UNITED STATES GEOLOGICAL SURVEY

UNDELETE

A Program to Recover
Deleted RSX-11 Disk Files

Program Logic Manual

Lawrence M. Baker

Open-File Report 86-418

1986

Abstract

Undelete is a non-privileged utility program for the in-place restoration of accidentally deleted disk files for the DEC PDP-11 RSX-family operating systems. Files are restored based on a user-specifed search pattern, which provides a flexible means of wild card matching by character or by field within a file specification. A dry run option is available to identify candidates for restoration without actually modifying the disk file structure.

1.0 Introduction

This document is a companion to the Undelete User's Manual and Installation Guide, which should be read first. It is assumed that the reader is familiar with the operation of Undelete and is generally familiar with the DEC PDP-11 RSX-family operating systems and the Files-11 on-disk structure maintained by the Files-11 Ancillary Control Processor, F11ACP.

2.0 Method of Operation

The on-disk file structure used by the RSX-family of operating systems stores auxiliary information about the disk contents (i.e., ownership, protection, size , etc.) in a special file called the index file. The index file contains one entry for each file stored on the disk, called a primary file header, or more simply, a file header. Each file header occupies an entire disk block (256 words), and has fields for header information (e.g., ownership, protection, and attributes), identification information (e.g., name, creation date, and revision date), and mapping information (e.g., retrieval pointers that specify where on the disk the pieces of the file are located). If the map area is insufficient to store all the retrieval pointers, additional extension file headers are allocated and chained together as needed. Even though it is a special file, the index file may be opened for read or write access like any other file, except that record I/O operations are undefined and all I/O must be performed directly using block I/O by Virtual Block Number (VBN).

Undelete uses only primary file headers to locate a possible candidate for restoration. Extension file headers are located using the information contained in the file header map area. Since the file system only modifies the map area of extension file headers, never the header or identification areas, this is the only safe method of correctly locating extension file headers that belong with the primary file header. It also avoids the accidental restoration of an extension file header that happens to match the search criteria when there is either no corresponding primary file header, a missing antecedent extension file header, or stale information in the header or identification areas as a result of a change to the primary file header that was not propagated to the extension file header.

When a file is deleted, the file system may or may not be able to carry out the request immediately (depending on the number of file accesses currently in progress). To keep it's bookeeping straight, the file system marks the file for deletion (in the system characteristics byte of the file header, H.SCHA), which prevents any further access to the file, and actually performs the delete operation when the last access terminates. The file system destroys very little in the process of deleting a file:

1. The disk blocks used by the file are marked "available" in the volume storage bit map, but are left undisturbed on the disk;

2. The primary file header and any extension file headers are marked "available" in the index file bit map;

3. The file header fields for the file number (H.FNUM) and the checksum (H.CKSM) are cleared; the other fields in the primary and extension file headers are left undisturbed; and

4. If the file was accessed via a directory (UFD) entry, the directory entry is removed.

Restoration of files whose contents have otherwise not been altered can be accomplished by simply reversing these steps:

1. The file number is implied by the file header's position in the index file;

2. The file header checksum is re-calculated once the file number is restored;

3. The index file bit map is modified to mark the primary and extension file headers "in-use" again;

4. The volume storage bit map is modified to mark as "in-use" again the disk blocks allocated to the file (whose addresses are mapped by the retrieval pointers in the primary and extension file headers); and

5. A directory entry is created for the file so that it may be easily accessed (not necessarily in the original UFD).

Undelete performs the first three steps of the restoration operation and relies on the standard RSX VFY utility to perform the last two steps. (VFY marks the disk blocks as "in-use" using the update option, and creates directory entries for them in UFD [1,3] using the lost file scan option.) VFY also helps reconcile any problems created when some of the disk blocks that were contained in the accidentally deleted files have since been re-used in other files (called "multiply allocated disk blocks" by VFY).

A detailed description of Undelete's operation follows, organized by Fortran subprogram. See the appendices for listings of the Fortran source module, include files, and command file.

3.0 Program UNDELE

1. The user is queried for either the dry run option or actual file recovery. The dry run option causes the logical variable "enable" to be set .FALSE., which is then used to conditionalize calls to PUTBLK and PUTCBK.

2. The user is prompted for the disk name. The form of the response is typically the standard RSX device name format ddnn:, where dd is the device mnemonic and nn is the optional unit number. The only syntax checking performed by Undelete is to make sure the response is terminated by a colon.

3. The disk device name supplied is concatenated with the name of the index file, [0,0]INDEXF.SYS, to form the filename used in the Fortran OPEN statement, which validates the disk name in the process of performing the open. The BUFFERCOUNT=-1 option is used in the OPEN statement to disable record level access to the index file, since all I/O must be performed using QIOs to the operating system directly. If this is not a dry run and the index file cannot be opened for write access, a message is printed and an attempt is made to continue as a dry run. If the index file cannot be opened for read-only access, a message is printed and the user is prompted for the disk name again.

4. The volume home block (VBN 2) is read. If an error occurs, Undelete is terminated with the error status. Otherwise, the size of the index file bit map (H.IBSZ) and the maximum number of files (H.FMAX) is printed on the user's terminal.

5. The search pattern to be used to match with available file headers is constructed (see Function MAKPAT).

6. For each possible file header (1 through H.FMAX):

    1. The header is skipped if it is not marked available (see Function HDRFRE).

    2. The file header is read. If an error occurs, a check is made to determine if an end-of-file was encountered. If so, Undelete is terminated normally. Otherwise, a message is printed and the header is skipped.

    3. The header is skipped if it has never been used before (i.e., the creation date (I.CRDT, I.CRTI) and modification date (I.RVDT, I.RVTI) fields are zero).

    4. The header is skipped if it is not a primary file header, i.e., the extension segment number (M.ESQN) is non-zero. (Extension file headers are dealt with separately in UNDHDR.)

    5. The header is skipped if it does not match the search pattern (see Function MATCH).

6. A message is printed and the header is skipped if it is corrupted (the first word (H.IDOF, H.MPOF) does not contain the value '027027'O (a fixed constant value for all Files-11 structure level 1 volumes) or the file number (H.FNUM) or checksum (H.CKSM) words are not zeroed out).

7. Recovery is attempted for the file header (see Function UNDHDR).

7. The index file is closed.

8. A message is printed with the total number of files that were recovered and instructions for running the VFY utility, or just the number of files that will be recovered, if it was a dry run.

9. Undelete is terminated by calling EXIT with the error status.

4.0 Function UNDHDR

UNDHDR performs the undelete operation for primary file headers and any surviving extension file headers.

1. The file number in the header (H.FNUM) is restored (it is the same as the header number).

2. The marked for delete characteristic (SC.MDL) in the system characteristics byte (H.SCHA) is reset.

3. If this is not a dry run, the file header is re-written with a new checksum. If an error occurs, a message is printed and the error is returned to the caller.

4. The header is marked "in-use" in the index file bit map (HDRFRE told us where), and, if this is not a dry run, the index file bit map is re-written. If an error occurs, a message is printed and the error is returned to the caller.

5. The total number of files recovered is incremented.

6. If the header has extension file headers:

   1. The next extension file header is located.

   2. If the extension file header is corrupt or has since been reused, a message is printed and the previous header is modified to truncate the extension header chain (M.EFNU=0, M.EFSQ=0). (Unfortunately this makes the rest of the file unrecoverable.)

   3. The extension file header is restored, as above, and, if this is not a dry run, the header and index file bit maps

are re-written. If an error occurs, a message is printed and the extension header chain is truncated.

5.0 Function MAKPAT

MAKPAT constructs the search pattern for MATCH from the user's response.

1. The user is prompted for the search pattern ([group,member]name.type;version). If an end-of-file is read, an error is returned to the caller.

2. The final search pattern is initialized with the "?" wild card character.

3. The user's response is converted to upper case.

4. If the group field was specified, and at least one non-wilcard was typed, then the preloaded "?"s are changed to "0"s, and the group number is right justified in the search pattern.

5. If the member field was specified, and at least one non-wilcard was typed, then the preloaded "?"s are changed to "0"s, and the member number is right justified in the search pattern.

6. If the name field was specified, it is copied into the search pattern until a period or a semicolon is found.

7. If the type field was specified, it is copied into the search pattern until a semicolon is found.

8. If the version number field was specified, and at least one non-wilcard was typed, then the preloaded "?"s are changed to "0"s, and the version number is right justified in the search pattern.

9. Any "*"s are converted to "?"s and the characters in the final search pattern are validated (all numeric fields must be octal or "?", and all alphanumeric fields must be RAD50 or "?"). If there are any syntax errors or illegal characters, a message is printed and the user is prompted for the search pattern again.

10. The final search pattern is printed on the user's terminal and success is returned to the caller.

6.0 Function MATCH

MATCH determines if a file header qualifies for recovery by comparing the file identification fields with the search pattern specified by the user. The format of the 24 character search pattern is:

```
                    "gggmmmfffffffffftttvvvvvv"

        g ( 1- 3) = Owner's group number (octal)
        m ( 4- 6) = Owner's member number (octal)
        f ( 7-15) = File name
        t (16-18) = File type
        v (19-24) = File version number (octal)
```

"?" in any position matches any character in the actual file specification. All numeric fields not wild carded must have leading zeros in the search pattern. The file name and file type fields must contain legal RAD50 characters (upper case letters and digits).

1. Assume the match will fail.

2. The UIC group (H.PROG) and member (H.PROJ) numbers are converted to octal. If the pattern match to the group number fails, return. If the pattern match to the member number fails, return.

3. The file name (I.FNAM) is converted from RAD50 to ASCII. If the pattern match to the file name fails, return.

4. The file type (I.FTYP) is converted from RAD50 to ASCII. If the pattern match to the file type fails, return.

5. The version number (I.FVER) is converted to octal. If the pattern match to the version number fails, return.

6. The pattern match is successful. A PIP-style directory line is printed on the user's terminal.

NOTE

The user may want to insert a query capability here as an option, and return no match if the user declines to restore this file (e.g., the type of prompt used by PIP and SRD for selective file deletions).

7.0 Function HDRFRE

HDRFRE returns the availability status of a file header (.TRUE. = header is marked "available-for-use") by examining the corresponding "in-use" flag in the index file bit map. Header number 1 corresponds to bit number 0 in the bit map; there are 4096 bits to a disk block (256 words), starting at VBN 3.

To optimize program performance, a write-through cache is maintained by retaining the validity of the VBN of the last bit map block read throughout the program. No read is performed if the VBN needed is the same as the last one read. (It's contents may have changed due to setting "in-use" flags, but the memory copy is always the same as the disk block, since a write is performed whenever an "in-use" flag is set.)

1. The desired bit map block number is calculated from the file header number.

2. If the desired bit map block is not the one in memory, the new block is read. If an error occurs, the bit map block number is set to -1. Otherwise, the bit map block number is updated.

3. If the bit map block number is the same as the desired bit map block:

    1. The index of the word containing the header's "in-use" flag and the mask value for testing/setting the state of the "in-use" flag is calculated.

    2. The complement of the "in-use" flag is returned to the caller.

4. Otherwise "header-in-use" is returned to the caller.

8.0 Functions GETBLK, PUTBLK, and PUTCBK

GETBLK reads a virtual block from the index file. PUTBLK writes a virtual block to the index file. PUTCBK computes the checksum of words 1 through 255 into word 256, then writes a virtual block to the index file.

1. If entry was at GETBLK, the I/O function code is set to IO.RVB.

2. If entry was at PUTCBK, the checksum of words 1 through 255 is calculated and placed into word 256 of the buffer. Execution continues at the PUTBLK entry.

3. If entry was at PUTBLK, the I/O function code is set to IO.WVB.

4. The operating system WTQIO subroutine is called with the appropriate I/O function code.

5. The value of the directive status word, if the operating system rejected the request, or the I/O status byte, if the request was accepted, is returned to the caller.

9.0 Subroutine UPCASE

UPCASE converts a string to upper case characters.

For each character in the string:

1. The character is converted to its integer equivalent.

2. Comparison is made with the integer equivalents for lower case a and z.

3. If it's lower case, the character is replaced with the corresponding single character substring from the upper case alphabet in ALPHA.

10.0 Acknowledgements

I would like to thank Jon Berger for the circumstances that necessitated writing Undelete, Richard Sipura for inspiring me to update it and finally write it up, and to my reviewers, Tim MacDonald and Gary Maxwell.

11.0 References

1. IAS/RSX-11 I/O Operations Reference Manual (Order No. AA-M176A-TC), Appendix A, "File Descriptor Block", Appendix E, "Index File Format", and Appendix F, "File Header Block Format".

2. RSX-11M/M-PLUS Utilities Manual (Order No. AA-L681A-TC), Chapter 9, "File Structure Verification Utility (VFY)".

3. Baker, Lawrence M., 1986, Undelete: A Program to Recover Deleted RSX-11 Disk Files, User's Manual and Installation Guide: U. S. Geological Survey Open-File Report 86-375, 8 p.

APPENDIX A

Fortran Source Module

A.1 Fortran Source Module (Undelete.ftn)

```
      Program UNDELE
C
C Undelete - Restore deleted files from an RSX-11 disk.
C
C This is not a privileged program!  It should only be run by someone
C knowledgeable about the system and Files-11 disk structures, and
C should normally NOT be INStalled for general use.  There must be no
C other activity on the disk and it must be MOUnted/UNL.  Naturally,
C this should only be run from a privileged terminal.  (Fortran-77 is
C required to perform 32-bit integer arithmetic.)
C
C Search patterns for recovery are specified in a form similar to the
C SRD /SElectentry option:  ? and * are single character wild cards
C and a stem search is automatically performed.  Any unspecified fields
C default to wild cards.  Note that the special version numbers, 0 and
C -1, are NOT supported.
C
C For example:
C
C      To recover all the .FTN files owned by [100,2], specify the
C      search pattern [100,2]*.FTN.
C
C      To recover all files beginning with BP2 owned by any account,
C      you may simply specify BP2.
C
C See the comments in the code for possible modifications ('NOTE').
C There should not be any required to run on an RSX-11M V3.1 system or
C on an RSX-11M-Plus V2.1 system (and possibly on a VAX in compatibili-
C ty mode).
C
C Warning -- UNDELETE has been used exactly once by the author in a
C real live situation (it worked).  It was written for this emergency
C and has been tested successfully on floppy disks and virtual disks,
C and has successfully recovered files at other installations.  It is
C recommended that you use any other source of a backup copy of a file
C before attempting to UNDELETE it, since it is always risky to mess
C with the volume structure.  However, when it has been actually
```

```
C applied in real disasters, UNDELETE has always worked, and you can
C have every confidence in its ability to correctly manipulate the
C index file structure.
C
C To compile:
C
C      >F77 Undelete,Undelete/-sp=Undelete
C
C To task build:
C
C      >TKB Undelete,Undelete/-sp=Undelete
C
C Undelete uses QIOs to manipulate the index file and normal Fortran
C Read and Write statements for terminal I/O, so your Fortran-77 OTS
C can be either the FCS or the RMS flavor.
C
C References:
C
C 1.  IAS/RSX-11 I/O Operations Reference Manual (Order No.
C     AA-M176A-TC), Appendix A, "File Descriptor Block", Appendix E,
C     "Index File Format", and Appendix F, "File Header Block Format".
C 2.  RSX-11M/M-PLUS Utilities Manual (Order No. AA-L681A-TC), Chapter
C     9, "File Structure Verification Utility (VFY)".
C
C Larry Baker
C U. S. Geological Survey
C 345 Middlefield Road  M/S 977
C Menlo Park, CA  94025
C (415) 323-8111 x2688
C
C Although this program has been tested by the Geological Survey,
C United States Department of the Interior, no warranty, expressed or
C implied, is made by the Geological Survey as to the accuracy and
C functioning of the program and related program material nor shall
C the fact of distribution constitute any such warranty, and no respon-
C sibility is assumed by the Geological Survey in connection therewith.
C
C Modification history:
C
C 21-Jan-80  L. M. Baker     Original version (RSX-11M V3.1)
C 22-May-86  L. M. Baker     Removed references routines in UserLib.
C 26-May-86  L. M. Baker     Use Fortran-77 character variables and
C                               Parameter statements for constants.
C 28-May-86  L. M. Baker     Add special case code to handle extension
C                               file headers.
C 15-Jun-86  L. M. Baker     Split restoration of primary and extension
C                               file headers into a separate subroutine.
C
      Implicit Integer (A-Z)
C
      Include  'IdxDef/List'     ! Define index file offsets
      Include  'HomDef/List'     ! Define home block offsets
      Include  'HdrDef/List'     ! Define file header block offsets
      Include  'UndCom/List'
C
```

```
      Integer*4   jhfmax, jhdrno, jhvbn
      Integer*2   hdrbuf(256), hombuf(256)
      Integer*2   ihdrno
      Logical     HDRFRE, MATCH
      Character   reply*72
C
C...  Don't need both at the same time
      Equivalence  (hombuf, mapbuf)
      Equivalence  (jhdrno, ihdrno)
C
C...  Prompt user for dry run option and disk name
C
 1000 Write (TTOUT,505)
  505 Format (/'$Do you want a dry run (no disk modifications',
     1         ' attempted) [Y/N,CR=Y]? ')
      Read (TTIN,502,End=9900) nchars, reply
  502 Format (Q, A)
      If (nchars .le. 0) Then
         reply(1:1) = 'Y'
      End If
      enable = ((reply(1:1) .eq. 'n') .or. (reply(1:1) .eq. 'N'))
      Write (TTOUT,501)
  501 Format (/'$Enter disk name (ddnn:) ')
      Read (TTIN,502,End=9900) nchars, reply
C...  Do a little syntax checking
      If (nchars .gt. 0) Then
         If (reply(nchars:nchars) .ne. ':') Then
            Write (TTOUT,503) reply(1:nchars)
  503       Format (/' ', A, ' must be of the form ddnn:, re-enter.')
            Goto 1000
         End If
      Else
         reply  = 'SY:'
         nchars = 3
      End If
C
C...  Make sure this is really a disk
C
 1100 reply(nchars+1:) = INDEXF
      If (enable) Then
C...     BUFFERCOUNT=-1 disables all record level access to a file --
C...     all I/O to the index file is done using QIOs to F11ACP directly
         Open (Unit=IDXLUN,File=reply,Status='Old',Buffer Count=-1,
     1         Err=1190)
         Goto 1200
 1190    Write (TTOUT,504) reply(1:nchars+LEN(INDEXF))
  504    Format (/' Unable to open ', A, ' for write access;',
     1            ' continuing as a dry run.')
         enable = .FALSE.
 1200    Continue
      End If
      If (.not. enable) Then
         Open (Unit=IDXLUN,File=reply,Status='Old',Buffer Count=-1,
     1         Read Only,Err=1290)
         Goto 1300
 1290    Write (TTOUT,507) reply(1:nchars+LEN(INDEXF))
  507    Format (/' Unable to open ', A, '.')
         Goto 1000
 1300    Continue
      End If
C
C...  Read home block
C
      jcount = 0
      If (GETBLK(IDXLUN,HOMVBN,hombuf,ierr) .ne. ISSUC) Then
         Write (TTOUT,506) ierr, ierr
  506    Format (/' Error reading home block:  ', O6, ' (', I6, '.).')
         Goto 9000
      End If
C
C...  Get offset to start of file headers and maximum number of files
C...  Note:  H.FMAX is a 16-bit unsigned integer
C
 1400 ihibsz = hombuf(HIBSZ)
      jhdrno = 0
      ihdrno = hombuf(HFMAX)
      jhfmax = jhdrno
      Write (TTOUT,509) ihibsz, ihibsz, jhfmax, jhfmax
  509 Format (/' Index file bit map size (H.IBSZ):  ', O6,
     1         ' (', I5, '.)'/
     2         ' Maximum number of files (H.FMAX):  ', O6,
     3         ' (', I5, '.)')
C
C...  Make data pattern to match for header search
C
      If (MAKPAT(ierr) .ne. ISSUC) Then
         Goto 9000
      End If
C
C...  There will be a maximum of H.FMAX headers starting at VBN
C...  MAPVBN+H.IBSZ
C
      jhvbn = ihibsz + MAPVBN
      Do 3000 jhdrno = 1,jhfmax
C
C...     Is this file header marked available?
C
         If (.not. HDRFRE(jhdrno)) Then
            Goto 3000
         End If
C
C...     Yes, see if it qualifies for restoration
C
         If (GETBLK(IDXLUN,jhvbn,hdrbuf,ierr) .ne. ISSUC) Then
C...         The index file is probably shorter than H.FMAX
            If (ierr .eq. IEEOF) Then
               ierr = ISSUC
               Goto 9000
            Else
               Write (TTOUT,510) jhdrno, jhdrno, ierr, ierr
  510          Format (/' Error reading file header number ', O6,
     1                 ' (', I5, '.):  ', O6, ' (', I6, '.).')
               Goto 3000
            End If
         End If
C
C...     If this header has never been used before, the creation date
C...     and modification date fields are zeros
C
         Do 2100 j = IRVDT,IRVDT+12
            If (hdrbuf(j) .ne. 0) Then
               Goto 2200
            End If
 2100       Continue
         Goto 3000
 2200    Continue
C
C...     See if it looks like it might be saved ('027027'O in
C...     word 1 and zeroed out file number and checksum)
C
         If ((hdrbuf(HIDOF) .ne. '027027'O) .or.
     1        (hdrbuf(HFNUM) .ne.       0  ) .or.
     2        (hdrbuf(HCKSM) .ne.       0  )) Then
            Goto 3000
         End If
C
C...     Extension file headers are dealt with separately in UNDHDR
C
         If (hdrbuf(MESQN) .ne. 0) Then
            Goto 3000
         End If
C
C...     Match search pattern?
C
         If (.not. MATCH(jhdrno,hdrbuf)) Then
            Goto 3000
         End If
C
C...     Perform recovery for this file header
C
         junk = UNDHDR (jhdrno,jhvbn,ihdrno,hdrbuf,ierr)
C
 3000    jhvbn = 1 + jhvbn
C
 9000 Close (Unit=IDXLUN)
C
C... Print message if any files recovered
C
      If (jcount .le. 0) Then
         Write (TTOUT,516)
  516    Format (/' No matching file headers found.'/)
      Else
         If (enable) Then
            Write (TTOUT,515) jcount, reply(1:nchars), reply(1:nchars)
  515       Format (/' Undelete recovered ', I5, '. files.'//
     1                ' You must run the VFY utility to complete the',
     2                ' recovery of the disk:'//
     3                '      VFY>TI:,LB:=', A, '/UP'/
     4                '      VFY>TI:,LB:=', A, '/LO'/)
         Else
            Write (TTOUT,517) jcount, 'will be'
  517       Format (/' Undelete found ', I5, '. files for recovery.'/)
         End If
      End If
C
 9900 Call EXIT (ierr)
C
      End
      Integer Function UNDHDR (jhdrno, jhvbn, ihdrno, hdrbuf, ierr)
C
C UNDHDR - Perform undelete operation for file.
C
      Implicit Integer (A-Z)
C
      Include  'IdxDef/NoList'     ! Define index file offsets
      Include  'HdrDef/NoList'     ! Define file header block offsets
      Include  'UndCom/NoList'
C
      Integer*4  jhdrno, jhvbn
      Integer*2  ihdrno, hdrbuf(256)
      Integer*4  jpreno, jextno, jxvbn
      Integer*2  iextno
      Logical    HDRFRE
C
      Equivalence  (jextno, iextno)
C
C...  Assume successful operation
C
      ierr = ISSUC
C
C...  Set file number in file ID (16-bit unsigned integer)
C
      hdrbuf(HFNUM) = ihdrno
C
C...  Reset marked-for-delete characteristic
C
      hdrbuf(HUCHA) = IAND (hdrbuf(HUCHA),NOT(SCMDL))
C
C...  Re-write header with checksum
C
      If (enable) Then
         If (PUTCBK(IDXLUN,jhvbn,hdrbuf,ierr) .ne. ISSUC) Then
C...        Error rewriting file header to index file
            Write (TTOUT,514) jhdrno, jhdrno, ierr, ierr
  514       Format (' Error rewriting file header number ', O6, ' (',
     1              I5, '.) to index file:  ', O6, ' (', I6, '.).')
            Goto 9000
         End If
      End If
C
```

```
C...  Set bit map in index file (HDRFRE told us where)
C
      mapbuf(wordno) = IOR (mapbuf(wordno),mask)
C
      If (enable) Then
         If (PUTBLK(IDXLUN,jmapno,mapbuf,ierr) .ne. ISSUC) Then
C...        Error rewriting index file bit map
            Write (TTOUT,513) jhdrno, jhdrno, ierr, ierr
 513        Format (' Error rewriting index file bit map for header',
     1                ' number ', O6, ' (', I5, '.):  ', O6, ' (', I6,
     2                '.).')
            Goto 9000
         End If
      End If
C
C...  Count this one in the total
C
      jcount = 1 + jcount
C
C...  Start of extension file header processing
C
      jpreno = jhdrno
C
C...  Look for the next extension file header
C
 2000 If ((hdrbuf(MEFNU) .ne. 0) .and.
     1    (hdrbuf(MEFSQ) .ne. 0)) Then
C
C...     Is this file header marked available?
C
         jextno = 0
         iextno = hdrbuf(MEFNU)
         jxvbn  = jextno + MAPVBN
         If (.not. HDRFRE(jextno)) Then
            Goto 4000
         End If
C
C...     Yes, see if it qualifies for restoration
C
         If (GETBLK(IDXLUN,jxvbn,hdrbuf,ierr) .ne. ISSUC) Then
            Goto 4000
         End If
C
C...     See if it looks like it might be saved ('027027'O
C...     in word 1 and zeroed out file number and checksum)
C
         If ((hdrbuf(HIDOF) .ne. '027027'O) .or.
     1       (hdrbuf(HFNUM) .ne.        0  ) .or.
     2       (hdrbuf(HCKSM) .ne.        0  )) Then
            Goto 4000
         End If
C
C...     Perform recovery for this file header
C
C...     Set file number in file ID (16-bit unsigned integer)
```

```
C
        hdrbuf(HFNUM) = iextno
C
C...    Reset marked-for-delete characteristic
C
        hdrbuf(HUCHA) = IAND (hdrbuf(HUCHA),NOT(SCMDL))
C
C...    Re-write header with checksum
C
        If (enable) Then
           If (PUTCBK(IDXLUN,jxvbn,hdrbuf,ierr) .ne. ISSUC) Then
C...          Error rewriting file header to index file
              Write (TTOUT,514) jextno, jextno, ierr, ierr
              Goto 4000
           End If
        End If
C
C...    Set bit map in index file (HDRFRE told us where)
C
        mapbuf(wordno) = IOR (mapbuf(wordno),mask)
        If (enable) Then
           If (PUTBLK(IDXLUN,jmapno,mapbuf,ierr) .ne. ISSUC) Then
C...          Error rewriting index file bit map
              Write (TTOUT,513) jextno, jextno, ierr, ierr
              Goto 4000
           End If
        End If
C
C...    Look for the next extension file header
C
        jpreno = jextno
        Goto 2000
C
C...    Extension header got clobberred or other error encountered
C...    processing extension header -- truncate extension header chain
C...    at last good header (makes rest of file unrecoverable, sorry)
C
C...    Restore previous file header
C
 4000   Write (TTOUT,508) jextno, jextno, jhdrno, jhdrno,
     1                    jpreno, jpreno
  508   Format (' Warning:  Extension file header number ', O6, ' (',
     1          I5, '.) for file header number ', O6, ' (', I5,
     2          '.) is corrupted;'/
     3          ' extension chain terminated at the previous file',
     4          ' header number ', O6, ' (', I5, '.).')
        jextno = jpreno
        jxvbn  = jextno + MAPVBN
        If (GETBLK(IDXLUN,jxvbn,hdrbuf,junk) .ne. ISSUC) Then
C...       The index file is probably shorter than H.FMAX
           If (junk .ne. IEEOF) Then
              Write (TTOUT,510) jextno, jextno, junk, junk
  510         Format (/' Error reading file header number ', O6,
     1                ' (', I5, '.):  ', O6, ' (', I6, '.).')
           End If
```

```
               Goto 9000
            End If
            hdrbuf(MEFNU) = 0
            hdrbuf(MEFSQ) = 0
            If (enable) Then
               If (PUTCBK(IDXLUN,jxvbn,hdrbuf,junk) .ne. ISSUC) Then
C...              Error rewriting file header to index file
                  Write (TTOUT,514) jextno, jextno, junk, junk
               End If
            End If
C
C...     End of extension file header processing
C
         End If
C
 9000 UNDHDR = ierr
C
         Return
         End
         Integer Function MAKPAT (ierr)
C
C MAKPAT - Make search pattern from user response.
C
         Implicit Integer (A-Z)
C
         Include 'UndCom/NoList'
C
         Integer    nlegal(24)
         Character  reply*40, char, legalc*38
C
C...     Number of legal characters, by index into patrn
         Data  nlegal/6*10,12*38,6*10/
C...     Legal characters (1-10 = octal;1-38 = RAD50)
         Data  legalc/'*?0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'/
C
         ierr = IEBAD
C
 1000 Write (TTOUT,507)
  507 Format (/'$Enter search pattern ([grp,mem]name.typ;version',
     1   ',CR=all) ')
         Read (TTIN,502,End=9000) nchars, reply
  502 Format (Q, A)
C...     Fill pattern with wild cards
         patrn = PATRN0
         If (nchars .gt. 0) Then
C...        Convert to upper case characters
            Call UPCASE (nchars,reply)
C...        Fill in 6 character owning UIC
            j = 1
            If (reply(j:j) .eq. '[') Then
               j = j + 1
C...           Find group number
               If (j .gt. nchars) Then
                  Goto 9900
               End If
            i = INDEX (reply(j:),',')
            If ((i .eq. 0) .or. (i .gt. 4)) Then
               Goto 9900
            End If
            If (i .gt. 1) Then
C...           Change preloaded ?'s to 0's if at least one non-wilcard
C...           is found
               Do 1100 k = j,j+i-2
                  If ((reply(k:k) .ne. '*') .and.
     1                (reply(k:k) .ne. '?')) Then
                     patrn(1:3) = '000'
                     Goto 1200
                  End If
 1100             Continue
C...           Right justify group number to O3.3 format
 1200          patrn(5-i:3) = reply(j:j+i-2)
            End If
            j = j + i
C...        Find member number
            If (j .gt. nchars) Then
               Goto 9900
            End If
            i = INDEX (reply(j:),']')
            If ((i .eq. 0) .or. (i .gt. 4)) Then
               Goto 9900
            End If
            If (i .gt. 1) Then
C...           Change preloaded ?'s to 0's if at least one non-wilcard
C...           is found
               Do 1300 k = j,j+i-2
                  If ((reply(k:k) .ne. '*') .and.
     1                (reply(k:k) .ne. '?')) Then
                     patrn(4:6) = '000'
                     Goto 1400
                  End If
 1300             Continue
C...           Right justify member number to O3.3 format
 1400          patrn(8-i:6) = reply(j:j+i-2)
            End If
            j = j + i
         End If
C...     Fill pattern with 9 character "name" part until a period or
C...     semicolon is found
         If (j .gt. nchars) Then
            Goto 2000
         End If
         i = INDEX (reply(j:),'.')
         If (i .eq. 0) Then
            i = INDEX (reply(j:),';')
         End If
         If (i .eq. 0) Then
            i = nchars - j + 2
         End If
         If (i .gt. 10) Then
            Goto 9900
         End If
         patrn(7:7+i-2) = reply(j:j+i-2)
         j = j + i
C...     Fill pattern with 3 character "type" part until a semicolon is
C...     found
         If (j .gt. nchars) Then
            Goto 2000
         End If
         i = INDEX (reply(j:),';')
         If (i .eq. 0) Then
            i = nchars - j + 2
         End If
         If (i .gt. 4) Then
            Goto 9900
         End If
         patrn(16:16+i-2) = reply(j:j+i-2)
         j = j + i
C...     Fill pattern with 6 character "version" part
         If (j .gt. nchars) Then
            Goto 2000
         End If
         i = nchars - j + 2
         If (i .gt. 7) Then
            Goto 9900
         End If
         If (i .gt. 1) Then
C...        Change preloaded ?'s to 0's if at least one non-wilcard is
C...        found
            Do 1500 k = j,j+i-2
               If ((reply(k:k) .ne. '*') .and.
     1             (reply(k:k) .ne. '?')) Then
                  patrn(19:24) = '000000'
                  Goto 1600
               End If
 1500          Continue
C...        Right justify version number to 06.6 format
 1600       patrn(26-i:24) = reply(j:j+i-2)
         End If
C
C...     Convert * to ? and validate characters (must be RAD50 or ?)
C
 2000    Do 2100 j = 1,24
            i = INDEX (legalc(1:nlegal(j)),patrn(j:j))
            If (i .eq. 1) Then
C...           Replace * with ?
               patrn(j:j) = '?'
            Else If (i .eq. 0) Then
C...           Illegal character in pattern
               Write (TTOUT,508) patrn(j:j), reply(1:nchars)
  508          Format (/' Illegal character, ', A, ', in ', A,
     1                  ', re-enter.')
               Goto 1000
            End If
 2100       Continue
      End If
```

```
C
      Write (TTOUT,511) patrn(1:3), patrn(4:6), patrn(7:15),
     1                  patrn(16:18), patrn(19:24)
  511 Format (/' Search pattern:  [', A, ',', A, ']', A, '.', A, ';',
     1        A/)
      ierr = ISSUC
C
 9000 MAKPAT = ierr
      Return
C
C...  Pattern not of the form [grp,mem]name.typ;version
 9900 Write (TTOUT,512) reply(1:nchars)
  512 Format (/' ', A, ' not of the form [grp,mem]name.typ;version,'
     1        ' re-enter.')
      Goto 1000
C
      End
      Logical Function MATCH (jhdrno, hdrbuf)
C
C MATCH - Matches file headers selected for recovery.
C
C         jhdrno - Index file header number.
C         hdrbuf - Current file header (256 words).
C
C Format of 24 character match pattern ("?"=wild card):
C
C              "gggmmmffffffffftttvvvvvv"
C
C                     g ( 1- 3) = Owner's group number
C                     m ( 4- 6) = Owner's member number
C                     f ( 7-15) = File name
C                     t (16-18) = File type
C                     v (19-24) = Version number
C
C All octal numbers not wild carded must have leading zeros.
C
      Implicit Integer (A-Z)
C
      Include  'HdrDef/NoList'     ! Define file header block offsets
      Include  'UndCom/NoList'
C
      Integer*4  jhdrno, jused
      Integer*2  hdrbuf(256), iused(2), iuic, group, member
      Character  name*24, digits*8, crdate*12
      Logical*1  luic(2), lgroup, lmembr
C
      Equivalence  (jused,   iused )
      Equivalence  (iuic,    luic  )
      Equivalence  (group,   lgroup)
      Equivalence  (member,  lmembr)
C
      Data  digits/'01234567'/
C
      MATCH = .FALSE.
C
```

```
C...  Convert UIC to group and member numbers
C
 1100 iuic   = hdrbuf(HPROG)
C
      group  = 0
      lgroup = luic(2)
      Do 1110 j = 3,1,-1
         i = MOD(group,8) + 1
         name(j:j) = digits(i:i)
 1110    group = group / 8
C
C...  Attempt pattern match to group number
C
      Do 1120 j = 1,3
         If (patrn(j:j) .ne. '?') Then
            If (patrn(j:j) .ne. name(j:j)) Then
               Goto 8100
            End If
         End If
 1120    Continue
C
      member = 0
      lmembr = luic(1)
      Do 1130 j = 6,4,-1
         i = MOD(member,8) + 1
         name(j:j) = digits(i:i)
 1130    member = member / 8
C
C...  Attempt pattern match to member number
C
      Do 1140 j = 4,6
         If (patrn(j:j) .ne. '?') Then
            If (patrn(j:j) .ne. name(j:j)) Then
               Goto 8100
            End If
         End If
 1140    Continue
C
C...  Convert file name from RAD50 to ASCII
C
      Call R50ASC (9,hdrbuf(IFNAM),name(7:15))
C
C...  Attempt pattern match to file name
C
      Do 1150 j = 7,15
         If (patrn(j:j) .ne. '?') Then
            If (patrn(j:j) .ne. name(j:j)) Then
               Goto 8100
            End If
         End If
 1150    Continue
C
C...  Convert file type from RAD50 to ASCII
C
      Call R50ASC (3,hdrbuf(IFTYP),name(16:18))
```

```
C
C...   Attempt pattern match to file type
C
       Do 1160 j = 16,18
          If (patrn(j:j) .ne. '?') Then
             If (patrn(j:j) .ne. name(j:j)) Then
                Goto 8100
             End If
          End If
 1160     Continue
C
C...   Convert version number to octal characters
C
       verno = hdrbuf(IFVER)
       Do 1170 j = 24,19,-1
          i = MOD(verno,8) + 1
          name(j:j) = digits(i:i)
 1170     verno = verno / 8
C
C...   Attempt pattern match to version number
C
       Do 1180 j = 19,24
          If (patrn(j:j) .ne. '?') Then
             If (patrn(j:j) .ne. name(j:j)) Then
                Goto 8100
             End If
          End If
 1180     Continue
C
C...   Successful pattern match, type PIP-style directory info for user
C
       iused(2)  = hdrbuf(FEFBK  )
       iused(1)  = hdrbuf(FEFBK+1)
       Write (crdate,101) (hdrbuf(j), j = ICRDT,ICRDT+5)
  101  Format (6A2)
       Write (TTOUT,501) jhdrno, hdrbuf(HFSEQ), name(1:3), name(4:6),
     1                  name(7:15), name(16:18), name(19:24), jused,
     2                  crdate(2:3), crdate(4:6), crdate(7:8),
     3                  crdate(9:10), crdate(11:12)
  501  Format (' (', O6, ',', O6, '): [', A, ',', A, ']', A, '.', A,
     1         ';', A, 2X, I8, '.', 2X, A, '-', A, '-', A, 1X, A, ':', A)
C
C...   NOTE:  You may want to insert a query capability here as an
C...   option.
C
       MATCH = .TRUE.
C
 8100  Return
       End
       Logical Function HDRFRE (jhdrno)
C
C HDRFRE - Returns availability status of header number jhdrno
C          (Integer*4) from the index file bit map.
C
C To optimize program performance, a write-through cache is maintained
```

```
C by retaining the validity of the VBN of the last bit map block read
C throughout the program.  No read is performed if the VBN needed is
C the same as the last one read.  (It's contents may have changed due
C to setting "in-use" flags, but the memory copy is always the same as
C the disk block, since a write is performed whenever an "in-use" flag
C is set.)
C
      Implicit Integer (A-Z)
C
      Include  'IdxDef/NoList'     ! Define index file offsets
      Include  'UndCom/NoList'
C
      Integer*4  jhdrno, jbitno, newblk
      Integer*2  masks(16), ibitno
C
      Data  masks/'0001'X,'0002'X,'0004'X,'0008'X,
     1            '0010'X,'0020'X,'0040'X,'0080'X,
     2            '0100'X,'0200'X,'0400'X,'0800'X,
     3            '1000'X,'2000'X,'4000'X,'8000'X/
C
C...  Header number 1 is bit 0 in the bit map
      jbitno = jhdrno - 1
C...  4096 bits to a disk block starting at MAPVBN
      newblk = MAPVBN + (jbitno/4096)
C...  If memory cache is not the correct bit map block, read new block
      If (newblk .ne. jmapno) Then
         If (GETBLK(IDXLUN,newblk,mapbuf,ierr) .ne. ISSUC) Then
            Write (TTOUT,501) jhdrno, jhdrno, ierr, ierr
  501       Format (' Error reading index file bit map for header',
     1              ' number ', O6, ' (', I5, '.):  ', O6, ' (', I6,
     2              '.);'/
     3              ' recovery not attempted.')
C...        If an error occurs, invalidate bit map block number
            jmapno = -1
         Else
            jmapno = newblk
         End If
      End If
      If (newblk .eq. jmapno) Then
C...     Calculate word in mapbuf and bit in word
         ibitno = MOD (jbitno,4096)
         wordno = (ibitno/16) + 1
         ibitno = (MOD(ibitno,16)) + 1
         mask   = masks(ibitno)
         HDRFRE = IAND(mapbuf(wordno),mask) .eq. 0
      Else
         HDRFRE = .FALSE.
      End If
C
      Return
      End
      Integer Function GETBLK (lun, iblk, ibuf, ierr)
C
C GETBLK - Reads a virtual block from the file open on lun.
C PUTBLK - Writes a virtual block to the file open on lun.
```

```
C PUTCBK - Computes the checksum of words 1 through 255 into word 256,
C          then writes a virtual block to the file open on lun.
C
C          lun    - Fortran Logical Unit Number
C          iblk   - Integer*4 Virtual Block Number (from 1)
C          ibuf   - Block buffer (256 words)
C          ierr   - RSX I/O status (also returned as the function value)
C
      Implicit Integer (A-Z)
C
      Integer    IORVB, IOWVB, ISSUC
      Parameter  (IORVB = '010400'O)
      Parameter  (IOWVB = '011000'O)
      Parameter  (ISSUC =       1  )
C
      Integer*2  ibuf(256), iblk(2), ioparm(6), cksum
      Logical*1  liosb(4)
C
      iocode = IORVB
      Goto 2000
C
      Entry PUTCBK (lun, iblk, ibuf, ierr)
C
C...  Calculate checksum, then fall through into PUTBLK
      cksum = 0
      Do 1000 j = 1,255
 1000    cksum = cksum + ibuf(j)
      ibuf(256) = cksum
C
      Entry PUTBLK (lun, iblk, ibuf, ierr)
C
      iocode = IOWVB
C
 2000 Call GETADR (ioparm(1),ibuf)
      ioparm(2) = 512
      ioparm(4) = iblk(2)
      ioparm(5) = iblk(1)
      Call WTQIO (iocode,lun,lun,,liosb,ioparm,ierr)
      If (ierr .eq. ISSUC) ierr = liosb(1)
C
 9000 PUTBLK = ierr
C
      Return
      End
      Subroutine  UPCASE (n, s)
C
C...  Convert string s to upper case characters
C
      Integer    n, ic
      Character  s*255, ALPHA*26
      Data       ALPHA/'ABCDEFGHIJKLMNOPQRSTUVWXYZ'/
C
      Do 1000 j = 1,n
C...     Convert the next character to its integer equivalent
         ic = ICHAR(s(j:j))
```

```
C...      Compare it to the integer equivalents for lower case a and z
          If ((ic .ge. ICHAR('a')) .and. (ic .le. ICHAR('z'))) Then
C...         If it's lower case, replace it with the corresponding single
C...         character substring from the upper case alphabet in ALPHA
             i = ic - ICHAR('a') + 1
             s(j:j) = ALPHA(i:i)
          End If
 1000     Continue
C
      Return
      End
```

APPENDIX B

Include Files

B.1 Index File Offset Definitions (IdxDef.ftn)

```
C
C IdxDef.ftn - Index file offset definitions
C
      Character   INDEXF*17
      Parameter   (INDEXF = '[0,0]INDEXF.SYS;1')      ! Index file name
      Integer*4   HOMVBN, MAPVBN
      Parameter   (HOMVBN =    2)       ! Home block
      Parameter   (MAPVBN =    3)       ! Index file bit map
C
```

B.2 Home Block Offset Definitions (HomDef.ftn)

```
C
C HomDef.ftn - Home block offset definitions
C
      Integer     HIBSZ,  HFMAX
      Parameter   (HIBSZ  =    1)       ! Index bitmap size
      Parameter   (HFMAX  =    4)       ! Maximum files allowed
C
```

B.3 File Header Block Offset Definitions (HdrDef.ftn)

```
C
C HdrDef.ftn - File header block offset definitions
C
      Integer     HIDOF,  HFNUM,  HFSEQ,  HPROG,  HUCHA,  HCKSM
      Integer     FHIBK,  FEFBK
      Integer     IFNAM,  IFTYP,  IFVER,  IRVDT,  ICRDT
      Integer     MESQN,  MEFNU,  MEFSQ
      Integer     SCMDL
      Parameter   (HIDOF  =    1)       ! Identification area offset/
                                        ! Map area offset
      Parameter   (HFNUM  =    2)       ! File number
      Parameter   (HFSEQ  =    3)       ! File sequence number
      Parameter   (HPROG  =    5)       ! Member number/
```

```
                                                ! Group number
      Parameter    (HUCHA   =   7)        ! User-controlled characteristics/
                                                ! System-controlled characteristics
      Parameter    (SCMDL   = '100000'O) ! File is "marked-for-delete"
      Parameter    (FHIBK   =  10)        ! Highest VBN allocated
      Parameter    (FEFBK   =  12)        ! End-of-file block number
      Parameter    (IFNAM   =  24)        ! File name
      Parameter    (IFTYP   =  27)        ! File type
      Parameter    (IFVER   =  28)        ! File version number
      Parameter    (IRVDT   =  30)        ! Revision date
      Parameter    (ICRDT   =  36)        ! /Creation date
      Parameter    (MESQN   =  46)        ! Extension segment number/
                                                ! Extension relative volume number
      Parameter    (MEFNU   =  47)        ! Extension file number
      Parameter    (MEFSQ   =  48)        ! Extension file sequence number
      Parameter    (HCKSM   = 256)        ! Checksum of words 1 through 255
C
```

B.4 Miscellaneous Definitions and Fortran Commons (UndCom.ftn)

```
C
C UndCom.ftn - Undelete miscellaneous definitions and common areas
C
      Integer      ISSUC, IEBAD, IEEOF
      Parameter    (ISSUC   =   1)
      Parameter    (IEBAD   =  -1)
      Parameter    (IEEOF   = -10)
C
      Integer      IDXLUN, TTIN,   TTOUT
      Parameter    (IDXLUN  =   1)
      Parameter    (TTIN    =   5)
      Parameter    (TTOUT   =   5)
C
      Character    PATRN0*24
      Parameter    (PATRN0 = '????????????????????????')
C
      Integer*4    jcount, jmapno
      Integer*2    mapbuf(256), wordno, mask
      Logical      enable
      Character    patrn*24
C
      Common /UNDCOM/  enable, jcount, jmapno, mapbuf, wordno, mask
      Common /UNDCON/  patrn
C
C...  Initialize the index file bitmap block context for HDRFRE
      Data  jmapno/-1/
C
```

APPENDIX C

Command File

C.1 Command File (Undelete.cmd)

```
.Enable Substitution
;
; Undelete.cmd -- Compile and task build file un-deleter
;
.SetS mcr ".;"
.SetS dcl ".;"
.If <CLI> eq "MCR" .SetS mcr ""
.If <CLI> eq "DCL" .SetS dcl ""
.;
.;'mcr'Pip Undelete.obj;*/de/nm,.lst;*
.;'dcl'Delete Undelete.obj;*,.lst;*
'mcr'F77 Undelete,Undelete/-sp=Undelete'P1'
'dcl'Fortran/F77'P1' Undelete /List
;
.;'mcr'Pip Undelete.tsk;*/de/nm,.map;*
.;'dcl'Delete Undelete.tsk;*,.map;*
'mcr'.SetS tkb "Tkb"
'dcl'.SetS tkb "Link"
'mcr'.IfIns ...Ftb .SetS tkb "Ftb"
'dcl'.IfIns ...Ftb .SetS tkb "Link/Fast"
'mcr'.SetS fp     "/fp"
'dcl'.SetS fp     "/Code:FPP"
.SetS f4peis ""
.TestFile LB:[1,1]F4PEIS.obj
'mcr'.If <FILERR> eq 1 .SetS fp     "/-fp"
'dcl'.If <FILERR> eq 1 .SetS fp     "/Code:NoFPP"
.If <FILERR> eq 1 .SetS f4peis ",LB:[1,1]F4PEIS"
'mcr''tkb' Undelete'fp',Undelete/-sp=Undelete'f4peis'
'dcl''tkb''fp' Undelete'f4peis' /Map:Undelete
.;
.; Documentation needs DSR with right offset 5 characters for printing
.; on an Imagen laser printer in portrait orientation.
.;
.;'mcr'.IfIns ...DSR Pip Undelete.doc;*/de/nm,UndPLM.doc;*
.;'dcl'.IfIns ...DSR Delete Undelete.doc;*,UndPLM.doc;*
.;'mcr'.IfIns ...DSR DSR Undelete=Undelete/right:5
.;'dcl'.IfIns ...DSR MCR DSR Undelete=Undelete/right:5
```

```
.;'mcr'.IfIns ...DSR DSR UndPLM  =UndPLM  /right:5
.;'dcl'.IfIns ...DSR MCR DSR UndPLM  =UndPLM  /right:5
;
```