Worldwide Response Center

HP 3000 APPLICATION NOTE #60

TurboIMAGE's I-FILES and J-FILES

HEWLETT PACKARD

September 1, 1989
Document P/N 5959-7368

RESPONSE CENTER APPLICATION NOTES

HP 3000 APPLICATION NOTES *are published by the Worldwide Response Center twice a month and are distributed with the Software Status Bulletin. These notes address topics where the volume of calls received at the Center indicates a need for addition to or consolidation of information available through HP support services.*

Following this publication you will find a list of previously published notes and a Reader Comment Sheet. You may use the Reader Comment Sheet to comment on the note, suggest improvements or future topics, or to order back issues. We encourage you to return this form; we'd like to hear from you.

TurboIMAGE's I-FILES AND J-FILES

When running applications that access a data base, has a message like the following ever appeared:

```
INTERNAL ERROR ENCOUNTERED
PLEASE SEND :STORE TAPE OF PRIVILEGED FILE I#######.PUB.WHATEVER
TO HEWLETT-PACKARD.  TO DELETE THIS FILE, LOG ON IN PUB.WHATEVER
AND RUN DBDRIVER.PUB.SYS,PURGE
```

This message might look pretty forbidding, but it is actually a flag and a tool for finding problems in data bases. If you have ever wondered about the how's, why's, and so-what's of I-FILEs, this article will be of great importance to you. In this article, we will introduce you to the I-FILE and discuss how to analyze it.

This article describes:

- The TurboIMAGE/V and TurboIMAGE/XL control block structures
- An introduction to the uses of the new I-FILE and J-FILE of TurboIMAGE/XL
- The events that could generate an I or J file
- The I-FILE and J-FILE contents
- Sample I and J files.

This article concludes with a case study illustrating how I-FILE analysis played a major role in problem identification and resolution.

I. GLOBAL STRUCTURES OF TurboIMAGE/V AND TurboIMAGE/XL

An I-FILE consists of unformatted TurboIMAGE data as well as some additional user-stack information. The TurboIMAGE data is captured in control blocks. TurboIMAGE on both V/E and XL machines, use control block structures to control and coordinate access to a data base. In order to analyze an I-FILE, it is imperative that you understand the information contained in TurboIMAGE control blocks.

A. TurboIMAGE/V CONTROL BLOCKS

TurboIMAGE/V control blocks are extra data segments used by the TurboIMAGE intrinsics to store data. The stored data allows us to coordinate multiple access to a database. By using multiple control blocks, TurboIMAGE overcomes the constraints of the MPE/V extra data segment size limitations. The use of multiple control blocks also allows for more information within each data segment. The three control blocks we will discuss are: the DBG, DBB, and DBU.

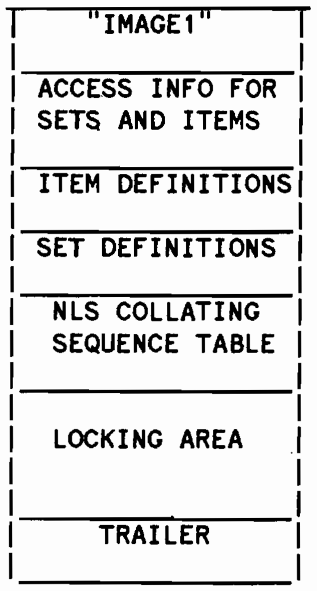
1. DBG - DATA BASE GLOBALS

The DBG data segment contains an area used for locking as well as global schema information from the root file. The DBG allows TurboIMAGE intrinsics to quickly access required security information since these structures are in memory and pointers are maintained in the user's stack. The DBG contains root file information such as: access tables, set definitions, item definitions, and Native Language Support collating sequences. This extra data segment is created before any of the other TurboIMAGE extra data segments at the time the data base is first opened and will be the last TurboIMAGE extra data segment purged at the time the last user in the data base does a DBCLOSE.

Each TurboIMAGE extra data segment begins with a "tag". This tag not only allows us to identify that the segment belongs to TurboIMAGE, but also indicates which control block it is.

The DBG begins with the literal value "IMAGE1". The DBG layout is pictured below:

DATA BASE GLOBALS
DBG



2. DBB - DATA BASE BUFFERS

The DBB control block contains areas for tasks such as logging management and buffer management. Since all TurboIMAGE I/O is done through buffers, the best performance is gained by coordinating the use of the DBB buffers. The DBB allows us to do that by providing a control block whose main purpose is to house and facilitate the coordination of all buffer usage. Logging, like buffer management, also requires extensive coordination. Whether we are doing ILR logging, user logging, or both, we need a monitored environment to ensure the proper application of changes. This activity also takes place within the DBB.

This extra data segment is created at the time of the first DBOPEN after the DBG is created and will be purged at the DBCLOSE of the last user accessing the data base.

The DBB begins with a tag of "IMAGE3". The DBB layout is pictured below:

```
           DATA BASE BUFFERS
                  DBB

         ___________________
        |     "IMAGE3"      |
        |___________________|
        | LOGGING/BUFFER    |
        |   GLOBAL INFO     |
        |___________________|
        | ILR INFORMATION   |
        |___________________|
        |  I/O BUFFERS      |
        |                   |
        |                   |
        |                   |
        |                   |
        |                   |
        |                   |
        |___________________|
        |     TRAILER       |
        |___________________|
```

3. DBU - DATA BASE USER DATA SEGMENT

The DBU data segment contains specific user information. A DBU will be created each time a DBOPEN is performed on a database. For example, if the same process opens a database twice, two DBU's will be created for that process. This is done to keep process specific access information for each open of a data base discrete. The data retained in the DBU is item and set access information. The DBU also contains the current list maintained for the user from the last intrinsic performed.

The DBU begins with "IMAGE2" as a tag to allow identification as a TurboIMAGE extra data segment. The layout of DBU follows below:

```
        DATA BASE USERS
              DBU

     ____________________
    |     "IMAGE2"       |
    |____________________|
    | SECURITY ACCESS    |
    | TABLE FOR THIS     |
    |       DBOPEN       |
    |                    |
    |____________________|
    |   CURRENT LIST     |
    |____________________|
    |     TRAILER        |
    |____________________|
```

B. TurboIMAGE/XL CONTROL BLOCK STRUCTURES OR "MAPPED FILES"

On the XL machines, the TurboIMAGE control block structures are used for the same purpose as on MPE/V: to store process information and coordinate access. Unlike MPE/V, though, the DBG and DBB have been combined into a single "mapped file" called <dbname>GB. The DBU is also a mapped file which is, however, temporary and unnamed.

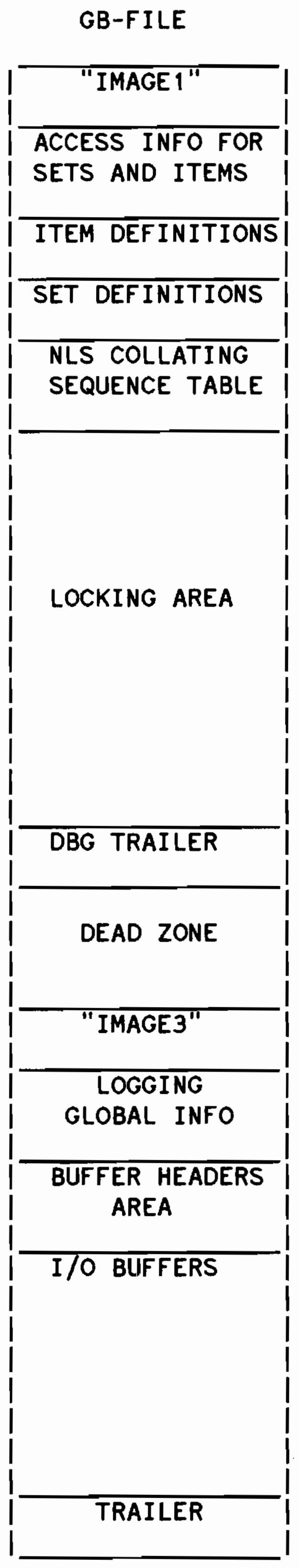
1. WHY "MAPPED" FILES ARE USED BY TurboIMAGE/XL

A "mapped file" is a file which can reside in memory or on disc or a combination of memory and disc. The advantage of "mapped files" is that they can be accessed as virtual memory, which allows for high speed performance. This performance increase is accomplished by taking advantage of the new architecture's *Demand Paging* to to page and prefetch (the file into main memory). No longer is there any need for File System buffers. Main memory serves as the only buffer needed by the new file system. Another reason for better performance is that these "mapped files" are accessed and controlled by the use of specific pointers to direct reads within the file. These reads are controlled and maintained by the special intrinsics: FPOINT and FCONTROL. The FREAD and FWRITE intrinsics used by TurboIMAGE/V therefore are not required by TurboIMAGE/XL.

2. GB-FILE

The information contained in the GB-FILE is the same information contained in TurboIMAGE/V control blocks. The major difference is that the DBG and DBB are combined into one "mapped" file. This "mapped file" has a file code of -403 and is binary. Though we call it the "GB-File", its true name is <dbname>GB where <dbname> is the name of the database.

At a glance the <dbname>GB looks like this:



3. XL - DBU

The TurboIMAGE/XL DBU is also a mapped file. However, it is not permanent and does not have an MPE file name or file code. It is a temporary file local to the process that created it with a DBOPEN. Its structure is the same as TurboIMAGE/V DBU structure.

II. I-FILE and J-FILE STRUCTURES

A. I-FILEs NAMING CONVENTION

This section presents how the TurboIMAGE/V control blocks and other user information are represented in I-FILEs. Unlike cars, homes, and people, I-FILEs do not come in a variety of shapes, colors, and sizes. They always have the same form. I-FILEs derive their name from the fact they begin with the letter "I" coming from IMAGE. The next seven characters of the file name represent the Julian date and military time of the I-FILE creation. The time is based on a 24-hour clock. This naming convention is similar to the naming convention used by EDITOR and TDP for K-FILES. An I-FILE called I0031300 indicates an I-FILE that was created on the third day of the year at 1:00 pm. The I-FILE naming algorithm is illustrated below:

```
"I<ddd><tttt>"  where <ddd> = Julian date (1:365)
                and <tttt>= Military time (0001:2400)
```

The I-FILE is created in the logon group and account of the user encountering the TurboIMAGE abort. This can be a different group and account that in which the database resides. Upon creation of an I-FILE, TurboIMAGE will inform the user that an I-FILE was produced. If the I-FILE cannot be found, chances are the user is looking in the wrong group and account. Usually a :LISTF I#######.@.@,2 will locate the file.

Another distinguishing feature of an I-FILE is its physical characteristics. I-FILEs are created with a negative file code which means they are privileged files. This lends extra file protection to the I-FILE since they cannot be purged, renamed, or modified accidentally.

B. WHAT IS A J-FILE AND WHY IS IT ONLY ON THE XL?

There is a slight twist to the I-FILE concept on the XL machine. A new data structure called the "J-FILE" was introduced along with the redefinition of the I-FILE. The J-FILE is not really a new data structure. It is just a new name for the TurboIMAGE/V I-FILE. The J-FILE naming convention is the same as the I-FILE so a J-FILE called J0041400 was created on the fourth day of the year at 2:00 p.m. Like the I-FILE, the J-FILE contains the TurboIMAGE control blocks related to the aborted process: the <DBNAME>GB and the DBU.

The MPE/XL I-FILE contains a stack marker trace and stack dump of the aborting process. This I-FILE is built based on a script file: IMAGEDMP.PUB.SYS. A quick look at that file will reveal a series of DEBUG commands. These commands are used to create a file containing information specific to the environment in which the abort occurred. The IMAGEDMP.PUB.SYS file has been written by experienced personnel to capture:

- Environmental variables
- Stack marker traces
- NM registers
- CM globals and registers

C. WHY WOULD AN *I-FILE BE PRODUCED?

*(For purposes of this discussion, I-FILEs will refer to both I and J-FILEs for the MPE/XL machine and I-FILEs on the MPE/V machine.)

An I-FILE will be produced only when certain TurboIMAGE aborts occur which call the intrinsic DBABORT AND the database being accessed has been enabled for dumping via DBUTIL. To enable a database for "dumping":

```
1) Logon as the data base creator.

2) :RUN DBUTIL.PUB.SYS

3) >>ENABLE <dbname> FOR DUMPING.
```

Once the database has been enabled for dumping, an I-FILE will be created when one of the following conditions exist:

- INTERNAL ERROR(S) from MPE File System Intrinsics
- INTERNAL INCONSISTENCY in the data base or extra data segments discovered by Image Library Procedures.
- A faulty user calling sequence
- A call from a user process with the hardware DB register not pointing to the process stack.

MPE file intrinsics such as FOPEN, FREADLABEL, FREADDIR, FWRITELABEL, FWRITEDIR, or FCLOSE impact the structure and contents of files. Any type of error when calling these procedures may result in a TurboIMAGE abort. This type of error would be reported as either INTERNAL ERROR or INTERNAL INCONSISTENCY. Typically the user may find or more of the following:

- Corrupt User/File Labels
- Corrupt Chain Head
- Broken Detail Chains
- Corrupted Delete Chains
- Virtual Memory Problems
- Extra Data Segment Corruption Due to Privileged Mode Programming

Why do these conditions cause TurboIMAGE to abort? Let us examine each condition to see what is taking place and why.

At the time of FOPEN, checks are done on several areas of the file label to ensure integrity. Two of these checks are: CHECKSUM and COLDLOAD ID. Either of these two might return an error condition which would signal corruption.

Corrupted chains due to bad chain heads, bad detail records pointers, or broken delete chains can be detected by FREADDIR or FWRITEDIR. A TurboIMAGE chain is simply a structure containing pointers within the dataset. An I-FILE will be produced when these pointers do not point to a record or an address within the file or when the pointed-to records within the file are illogical (bit-map indicates that the record which is to be updated is not in use.)

Virtual Memory problems can be sensed during an FOPEN or at a time when the DBB is swapped out of memory to enlarge or shrink the extra data segment size. The number of buffers used per accessor is determined through the DBUTIL command SET BUFFSPECS. By varying the number of buffers used per accessor, TurboIMAGE has to swap the DBB in and out of virtual memory. This swapping increases the probability of problems with virtual memory. The problem will be a lack of available space to copy the DBB from or to Virtual Memory. By setting the BUFFSPECS in DBUTIL to a constant value, say 100(1/120) this swapping will be avoided as much as possible.

Privledged mode programs in some rare cases could corrupt extra data segments, user stacks, and other memory structures. The corruption experienced during PM processing is usually not intentional but is due to improper programming or poor coding techniques. An example might include an incorrectly set pointer. Consequences of this corruption will cause the intrinsics operating at the time to abort unexpectedly since they return or receive unexpected results.

It is important to remember that the conditions causing an I-FILE to be created signal a *serious* problem. When this problem occurs, it is informing the user that processing in this situation is questionable and that further analysis should be done. *When an I-FILE is created, all users should exit the database and the extent of the database corruption should be determined before users are allowed to access the database.* Users will receive a message indicating that further access is not allowed and that the only TurboIMAGE intrinsic that will be performed is a DBCLOSE of the database in question.

Though all these examples vary from case to case, the cause and effect relationship is still the same for each case:

> I-Files are created by ->
>
> > File Intrinsic Failures due to ->
> >
> > > Some type of corruption

A later section of this article contains two I-FILEs: one resulting from Privileged Mode program corruption and another resulting from broken chains. The third example is an I-FILE and a J-FILE generated by TurboIMAGE/XL showing a bad pointer, which is possibly due to a broken chain.

III. ANALYSIS

Now that you know what an I-FILE is and why it is created, it is time to ask yourself what can be done with it. Since an I-FILE cannot be edited or listed, how can it help? More importantly, how can I read it?

As you might have guessed an I-FILE can help by telling you almost everything you need to know about the state of the database at the time of the I-FILE was created. The next question should be: "How can I use an I-FILE to isolate the problem. Since an I-FILE cannot be edited or listed, how can it help me? More importantly, how can I read it?

There is a Hewlett Packard tool that will help you "dump" the I-FILE for inspection: DBDRIVER.PUB.SYS. DBDRIVER may be used for a wide variety of diagnostic purposes, however in order to print an I-FILE the program must be run with the "CLONE" entry point.

Running DBDRIVER

To access and dump the I-FILE with the DBDRIVER program, the user must satisfy one of two conditions:

- Be the creator of the I-FILE, or
- Have PM capability

It is this program (DBDRIVER) that allows a user to "dump" an I-FILE. The formal file designator for the output of DBDRIVER is DBDRLIST. By default, DBDRLIST is directed to "LP". The user can redirect the dump to the terminal or to a disc file by issuing a file equation:

```
Terminal------------>  :FILE DBDRLIST=$STDLIST
Disc---------------->  :FILE DBDRLIST;DEV=DISC;SAVE
Deferred Spoolfile-->  :FILE DBDRLIST;DEV=LP,1,1
```

The input file can be file equated as well as the output file. This will enable the user to be signed on to a different account and/or group than that in which the I-FILE resides. Any formal file designator may be used:

```
:FILE xyz=######.PUB.WHATEVER
```

where *xyz is to be used when DBDRIVER prompts for the I-File name (see below) and xyz is equated to a valid I-File.

To "dump" the file, run the DBDRIVER program as indicated below:

```
:RUN DBDRIVER.PUB.SYS,CLONE
```

The user will be prompted for the name of the file to be "cloned". The group and account cannot be specified so make sure you are logged into the group and account in which the I-FILE exists or use a file equation as mentioned above.

Please remember throughout this discussion that a complete analysis of an I-FILE should be conducted by a trained Hewlett-Packard employee and that the information discussed here will allow the user a cursory glance at the problem.

As explained earlier, the I-FILE contains a short description of the problem, a stack trace, the user stack, and the TurboIMAGE control blocks associated with that particular database.

A typical analysis of an I-FILE would start with interpreting the ASCII message found at the beginning of the I-FILE. These messages will include the name of the aborted TurboIMAGE intrinsic along with the database name and abort location. For example:

```
ABORT: DBFIND ON DATABASE TESTDB.PUB.MYACCT;
IMAGE ABORTS AT PROCEDURE: 000737; ADDRESS: 042046
INTERNAL TRAP ENCOUNTERED.
```

Although this message does not give a comprehensive report of the problem, it does tell you that a problem occurred when executing a DBFIND on the database called TESTDB.PUB.MYACCT. At this point, it might be in the best interest of the user to run some type of diagnostic program against the database to verify consistency. DIOGENES, a database diagnostic program usually found in the TELESUP account, will help identify any type of structural inconsistency.

The other information displayed in the message in this example may not be helpful to users since TurboIMAGE source code is not available to reference the offset. But once again, this message does identify the database affected.

Other sample messages contained in I-FILEs might read:

```
LOST FREE SPACE in DATASET ## or UNABLE TO COMPLETE ILR FUNCTION.
```

After examining the ASCII text, the next step in I-FILE analysis is to review the error cells of the DBB. The following table illustrates the meaning of each cell and its placement within the DBB:

Cell Contents	MPE/V	MPE/XL	Comments
• Error Number	%100	$168-$169 bytes	The TurboIMAGE error number corresponding to the offending aborted procedure.
• Error Dataset	%101	$16a-$16b bytes	The dataset in which the error occurred.
• FS Error	%102	$16c-$16d bytes	The file system error returned by the TurboIMAGE intrinsic if appropriate.
• File Number	%103	$16e-$16f bytes	The file number assigned to the dataset in question by the file system. Since TurboIMAGE datasets are globally opened, this number will be negative.

IV. EXAMPLES OF I-FILE AND J-FILE ANALYSIS

Following are examples of parts of actual I-FILEs and J-FILEs with a discussion of the most pertinent areas of concern.

A. TurboIMAGE/V

1. EXAMPLE OF I-FILE SHOWING BROKEN CHAIN

This example shows portions of an I-FILE generated due to bad pointers in a sorted chain. The user who received the error attempted to do a DBPUT to an entry with a pointer that pointed outside the dataset. Because the database was enabled for dumping when the user attempted the DBPUT, the same message was displayed to the user as appears at the beginning of this I-FILE.

(For brevity and readability in this article, only the pertinent information is displayed.)

```
ABORT: DBPUT    ON DATA BASE EXAMP.PUB.SYS;IMAGE ABORTS AT PROCEDURE: 000627; A
IMAGE ABORTS AT PROCEDURE: 000627: ADDRESS: 42730
CRITICAL READ ERROR ON DATA SET #5.
END OF FILE (FSERR 0)
```

The following is a stack dump of the aborting procedure. This information is useful only if a map of the procedure is available.

```
***    STACK  DISPLAY    ***       ID # 4
              S=051007   DL=177650   Z=05361
      Q=051013 P=006720  LCST= S067  STAT=P,1,0,L,0,
      Q=050570 P=002727  LCST= S072  STAT=P,1,0,L,0,
      Q=050316 P=012166  LCST=  020  STAT=U,1,1,L,0,
      Q=030206 P=017201  LCST=  021  STAT=U,1,1,L,0,
      Q=027723 P=020247  LCST=  021  STAT=U,1,1,L,0,
      Q=027717 P=177777  LCST= S156  STAT=P,1,0,L,0,
```

The following is the actual stack of the aborting program.

```
..DB..                                        OCTAL                                                          ASCII
-00130  000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
-00114  000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
-00100  000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
-00064  000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
-00050  000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
-00034  000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
-00020  000000 000000 000000 000000 000000 000000 000000 000000 022422 000000 177777 000000.. .. .. .. .. .. .. .. $. .. .. ..
-00004  000000 177777 000000 177777 000000 000526 002024 002044 002072 002476 003102 003112.. .. .. .. .. .V .. .$ .: .> .B .J
 00010  003144 003164 003204 004724 004744 005064 005470 005510 005544 000000 000000 000000.d .t .. .. .. .4 .8 .H .d .. .. ..
 00024  000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
```

The following is the DBB which we can tell from the value "IM AG E3" which appears as the header at the beginning of this block of data. This data segment contains the File System error referred to in the ANALYSIS section. In this data segment words %100-%103 contain this important tracing information.

```
00000 044515 040507 042463 000367 000426 015600 002401 000000 000000 000000 000001 000000 IM AG E3 .. .. .. .. .. .. .. .. ..
00014 000000 000000 000000 000000 000000 000002 000000 042530 040515 050040 050125 041040.. .. .. .. .. .. .. EX AM P  PU B
00030 020040 020040 046525 051120 044131 020040 024704 000006 000000 000000 176000 000000       SY S    ). .. .. .. .. ..
00044 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
00060 000000 000000 177777 000000 000000 177777 000430 002401 000000 000000 000000 000003.. .. .. .. .. .. .. .. .. .. .. ..
```

Following are words %100-%103:

```
00074  000006 000000 000000 000000 177775 000005 000000 177770 000000 002001 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
```

Word %100:   IMAGE ERROR NUMBER   %177775 #-3 FREADDIR FAILURE
This means that when attempting to access an address, an error condition occurred. For example the pointer was outside the bounds of this dataset.

Word %101:   SET NUMBER   %000005 #5 DATA SET 5
The error occurred while attempting to access dataset 5.

Word %102:   FILE SYSTEM ERROR   %000000 #0 FSERR 0
An FSERR 0 (END OF FILE) occurred while trying to find the values pointed to in a chain.

Word %103:   FILE NUMBER   %177770 #-8 GLOBAL AFT ENTRY 8
The file number assigned to this file in the GLOBAL Available
File Table, which holds entries for each dataset opened, is 8.

```
00110   000000 000000 000000 000000 000000 000000 000223 000213 000316 000416 000470 000470.. .. .. .. .. .. .. .. .. .. .8 .8
00124   006564 000102 000074 000044 000000 006074 010000 000000 000000 000000 000000 000000.t .B .< .$ .. .< .. .. .. .. .. ..
00140   000000 000000 000000 000771 000006 000000 000002 000002 000001 000003 000000 000000.. .. .. .. .. .. .. .. .. .. .. ..
```

The following is the DBG which begins with the literal "IM AG E1". Notice as well the TurboIMAGE version is contained in the header of this extra data segment.

```
00000   044515 040507 042461 000426 000000 011554 041456 030060 032061 000367 000430 000000IM AG E1 .. .. .1 C. 00 41 .. .. ..
00014   000000 000000 000000 000000 000405 000417 000413 000567 000722 000725 001147 001547.. .. .. .. .. .. .. .w .. .. .g .g
00030   000001 000000 000000 000002 000000 000000 042530 040515 050040 050125 041040 020040.. .. .. .. .. .. EX AM P  PU B
00044   020040 046525 051120 044131 020040 000000 000000 000000 000000 000000 000000 000000   SY S    .. .. .. .. .. .. .. ..
00060   000000 000000 000000 000000 000000 000000 020040 020040 020040 020040 040100 100200.. .. .. .. .. ..          @@ ..
```

B. TurboIMAGE/XL

1. GETTING ABORT INFORMATION FROM TurboIMAGE/XL I-FILEs

The previous example featured an I-FILE on TurboIMAGE/V. This next example uses the I-FILE and J-FILE on TurboIMAGE/XL.

The J-FILE contains the same data structures as the TurboIMAGE/V I-FILE. But what does the MPE/XL I-FILE contain?

Remember that this file is produced by the Script contained in the file IMAGEDMP.PUB.SYS and that the IMAGEDMP file should not be changed. Hewlett Packard support personnel DO have the ability to customize this file for the purpose of retaining special information required to fully analyze the abort. It is very important not to alter the variables, since changing or adding new parameters could cause a system abort.

Sometimes there is detailed error message information that can be useful in determining the problem. By using DEBUG, we can "map" into the MPE/XL I-FILE by the use of the following commands, and dump out these error messages.

```
:DEBUG

HPDEBUG Intrinsic at: a.06d3298 hxdebug+12c

$1 ($2f) nmdebug >MAP I0041403 FILECODE -403
1  I0041403.EXAMP.SYS    755.0 Bytes=35100
$2 ($ac) nmdebug >  dv 755.0,#150,b
VIRT $755.0          $41424f52 543a2020 44425055 54202020  ABOR T:   DBPU T
VIRT $755.10         $204f4e20 44415441 20424153 45205445   ON  DATA  BAS E TE
VIRT $755.20         $53544442 2e47524f 55502e41 4343543b  STDB .GRO UP.A CCT;
VIRT $755.30         $20202020 20202020 20202020 20202020
VIRT $755.40         $20202020 20202020 20202020 20202020
VIRT $755.50         $20202020 20202020 20202020 20202020
VIRT $755.60         $20202020 20202020 20202020 20202020
VIRT $755.70         $20202020 20202020 20202020 20202020
VIRT $755.80         $20202020 54555242 4f494d41 47452f58       TURB OIMA GE/X
VIRT $755.90         $4c204142 4f525453 20415420 50524f43  L AB ORTS  AT  PROC
VIRT $755.a0         $45445552 453a2024 30303030 30313937  EDUR E: $ 0000 0197
VIRT $755.b0         $3b204144 44524553 533a2024 30303565  ; AD DRES S: $ 005e
VIRT $755.c0         $39343038 20202020 20202020 20202020  9408
VIRT $755.d0         $20202020 20202020 20202020 20202020
VIRT $755.e0         $20202020 20202020 20202020 20202020
VIRT $755.f0         $20202020 20202020 20202020 20202020
VIRT $755.100        $20202020 20202020 54555242 4f494d41            TURB OIMA
VIRT $755.110        $47452f58 4c204142 4f525453 204f4e20  GE/X L AB ORTS  ON
VIRT $755.120        $44424220 434f4e54 524f4c20 424c4f43  DBB  CONT ROL  BLOC
VIRT $755.130        $44424220 20202020 20202020 20202020  K
VIRT $755.140        $20202020 20202020 20202020 20202020
VIRT $755.150        $20202020 20202020 20202020 20202020
VIRT $755.160        $20202020 20202020 20202020 20202020
VIRT $755.170        $20202020 20202020 20202020 20202020
VIRT $755.180        $20202020 20202020 20202020 4c4f5354                 LOST
VIRT $755.190        $20465245 45205350 41434520 494e2044   FRE E SP ACE  IN D
VIRT $755.1a0        $41544120 53455420 2331312e 20202020  ATA  SET  #11.
VIRT $755.1b0        $20202020 20202020 20202020 20202020
VIRT $755.1c0        $20202020 20202020 20202020 20202020
VIRT $755.1d0        $20202020 20202020 20202020 20202020
VIRT $755.1e0        $20202020 20202020 20202020 20202020
VIRT $755.1f0        $20202020 20202020 20202020 20202020
VIRT $755.200        $20202020 20202020 20202020 20202020
VIRT $755.210        $454e4420 4f462046 494c4520 20284653  END OF F ILE    (FS
VIRT $755.220        $45525220 30292020 20202020 20202020  ERR 0)
VIRT $755.230        $20202020 20202020 20202020 20202020
VIRT $755.240        $20202020 20202020 20202020 20202020
VIRT $755.250        $20202020 20202020 20202020 20202020
```

$1. To dump this file out we opened the I-FILE as a mapped file (with a file code of -403).

$2. We then display the first #150 bytes of the I-FILE and discover the errors that the running program received:

ABORT DBPUT ON DATA BASE TESTDB.GROUP.ACCT
TURBOIMAGE/SL ABORTS AT PROCEDURE: $00000197;ADDRESS: $005e9408
TURBOIMAGE/XL ABORTS ON DBB CONTROL BLOCK
LOST FREE SPACE IN DATA SET #11
END OF FILE (FSERR 0)

This is actually the beginning of the dump of the aborting procedure. The user who was running the program received this message, however in most cases this message is not seen by anyone but the aborting user. Often this is due to the user running a block mode application with a small window for error messages.

2. GETTING FILE SYSTEM ERRORS FROM AN MPE/XL J-FILE

The J-FILE contains both the DBG and the DBB of the TurboIMAGE/V, otherwise known as the GB-FILE on MPE XL. J-FILEs may be examined by running DEBUG. To use DEBUG to read the privileged J-FILE the user must have PM capability.

Bytes $168 through $16f, on MPE/XL operating systems of 1.1, of the DBB contain the File System error information that was contained in words %100 through %103 of the TurboIMAGE/V DBB (see III. ANALYSIS). Using Debug on XL, we can open the J-FILE as a mapped file, find the start of the DBB in the J-FILE, and from there read the File System error cells.

A step-by-step example follows:

```
:DEBUG
HPDEBUG Intrinsic at: a.006d3298 hxdebug+$12c

$1 ($2f) nmdebug > MAP J0041403 FILECODE -#403
1  J0041403.SANDRA.CT    354.0 Bytes=35100

$2 ($2f) nmdebug > DV 354.0,10,B
VIRT $354.0         $ 494d4147 45310000 e5980000 e5980000  IMAG E1.. .... ....
VIRT $354.10        $ 000073da 0000432e 30303437 e598e7b4  ..s. ..C. 0047 ....
VIRT $354.20        $ 00000000 e59802fc e598026c e598058a  .... .... ...1 ....
VIRT $354.30        $ e598071a e59807ae e59818dc e59820bc  .... .... .... .. .

$3 ($2f) nmdebug > DV 354.73DA*2,10,B
VIRT $354.e7b4      $ 494d4147 45330000 e598e7b4 e5980000  IMAG E3.. .... ....
VIRT $354.e7c4      $ 00010000 00000000 00000000 00000000  .... .... .... ....
VIRT $354.e7d4      $ 00000000 00000000 00000000 00000000  .... .... .... ....
VIRT $354.e7e4      $ 00000000 ffff0000 00000000 00000000  .... .... .... ....


$4 ($2f) nmdebug > DV 354.73DA*2+168,2  <<<remember on 1.0 to use 8A>>>
VIRT $354.e83c     $ fffd000d 000c0013

$5 ($2f) nmdebug > E
:
```

Refer to the Debug commands by the hex number at the beginning of each command line.

$1. We open J-FILE J0041403 as a mapped file, passing the file code, -403.

$2. We display the first $10 words, in hex and ASCII. The control block "tag", IMAGE1, is plainly visible. From word $4 (0 relative) we get the length of the DBG, $73da HALF-WORDS, not words or bytes. The DBB should trail the DBG, and should start at that address.

$3. We display starting at BYTE $73da*2; we have multiplied the half-word figure by two, to get bytes. We display in hex and ASCII, and the DBB "tag", IMAGE3, tells us we made it to the right place.

$4. In the XL J-FILEs, the File System error cells are in half-words starting at byte $168, as mentioned above. We display them, and in this case we can learn the following:

Byte $168-169 IMAGE ERROR NUMBER $fffd #-3 FREADDIR FAILURE
This means that when attempting to access an address, an error condition occurred. Sound familiar?

Byte $16a-16b SET NUMBER $000d #13 DATA SET 13
The error occurred while attempting to access dataset 13.

Byte $16c-16d FILE SYSTEM ERROR $000c #12 FSERR 12
An FSERR 12 (RECORD NUMBER OUT OF RANGE) occurred. This probably means the bad pointer was a negative number due to the sign bit being on.

Byte $16e-16f FILE NUMBER $0013 # 13 FILE NUMBER 13
ON the MPE/XL machine this cell has no real meaning since we no longer have an AFT TABLE.

V. Case Study: I-FILE Analysis Speeds Up Problem Identification

EXAMPLE OF I-FILE WITH PRIVILEGED MODE CODE VIOLATION

Using an actual call history, we will examine how the use of I-FILEs play a major role in diagnosing a particular problem.

A user reported there were sporadic instances when Query would abort when executing a TurboIMAGE intrinsic against a database. The following error message would be displayed on the users terminal:

```
ABORT:  DBDELETE ON DATABASE TESTDB.GROUP.ACCOUNT;

IMAGE ABORTS AT PROCEDURE:  000737;  ADDRESS: 042046

READ ERROR ON DATA SET #8
```

```
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
| File Name is TESTDB08.GROUP.ACCOUNT             |
| FOPTIONS: SYS,BINARY,FORMAL,F,NOCCTL,DEQ        |
|           NOLABEL                               |
| AOPTIONS: IN/OUT,NOMR,NOLOCK,SHR,NOBUF          |
|           NOMULTI,NOWAIT,NOCOPY                 |
| DEVICE TYPE: 3      DEVICE SUBTYPE: 8           |
| LDEV: 2       DRT: 89       UNIT: 0            |
| RECORD SIZE: 128    BLOCK SIZE: 128   (WORDS)  |
| EXTENT SIZE: 4      MAX EXTENTS: 1             |
| RECPTR: 0           RECLIMIT: 2                |
| LOGCOUNT: 0            PHYSCOUNT: 0            |
| EOF AT: 2           LABEL ADDR: %00200247636   |
| FILE CODE: -401   ID IS USER      ULABELS: 1   |
| PHYSICAL STATUS: ????????????????              |
| ERROR NUMBER: -3    RESIDUE: 0                 |
| BLOCK NUMBER: 0            NUMREC: 1           |
+-------------------------------------------------+

ABORT :QUERY.PUB.SYS.%15.%12243:SYSL.%75.%11176
PROGRAM ERROR #18 :PROCESS QUIT .PARAM = 18260
```

PROGRAM TERMINATED IN AN ERROR STATE. (CIERR 976)

Day after day the customer experienced similar errors on different datasets in different databases. No common thread tied these aborts together. So in order to collect more data about the aborting processes, we suggested the user enable all their databases for dumping. At this point, I-FILES were produced each time an abort occurred.

What follows is a step-by-step analysis of the problem. For brevity and readability, only the pertinent information from the I-FILE is displayed.

THE ACTUAL I-FILE ITSELF

We began analyzing the I-FILE by studying the ASCII message contained in the first part of the file.

```
ABORT:  DBDELETE ON DATA BASE TESTDB.GROUP.ACCOUNT;
IMAGE ABORTS AT PROCEDURE: 000737; ADDRESS: 042046
INTERNAL IMAGE TRAP ENCOUNTERED.
```

We were able to verify procedure %737 was DBDELETE and that offset %042046 was a call to DBABORT by studying the TurboIMAGE code. Following the ASCII description was a dump of the stack markers.

```
***   STACK DISPLAY   ***        ID # 15

                  S=003355  DL=177650  Z=00636
         Q=003361 P=006720 LCST= S357 STAT=P,1,0,L,0,CCG X=000011

         Q=003136 P=002045 LCST= S356 STAT=P,1,0,L,0,CCG X=000004
         Q=003115 P=001534 LCST= S356 STAT=P,1,0,L,0,CCL X=000002
         Q=003072 P=004013 LCST= S356 STAT=P,1,0,L,0,CCL X=000006
         Q=003042 P=003147 LCST= S356 STAT=P,1,0,L,0,CCL X=000002
         Q=003023 P=001731 LCST= S363 STAT=P,1,0,L,0,CCL X=000003
         Q=002551 P=003726 LCST=  001 STAT=U,1,1,L,0,CCG X=000017
         Q=002512 P=177777 LCST= S156 STAT=P,1,0,L,0,CCG X=000000
```

This also allowed us to verify that DBDELETE was calling DBABORT. Following the dump of the stack markers was the user stack itself. At this point we were able to verify that the parameters passed from QUERY to DBDELETE were correct.

The TurboIMAGE control blocks followed the user stack dump. Rather than listing all control blocks (some of which are over %10000 words), we will discuss only those pertinent to user analysis.

After finding the DBB by searching for the "IM AG E3" tag,we preceded to dump out the error cells in words %100 - %103:

```
00074  000025 000000.000000 000000 177775 000010 000110 177132 000000 002001 000000 000000.. .. .. .. .. .. .. .H .Z .. .. .. ..
```

```
Word %100:     IMAGE ERROR NUMBER   %177775 #-3 FREADDIR FAILURE
  While attempting to access an address, an error condition occurred.
  The error number indicates a FREADDIR Failure.

Word %101:     SET NUMBER          %000010 #8  DATA SET 8
  The error occurred while attempting to access dataset #8 (TESTDB08).

Word %102:     FILE SYSTEM ERROR    %000110 #0  FSERR 72
  This error indicates an INVALID FILE NUMBER was being used during the
  particular intrinsic in question.

Word %103:     FILE NUMBER          %177132 #-422 GLOBAL AFT ENTRY 422
  The file number assigned to this file (TESTDB08) in the GLOBAL Available
  File Table was -422.
```

As we collected I-Files from each aborted process, we noticed that while the error number and dataset were different, the MPE V assigned file number was always the same: -422. This was significant, the same file number was involved even though the databases and datasets were different. Based on our previous experience, we knew there was a product which had in the past FCLOSEd out datasets due to a defect in its code. Though that particular defect FCLOSEd files assigned with a number of -1, we decided to do some further checking into that same product. Checking through the source code for this product, it was discovered that in rare circumstances an FCLOSE of file number -422 could occur.

The point of this explanation is that I-FILEs allowed us to identify similarities across aborts. This was done by analyzing words %100 - %103 of the DBB. In all of the I-FILEs produced in this case, word %103 was equal to %177132 (-422). Thus we knew we were dealing with a localized problem. Something was FCLOSing globally opened files whose file number was -422. Since we knew that the customer was running the same product that had in the past showed similar problems, we decided to pursue this line of investigation which correctly led us to the problem and the subsequent solution.

BACK ISSUE INFORMATION

Following is a list of the Application Notes published to date. If you would like to order single copies of back issues please use the *Reader Comment Sheet* attached and indicate the number(s) of the note(s) you need.

Note #	*Published*	*Topic*
1	*2/21/85*	*Printer Configuration Guide (superseded by note #4)*
2	*10/15/85*	*Terminal types for HP 3000 HPIB Computers (superseded by note #13)*
3	*4/01/86*	*Plotter Configuration Guide*
4	*4/15/86*	*Printer Configuration Guide - Revised*
5	*5/01/86*	*MPE System Logfile Record Formats*
6	*5/15/86*	*Stack Operation*
7	*6/01/86*	*COBOL II/3000 Programs: Tracing Illegal Data*
8	*6/15/86*	*KSAM Topics: COBOL's Index I/O; File Data Integrity*
9	*7/01/86*	*Port Failures, Terminal Hangs, TERMDSM*
10	*7/15/86*	*Serial Printers - Configuration, Cabling, Muxes*
11	*8/01/86*	*System Configuration or System Table Related Errors*
12	*8/15/86*	*Pascal/3000 - Using Dynamic Variables*
13	*9/01/86*	*Terminal Types for HP 3000 HPIB Computers - Revised*
14	*9/15/86*	*Laser Printers - A Software and Hardware Overview*
15	*10/01/86*	*FORTRAN Language Considerations - A Guide to Common Problems*
16	*10/15/86*	*IMAGE: Updating to TurboIMAGE & Improving Data Base Loads*
17	*11/01/86*	*Optimizing VPLUS Utilization*
18	*11/15/86*	*The Case of the Suspect Track for 792X Disc Drives*
19	*12/01/86*	*Stack Overflows: Causes & Cures for COBOL II Programs*
20	*1/01/87*	*Output Spooling*
21	*1/15/87*	*COBOLII and MPE Intrinsics*
22	*2/15/87*	*Asynchronous Modems*
23	*3/01/87*	*VFC Files*
24	*3/15/87*	*Private Volumes*
25	*4/01/87*	*TurboIMAGE: Transaction Logging*
26	*4/15/87*	*HP 2680A, 2688A Error Trailers*
27	*5/01/87*	*HPTrend: An Installation and Problem Solving Guide*
28	*5/15/87*	*The Startup State Configurator*
29	*6/01/87*	*A Programmer's Guide to VPLUS/3000*
30	*6/15/87*	*Disc Cache*
31	*7/01/87*	*Calling the CREATEPROCESS Intrinsic*
32	*7/15/87*	*Configuring Terminal Buffers*
33	*8/15/87*	*Printer Configuration Guide*
34	*9/01/87*	*RIN Management (Using COBOLII Examples) (A)*
34	*10/01/87*	*Process Handling (Using COBOLII Examples) (B)*
35	*10/15/87*	*HPDESK IV (Script files, FSC, and Installation Considerations)*
34	*11/01/87*	*Extra Data Segments (Using COBOLII Examples) (C)*
36	*12/01/87*	*Tips for the DESK IV Administrators*
37	*12/15/87*	*AUTOINST: Trouble-free Updates*
38	*1/01/88*	*Store/Restore Errors*
39	*1/15/88*	*MRJE Emulates a HASP Workstation*
40	*2/01/88*	*HP 250 / 260 to HP 3000 Communications Guidelines*
41	*4/01/88*	*MPE File Label Revealed - Revised 6/15/88*
42	*7/15/88*	*System Interrupts*
43	*7/15/88*	*Run Time Aborts*
44	*8/01/88*	*HPPA Pathing Conventions For HP3000 900 Series Processors*
45	*8/15/88*	*Vplus & Multiplexers*
46	*8/15/88*	*Setting Up An HPDesk/HPTelex For The First Time*

47	9/15/88	*Customizing Database Data Items & Changing Passwords in JCL Files*
48	11/15/88	*Printer Configuration (Revision #4)*
49	12/01/88	*Configuring DATACOMM Products Into MPE*
50	12/15/88	*VFC's For Serial Printers*
51	1/01/89	*Terminal Types For The HP 3000 HPIB Computers*
52	1/15/89	*Configuring MRJE*
53	2/01/89	*Using Special Characters on the 700/9x Series Terminals*
54	4/01/89	*Improving Database Performance*
55	4/15/89	*Customized Message Catalogs And Help Facilities*
56	5/01/89	*BRW Tips For Beginners*
57	6/15/89	*Configuring The HP 2334A Plus & Hp 2335A As A Statistical Multiplexer*
58	7/01/89	*HPPA Pathing Conventions For HP3000 900 Series Processors (Update)*
59	8/01/89	*HP 2334A and HP 2335A Configuration Recipes*
60	9/01/89	*TurboIMAGE's I-FILES and J-FILES*

READER COMMENT SHEET

Worldwide Response Center Support
HP 3000 Application Note 60: TurboIMAGE's I-FILES and J-FILES
(September 1, 1989)

We welcome your evaluation of this Application Note. Your comments and suggestions help us to improve our publications. Please explain your answers under Comments, below, and use additional pages if necessary.

Is this Application Note technically accurate?	☐ Yes	☐ No
Are the concepts and wording easy to understand?	☐ Yes	☐ No
Is the format of this Application Note convenient in size, arrangement and readability?	☐ Yes	☐ No

Comments and/or suggestions for future Application Notes:

This form requires no postage stamp if mailed in the U.S. For locations outside the U.S., your local HP representative will ensure that your comments are forwarded.

FROM: **Date** __________

Name ____________________

Company ____________________

Address ____________________

____

____

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 95, MT. VIEW, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Application Note Comments
Hewlett-Packard Manufacturing Specifications
690 E. Middlefield Road
Mail Stop 30-0
Attention: AN ORDERS
Mt. View, CA 94043

FOLD FOLD