Charles Van Ausdall

Abstract

It is not our intent to define nor exhaust the possible approaches to transaction logging but to point out the importance of and need for such a facility, specifically, in an on-line environment utilizing the HP 3000 Data Base subsystem IMAGE.

I. What is transaction lossing?

II. Why lose transactions?
(batch/on-line implications)
A. Rerun/recovery
B. Audit capability
C. System evaluation, whoenteredhowmanyofwhatwhen
1. Who - operator/process ID
2. Enteredhowmany - provided by txn. count
3. Ofwhat - transaction ID
4. When - time and date stamp

III. What should we lose?
(as a function of why we lose)
A. All transactions
B. Only actual modifications to the "data base"

IV. Who loss the "what" we choose to lose?
(optimization of utility)
A. Users responsibility?
B. Vendors "opportunity"?

V. How do we lose the "what"?
(the medium chosen as a function of resources/hardware)
A. Tape or disc as "system" resources
B. Unsupported (by vendor) devices configured as terminals

VI. Summary/Proposal
A. Reliability of interested systems
B. Efficiency
C. Operational considerations

Transaction Logging for On-Line Systems
Integrity of Image Data Bases

Welcome to the wonderful world of on-line computing!!! I am sure that by now we all believe that on-line processing is possibly the greatest thing since peanut butter. However, as with all "good things" we sometimes find "nits" which only come to light after making the commitment to innovate. One particularly important such item is the management of system integrity for such an amorphous entity as an on-line computing facility.

For our purposes let us define and limit "on-line" systems to be those facilities providing an operator, typically the end user, with the capability of dealing directly with the computer. Further, let us provide this capability in an interactive mode, allowing the user to enter data and providing on the spot results. This process is to be differentiated from conventional batch systems which accept data in bulk quantities prepared in advance in some computer compatable nonconsumable medium. This data is typically gathered by the user, "punched" by a data entry operator via such devices as keypunch, key-tape, key-disc, etc... and finally submitted by a computer operator to the "machine" for processing. The results are then routed back to the user to be verified and possibly resubmitted to Data Entry for another run.

There are many variations and combinations of the batch/on-line mix and as many valid arguments for the use of each. Our primary interest here is in the implications of the user dealing directly and interactively with the computer.

Data Integrity

Since the user enters data directly to the computer there is typically no "computer compatable nonconsumable medium" produced. This would be the case with video type terminals. This point is significant in that it implies the possible loss of the capability to rerun the process. If we are operating in an environment in which data is entered from a source document, it would be necessary to manually reenter all data processed since the last backup. In a non-source environment, however, i.e., phone order processing recovery may not be possible.

The process of lossing each transaction as it occurs would provide us with the required history of values and sequence in which data was processed against the data base. This "lose" together with periodic system backup procedures should provide us with the information to restore our "environment" to its status at any given point in time.

A second important consideration to the question of data integrity is the implication of the multi programming environment in which several processes may be concurrently effecting the data environment. This is particularly important if the sequence in which transactions are processed against the data base effect their outcome. In this situation, the result of any one transaction may depend on either the value of elements already in the data base or the possibility that another transaction against the data base may be pending.

The above considerations may be placed in perspective by the following question: "How do we recover our data base following a system failure? (hardware or software)". We are not so much concerned with who or what caused the failure but without ability to pick up the pieces after the cause has been remedied.

Providing Integrity

We have attempted to handle this situation in our shop in a straight forward manner. Simply stated: ...We do not allow system crashes... This policy, however, has not proven to be entirely successful and we, therefore, have been forced to consider alternatives.

Periodic backup provides a partial answer allowing us to recover up to some point in time T(n) at which the system was last backed up. (We have even implemented an S.O.P. - Standard Operating Procedure - to provide the backups). However, we have not been successful in getting the "system" to cooperate in scheduling its crashes. (What about the transactions from T(n) to the time of crash?)

Logging

Our next approach was to explore the process of Transaction Logging. In a nutshell, Transaction Logging provides a "copy" in which it occurs. These transactions are then available and together with the backup from time T(n) provide the information to restore the data environment to the status at the exact time of failure (assuming, of course, that some facility is provided to "rerun" there transactions against the data base).

In our survey we found several questions which must be asked to define this "Logging" process.

What do we lose? Our answer is primarily a function of why we lose. If we lose transactions for data recovery only then we need lose only those transactions which modify our data base and more exactly only that data which has been actually updated. If we would also like to analyze all activity against the system we must necessarily store more transactions with more information. This approach could provide information for audit trail or system evaluation as in whoenteredhowmanyofwhatwhen.

Having defined the why and what we move to the next logical process, the definition of how we implement logging and implicitly the restore procedures. We considered several avenues of approach. At the outset lossing appeared to be a relatively application dependent process, similar to yet sufficiently different for each process to require many lose and restore procedures and involve much effort to create and maintain the recovery software.

After studying the data storage technique chosen, HP Image 3000, we developed what appeared to be a generalized lossing/recovery process using the data base itself to define the structure and content of the data base at recovery time. Our approach was to provide user callable procedures with which to accomplish the file handling and transaction record output. However, in our attempt to implement this facility we have encountered some difficulties yet to be resolved to our satisfaction:

1. Multiple processes must be able to concurrently share a single file (of variable record length to allow maximum blocking efficiency) and to maintain a single I/O buffer for that file to insure that records are lossed in the sequence the transactions are processed. The overhead (coding, maintenance and execution) required to define, open and write to the lose file might possibly be more efficiently performed by IMAGE.
2. Where do we create the lose file? Preferably offline to cassette or discette or some similar relatively inexpensive medium.

3. User written lossing procedures cannot inteface with the Data Base inquiry subsystem QUERY. Our choice must then be to either strictly control the use of QUERY since QUERY modifications to the Data Base would not be lossed or to prohibit the use of this program.

In Summary

The subject of integrity in the data environment of an on-line computer facility should weigh heavily in the design of that facility and in its daily management and operation. Data base management technology is one area in which there is currently great interest and with which an attempt has been made to integrate user data into a common working structure to improve cohesion, reliability and availability. It is felt that by providing a lossing and restore facility to the data base structure would greatly improve the reliability of on-line processing, and add significantly to the market potential of the system.

Further, on the basis of our experience in attempting to provide this capability, we believe that optimisation requires that the lossing utility take advantage of operating under system rather than user control.

We feel the subject of data integrity to be of sufficient importance to the user in an on-line environment to warrant the interest of the vendor and request endorsement by the HP 3000 International Users Group in the request that Hewlett-Packard initiate a hi-priority project to provide a transaction lossing facility as an enhancement to the IMAGE/QUERY 3000 Data Base Management Subsystem.

Following are our lossing procedure definitions and the code developed to perform the lossing function. Please note this code was developed with the concept of sharing a common buffered variable record length disc file. It is currently our belief that the process should lose to an unbuffered shared device possibly a tape cartridge to provide an inexpensive offline lossing medium.

```
*************************************************************************
*     DATA BASE LOG  06.08.76:C.R.Van Ausdall                           *
*************************************************************************
      All activity which in any way changes any of the supported
      (by program reference) H.P. 3000 IMAGE data bases with the
      exception of H.P. QUERY access will be logged to provide the
      capacity to restore data base integrity subsequent to system
      or program crashes

      * the format of the logged record will be as follows:
        1) Current time and date from system clock
             DATE YYDDD (Julian date from procedure CALENDAR     )
                                                      1word-integer
             TIME HHMMSSTT (from procedure CLOCK)   2word-integer
             (see system procedure definitions for description   )
             (of format in MPE OPERATXNG SYSTEM reference manual)
        2) DATA BASE NAME                              6byte-character
        3) DATA SET NAME                              16byte-character
        4) MODE - add, change, or delete code          1word-integer
        5) STATUS - contents of status array returned by
                     last D.B. call                   10word-integer
                     (see IMAGE reference and D.B. Call ie...DBPUT)
                     (for definition of format                    )
        6) LIST - a concatenated string of item names passed
                   as a parameter to the D.B. call ie...DBPUT
                     (see IMAGE reference for definition          )
        7) LIST LENGTH - an integer, the value of which is the
                   length of the array LIST including delimeters
                   and terminator.
        8) BUFFER - is an array containing the values of the
                   fields defined by the array LIST in a
                   corresponding sequence
        9) BUFFER LENGTH - an integer, the value of which is
                   the length of the array BUFFER.

      * procedure DBLOG(MPEFNO,DBASE,DSET,DMODE,DSTAT,DLIST,DLSTLEN,
                        DBUFF,DBUFLEN)

        ARRAY(word) : DBASE, DSET, DSTAT, DLIST, DBUFF
        INTEGER : MPEFNO, DMODE, DLSTLEN, DBUFLEN

        All variables with the following exceptions are defined
        in the IMAGE reference manual.
        MPEFNO - the mpe file number of the logging file
                  as opened by DBOPNLOG.
        DMODE - the value of mode is determined by the
                transaction against the data base which
                preceeded (triggered) the call to DBLOG.
                This may have been a DBPUT, DBUPDATE, DBDELETE
                MODE = 1 if a DBPUT is to be logged.
                MODE = 4 if a DBUPDATE is to be logged.
                MODE = 7 if a DBDELETE is to be logged.
        This procedure will actually write a log record to the
        Data Base logging file LOGFIL.
```

```
* procedure DBOPNLOG(MPEFNO)

  INTEGER : MPEFNO
```

This procedure defines and opens a variable length MPE file designated as the file LOGFIL.group.account, and returns the MPE file number to the calling routine, to be used in calls to the logging procedure DBLOG.

```
$CONTROL USLINIT,MAP,LABEL
      PROGRAM DRIVLOG
C
C*************************************************************************
C
C     This program is an example of the use of
C     D.B.  Logging procedures DBOPNLOG and DBLOG...06/30/76:cv
C
C*************************************************************************
C
      IMPLICIT INTEGER (A-Z)
      CHARACTER *1
     ^   DBASE *26,
     ^   DSET *16,
     ^   DSTAT *20,
     ^   DLIST *24,
     ^   DBUFF *24
      DIMENSION
     ^   IBASE (13),
     ^   ISET  ( 8),
     ^   ISTAT (10),
     ^   ILIST (12),
     ^   IBUFF (12)
      EQUIVALENCE
     ^   (IBASE(1),DBASE),
     ^   (ISET (1),DSET ),
     ^   (ISTAT(1),DSTAT),
     ^   (ILIST(1),DLIST),
     ^   (IBUFF(1),DBUFF),
     ^   (ISTAT(2),I)
      DISPLAY "BEGIN DRIVLOG...DBLOG DRIVER PROGRAM"
C
C     OPEN LOG FILE AS SHARED AND RETURN MPE FILE NUMBER.
      CALL DBOPNLOG (MPEFNO)
      DISPLAY "I'VE OPENED THE LOGGING FILE"
      DISPLAY "PLEASE ENTER DATA BASE NAME XXXXXXXX"
      ACCEPT DBASE
      DSET = "SET NUMBERXXXXXX"
      DLIST = "LIST XXXXX123456789 1234"
      DBUFF = "BUFF XXXXX123456789 1234"
      LSTLEN = 12
      BUFLEN = 12
      DO 30 I = 1,100,2
      DO 30 J = 1,7,3
      MODE = J
C
C     KILL TIME...
      DO 20 K = 1,100
```

```
048000          Y = Y+1
049000          Y = Y-1
050000   20     CONTINUE
051000          CALL DBLOG (MPEFND,IBASE,ISET,MODE,ISTAT,ILIST,
052000                      \LSTLEN\,IBUFF,\BUFLEN\)
053000          REC =  REC + 1
054000   30     CONTINUE
055000          DISPLAY "END DRIVLOG"
056000          DISPLAY REC, " RECORDS LOGGED."

GE 0002   DRIVLOG
057000          STOP
058000          END

SYMBOL MAP
ME              TYPE        STRUCTURE    ADDRESS                       NAME        T
FLEN            INTEGER     SIMPLE VAR   Q+ 12                         DBASE       C
LOG                         SUBROUTINE                                 DBOPNLOG
UF              CHARACTER   SIMPLE VAR   Q+ 11,I                       DLIST       C
ET              CHARACTER   SIMPLE VAR   Q+  4,I                       DSTAT       C
                INTEGER     SIMPLE VAR   Q+  7,I                       IBASE       I
UFF             INTEGER     ARRAY        Q+ 10,I                       ILIST       I
ET              INTEGER     ARRAY        Q+  3,I                       ISTAT       I
                INTEGER     SIMPLE VAR   Q+ 14                         K           I
TLEN            INTEGER     SIMPLE VAR   Q+ 19                         MODE        I
EFND            INTEGER     SIMPLE VAR   Q+ 13                         REC         I
                INTEGER     SIMPLE VAR   Q+ 15

 LABEL MAP
 STATEMENT    CODE     STATEMENT    CODE     STATEMENT    CODE     STATEMENT    CODE     STA
   LABEL     OFFSET      LABEL     OFFSET      LABEL     OFFSET      LABEL     OFFSET      L

    20         324         30        345

**** NO ERRORS,   NO WARNINGS!    PROGRAM UNIT COMPILED   ****
MPILATION TIME    2.344 SECONDS    ELAPSED TIME     5.706 SECONDS
TAL COMPILATION TIME   0:00:03
TAL ELAPSED TIME       0:00:07
```

```
PAGE 0001   HEWLETT-PACKARD 32100A.06.0  SPL  WED, JUL 14, 1976,  7:26 PM  (C) COP
0
00002000  00000 0   $CONTROL NOWARN,INNERLIST
00003000  00000 0   $CONTROL MAP,SUBPROGRAM
00004000  00000 0   <<                                                        >>
00005000  00000 0   << DBOPNLOG...06/29/76:CV                                 >>
00006000  00000 0   << THIS PROCEDURE WILL OPEN AN MPE VARIABLE LENGTH FILE   >>
00007000  00000 0   << 'LOGFIL' TO BE USED BY PROCEDURE 'DBLOG' TO RECORD     >>
00008000  00000 0   << TRANSACTIONS AGAINST AN IMAGE DATA DBASE.  THIS LOG FILE >>
00009000  00000 0   << TOGETHER WITH A STORE TAPE OF THE DATABASE SHOULD PROVIDE>>
00010000  00000 0   << DATA INTEGRITY VIA A RESTORE PROGRAM "DBRERUN" TO RESTORE>>
00011000  00000 0   << A CRASHED DATA DBASE TO THE STATUS PRIOR TO THE CRASH.  >>
00012000  00000 0   <<                                                        >>
00013000  00000 0   << NOTE THAT ONLY TRANSACTIONS WHICH ACTUALLY MODIFY THE  >>
00014000  00000 0   << CONTENT OF THE DATA BASE SHOULD BE LOGGED...IE DBUPDATE >>
00015000  00000 0   << ...DBPUT AND DBDELETE.                                 >>
00016000  00000 0   <<                                                        >>
00017000  00000 0   << PROCEDURES FOR RESTORING FROM THE LOG FILE WILL BE FOUND >>
00018000  00000 0   << IN THE LOGGING SYSTEM REFERENCE AND OR THE RELOAD PROG..M>>
00019000  00000 0   << "DBRERUN".                                             >>
00020000  00000 0   <<                                                        >>
00021000  00000 0   BEGIN
00022000  00000 1   PROCEDURE DBOPNLOG (MPEFNO);
00023000  00000 1   INTEGER MPEFNO;
00024000  00000 1      BEGIN
00025000  00000 2      INTEGER E;
00026000  00000 2      BYTE ARRAY LOGFIL(0:6);
00027000  00000 2      INTRINSIC FOPEN,FCHECK;
00028000  00000 2      MOVE LOGFIL:="LOGFIL ";
                                          00000    ADDS,003             035003
                                          00001    LRA    S- 000        171700
                                          00002    LSL ,000,001         010201
                                          00003    STOR   Q+ 002        051402
                                          00004    ADDS,003             035003
                                          00005    LRA    Q+ 002,I      17340
                                          00006    LRA    P+ 003        170003
                                          00007    LSL ,000,001         010201
                                          00010    BR     P+ 000        140000
                                          00011    INSERT OR FIXUP      046117
                                          00012    INSERT OR FIXUP      043506
                                          00013    INSERT OR FIXUP      044514
                                          00014    INSERT OR FIXUP      020040
                                          00015    LDI ,007             021007
                                          00016    MVB ,000,003         020043
00029000  00017 2      MPEFNO:=FOPEN(LOGFIL,1,%345);
                                          00017    ZERO, NOP            000600
                                          00020    LOAD   Q+ 002        041402
                                          00021    LDI ,001             021001
                                          00022    LDI ,345             021345
                                          00023    ADDS,013                       035013
                                          00024    LOAD   P+ 000                  040000
                                          00025    PCAL,000                       000000
                                          00026    STOR   Q- 004,I                053604
00030000  00027 2          IF <> THEN BEGIN
                                          00027    BE     P+ 000                  141200
00031000  00030 3                     FCHECK(MPEFNO,E);
                                          00030    LOAD   Q- 004,I                043604
                                          00031    LRA    Q+ 001                  171401
                                          00032    ADDS,003                       035003
1
PAGE 0002   HEWLETT-PACKARD
0
                                          00033    LDI  ,030                      021030
                                          00034    PCAL,000                       000000
00032000  00035 3                     END;
00033000  00035 2       END;
                                          00035    EXIT,001                       031401


        IDENTIFIER           CLASS          TYPE        ADDRESS

     E                     SIMP. VAR.     INTEGER     Q +001
     FCHECK                PROCEDURE
     FOPEN                 PROCEDURE      INTEGER
     LOGFIL                ARRAY          BYTE        Q +002
     MPEFNO                SIMP. VAR.     INTEGER     Q -004


00034000  00000 1    END.


        IDENTIFIER           CLASS          TYPE        ADDRESS

     DBOPNLOG              PROCEDURE


 PRIMARY DB STORAGE=%000;    SECONDARY DB STORAGE=%00000
 NO. ERRORS=000;             NO. WARNINGS=000
 PROCESSOR TIME=0:00:03;     ELAPSED TIME=0:00:12
```

```
PAGE 0001   HEWLETT-PACKARD 32100A.06.0  SPL  WED, JUL 14, 1976,  7:26 PM  (C) COP
0
00001000  00000 0   $CONTROL NOWARN,INNERLIST
00002000  00000 0   $CONTROL MAP,SUBPROGRAM
00003000  00000 0   <<                                                              >>
00004000  00000 0   << DBLOG...06/24/67:CV                                          >>
00005000  00000 0   << This procedure will log transactions against an IMAGE        >>
00006000  00000 0   << Data Base to a previously defined and opened Shared MPE      >>
00007000  00000 0   << variable length file referenced by the MPE file number       >>
00008000  00000 0   << MPEFNO.  This LOGFIL together with the STORE tape of         >>
00009000  00000 0   << the Data Base taken prior to D.B. modifications should       >>
00010000  00000 0   << provide 'FAIL SOFT' capability and allow restoration of      >>
00011000  00000 0   << data integrity to the instant prior to the failure.          >>
00012000  00000 0   <<                                                              >>
00013000  00000 0   << NOTE THAT ONLY TRANSACTIONS WHICH ACTUALLY MODIFY THE        >>
00014000  00000 0   << CONTENT OF THE DATA BASE SHOULD BE LOGGED...IE DBUPDATE      >>
00015000  00000 0   << ...DBPUT AND DBDELETE.                                       >>
00016000  00000 0   <<                                                              >>
00017000  00000 0   << PROCEDURES FOR RESTORING FROM THE LOG FILE WILL BE FOU       >>
00018000  00000 0   << IN THE LOGGING SYSTEM REFERENCE AND OR THE RELOAD PROGRAM>>
00019000  00000 0   << "DBRERUN".                                                   >>
00020000  00000 0   <<                                                              >>
00021000  00000 0   BEGIN
00022000  00000 1   PROCEDURE DBLOG(MPEFNO,DBASE,DSET,DMODE,DSTAT,DLIST,DLSTLEN,
00023000  00000 1                       DBUFF,DBUFLEN);
00024000  00000 1   VALUE DLSTLEN,DBUFLEN;
00025000  00000 1   INTEGER DMODE,MPEFNO,DLSTLEN,DBUFLEN;
00026000  00000 1   ARRAY DBASE;
00027000  00000 1   ARRAY DSET;
00028000  00000 1   ARRAY DSTAT;
00029000  00000 1   ARRAY DLIST;
00030000  00000 1   ARRAY DBUFF;
00031000  00000 1      BEGIN
00032000  00000 2      INTEGER DATE,LENGTH;
00033000  00000 2      ARRAY LOGREC(0:1023);
00034000  00000 2      LOGICAL PARM,WAIT;
00035000  00000 2      INTEGER POINTER SDATE := @LOGREC(0);
00036000  00000 2      INTEGER POINTER LMODE := @LOGREC(24);
00037000  00000 2      DOUBLE POINTER STIME  := @LOGREC(1);
00038000  00000 2      INTRINSIC FWRITE,FUNLOCK,FLOCK;
00039000  00000 2      INTRINSIC FCHECK,FCONTROL;
00040000  00000 2      INTRINSIC CALENDAR,CLOCK;
00041000  00000 2      <<  BUILD LOG RECORD AFTER GETTING TIME STAMP
00042000  00000 2      PARM                     := TRUE;
                                          00000    ADDS,011              035011
                                          00001    LRA    S- 000         171700
                                          00002    STOR   Q+ 003         051403
                                          00003    LOAD   P+ 000         040000
                                          00004    ADDS,000              035000
```

```
                                                     00005       LOAD    Q+ 003                  041403
                                                     00006       STOR    Q+ 006                  051406
                                                     00007       LOAD    Q+ 003                  041403
                                                     00010       ADDI,030                        022430
                                                     00011       STOR    Q+ 007                  051407
                                                     00012       LOAD    Q+ 003                  041403
                                                     00013       ADDI,001                        022401
                                                     00014       STOR    Q+ 010                  051410
                                                     00015       LDNI,001                        025001
                                                     00016       STOR    Q+ 004                  051404
1
PAGE 0002   HEWLETT-PACKARD
0
00043000   00017 2         WAIT                   := TRUE;
                                                     00017       LDNI,001                        025001
                                                     00020       STOR    Q+ 005                  051405
00044000   00021 2         LENGTH                 := DLSTLEN + DBUFLEN + 35;
                                                     00021       LOAD    Q- 006                  041606
                                                     00022       ADDM    Q- 004                  071604
                                                     00023       ADDI,043                        022443
                                                     00024       STOR    Q+ 002                  051402
00045000   00025 2         SDATE                  := CALENDAR;
                                                     00025       ZERO,  NOP                      000600
                                                     00026       PCAL,000                        000000
                                                     00027       STOR    Q+ 006,I                053406
00046000   00030 2         STIME                  := CLOCK;
                                                     00030       DZRO,  NOP                      000700
                                                     00031       PCAL,000                        000000
                                                     00032       STD     Q+ 010,I                163410
00047000   00033 2         MOVE LOGREC(3)         := DBASE,(13);
                                                     00033       LDXI,003                        021403
                                                     00034       LRA     Q+ 003,I,X              177403
                                                     00035       LRA     Q- 013,I                173613
                                                     00036       LDI ,015                        021015
                                                     00037       MOVE,004,003                    020023
00048000   00040 2         MOVE LOGREC(16)        := DSET,(8);
                                                     00040       LDXI,020                        021420
                                                     00041       LRA     Q+ 003,I,X              177403
                                                     00042       LRA     Q- 012,I                173612
                                                     00043       LDI ,010                        021010
                                                     00044       MOVE,004,003                    020023
00049000   00045 2         LMODE                  := DMODE;
                                                     00045       LOAD    Q- 011,I                043611
                                                     00046       STOR    Q+ 007,I                053407
00050000   00047 2         MOVE LOGREC(25)        := DSTAT,(10);
                                                     00047       LDXI,031                        021431
                                                     00050       LRA     Q+ 003,I,X              177403
                                                     00051       LRA     Q- 010,I                173610
                                                     00052       LDI ,012                        021012
                                                     00053       MOVE,004,003                    020023
```

```
00051000   00054 2        MOVE LOGREC(35)         := DLIST,(DLSTLEN);
                                            00054     LDXI,043                       021443
                                            00055     LRA    Q+ 003,I,X              177403
                                            00056     LRA    Q- 007,I                173607
                                            00057     LOAD   Q- 006                  041606
                                            00060     MOVE,004,003                   020023
00052000   00061 2        MOVE LOGRED(35+DLSTLEN):= DBUFF,(DBUFLEN);
                                            00061     LDI  ,043                      021043
                                            00062     ADDM   Q- 006                  071606
                                            00063     STAX, NOP                      004300
                                            00064     LRA    Q+ 003,I,X              177403
                                            00065     LRA    Q- 005,I                173605
                                            00066     LOAD   Q- 004                  041604
                                            00067     MOVE,004,003                   020023
00053000   00070 2        FLOCK(MPEFNO,WAIT);
                                            00070     LOAD   Q- 014,I                043614
                                            00071     LOAD   Q+ 005                  041405
                                            00072     PCAL,000                       000000
00054000   00073 2        FWRITE(MPEFNO,LOGREC,LENGTH,0);
                                            00073     LOAD   Q- 014,I                04361.
I
PAGE 0003    HEWLETT-PACKARD
0
                                            00074     LOAD   Q+ 003                  041403
                                            00075     LOAD   Q+ 002                  041402
                                            00076     ZERO, NOP                      000600
                                            00077     PCAL,000                       000000
00055000   00100 2        FCONTROL(MPEFNO,6,PARM);
                                            00100     LOAD   Q- 014,I                043614
                                            00101     LDI  ,006                      021006
                                            00102     LRA    Q+ 004                  171404
                                            00103     PCAL,000                       000000
00056000   00104 2        FUNLOCK(MPEFNO);
                                            00104     LOAD   Q- 014,I                043614
                                            00105     PCAL,000                       000000
00057000   00106 2        END;
                                            00106     EXIT,011                       031411


        IDENTIFIER            CLASS          TYPE        ADDRESS

     CALENDAR             PROCEDURE      LOGICAL
     CLOCK                PROCEDURE      DOUBLE
     DATE                 SIMP. VAR.     INTEGER      Q +001
     DBASE                ARRAY          LOGICAL      Q -013
     DBUFF                ARRAY          LOGICAL      Q -005
     DBUFLEN              SIMP. VAR.     INTEGER      Q -004
     DLIST                ARRAY          LOGICAL      Q -007
     DLSTLEN              SIMP. VAR.     INTEGER      Q -006
     DMODE                SIMP. VAR.     INTEGER      Q -011
```

```
    DSET                 ARRAY          LOGICAL      Q -012
    DSTAT                ARRAY          LOGICAL      Q -010
    FCHECK               PROCEDURE
    FCONTROL             PROCEDURE
    FLOCK                PROCEDURE
    FUNLOCK              PROCEDURE
    FWRITE               PROCEDURE
    LENGTH               SIMP. VAR.     INTEGER      Q +002
    LMODE                POINTER        INTEGER      Q +007
    LOGREC               ARRAY          LOGICAL      Q +003
    MPEFNO               SIMP. VAR.     INTEGER      Q -014
    PARM                 SIMP. VAR.     LOGICAL      Q +004
    SDATE                POINTER        INTEGER      Q +006
    STIME                POINTER        DOUBLE       Q +010
    WAIT                 SIMP. VAR.     LOGICAL      Q +005

0058000   00000 1    END.

       IDENTIFIER            CLASS            TYPE          ADDRESS

    DBLOG                PROCEDURE


PRIMARY DB STORAGE=%000;    SECONDARY DB STORAGE=%00000
NO. ERRORS=000;             NO. WARNINGS=000
PROCESSOR TIME=0:00:05;     ELAPSED TIME=0:00:20
```