UNIX* RTR OPERATING SYSTEM FIELD UPDATE FACILITIES

SOFTWARE SUBSYSTEM DESCRIPTION

AT&T 3B20D COMPUTER

CONTENTS PAGE

*Trademark of AT&T Bell Laboratories

1. GENERAL

1.01 This section describes field update concepts and defines procedures for implementing updates. These updates are used to maintain the AT&T 3B20D computer which runs the UNIX RTR operating system, release 1.

1.02 Whenever this section is reissued, the reason(s) for reissue will be listed in this paragraph.

1.03 Field update is the software subsystem that installs changes (updates) to files and their main memory images on the 3B20D computer. The update information, applied by field update, is contained in an update package called a broadcast warning message (BWM). A BWM contains all the information needed to fix a system problem or provide a program enhancement.

2. OVERVIEW

2.01 Field update provides the software needed to make changes to UNIX RTR operating systems at a field site with only minimal service interruption. The two kinds of UNIX RTR files where changes may be applied are: data files and executable files.

2.02 Data files are normally updated by field update using the technique of file replacement where a new file is substituted for an old one.

2.03 Executable files (pfiles, a.out files, or shared libraries) and the processes created from them may be updated by file replacement or by function replacement where only parts of the process are changed. Three types of executable processes may be updated: killable, quasikillable, and nonkillable processes.

2.04 Field updates are delivered to the field either on tape or directly to a special update directory on the 3B20D computer disk via the software change administration and notification system (SCANS). There are three types of field updates:

(1) **File Replacement** installs a new file on secondary storage (disk), possibly replacing an old file

(2) **Function Replacement** changes global data of, adds a new function to, or replaces an old function in an executable process

(3) **Byte Replacement** loads specified data values into locations in an executable process.

Field update does **not** update data bases, special device files, or hardware components.

2.05 A BWM consists of a set of files which are placed in a directory named for the BWM number. These files are:

(a) **Header File:** The header file contains administrative information about the other files of the BWM and includes their path names, size in bytes, and check sums.

(b) **Messages File:** The messages file contains input messages to be used to apply the Field update files. The information is displayed in a script format. If the BWM is for information only, the messages file is not present.

(c) **SCANS File:** The SCANS file, an administrative file, contains information on the number of blocks on the disk required for each update file and for the package as a whole. It also contains:

(1) Number of blocks on the 3B20D disk required for each update file and for the package as a whole

(2) A description of the problems addressed by the BWM and the effect of the problem on the system

(3) A description of the solution to the problem and each piece of software affected by the solution

(4) A description of the effects of the updates in the package.

(d) **Zero or More Update Files:** These files contain the actual update information. For BWMs that are 'information only', the update files do not exist.

2.06 The field update process is comprised of the following four procedures:

(a) **Applying Change:** Each update is initially installed as temporary.

(b) **Soaking:** During this period, the update is tested.

(c) **Backing Out Change:** After the test period, a decision is made to either 'back out' the update or to make it permanent. The update must be backed out if it does not work or if it adversely affects the system.

(d) **Making Change Permanent:** The update is made permanent when field update overwrites a temporary update disk image onto the official disk image.

2.07 The field update system automatically maintains a database to record all changes applied by field update input messages.

3. FIELD UPDATE

A. Field Update Features

3.01 Field update provides the capability to apply BWMs or emergency fixes to a field site. An update may be applied to either an incore process or to the disk image from which subsequent core images will be created.

3.02 The means of implementing an update is dependent upon the following conditions.

- If the change is to apply to a core image or the disk image of a process

- If the process to be changed is killable, quasikillable, or nonkillable.

Units of Update

3.03 Field update normally works at a high level. The update replaces a whole file or a set of functions in a process as a single unit of change instead of forcing the user to specify absolute core addresses and contents. However, specification of absolute core addresses and contents also can be done, if necessary.

Types of Update

3.04 There are three types of updates: file replacement, function replacement, and byte replacement.

File Replacement

3.05 File replacements affect only disk images. Any file replacement must be known to the system. That is, the file must have an entry in the system generation data base; and if the file is always contiguous, it must have a preallocated shadow file. (A shadow file is an always-contiguous file that exists to insure that sufficient contiguous space is available for an update.) For field update to install a new file, the new file must first have an entry in the system generation data base. File replacement also includes replacing an entire disk partition.

Function and Byte Replacement

3.06 Function and byte replacements affect **updateable object files**, those files that can have executable main memory images. This file type represents processes, shared libraries, or the kernel itself, and may be either killable, quasikillable, or nonkillable.

3.07 A killable process or shared library may die and leave main memory without disrupting service. This killable process, which terminates by itself, is restarted by an input message or by another process. The next time the process is started after an update has been applied to the disk image, the created core image will contain the update.

3.08 A quasikillable process, which never terminates, is a monitored killable process under control of the user-level automatic restart process. Whenever one of these processes dies, the process is restarted automatically by the user-level automatic restart process. To update one of these processes, the change is applied to the disk image. Then the process must be killed [by using the program documentation standard (PDS) input message or its equivalent]. A new core image is then automatically created from the disk image thus implementing the update.

3.09 A nonkillable process causes service interruption when terminated. Field update can automatically detect if a process is nonkillable from information on the disk. In general, nonkillable processes are members of the boot file and are brought up once at boot time.

3.10 Function and byte replacements to nonkillable processes affect main memory and disk images. Function and byte replacements to killable processes affect only the disk images. File replacements affect only disk images.

3.11 Any effect of an update is seen only when main memory is changed. Thus, a function or byte replacement to a nonkillable process becomes effective once the update has been applied. A file replacement to an updateable object file becomes effective only when the new file has an image in main memory. Thus, making the file replacement changes evident requires a boot for nonkillable processes, killing the current invocation of a quasikillable process, and reexecuting a killable process.

Updating Boot Images

3.12 For an updateable object file that is in the boot image, it is not sufficient that an update change only the disk image of the file. That updated file must be incorporated into the boot image. Field update automatically recreates the application boot image. However, the boot image that contains only the operating system must be rebuilt by explicit command. This procedure guards against a faulty update automatically corrupting both boot images.

Duration of Update

3.13 Before any update can be made permanent (official), it must be installed with temporary status. This allows a change to be backed out after a trial period, if required.

3.14 All updates to main memory have an interval called the **wait time** during which any phase 1 fault will remove the update. The default is 5 minutes, but the time interval can be changed by a data

field in the update input message. Once the wait time for an update has expired (soaked) and the changes are effective and not harmful, then the update may be made a permanent part of the system.

3.15 In the case of a nonkillable process, a new update is applied to both the core image and to a temporary copy of the disk image. The effect of the update can be almost immediate.

> ***Note:*** A system boot will backout temporary updates since the core image will be recreated from the unchanged official disk image.

3.16 A killable process is updated by creating a temporary disk image. The official disk image is then temporarily overridden. A change to a killable process does not take effect until the process is started from this new disk image.

3.17 Once an update has been soaked and the changes made considered effective and safe, then the update may be made a **permanent** part of the system. When making a file replacement permanent, it must be noted that changes made may not be evident.

3.18 When a function replacement to a nonkillable process is made permanent, the disk image of that process is also permanently changed. If that process is part of the boot image, then the application boot image is also rebuilt.

3.19 Updates to both killable and nonkillable processes can be made permanent by using the **UPD:PERM:UPNM** input message. When this happens, the official disk image is, in effect, overwritten with the temporary disk image created by field update. Field update does not allow a permanent update to be backed out. Figure 1 diagrams the states through which an update moves.

Update Data Formats

3.20 Update information arrives at the field site as part of a BWM. There are three possible formats. One is a full disk image of the executable file or the data file to be updated that replaces the old disk image. Second is a .m file, a subset of the process to be updated in the form of a special partial disk image. Only executable files may be updated using .m files. The third format is BWM text describing address-data couplet overwrites which are applied using the **UPD:APPLY:BYTER** input message. This message allows temporary address-data couplet over-

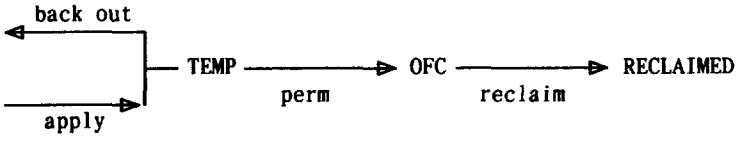

Fig. 1—Field Update States

writes. These overwrites change the words at the specified virtual address within the process.

Update Accounting

3.21 Field update maintains a database to record all changes applied by field update input messages. Field update also maintains a database containing information about new disk images, .m file updates, and byte replacements. This is useful when the BWM status of a data file or process is in question.

3.22 Field update recognizes three prefixes: BWM, for general updates; CFT, for craft-supplied updates; and DMT, for DMERT-only updates. The **UPD:DISPLAY** input message uses the database to list status information.

3.23 In addition, every update has a sequence number in the database. This number controls the order of conversion of status from temporary to permanent and backouts of temporary changes.

3.24 The database maintains information throughout a single software release. When a new release (either point or generic) is sent to the field, it is in the form of a disk image and requires none of the earlier updates.

B. Broadcast Warning Message (BWM) Format

3.25 The update information that is applied by field update is contained in an update package called a BWM. More than one BWM may be included in a field update package. The BWM is identified by an identification number of the following form: XXXYY-NNNN where XXX is BWM for general updates, CFT for craft-updates, and DMT for DMERT-only updates; YY is last two digits of the current year; and NNNN is the next sequential change number for this system within the year. Thus BWM84-0015 is the fifteenth general update package for 1984.

3.26 The BWM format (Fig. 2) follows that of a UNIX operating system directory structure with the following files:

- Header file
- Messages file
- SCANS information file
- Zero or more update files.

The header file and SCANS information file must be present in each BWM. If the BWM is of an information-only nature, no update file or messages file is present.

Header File

3.27 The header file (Fig. 3) contains the necessary information for verifying the BWM before starting the update process. The header file is a character file containing a number of data field identifiers separated by field identifiers. Each field identifier consists of a unique 3-character string followed by the colon character(:). The data field follows the appropriate field identifier and ends in the newline character \n.

3.28 Field identifiers are uppercase characters. Data fields can contain characters from the restricted set [A-Z, 0-9, (,), -] except for the data field specifying the name of a file where normal UNIX operating system file-naming conventions apply. Blank characters are not allowed as fill or embedded characters anywhere within the header file. The appearance of the data fields (ie, field header field identifier:contents) within the header file does not imply any fixed order. Each data field can be uniquely identified by field identifier, so a fixed order is not mandatory. A description of the field identifiers and the contents of the associated data fields are in Table A.

Messages File

3.29 The messages file (Fig. 4) is a character file containing all input messages for applying the BWM. The PDS format input messages are initially displayed at the Switching Control Center System work station or on the target system to inform maintenance personnel of the required input.

BWM SCANS Information File

3.30 The SCANS information file (Fig. 5) contains all the necessary information associated with queries and distribution of the BWM. The current list of information from SCANS that is available to maintenance personnel on a per-BWM basis includes:

- BWM identification

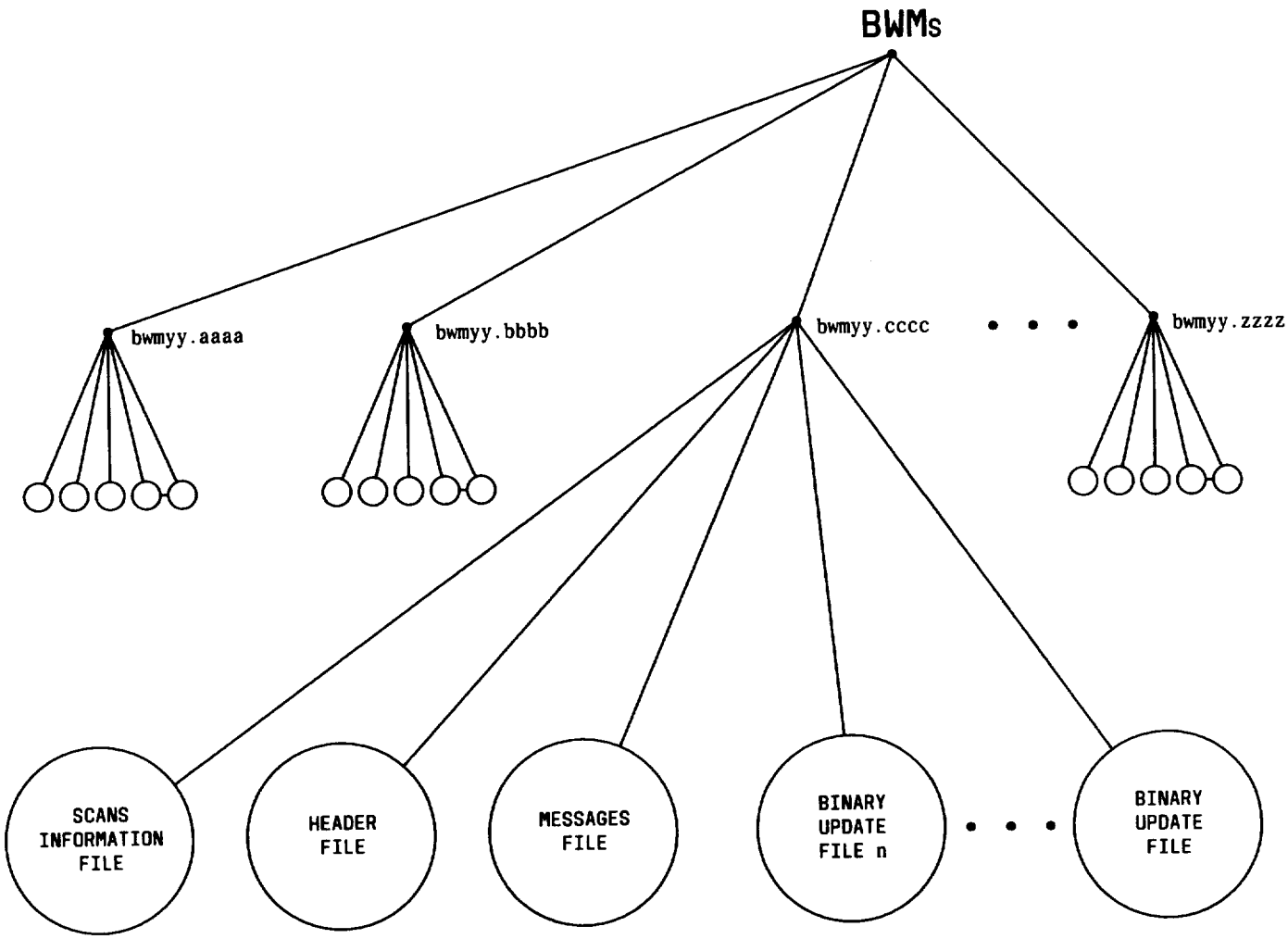

Fig. 2—BWM Structure

- Change notice number
- Identification of target system
- Affected generics
- Affected features
- Description of change.

Zero or More Update Files

3.31 An update file, which contains the actual changed data, is usually referred to by one or more input messages in the messages file.

C. BWM States

3.32 The states a BWM moves through during a typical update session are verified, temporary, permanent, cleared, and purged (Fig. 6). States that

COLUMN 1	EXPLANATION
BWM:81-0001	BWM Number
NOG:1	Number of Generics
GEN:3B20D<3>,9.2,12	Dummy DMERT Generic
NOF:3	Number of Update Files
UFN:F0001.m,26,512	File Name, Size, and Sum
UFN:F0002.m,31,819	(Some names changed to
UFN:F0003.m,74,426	avoid naming conflicts)
MSG:MSGS,1198,893	Messages File
SC2:SCANS,1101,343	SCANS Information File

Fig. 3—Sample BWM Header File

may also be encountered during a field update are omitted, reclaimed, and pending.

States A BWM Moves Through Normally

3.33 The following list defines the sequence of the states in which a BWM moves through during a typical update session.

(1) **Verified** — The verify operation confirms the appropriate files exist in the BWM directory and that they are in the proper format. Ordinarily verification of the BWM is done before any other operation. This state applies only to the BWM package as a whole.

(2) **Temporary** — Changes to a running system are suspect until they are proven not to affect service adversely. Therefore, updates are applied in a temporary state (except as noted) so they may be removed if judged unsatisfactory. The one exception to an update being applied in a temporary state is an update to a file partition, for which there is no mechanism for a temporary state. An update is removed (backed out) by explicit command. A BWM is temporary if it contains only temporary or temporary and omitted updates. Such a BWM may be backed out, made permanent, or have more updates added to it. The temporary state can apply either to a BWM as a whole or to individual updates.

(3) **Permanent** — Once the wait time for an update has expired and the changes are effective and not harmful, then the update may be made a permanent part of the system. Making a function replacement to a nonkillable process permanent also permanently changes the disk image of that process. If this process is part of the boot image, then the application boot image is automatically rebuilt. A BWM is permanent if it contains only permanent or permanent and omitted updates. Such a BWM is inactive. The permanent state can apply either to a BWM as a whole or to individual updates.

(4) **Cleared** — The cleared state means that the BWM files have been removed from the BWM directory. A BWM may be cleared only if it is permanent and all of its component function replacement updates have been reclaimed. The cleared state applies only to the BWM package as a whole.

(5) **Purged** — The purged state means that all history of the BWM has been removed from the field update data base. A BWM may be purged only after it has been made permanent and its updates have been reclaimed. The purged state applies only to the BWM package as a whole.

Other BWM States

3.34 The following list defines other states that may be encountered during field update. All of these states can apply to a BWM as a whole or to individual updates.

TABLE A

FIELD IDENTIFIERS AND CONTENTS OF ASSOCIATED DATA FIELDS

FIELD IDENTIFIERS	DESCRIPTION OF DATA FIELD CONTENTS
BWM	***Broadcast Warning Message Identification:*** Seven characters specifying the BWM identifier in the syntax of YY-NNNN where YY is the year and NNNN is the next sequential BWM number issued against the target system in the year; padded with zeros on the left. This also acts as the directory name for the BWM.
NOG	***Number of Generics Affected:*** One or two numeric characters indicating the number of subsequent generic fields to follow.
GEN*	***Generic Identification:*** Four subfields separated by commas in the following format: GENERIC-NAME,ISSUE-NAME,SEQ-NO,FEATURE-PKG The GENERIC-NAME subfield can be from one to ten alphanumeric characters from the restricted set [A-Z,0-9,(,),-]. The ISSUE-NAME subfield consists of two portions separated by a period in the form: NNM. MMM. The NNN portion is called the issue and the MMM portion is called the point issue. Both portions must appear in the issue name and can be from one to three numeric characters. If all issues of a particular generic are affected by the BWM (only for informational bulletins), the characters "ALL" may be substituted for the issue portion of the issue name. When ALL is specified, the point issue number is disregarded. If all point issues of a particular generic issue are affected by the BWM, the characters 999 may be substituted for the point issue portion of the issue name. The SEQ-NO subfield is from one to four numeric characters that represent the sequence number of the update as assigned by the system test group and applied to the generic represented by the GENERIC-NAME and full issue number (NNN) as specified in ISSUE-NAME. A sequence number of 9999 overrides the sequence checking performed on the 3B20D. The FEATURE-PKG subfield of the generic identification is optional and represents a variable length alphanumeric name of a particular feature package associated with the named generic. If no feature package field is included, the preceding comma (,) need not be present. Examples of feature package names are BLVFY, RZMN, and RSM.

* There must be as many GEN fields as the value of NOG.

TABLE A (Contd)

FIELD IDENTIFIERS AND CONTENTS OF ASSOCIATED DATA FIELDS

FIELD IDENTIFIERS	DESCRIPTION OF DATA FIELD CONTENTS
NOF	***Number of Update Files in BWM:*** One to three numeric characters indicating the number of update files to follow. This number must agree with the total number of system update files in the BWM.
UFN†	***Update File:*** Three sections separated by commas in the following format: FILENAME,SIZE,CHECKSUM FILENAME is the 1 to 14 alphanumeric character name of the system update file in the BWM. This name must conform to the UNIX operating system file-naming convention. SIZE is the (decimal) number of bytes (ten numeric characters maximum) specifying the size of the update file. CHECKSUM is a (decimal) number of from one to ten numeric characters specifying the result of the checksum algorithm applied to this update file.
MSG	***Message File:*** Has the same format as the UFN field with the exception that the FILENAME is replaced by **MESSAGE.**
SC2	**SCANS File:** Has the same format as the UFN field with the restriction that the FILENAME must be SCANS. This field may exist in the header file even though the SCANS information file SCANS is not present in the BWM structure.

† There must be as many UFN fields as the value of NOF.

- **Omitted** — A BWM is omitted if all its updates are omitted. Such a BWM is considered inactive, and updates may not be applied against it. The omitted state is not used typically unless a trouble condition occurs.

- **Reclaimed** — For function replacement, field update uses portions of the updated file or main memory image to effect the update. Once these portions and the replaced functions are no longer needed, they may be reclaimed for later use. Only those files that have permanent function replacements may be reclaimed.

- **Pending** — If an update operation fails, that update is in a pending state. A pending update must be resolved before any other operations can take place.

A system initialization (boot) is the equivalent of a back out in that temporary and pending temporary updates are removed from the system and the field update data base. After a boot, pending permanent updates remain so marked in the data base. The effects of the update are therefore determined by how far the update got before it was interrupted by the boot.

Dealing With Pending Updates

3.35 When an update operation fails, that update is marked pending in the field update data base. Except in the case of **pending reclaim**, the associ-

```
"APPLY.*****************************************************************
"
"       MR:d84000001
"
"       this BWM updates /fldupd/sndnt.p with two minimal files.
"            ntllc.m and ntl3c.m, and updates /fldupd/sknnt.p
"            with the minimal file nt2la.m
"
UPD: VFY:DATA , BWM =  CFT84-0001";
"
UPD: APPLY:DATA , FUNCR, UPNM = "CFT84-0001", FN = "/fldupd/sndnt.p"?
UF = "/etc/bwm/CFT84-0001/ntllc.m", WTIME=5;
UPD: APPLY:DATA , FUNCR, UPNM = "CFT84-0001", FN = "/fldupd/sndnt.p"?
UF = "/etc/bwm/CFT84-0001/ntl3c.m", WTIME=4;
"
UPD: APPLY:DATA , FUNCR, UPNM = "CFT84-0001", FN = "/fldupd/sknnt.p"?
UF = "/etc/bwm/CFT84-0001/nt2la.m";
"
"SOAK.*****************************************************************
"
UPD:DISPLAY:DATA, VLEV=0&2;
"
UPD:FTRC:DATA, FN="/fldupd/sndnt.p",FCN="control";
"
"BKOUT.****************************************************************
"       If the fix results in the need to reboot their system, the fix will
"       have been backed out automatically. If the fix does not result
"       in a reboot but otherwise does not work correctly, it can be backed
"       out by entering the command[s]:
"
UPD: BKOUT:DATA , UPNM = "CFT84-0001";
"
"PERM.*****************************************************************
"
UPD: PERM:DATA , UPNM = "CFT84-0001";
"
UPD: RECLAIM:DATA , FN = "/fldupd/sndnt.p";
"
UPD:DISPLAY:DATA, FN="/fldupd/sndnt.p", VLEV=0&2;
```

Fig. 4—Sample BWM Messages File

ated BWM is also marked pending. A pending update in a pending BWM must be resolved before any other update operations can take place. The following options, in the order they should be attempted, are available for dealing with pending updates.

(1) **Continue** the operation: Reinvoking the update command with the single data keyword **CONT** will pick up the operation from the point of failure.

(2) **Back out** the update: The command **UPD:BKOUT:CONT** will back out a pending temporary update.

(3) **Reset** the operation: The command **UPD:RESET** will take the operation back to its starting state where the command can be reinvoked. Resetting performs no update operations and may leave temporary files created as part of the failed update attempt.

(4) **Omit** the update: The command **UPD:OMIT** will mark the update as omitted, removing the pending state from the data base but not undoing any effects the interrupted update may have made. An omit changes the definition of the BWM by removing an update from the BWM.

```
BWM NUMBER.
 81-0001
SYSTEM IDENTIFICATION.
 3B20D
BWM TYPE.
 BIN
PROFILE MATCH.
 3B20D<1>,9.2,12
SUBJECT.
 3B20D DMERT Field Update
PERSON TO CONTACT.
 WESTERN DIAGNOSTIC CENTER.
KEYWORDS.
 NONE
SPECIAL INSTRUCTIONS.
 NONE
GENERAL INFORMATION.
 Introduction:
        A high level description of the problem that this
        BWM addresses goes here.

 Problem Description:
        Each problem is described here as to how it affects
        the system. A list of associated MRs also goes here.

 Fix Description:
        The solution to the problem is described here with
        reference to each 3B20 product and source file affected.

 Test Procedures:
        This section is used by system test to describe how
        the BWM is tested. It is primarily for the information
        of applcations' developers and system testers.
MNEMONIC CODE.
 NONE
HASH SUMS.
 NONE
SOFTWARE CHANGE SIZE.
 TOTALSIZE 2583 10
 810001/HDR 153 1
 810001/SCANS 1101 3
 810001/MSGS 1198 3
 810001/F0001.m 26 1
 810001/F0002.m 31 1
 810001/F0003.m 74 1
OVERWRITE DATA.
 BINARY
END OVERWRITE.
END BWM.
```

Fig. 5—Sample BWM SCANS Information File

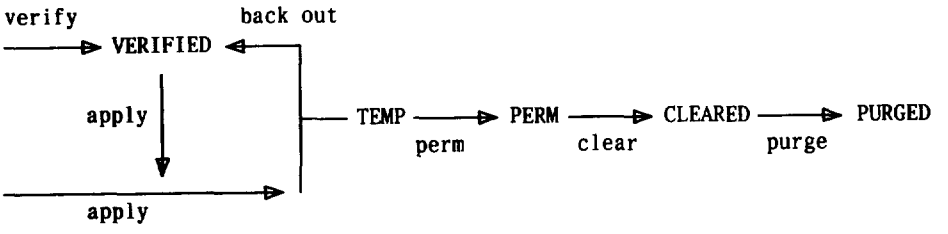

Fig. 6—BWM States

D. Emergency Fixes

3.36 An emergency fix is implemented when an unexpected problem with the release program occurs and there is not enough time for the normal BWM development and issue processes.

3.37 Emergency fixes are assigned a sequential craft number similar to the BWM number; but instead of BWMYY-NNNN, the update number is CFTYY-NNNN. The program update subsystem provides emergency fixes with the same status and options as BWMs (ie, make temporary, make permanent, or back out).

E. Applying Update

Receiving a Field Update

3.38 The domain of field update begins at the point where a field update input message is received (whether entered from a terminal or generated by software). At this point, the update data is already loaded onto a disk and thus has a path name to which the input message can refer.

Applying Changes

3.39 Every change that is applied via **UPD:APPLY** to a process is first applied as a temporary change to either the core or disk image. Only after a change is in temporary status can it be made permanent (via **UPD:PERM** input message). If the temporary change is made to core and a phase 1 initialization occurs within the specified wait time after the core is changed, the update is backed out automatically.

3.40 The BWM text specifies the order of application of changes and specifies **whether, which,** and **when** any processes are to be killed and restarted in order to cause the changes to take effect.

Soaking

3.41 Soaking refers to the period of time after a temporary change is installed prior to its being made permanent (official). During this interval, its effects may be observed, and if necessary, the change may be backed out.

3.42 Since an update to a nonkillable process changes the active core image, any observable effects of the change could be immediately apparent. On the other hand, an update to a killable process

does not alter the core image, and so no change in the process behavior can possibly occur until a new instance of the process is created from the updated disk image.

3.43 The soak period information, part of the BWM text, provides an approximate observation time and indicates whether any observable symptoms will be present after the change.

Displaying Information About Updates

3.44 The status of all known updates (ie, 'permanent' or 'temporary'), the path names of temporary disk images, and the order of application can be determined by using the **UPD:DISPLAY** input message. The prime use of this input message is to determine if a given update has been applied to a process. If the update is in a temporary state, this input message also reveals subsequent temporary updates that would be affected if this update was backed out.

Backing Out Updates

3.45 After the soak period, the decision to either back out the updates or make them permanent must be made. Only temporary updates can be backed out. If an intermediate update must be backed out, all updates following that one throughout the whole system must also be backed out and the backouts must occur in reverse order. For example, assume there are currently a total of 103 updates applied to the system and the last four are still temporary. If a **UPD:BKOUT** input message is issued for update 101, the field update system will automatically back out update 103, then 102, then 101.

3.46 Note, also, that if a group of updates is applied using the same BWM (or CFT or DMT) name, they will be backed out as a group. This is true even if separate input messages are issued that refer to the same update name.

Making Updates Permanent

3.47 Changes resulting from the **UPD:APPLY** input message are made permanent via the **UPD:PERM** input message. Updates must be made permanent in the same order they were applied. The effect, regardless of whether the process is nonkillable or killable, is that the temporary disk image is used to overwrite the official disk image and the history file entry for the process is changed to reflect the new update status. For killable processes, the way to cause an update to take effect is to terminate and restart any currently active instances of the process. Changes resulting from the **UPD:APPLY** input message; ie, those temporary changes that were applied as address-data couplet overwrites, are made permanent in the same way as for .m update file changes applied via **UPD:PERM**.

Reclaiming Space Freed By Updates

3.48 After an update has been applied to a process and has been made permanent, it is possible that one or more fragments of patch space or the original text space are no longer being used. The field update input message **UPD:RECLAIM** is provided to 'audit' all processes currently updated. If no temporary updates are outstanding for a process, patch and/or text space is freed and are marked as being available for holding new patches.

Field Update Messages

3.49 Field update messages are input messages implemented in the UNIX RTR operating system. Refer to the 3B20D computer input/output message manual for a complete explanation of syntax rules and parameter meanings. The following messages are used in performing field updates:

(a) Temporary updates for byte, file, or function replacement are applied by using one of the following messages:

(1) Byte replacement update associated with a BWM is applied using input message **UPD:APPLY:BYTER**. **BYTER** writes into sequential memory locations in processes.

(2) File replacement update associated with a BWM is applied using input message **UPD:APPLY:FILER**. **FILER** replaces an old file with a new file. The new file retains the old file's characteristics (mode, owner, etc.).

(3) Function replacement update associated with a BWM is applied using input message **UPD:APPLY:FUNCR**. **FUNCR** changes global data, adds a new function, or replaces an old function in an executable process.

(b) Input message **UPD:PERM** makes updates associated with a temporary BWM permanent.