number 32260-90010
July, 1980

The topic of this new manual is ways to customize Materials Management/3000. A step by step approach for customizing data, screens, security, system values, messages, and reports.

System Operation
part number 32260-90011
July, 1980

Subjects covered by this new manual include: how to operate Materials Management/3000 on a day to day basis; how to start and stop the system; how to review and reply to messages.

A New Manual For FLEXIBLE DISCCOPY/3000

FLEXIBLE DISCCOPY/3000 User's Guide
part number 32199-90001
August, 1980

This manual documents FLEXIBLE DISCCOPY/3000 (DISCCOPY), a new HP product that allows HP 3000 Series 30/33 users to process data on flexible discs created by IBM 3741-compatible equipment.

The manual is tutorial in content and style. It explains how to use DISCCOPY in jobs and sessions for single-volume and multiple-volume conversions. A section entitled "A Closer Look at FLEXIBLE DISCCOPY/3000" gives the reader a basic understanding of how DISCCOPY operates. Another section entitled "DISCCOPY and Files" explains how DISCCOPY interacts with the MPE File System.

The manual also serves as an easy-to-use reference book. DISCCOPY's many messages are discussed in detail, with further DISCCOPY actions predicted and user responses suggested. The manual's appendices include other reference information such as track specifications, formal file designators, and a glossary of terms.

NEW EDITIONS

EDIT/3000 Reference Manual 3rd Edition
part number 03000-90012
August, 1980

This new edition documents the new COPY command and new options to the ADD (ADD*) and the SET (SET LINES) commands. The COPY command copies text from one location to another in the work file. The ADD* command will cause text to be added immediately following the line on which the pointer is situated. The LINES parameter of the SET command specifies the number of lines that will be printed between page ejects on an offline listing.

Software Pocket Guide
part number 30000-90049
February, 1980

A new edition of the MPE Software Pocket Guide has been released to reflect 1918 system enhancements. The new pocket guide includes not only the operating system but also all subsystems. It has been re-formatted in a binder so that it can be updated on a regular basis.

MRJE/3000 Reference Manual 3rd Edition
part number 32192-90001
July, 1980

This new edition reflects changes to the MRJE/3000 product as well as providing a more detailed definition of the product than is found in the preceeding editions.

MRJE /3000 will now run on the INP. Output for jobs submitted through MRJE will now be owned by USER.ACCT of the submitter whenever that output is directed to a spooled output device. All host console messages may now be logged to a permanent disc file, if desired. The console command, =MRJE, has been replaced by a session command :MRJECONTROL; use of :MRJECONTROL may now be allowed to users other than the operator of the system console.

DS/3000 Reference Manual 2nd Edition
part number 32190-90001
September 1980

This edition includes documentation of the new Network File Transfer (NFT) feature which has been added to DS/3000. There is also expanded coverage of Distributed Systems applications, as well as general updating of some of the examples.

UPDATES

HP 3001A Intelligent Network Processor (INP) Installation & Service Manual — Update #1
part number 30010-90001
June, 1980

This update documents new settings for the function switches and adds MRJE information to the configuration dialogue.

HP 30010A/30020A Intelligent Network Processor (INP) Diagnostic Procedures Manual — Update #1
part number 30010-90002
June, 1980

This update recognizes the implementation of Test 5.3, Extensive BISYNC.

HP 30020A Intelligent Network Processor (INP) Installation & Service Manual — Update #1
part number 30020-90001
June, 1980

This update documents new settings for the rocker switches and adds MRJE information to the configuration dialogue.

UPDATES

RPG Reference Manual
part number 32104-90001
may, 1980

Update #4

Update #4 documents the two major enhancements to RPG: a more flexible mechanism for locking IMAGE data bases, and support for the calculation operations TIME and TIME2. It also contains numerous clarifications and changes.

V/3000 Reference Manual
part number 32209-90001
April, 1980

Update #1

In addition to many changes and clarifications, update #1 documents the five major enhancements to V/3000 data entry and forms management system:

- Multi-usage form families
- Three new information intrinsics
- Incremental compilation
- Forms files created non-KSAM
- VPOSTBATCH intrinsic (for improved recovery of batch files after system crash)

The update also includes a new, expanded index and reprinted listings of the ENTRY program for the end of the manual.

```
    XXXX        XX    XXXXXXXX    XX      XX         XXXX        XXXX
  XX    XX     XXXX      XX      XXXX     XX       XX    XX    XX    XX
 XX           XX  XX     XX     XX  XX    XX      XX      XX  XX
 XX          XX    XX    XX    XX    XX   XX      XX      XX  XX
 XX          XXXXXXXX    XX    XXXXXXXX   XX      XX      XX  XX  XXXXX
  XX    XX  XX      XX   XX   XX      XX  XX       XX    XX    XX    XX
    XXXX    XX      XX   XX   XX      XX  XXXXXXX    XXXX        XXXX

OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
OOOO    OOOOOOOO  OOOO        OOOO  OOOOOO  OOOOOOOOO    OOOOOOOO    OOO
OO  OOOO  OOOOO    OOOOOO  OOOOOO    OOOOO  OOOOOOO  OOOO  OOOO  OOOO  O
O  OOOOOOOOOOO  OO  OOOOO  OOOOO  OO  OOOO  OOOOOO  OOOOOO  OO  OOOOOOOO
O  OOOOOOOOOO  OOOO  OOOO  OOOO  OOOO  OOO  OOOOOO  OOOOOO  OO  OOOOOOOO
O  OOOOOOOOOO        OOOO  OOOO        OOO  OOOOOO  OOOOOO  OO  OO     O
OO  OOOO  OO  OOOOOO  OOO  OOO  OOOOOO  OO  OOOOOOO  OOOO  OOOO  OOOO  O
OOOO    OOOO  OOOOOO  OOO  OOO  OOOOOO  OO       OOOO    OOOOOOOO    OOO
OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO

    XXXX        XX    XXXXXXXX    XX      XX         XXXX        XXXX
  XX    XX     XXXX      XX      XXXX     XX       XX    XX    XX    XX
 XX           XX  XX     XX     XX  XX    XX      XX      XX  XX
 XX          XX    XX    XX    XX    XX   XX      XX      XX  XX
 XX          XXXXXXXX    XX    XXXXXXXX   XX      XX      XX  XX  XXXXX
  XX    XX  XX      XX   XX   XX      XX  XX       XX    XX    XX    XX
    XXXX    XX      XX   XX   XX      XX  XXXXXXX    XXXX        XXXX

OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
OOOO    OOOOOOOO  OOOOO      OOOOO  OOOOOO  OOOOOOOOO    OOOOOOOO    OOO
OO  OOOO  OOOOO    OOOOOO  OOOOOO    OOOOO  OOOOOOO  OOOO  OOOO  OOOO  O
O  OOOOOOOOOOO  OO  OOOOO  OOOOO  OO  OOOO  OOOOOO  OOOOOO  OO  OOOOOOOO
O  OOOOOOOOOO  OOOO  OOOO  OOOO  OOOO  OOO  OOOOOO  OOOOOO  OO  OOOOOOOO
O  OOOOOOOOOO        OOOO  OOOO        OOO  OOOOOO  OOOOOO  OO  OO     O
OO  OOOO  OO  OOOOOO  OOO  OOO  OOOOOO  OO  OOOOOOO  OOOO  OOOO  OOOO  O
OOOO    OOOO  OOOOOO  OOO  OOO  OOOOOO  OO       OOOO    OOOOOOOO    OOO
OOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO

    XXXX        XX     XXXXXX     XX      XX         XXXX        XXXX
  XX    XX     XXXX      XX      XXXX     XX       XX    XX    XX    XX
 XX           XX  XX     XX     XX  XX    XX      XX      XX  XX
 XX          XX    XX    XX    XX    XX   XX      XX      XX  XX
 XX          XXXXXXXX    XX    XXXXXXXX   XX      XX      XX  XX  XXXXX
  XX    XX  XX      XX   XX   XX      XX  XX       XX    XX    XX    XX
    XXXX    XX      XX   XX   XX      XX  XXXXXXX    XXXX        XXXX
```

KEY

Manuals that are new or have changed since the last edition of this catalog are noted by an asterisk (*) in the leftmost column. An asterisk in the "Price" column indicates that the price of the manual was not available at the time the catalog was printed.

If the V (version) column contains a #, the manual is applicable to systems running MPE III and to those running MPE C. Manuals which apply to MPE C systems only are listed under "MPE C MANUALS".

HP 3000 COMPUTER SYSTEMS

SYSTEM MANUALS

	Manual Title	V	Part Number	Price	Print Date	Up-date
	Using the HP 3000: An Introduction to Interactive Programming	#	03000-90121	8.50	4/79	
	General Information Manual (Series II/III)		30000-90008	5.25	9/79	
	MPE Commands Reference Manual		30000-90009	16.75	7/79	3/80
	MPE Intrinsics Reference Manual		30000-90010	20.00	4/78	3/80
	MPE Segmenter Reference Manual	#	30000-90011	3.50	2/77	
	MPE Debug/Stack Dump Reference Manual	#	30000-90012	4.50	9/76	6/77
	Series II/III Console Operator's Guide		30000-90013	7.50	3/80	

SYSTEM MANUALS (continued)

	Manual Title	V	Part Number	Price	Print Date	Up-dated
	System Manager/System Supervisor Manual		30000-90014	9.00	7/79	3/80
	Error Messages and Recovery Manual		30000-90015		###	
	HP 3000 Computer System Machine Instruction Set		30000-90022	6.75	2/80	
	MPE III System Utilities Reference Manual		30000-90044	4.50	3/77	3/80
	Index to MPE Reference Documents		30000-90045		###	
*	Software Pocket Guide		30000-90049	7.75	2/80	
	Using Files	#	30000-90102	8.50	4/78	
*	Series 30/33 Console Operator's Guide		30070-90025	12.75	3/80	

These manuals have been temporarily removed from circulation.

SUBSYSTEM MANUALS

	Manual Title	V	Part Number	Price	Print Date	Up-dated
*	EDIT Reference Manual	#	03000-90012	6.00	8/80	
	Trace Reference Manual	#	03000-90015	4.50	6/76	
	FCOPY Reference Manual	#	03000-90064	4.75	2/78	2/79
	Scientific Library Reference Manual		30000-90027	4.25	6/76	2/77
	Compiler Library Reference Manual		30000-90028	8.50	11/76	
*	FLEXIBLE DISCCOPY/3000		32199-90001	*	8/80	
	SORT Reference Manual	#	32214-90001	3.50	3/80	

LANGUAGE MANUALS

	Manual Title	V	Part Number	Price	Print Date	Up-dated
	BASIC for Beginners	#	03000-90025	6.00	11/72	
	BASIC/3000 Pocket Guide	#	03000-90050	1.25	9/74	
	System Programming Language Reference Manual	#	30000-90024	12.00	9/76	2/77
	System Programming Language Textbook	#	30000-90025	7.50	6/76	1/77
	BASIC Interpreter Manual		30000-90026	13.00	6/76	8/78
	FORTRAN Reference Manual		30000-90040	10.00	6/76	5/79
	SPL Pocket Guide	#	32100-90001	2.00	11/76	
	FORTRAN Pocket Guide	#	32102-90002	2.50	5/79	
	BASIC Compiler Reference Manual	#	32103-90001	3.00	11/74	6/76
*	RPG/3000 Compiler Reference Manual	#	32104-90001	22.00	2/77	5/80
	RPG Listing Analyzer	#	32104-90003	.50	2/77	
	APL Reference Manual		32105-90002	35.00	1/79	
	APL Pocket Guide		32105-90003	4.50	11/76	
	COBOL Reference Manual	#	32213-90001	12.00	7/75	1/79
	Using COBOL: A Guide for the COBOL Programmer	#	32213-90003	13.00	3/78	
	COBOL/II Reference Mnl.		32233-90001	19.00	12/79	
	COBOL/3000 to COBOL II /3000 Conversion Guide		32233-90005	3.25	12/79	

DATA COMMUNICATIONS MANUALS

	Manual Title	V	Part Number	Price	Print Date	Up-dated
	Guidebook to Data Communications	#	5955-1715	3.00	1/77	
	RJE/3000 Remote Job Entry (2780/3780 Emulator) Ref. Manual		30000-90047	12.75	11/79	
	Data Communications Handbook		30000-90105	13.50	10/78	
*	HP 30010A Intelligent Network Processor (INP) Installation & Service Manual		30010-90001	4.75	10/79	6/80
*	HP 30010A/30020A Intelligent Network Processor Diagnostic Procedures Manual		30010-90002	4.25	10/79	6/80
*	HP 30020A Intelligent Network Processor (INP) Installation & Service Manual		30020-90001	4.50	10/79	6/80
	HP 30032B Asynchronous Terminal Controller Instl. & Serv. Manual		30032-90004	14.00	1/74	7/76
	HP 30055A Synchronous Single-Line Controller (SSLC) Instl. & Serv. Manual	#	30055-90001	8.50	12/77	4/79
	Hardwired Serial Interface (HSI) Instl. & Service Manual		30360-90001	6.00	3/77	5/79

HP 3000 COMPUTER SYSTEMS

DATA COMMUNICATIONS MANUALS (continued)

	Manual Title	V	Part Number	Price	Print Date	Up-dated
*	DS/3000 Reference Manual		32190-90001	19.00	9/80	
	DS/3000 to DS/1000 Reference Manual for HP 3000 Users		32190-90005	7.25	1/78	
*	MRJE/3000 Reference Mnl.		32192-90001	8.75	7/80	
	MTS/3000 Reference Mnl.		32193-90002	10.00	11/79	4/80
	IML/3000 Reference Mnl.		32229-90001	*	4/80	

MANUFACTURING APPLICATIONS MANUALS

	Manual Title	V	Part Number	Price	Print Date	Up-dated
	EDC/3000 User Reference Manual		32380-90001	20.00	3/78	4/79
	EDC/3000 System Admin. Reference Manual		32380-90002	8.50	3/78	4/79
	EDC/3000 Programmer's Reference Manual		32380-90003	20.00	3/78	
	IOS/3000 User Reference Manual		32384-90001	25.00	3/78	
	IOS/3000 System Admin. Reference Manual		32384-90002	11.00	3/78	
	IOS/3000 Programmer's Reference Manual		32384-90003	23.50	3/78	
	MRP/3000 User-Admin. Reference Manual		32388-90001	19.50	8/78	11/79

MANUFACTURING APPLICATIONS MANUALS (continued)

	Manual Title	V	Part Number	Price	Print Date	Up-dated
	MRP/3000 Programmer's Reference Manual		32388-90002	13.00	9/78	
	SPC/3000 User Reference Manual		32392-90001	11.00	4/79	
*	Master Production Scheduling and Rough Cut Resource Planning		32260-90001	17.00	7/80	
*	Maintaining Parts and Bills of Material		32260-90002	17.00	7/80	
*	Maintaining Routings and Workcenters		32260-90003	11.00	7/80	
*	Material Issues and Receipts		32260-90004	14.75	7/80	
*	Maintaining Work Orders		32260-90005	15.00	7/80	
*	Managing Inventory Balances		32260-90006	12.00	7/80	
*	Maintaining Purchase Orders		32260-90007	14.00	7/80	
*	Material Requirements Planning		32260-90008	7.25	7/80	
*	Standard Product Costing		32260-90009	8.00	7/80	
*	System Customization		32260-90010	25.00	7/80	
*	System Operation		32260-90011	8.00	7/80	
*	Materials Mgt/3000 Manual Set		32263A	125.00	7/80	

TRANSACTION PROCESSING MANUALS

Manual Title		V	Part Number	Price	Print Date	Up-dated
	QUERY Reference Manual	#	30000-90042	9.00	6/76	5/79
	KSAM Reference Manual		30000-90079	14.50	5/79	
*	HP V/3000 Ref. Manual		32209-90001	14.50	1/80	4/80
	HP V/3000 Entry Program		32209-90003	2.50	1/80	
	Using HP V/3000		32209-90004	17.00	1/80	
	IMAGE Data Base Management Reference Manual		32215-90003	11.75	9/79	3/80

EDUCATIONAL APPLICATION MANUALS

Manual Title		V	Part Number	Price	Print Date	Up-dated
	Student Information System Reference Manual	#	32900-90001	13.00	9/74	8/76
	Student Information System Technical Mnl	#	32900-90005	32.00	3/75	
	Student Assignment System Reference Manual	#	32901-90001	15.50	8/78	
	Student Assignment System Technical Manual	#	32901-90005	9.75	8/78	
	College Information System Reference Manual	#	32902-90003	13.00	1/78	
	College Information System Technical Mnl.	#	32902-90005	10.50	2/78	

ADDITIONAL MANUALS

	Manual Title	V	Part Number	Price	Print Date	Up-dated
	HP 3000 Series System Support Log		03000-90117	20.00	2/80	
	HP 3000 CX to HP 3000 Series II Program Conversion Guide		30000-90046	3.50	6/76	
	Guide to a Successful Installation	#	30000-90135	7.00	12/79	
	Series III(32435A) Site Preparation Manual		30000-90145	2.00	1/79	4/79
	Series III(32435A) Site Planning Workbook		30000-90146	5.50	5/79	
	Technical Writer's Survival Kit		30000-90171	2.50	7/79	
	HP 3000 Computer System Site Planning and Preparation Guide		30000-90206	*	6/80	
	HP 3000 Computer System Site Planning Wkb		30000-90207	*	6/80	
	HP 3000 Computer System Site Planning Set		30000-60029	*	6/80	
*	Series 33 Installation Manual		30070-90021	5.25	10/78	1/80
	Series 33 Diagnostic Manual Set		30070-60068	55.00	9/78	6/80

ADDITIONAL MANUALS (continued)

Manual Title	V	Part Number	Price	Print Date	Up-dated
Series 30 Installation Manual		30080-90001	6.25	8/79	1/80
HP 2894A Card Reader Punch Operating Manual		30119-90009	11.50	10/76	
Line Printer Operating and Programming Manual		30209-90008	6.75	6/76	
IBM System/3 to HP 3000 Conversion Guide	#	32104-90004	10.75	7/78	

MPE C MANUALS

Manual Title	V	Part Number	Price	Print Date	Up-dated
BASIC Interpreter Reference Manual		03000-90008	9.75	7/75	
Compiler Library Reference Manual		03000-90009	11.50	2/76	
Scientific Library Reference Manual		03000-90010	5.75	7/75	
System Ref.Mnl. Series I		03000-90019	24.00	9/73	3/77
Software Pocket Guide		03000-90126	2.70	7/78	
IMAGE Data Base Management Reference Manual		30000-90041	7.00	12/76	5/78

MPE C MANUALS (continued)

Manual Title	V	Part Number	Price	Print Date	Up-dated
MPE Intrinsics Reference Manual		30000-90087	20.00	4/77	4/78
MPE Commands Ref. Mnl.		30000-90088	20.00	4/77	4/78
System Manager/System Supervisor Manual		30000-90089	12.50	4/77	4/78
Console Operator's Guide		30000-90090	11.00	4/77	4/78
General Information Manual (Series I)		30000-90091	9.25	4/77	
INDEX/3000 Reference Mnl		30000-90095	10.50	6/77	4/78
RJE/3000 (2780/3780 Emulator) Ref. Mnl. for Pre-Series II Systems		30130-90001	9.00	12/74	1/80
MPE System Utilities Reference Manual		32000-90008	2.05	10/75	
FORTRAN Reference Manual		32102-90001	10.00	3/76	
IBM 1130/1800 to HP 3000 FORTRAN Conversion Gd.		36995-90013	4.70	2/75	5/75

:MR$\overset{J}{\wedge}$ECONTROL, A New MPE Command

By Steve Stauss, General Systems Division

In the 2028 Installation Tape, the console operator command =MRJE has been replaced with the session command :MRJECONTROL. This command is used to control MRJE/3000 remote communication activities. Users may be ALLOWed to use it.

SYNTAX

```
                     START    [,hostid] [;trace function]
                     SIGNOFF  [,hostid]
:MRJECONTROL         KILL     [,hostid]
                     RETRIES  [,hostid], retrynum
                     TRACE    [,hostid], [trace options]
```

where "trace function" has the following syntax:

```
     trace on [trace options]
     trace, off ,
```

and where "trace options" has the following syntax:

```
     [,[ALL] [,[mask] [,[numentries][,[WRAP][,filename]]]]]
```

PARAMETERS

START	Opens a data communications link to a remote computer via the MRJE/3000 Subsystem. Initiates MRJE execution.
hostid	The name of a host system as defined by the MRJE/3000 manager. The name can be spelled out or abbreviated to its first character. If omitted, connection is made to the default system.
SIGNOFF	Closes the data communication link to the system specified by hostid. Sends a SIGNOFF record.
KILL	Immediately breaks all communication with the system specified by hostid. Used for abnormal line terminations.

TEAR OUT SECTION

RETRIES	Permits the user to override the CS retry counter which specifies how many times the system will repeatedly attempt to recover from transmission errors before terminating the connection. The default is 255 retries.
retrynum	Specifies the number of times the MRJE/3000 Subsystem will attempt to repeat a data transmission in case of data errors. Must be an integer between 1 and 255, inclusive.
TRACE ,ON ,OFF	Activates or deactivates the CS/3000 Trace facility.
ALL	Generates trace records for all line activity. Default: Trace records are written only when transmission errors occur.
mask	An octal number preceded by a percent symbol, that specifies the type of events to trace.

Bit	Meaning When ON
0	Not used by MRJE
1	Not used by MRJE
2	STN entries
3	OPR entries [default ON]
4	RCT entries [default ON]
5	RTX entries [default ON]
6	SCT, POL, SEL entries [default ON]
7	STX entries [default ON]

numentries	The maximum number of entries in a trace record. Should be an integer multiple of eight not greater than 248. If no trace file exists when you turn on the trace facility and you do not specify "numentries", the system will create a file to hold 24 entries. If the file already exists, its established number is used. Once created, the CS trace file must be purged in order to increase numentries.
WRAP	Causes trace entries that exceed the trace record size [.e., are greater than numentries] to overlay the prior trace entries. Default: Entries are discarded rather than wrapped over earlier entries.
filename	The trace file name. If a filename is specified with a group and account other than PUB.SYS, the file must already exist in order for the CS facility to be able to open it. Default: MRJETRCE.PUB.SYS.

TEAR OUT SECTION

OPERATION

The :MRJECONTROL START command starts the line opening procedure. If communication is over a private [leased] line, connection occurs almost immediately. If communication is over a dialup line, after receiving the dial message on the console you must dial the telephone number and press the DATA [or TALK] button when you hear the carrier tone. Once the line has been opened successfully, the SIGNON COMPLETED message will be received at the console.

You cannot initiate MRJE activity if the communications line is already in use by another data communications subsystem.

The :MRJECONTROL SIGNOFF command will disconnect the line to the host system. However, if data is being received or transmitted, MRJE will wait for all activity to terminate normally before disconnecting the line. No data will be lost.

The :MRJECONTROL KILL command immediately disconnects the line to the host system. Use with care as data may be lost.

EXAMPLE

The following example opens a line to the host system identified by the name BHOST. The first letter [B] identifies the configuration file name. It also specifies the system should create a file named [by default] MRJETRCB.PUB.SYS. The file can hold 250 entries in each trace record. In this example, all line events that pass through the default mask of %37 will be traced. (With a mask of %37, only STN entries will not be recorded.) The most recent entries are discarded, not wrapped.

```
    :MRJECONTROL START, BHOST; TRACE ON,ALL,,250
```

Printed in U.S.A. 7/80
Part No. 5951-6113

HEWLETT
PACKARD