NOAA Eastern Region Computer Programs
and Problems NWS ERCP - No. 43

AEX - Automatic Program Execution

Harold H. Opitz
Ohio River Forecast Center
Cincinnati, Ohio

Scientific Services Division
Eastern Region Headquarters
June 1988

U.S. DEPARTMENT OF COMMERCE / National Oceanic and Atmospheric Administration / National Weather Service

NOAA TECHNICAL MEMORANDUM

National Weather Service, Eastern Region Computer Programs and Problems

The Eastern Region Computer Programs and Problems (ERCP) series is a subset of the Eastern Region Technical Memorandum series. It will serve as the vehicle for the transfer of information about fully documented AFOS application programs. The format ERCP - No. 1 will serve as the model for future issuances in this series.

1 An AFOS version of the Flash Flood Checklist. Cynthia M. Scott, March 1981. (PB81 211252).

2 An AFOS Applications Program to Compute Three-Hourly Stream Stages. Alan P. Blackburn, September 1981. (PB82 156886).

3 PUPPY (AFOS Hydrologic Data Reporting Program). Daniel P. Provost, December 1981. (PB82 199720).

4 Special Search Computer Program. Alan P. Blackburn, April 1982. (PB83 175455).

5 Conversion of ALEMBIC$ Workbins. Alan P. Blackburn, October 1982. (PB83 138313).

6 Real-Time Quality Control of SAOs. John A. Billet, January 1983. (PB83 166082).

7 Automated Hourly Weather Collective from HRR Data Input. Lawrence Cedrone, January 1983 (PB83 167122).

8 Decoders for FRH, FTJ and FD Products. Cynthia M. Scott, February 1983. (PB83 176057).

9 Stability Analysis Program. Hugh M. Stone, March 1983. (PB83 197947).

10 Help for AFOS Message Comp. Alan P. Blackburn, May 1983. (PB83 213561).

11 Stability and Other Parameters from the First Transmission RAOB Data. Charles D. Little, May 1983. (PB83 220475).

12 TERR, PERR, and BIGC: Three Programs to Compute Verification Statistics. Matthew R. Peroutka, August 1983. (PB84 127521).

13 Decoder for Manually Digitized Radar Observations. Matthew R. Peroutka, June 1983. (PB84 127539).

14 Slick and Quick Data Entry for AFOS Era Verification (AEV) Program. Alan P. Blackburn, December 1983. (PB84 138726).

15 MDR--Processing Manually Digitized Radar Observations. Matthew R. Peroutka, November 1983. (PB84 161462) (Revised June 1985, PB85-220580/AS)

16 RANP: Stability Analysis Program. Hugh M. Stone, February 1984.(PB84 161447)

17 ZONES. Gerald G. Rigdon, March 1984. (PB84 174325)

18 Automated Analysis of Upper Air Soundings to Specify Precipitation Type. Joseph R. Bocchieri and Gerald G. Rigdon, March 1984. (PB84 174333)

(Continued on Inside Rear Cover)

NOAA Eastern Region Computer Programs and Problems
NWS ERCP - No. 43

AEX - Automatic Program Execution

Harold H. Opitz
Ohio River Forecast Center
Cincinnati, Ohio

Scientific Services Division
Eastern Region Headquarters
June 1988

ACKNOWLEDGEMENT

The author wishes to greatly acknowledge the following people for their support and cooperation in the development and implementation of this program.

I cannot say thank you enough.

Especially to Don Close and the staff at OHRFC

Dr. Charles N. Hoffeditz
Dr. Ward Seguin
Dave Brandon
Bob McLeod
Janet Soslow
Cindy Scott
Steve Hentz
Bill Gery
Steve Todd
Ken Mielke
Harry Lebowitz

AEX - Automatic Program Execution

Harold H. Opitz
Ohio River Forecast Center
Cincinnati, Ohio

The AEX package allows a site to run local applications programs on a scheduled basis, freeing users for other tasks. It consists of three programs: AEX.SV (this does the actual schedule monitoring and program execution), AEXSCHD.SV (used to create and edit the schedule file) and RBOOT.SV (used by AEX's optional auto-reboot feature). Each program will be documented separately.

AEX.SV

I. Introduction

A. Purpose of Program

AEX (Auto-EXecution) is an applications program that controls the execution of other programs, macros and/or RDOS commands according to a user-created schedule. It also emulates the RDOS Command Line Interpreter (CLI) to provide normal console interaction and can correctly process AFOS ADM RUN: commands as well.

B. Movtivation for Development

NWS operational sites run many applications programs, some routinely, others only under certain weather conditions. AFOS has no facility for scheduling such activity except the PROCEDURES. These can only schedule by time, involve AFOS activity and tie up the ADM where they are run.

Operational sites also need to keep AFOS up and running. There is no way for the system to detect problems and take action; human intervention is required. This is a particular problem at part-time sites.

C. Benefits to the User

AEX provides a logical, systematic and flexible means of scheduling background activity. Programs, macros and commands can be time-scheduled or triggered by the arrival or creation of a new AFOS product or RDOS file, on any combination of days of the week. Users can prioritize jobs in the order of their importance and relationship to an operational scheme. AFOS ADM alerts can be routed to different job stations to announce the completion of selected programs. Programs can still be manually run at either the console or any ADM.

In addition, AEX can monitor AFOS. It can be optionally set to reboot the entire system automatically if AFOS hangs or crashes (single-CPU sites only). The optional manual reboot feature also simplifies manual rebooting.

II. Methodology and Software Structure

A. Program Flow and Description

1. AEX Environment

AEX operates under Data General mapped RDOS in either the foreground or background memory partition. 24K of memory and about 100 blocks of disk space are required for execution.

Some background information on RDOS will help explain how AEX works. RDOS provides 5 levels of program execution, diagrammed in Figure 1. A program can be suspended and copied to disk while another executes at a higher level (a PUSH or SWAP). When the second program completes, the first resumes at the lower level (a POP).

In a standard AFOS setup, normally only the Background Monitor program BGMON is active, at level 0. BGMON handles any input through the console and also any RUN: commands from AFOS ADM's. It is a slightly modified version of the standard RDOS CLI (Command Line Interpreter) program. When someone executes a program, BGMON is temporarily suspended and the program executes at level 1. After the program finishes, BGMON is restored at level 0.

When AEX is started, the same thing happens: BGMON is swapped out and AEX occupies level 1. But AEX runs continually, monitoring its schedule and checking for AFOS RUN: commands and/or console activity. When it needs to execute something (a console command, for example, or a job from the schedule), it executes CLI at level 2. (It is necessary to use standard CLI instead of BGMON at level 2 in order to use the AEX AFOS interface options.) The CLI will handle the console commands; if a program is to be run, CLI will execute it a level 3. (This leaves one more level for any internal swaps the programs might have. Programs swapping more than one level will not function under AEX.) When the program finishes, control returns to the CLI at level 2 and then back to AEX at level 1.

The operator can deliberately execute the CLI at level 2 by using the AEX command CTRL-P (P for PUSH) and return to AEX with the command POP. The command CTRL-K will terminate AEX and return control to BGMON at level 0.

Although the above describes an AFOS system and AEX has many features designed to work with AFOS, AEX does not require AFOS to operate. Either standard CLI or BGMON may be used at level 0. Thus AEX can be used on RFC S/140's.

2. AEX Internal Operation

AEX is a multitasking program written in Data General FORTRAN IV and Assembly language.

AEX allows the scheduling of "jobs" for automatic execution. The program maintains a runtime schedule table containing "triggers" and their associated command lines. The table can hold a maximum of 102 of these entries (triggers plus their command lines). The trigger indicates what event will automatically queue the associated command line for execution. This could be the arrival of a new AFOS product, for instance, or a particular time. The command line may contain any combination of valid RDOS commands (i.e., programs, macros, CLI language, etc.) as prescribed in the Command Line Interpreter User's Manual, but must not exceed 124 bytes in length.

The accessory program AEXSCHD.SV is used to create and edit the schedule in file AEX.CF. AEX.CF also contains information about the system AEX is to run on, as well as user runtime options that determine AEX's behavior. These options are available:

AFOS monitoring (reboot the system or halt AEX if a problem is detected)

manual reboot command (control-B, control-B)

capture of AFOS ADM RUN:'s

job queueing on AFOS products, RDOS files and/or time

rerun of schedule-initiated jobs interrupted by control-A or a system crash

disable control-C

In addition a counter must be set to indicate how long AEX should wait before taking action after detecting an AFOS problem (for use with the AFOS monitoring option). For information on creating the schedule, see the AEXSCHD documentation. The structure of AEX.CF is covered in AEX part II C.

AEX runs in two modes: CYCLE and WAIT. During CYCLE mode AEX monitors the schedule and queues jobs for execution. It also checks

for any waiting AFOS ADM RUN: commands every eight seconds and after every scheduled job completes. AEX Version 8.6 runs continuously in CYCLE mode until a carriage return is received from the console. Then it immediately suspends schedule activity and enters WAIT mode in order to execute whatever command has been typed there. (Earlier versions of AEX used a timed alteration between CYCLE and WAIT and would return a "BUSY" to the console if someone tried to use it during CYCLE.)

Five or six tasks (depending on which runtime options have been selected) operate during both modes. Figure 3 shows the relationship among the tasks. Their methodology and function is described below. Figures 4-7 show the subroutines involved in those tasks calling subroutines; subroutines are further described in AEX part II D.

Task: MAIN

The MAIN task (Figure 4) actually has two functions. The program begins by first allocating all I/O channels and loading in the schedule table. Once the table is loaded and all the runtime parameters are set, MAIN queues the remaining system tasks.

The secondary function of the MAIN task is to constantly manage and monitor the execution paths of all subsequent tasks. All task states, task flags, inter-task messages and system interrupt functions are filtered through MAIN, as well as scheduler execution. This maintains the integrity of program flow. See Figure 3.

Task: AEXMN

This is the schedule manager/system state monitor task (Figure 5). It provides the controls for the scheduling and enqueueing of jobs and, if the AFOS monitoring option is enabled, provides information on the status of AEX/AFOS interground communication. This is used as an indicator of whether AFOS is up or down. This task also updates the current date/time in file AEX.TM every time a program runs or every minute, whichever is first. (This file is used during automatic reboots.) AEXMN maintains CYCLE mode.

During CYCLE, AEXMN manages the schedule table sequentially and by priority for automatic job initation. Each entry's schedule data line includes a job priority with 0 being the highest and 7 being the lowest. A pass through the schedule table begins with a sequential search from the top of the scheduler table at a specific priority. This priority is called the reference priority. All jobs are executed, first, according to their priority and, second, to their sequential position in the table. If two priority 0 jobs are ready to queue, the first in the list (closer to top of table) will execute first. (The second job will not necessarily be next. In the time it takes the first to execute, another priority 0 job ahead of the second job in the table may become ready.)

The first pass looks at triggers for priority 0 only. The second pass will look for triggers at priority 0 and 1; a third pass will look at triggers with priorities of 0, 1, and 2 and so on. At the end of priority 7, AEX starts over at priority 0 again.

When AEX queues a schedule job, it writes the command data line to the file CLI.CM and invokes CLI. CLI does the actual execution of the program or command. AEX also writes the job's "label" from the schedule data line on the console along with a one-letter code indicating how the job is scheduled. Possible codes for time-scheduled jobs are M (minute intervals), H (hourly intervals) and D (daily intervals). T indicates the job runs on the receipt of a new product or file, while S indicates execution when a product or file exists. When a scheduled job completes, AEX returns the "R (A)" AEX prompt. (The "(A)" indicates that AEX is operating and not BGMON/CLI.) For example, if the job "ROUNDUP" is scheduled to run every hour the console will show

```
ROUNDUP H
R (A)
```

whenever the job runs.

If a job is automatically initiated, AEX enters re-CYCLE mode after the job completes. This means that AEX saves its current reference priority level (e.g., PRI=2) and begins "recycling" through all higher level priorities before entering its current reference level. An example: AEX is looking for priority 2 and higher jobs (the reference priority is 2). It finds a job to queue and it executes. Upon return, AEX will begin searching back at priority 0, then priority 0 and 1, and finally priority 0, 1, and 2 before incrementing the reference priority.

In this scheme, the reference priority will not increment until AEX passes through all re-CYCLE priority levels and the current reference priority level without initiating any jobs. This nearly ensures a true priority job initiation scheme.

AEX will enter WAIT mode if a carriage return is received from the system console, allowing immediate execution of any manual console commands.

If the auto-rerun option has been enabled, a scheduled program that is interrupted by a manual control-A or a system crash will be rerun. (In the case of a control-A, any outstanding AFOS RUN:'s will be executed first.) This also indicated on the console, for example

```
ROUNDUP H (RERUN)
```

would appear if the ROUNDUP job had been interrupted. If the operator really wishes to cancel a job, use control-I (or restart AEX with AEX/I).

Task: RDCRT

This task attempts to emulate CLI and is dormant until WAIT mode is established. During WAIT mode, it monitors and reads the console keyboard for manual command entry and user interrupts (see Figure 6). "R (A)" appears once when this task first becomes active when AEX is started. Keyboard commands such as backspace, character rubouts, full line deletions (\), command line continuation characters, and the special AEX command control characters are supported. All commands are buffered until a carriage return or an AFOS ADM RUN: command is captured.

During CYCLE mode, the console is monitored for carriage returns, and control-I, control-A and control-C interrupts. Receiving a carriage return will put AEX into WAIT mode in order to execute any console commands that have been typed. A control-I will also put AEX into WAIT. (The duration of WAIT - if no further keystrokes are received - is controlled by the first parameter in the AEX.CF .START line. Minimum WAIT possible is 10 seconds.) RDOS system control-A and control-C keyboard interrupts can be entered at any time during program execution.

The RDCRT task provides 7 AEX user commands. These single stroke console commands provide a means of executing special AEX runtime commands. They are initiated by holding down the CONTROL key and entering the appropriate character. These commands are valid only during AEX WAIT mode except for control-I, control-A, and control-C, which may be entered any time.

Control-B Control-B — "B"ootstrap initiation. This will perform a manual bootstrap from any ground. The command must be entered as two control-B's in row and will only work if this option has been enabled. The bootstrap sequence is discussed under task AXBT.

Control-I — "I"nterrupt the current program CYCLE and return AEX to WAIT mode. AEX WAIT mode permits manual console keyboard command entry while automatic job initiation is suspended.

AEX's console response to Control-I is:

```
INT (A)
R (A)
```

Control-K "K"ill the AEX program and return to level 0 BGMON/CLI. This terminates AEX program scheduling. Response from AEX at the console will be:

```
AEX TERM
R                <- from level 0 BGMON/CLI
```

Control-P "P"ause (or suspend) AEX schedule monitoring <u>indefinitely</u> at the current reference priority and start the CLI at level 2. This allows extended use of CLI; helpful if the operator wishes to perform extensive manual keyboard command entry or wishes to temporarily suspend job scheduling. Enter the RDOS command 'POP' to resume AEX scheduling. "POP" may be entered regardless of the current directory location since AEX will return to the directory current at the time of the previous control-P. Response from AEX at the console is:

```
CLI.SV (A)
R                <- from level 2 CLI
```

Control-R "R"eset the reference priority. This command will reset the current reference level priority to 0 (zero) regardless of the current reference CYCLE and re-CYCLE priorities. Response from AEX at the console will be:

```
PRI= 0 (A)
R (A)
```

Control-U "U"pdate some or all schedule trigger entries. This command will update triggers to the current system date/time/event in the following sequences and manners:

(1) With the manually entered control-U, all triggers will be updated to the current date/time. The operator will see

```
UP= ALL (A)
R (A)
```

at the console during the manual update event. This command is used if AEX has not run for a while to prevent it from trying to catch up on all the missed jobs. It should also be executed as early as possible in a New Year (like AFOS, AEX does not keep track of the year). However, one should be aware of its consequences: any hourly jobs queued

for the current hour will be reset to the next hour and will not run until then.

(2) AEXMN's updating mechanism is also invoked internally after a change has been made to the schedule. When AEXSCHD is run to change/add/delete schedule entries, it indicates the changed entries by setting flags in an update table to -1 (unchanged entries are flagged with 0). Before the next CYCLE, AEXMN initializes the triggers of the changed entries only. The initialization gives the triggers a valid starting point so that AEX can determine when to execute their associated command lines. The operator will see

```
UP= SINGL (A)
R (A)
```

at the console, which notifies the operator that AEX is checking the table and updating only those triggers appropriately flagged.

Control-X E"X"ecute the scheduler program AEXSCHD.SV. AEX allows the operator to change/add/delete entries in the schedule table while it is running. The actual execution is through CLI using the command AEXSCHD/(1,0). The schedule table is automatically reloaded and updated (control-U, option 2) following a successful run. Response from AEX at the console will be:

```
AEXSCHD.SV (A)
R (A)
RELOADED (A)
RESTART (A)

INT (A)
UP= SINGL (A)
R (A)
```

After the reload, AEX resumes normal CYCLE and WAIT mode functions. No further user entry is required.

Task: IRUN

The IRUN task is a user optioned module which communicates with AFOS and captures AFOS ADM "RUN:" commands. IRUN will check for and execute any waiting "RUN:"'s through CLI (not BGMON) every eight seconds and after every scheduled job completes, in the order in which AFOS passes them to AEX. Execution and completion are acknowledged on

the console via the .WROPR system call. "R (A)" indicates the return to AEX. For example:

```
!!B (AEX) RP PIT/P HTS/P

!!B (AEX) RUN COMPLETE
R (A)
```

Valid RUN: commands end with a carriage return with a maximum length of 36 bytes. Invalid "RUN:"'s will be ignored.

ADM-initiated programs can be interrupted with control-A or control-C (unless control-C has been disabled) on the console. Control will return to AEX at level 1 or BGMON/CLI at level 0, respectively.

Task: INTR

AEX still allows the use of the RDOS program interrupts control-A and control-C. The INTR task (Figure 7) accepts and processes these keyboard entries. Control-I, however, is handled by task RDCRT. The following is an explanation of the fundamental differences between control-A/control-C and control-I. For any keyboard entry requiring a system interrupt, the highest priority will be given to control-C, followed by control-A, and last by control-I.

Control-I is captured by RDCRT and flagged in the MAIN task. This command is generally used for interrupting the AEX CYCLE and RECYCLE modes. MAIN follows by flagging all other tasks to suspend their current activity. RDCRT will enter WAIT mode and become fully active and ready to accept keyboard entry and/or will listen for AFOS RUN: queues. The current priority level will be maintained at the current reference level.

If a control-A is received when only AEX is running (level 1), INTR takes the following action: all active tasks will be suspended, any interground communication with AFOS is terminated, the trigger table is updated and AEX will simply be restarted from level 0 BGMON or CLI. A full restart will put the program through its initialization sequence and subsequently into CYCLE mode (with the reference priority at 0, the highest level).

A control-A at level 2 (i. e., CLI has been started with control-P but no other program is running) will just restart level 2 CLI.

If a program is running (level 3), control-A will, first, interrupt the currently executing progam or command. Subsequently, the system will pass through level 2 CLI ending at level 1 AEX in the directory where AEX was started. (Unless the executing program was started from CLI level 2 -- on the console after a control-P -- in which case control-A returns the user back to level 2 CLI in the directory where AEX was started.) Each level of interrupt will be acknowledged at the console and, after AEX gains control, AEX execution will continue from its point of suspension. The console will display:

```
INT          <- level 3 (CLI)

INT (A)      <- level 1 (AEX)
R (A)
```

Unless it has been disabled (see below), the keyboard control-C command will terminate AEX with the usual foreground/background break file handling characteristics as prescribed by RDOS.

If only AEX is running (level 1), a control-C will terminate AEX and return the user to level 0 BGMON/CLI. The console will display:

```
R (A)        <- level 1 (AEX)
R            <- level 0 (BGMON/CLI)
```

A (F)BREAK.SV file will have been written to disk containing the current level 1 core image.

At level 2 (i. e., CLI started with control-P but nothing else running), control-C restarts level 2 CLI. The system will produce a core-image file (F)BREAK.SV.

A level 3 control-C will interrupt the currently executing program or command. This interrupt will bring the user to level 0 BGMON/CLI ending in the directory where the control-C was initiated. (If the executing program was started from CLI level 2 - on the console after a control-P - control-C returns the user back to level 2 CLI.) System BREAK files will be available from both level 3 and level 1. The level 3 core-image copy may be found in the disk file XBREAK.SV, while the level 1 core-image will be in (F)BREAK.SV. This provides a one-stroke capability for terminating all multi-level executing programs while maintaining current core-images from different levels on disk. A level 3 control-C interrupt will produce the following console display:

```
BREAK        <- level 3 (program)
BREAK (A)    <- level 1 (AEX)

BREAK        <- level 0 (BGMON/CLI)
R            <- level 0 R-prompt
```

If the core-image files cannot be written to disk, AEX will not overwrite the disk. It proceeds as if a control-A had been received.

The control-C command may be disabled by setting the appropriate parameter in the control file AEX.CF. This is to prevent program traps (which AEX interprets the same way as control-C) from terminating both the program and AEX during unattended operation. See the AEXSCHD documentation for more details.

Task: AXBT (AEX System Bootstrap/Auto-Terminate Functions)

Manual System Restart

AEX provides an optional mechanism to restart RDOS from a master directory by-passing the conventional RDOS "FILENAME?" and date/time queries at the console. This feature will only function if the proper user runtime options have been enabled. (IMPORTANT USER NOTE: AEX assumes the system device code for performing a hipboot/bootstrap is octal 27.)

AEX invokes the current operating system save file using the form

PARDIR:MDIR:OS.SV

where the terms are defined as:

PARDIR	Parent Directory Disk	This will always be DZ0. It must be initialized.
MDIR	Master Directory	This can be any directory/partition and/or sub-directory/sub-partition as long as it is initialized prior to the AEX system boot.
OS.SV	Operating System Filename	The filename of the currently executing RDOS operating system. The operating system must reside in MDIR (no links to other locations).

The date/time will be set by the accessory program RBOOT.SV using the AEX date/time file, AEX.TM. RBOOT will then invoke a locally created macro, BG.MC, via BGMON/CLI to execute any commands or programs necessary (restarting AFOS, for instance).

The entire manual reboot proceeds as follows:

To initiate the manual bootstrap, the user enters control-B, control-B at the console keyboard. AEX then takes the following steps:

1. If the manual reboot option has not been enabled, AEX returns the the following message and enters WAIT mode.

```
MAIN SYS BOOT UNDECLARED
R (A)
```

   If the manual reboot option is enabled, AEX returns the following message at the console and initiates the system bootstrap (continue to step 2).

2. AEX then checks for foreground activity, looking for an active foreground program three times at 5-second intervals. If foreground is active, AEX displays this message on the console after every check:

FG ACTV

(If AEX is running in foreground, it still checks the foreground, not the background. It will see itself as active.)

3. AEX checks the CPU console switch settings for a valid entry at three 5-second intervals. The valid entries for the CPU switches are 177777 (-1) or 100027 octal.

If they are not set to 177777 or 100027, AEX will display on the console:

CPU

In this case, the software still reboots the system but queries the user for "FILENAME?", date and time in the usual fashion. RBOOT.SV is not utilized.

If the CPU switch settings are valid, AEX moves to the master directory and performs the following system file functions:

RESTART.SV --> is CHATR'd --> -P

RESTART.SV --> is RENAMED --> ARESTART.SV

RESTART.SV --> is LINKED to --> RBOOT.SV

If AEX cannot perform the following rename/link commands, the bootstrap will be cancelled and the error will be reported on the console. The system will return to level 0 BGMON/CLI.

5. AEX reboots RDOS. Because of the linking in (4), RDOS will by-pass the "FILENAME?" and date/time queries and start execution of RBOOT.SV. RBOOT.SV reads the system date/time from AEX.TM but adds a minute to cover the time it takes to restart RDOS. If the date or time cannot be set for any reason, the system uses the default values (00:00 on 1/1/68) and continues. Following the date/time settings, RBOOT performs the following system file reset function:

RESTART.SV --> is UNLINKED

ARESTART.SV --> is RENAMED back to --> RESTART.SV

RESTART.SV --> is CHATR'd --> +P

Last, RBOOT.SV invokes the macro BG.MC, via BGMON/CLI. This system macro should contain any valid RDOS command(s) and/or instruction(s) necessary to start the system. For more details, see the RBOOT documentation.

Automatic System Restart (AFOS only)

AEX can also be set up to perform an automatic bootstrap if it determines that AFOS is down. AEX assumes there is a problem with AFOS if it cannot receive a response from AFOS when attempting to retrieve RUN: commands or if it receives an indeterminate error from the BG.LB KSRCF (key search) library function. The AFOS error return from KSRCF does not indicate that AFOS is totally down/crashed. AFOS may still be "partially" up (i.e., synch comms, data storage, comms spooler, MCA, etc.) while other AFOS tasks are "down" (i.e., FICR, asynch queue, ICE, overlay loading, etc.). But something must be amiss with AFOS and a fresh restart of RDOS never hurts.

The following illustrates the automatic reboot process:

1. AEX asks for a key record or RUN: command from AFOS. If it is unsuccessful, then:

2. AEX will check for AFOS running MODIFY by checking the file status of MODIFY.DT, MODIFY.AS, AND MODIFY.TX. If these files are in use, AEX assumes MODIFY is running and will not reboot the system. AEX displays the following message:

   SYS AFOS M

3. If AFOS MODIFY is not running...

   a. and the auto reboot function has not been enabled, AEX will not reboot. The following message will be typed on the console until AFOS is up and operating.

      SYS AFOS N

   b. and the function is enabled, AEX continues to query the AFOS database once per CYCLE (using only the first product key that it encounters) for a user-set number of cycles (specified in AEX.CF). AEX will type on the console

      SYS AFOS #

after each unsuccessful try. (# indicates the cycle number; it can range from 0 to 7). No RUN:'s or AFOS-dependent jobs from the table will be queued until AFOS is back up.

4. If AEX reaches the user-set maximum number of unsuccessful tries, it reboots RDOS by the same process as for the manual reboot above.

5. If the operator needs to cancel the automatic bootstrap function, one may enter control-A or control-C (<u>if control-C is enabled</u>) at the keyboard. However, if AFOS remains down, AEX will reboot the system repeatedly.

Automatic Aex Termination

The user may also option AEX to perform an automatic program termination if it has been determined that AFOS is down. When AEX terminates, it will return the system to level 0 BGMON/CLI and prompt. The same logic is utilized as in the Automatic System Restart (AFOS only):

1. AEX will ask for a key record/run: command from AFOS. If an unsuccessful response is produced then:

2. AEX will check for AFOS running MODIFY by checking the file status of MODIFY.DT, MODIFY.AS, AND MODIFY.TX. If the files are in use, it is assumed MODIFY is running. In this situation, AEX will display the following message and the automatic termination will not occur:

SYS AFOS M

3. When AFOS modify is not running...

   a. The following message will appear at the user console if the auto function is disabled and will continue to appear until AFOS is up and operating (do not proceed further).

SYS AFOS N

   b. If the function is enabled, an operator message:

SYS AFOS #

will appear at the console where # is a value from 0 to 7. The # value notifies the operator how many times AEX has queried AFOS for information per cycle and has not received anything.

c. No more queries for RUN: will occur until AFOS is up.

d. Product key records will be requested only once from the first occurence in the trigger table of each cycle.

4. If an unsuccessful response is still produced, AEX begins counting the cycle at which the error occured. Since there are 8 cycles, the maximum number of errors may be 8.

5. The operator specifies via the scheduler how many times (0-7) AEX is to wait for AFOS before invoking the termination function. During this period, all jobs flagged 'AFOS ONLY' will be held in queue.

6. When the maximum number occurs, AEX terminates and returns the system to level 0 BGMON/CLI.

B. Equations and Algorithms

AEX does not use any special equations or algorithms.

C. Relationship Among Disk Files (see Figure 2)

AEX uses the control file AEX.CF continuously. This file is created and maintained locally using the accessory program AEXSCHD.SV. If any of the system reboot functions are to be used, another accessory program called RBOOT.SV is needed. This program utilizes the AEX.TM file created and updated by AEX. Finally, AEX requires CLI.SV and CLI.ER for level 2 execution if it is to handle AFOS ADM RUN: commands (BGMON will not work for this).

FILE: AEX.CF

AEX.CF (CF=control file) contains all the entries, command lines, triggers, runtime options, and system runtime file assignments. AEX reads the trigger table, file assignments, and runtime options into memory when it is started and periodically updates the trigger table in order to maintain a current control file. The file is generally updated when an execution is to take place, whether it is a manual or automatic initiation. AEX reads the command lines using block I/O when automatically executing a job. The AEX.CF file has the following structure:

Blocks 0-7

These contain the actual entries, their triggers, and a history block. The entries may be filenames, AFOS keys or ID labels. An ID label is an actual entry but is used for operator identification only. Triggers can be events by file/key time, events by file/key status or time scheduling. The history table

tracks the past 5 trigger events by time and date. The user can retreive this history information by using AEXSCHD to create the ASCII file AEX.HS.

Blocks 8-33

The command lines are stored in these blocks. AEX computes a pointer to the desired command using the entry number N ($0 > N \leq 102$) in the following equations:

Block Number= ((N-1)/4)+8	(relative block number RB)
Record in RB= (N)-((RB-8)*4)	(relative record in block)
Start of com= (record*64)-63	(byte pointer to start of command)

AEX reads the appropriate command and posts it to the system file CLI.CM for execution via CLI.

Block 34

This block stores all entry pointer update flags. If a user changed entry has been entered, AEX will automatically "UPDATE" the trigger to see if it is time to execute this particular job.

Block 35

The file's last block contains the system file assignments. It describes the type of system AEX is to execute on; AEX will not run properly if the file does not contain the correct file assignments. Please see the AEXSCHD.SV documentation for further details.

FILE: AEX.TM

AEX.TM is always located in the system's master directory. This file contains the current system time and date for the manual/automatic bootstrap feature. RBOOT.SV uses this file to set the date and time when the operating system is bootstrapped.

Support Program: AEXSCHD.SV

This program is used to create and edit the AEX.CF control file. It can be invoked through AEX (via control-X) as well as separately. AEXSCHD can also produce an ASCII file equivalent of the control file (AEX.OT) and a trigger runtime history file (AEX.HS) for management purposes. See the section on AEXSCHD for further details.

Support Program: RBOOT.SV

RBOOT.SV is used to bypass the RDOS "Filename?" and system date/time inqurles when the user performs either a manual system boot (control-B control-B) or AEX initiates an automatic system boot. RBOOT.SV reads the "old" date/time from AEX.TM, adds one minute and resets the current system date/time, then initiates a background macro, BG.MC, via BGMON.SV (or CLI.SV). See the RBOOT documentation for more information.

D. Subroutine Functional Description

AEX0 — AEX revision level value.

UTRIG — Transfers a 5 word frame to the trigger table and updates the history time blocks.

DTPAC — Packs the system date/time information into AEX formatting structures. Uses 6 integer words of space.

AKEY — Retreives and verifies AFOS key records from DATAKEY0. Also passes system status information about AFOS.

AEXKL — Module will terminate AEX via operator request (CTRL-K) or by automatic termination flag.

AEXB2 — Posts the updated blocks 2-7 into the AEX.CF file.

AFMOD — Checks the status of AFOS MODIFY.

CTRIG — Computes the relative block and byte pointers for command line extraction from AEX.CF file.

RTRIG — This module does the inital block load of AEX.CF at runtime and verifies for a valid file assignment block.

AXSYS — 6 entry points:

- AUTOS — Automatic restart of AEX if the task status from IRUN shows a problem.
- STAC — Initializes the control-A handler address.
- CNTL — This module executes when a keyboard control-A is entered.
- COMLN — Posts and invokes automatic execution via CLI.
- AEXTM — Encodes and posts the system date/time into the AEX.TM file.

AXPMT — 8 entry points:

- PRMTR — AEX prompt message to console
- PRMTB — AEX prompt 'BUSY' message to console.
- PRMTI — This will display the AEX interrupt prompt "INT (R)" on the console.

PRMTM Passes a system message to console. The module is passed a byte pointer and byte count for display.

FGRUN Displays the "!!F (AEX) cmdline" message on system console for ADM-initiated programs.

FGCOM Displays the "!!F (AEX) RUN COMPLETE" message on system console.

AXTDK Task kill routine via FORTRAN argument list.

AXTDR Readies a task via FORTRAN argument list.

GCAEX 4 entry points

GCONI Determines the system input console, allocates a channel to it and opens the channel.

GCONO Determines the system output console and performs the same functions as GCONI.

GCRND Deletes/creates a UFD in SYS.DR, allocates a channel and opens the given "filename".

GCOPN Allocates a channel and opens a given "filename". No UFD maintenance is performed in SYS.DR.

III. Cautions and Restrictions

AEX.SV may reside on any initialized disk directory and/or partition (directory/partition=D/P). A program link is no problem. The D/P where AEX is initiated will become the "AEX" D/P. Any D/P change, whether manual (DIR USER1) or from a program (CALL DIR), will not affect AEX since it will automatically return to the "AEX" D/P upon normal and abnormal system halts. AEX will also return to the "AEX" D/P following an RDOS CLI "POP" command or from program terminations such as FORTRAN "CALL EXIT", FORTRAN "STOP" or assembly ".RTN" and ".ERTN" instructions. The exceptions to this are system panics, program TRAP violations and user CLI interrupts (control-A/control-C).

AEX has exclusive use of the AEX.CF control file, which must reside in the "AEX" directory. No links to other directories are permited. No other access to AEX.CF is permitted: due to the nature of the information contained in the control file, absolute file integrity must be maintained. Any outside attempt to access AEX.CF will result in a "FILE IN USE" error message to the user, followed by an immediate abnormal program termination and return to level 0 BGMON/CLI.

IV. References

Bunevitch, Richard (personal communication).

Data General Corporation, 1975: Eclipse-Line Real Time Disk Operating System Reference Manual (093-000129-01).

----------, 1978: RDOS/DOS Command Line Interpreter Users Manual (093-000109-01).

----------, 1979: Real Time Disk Operating System (RDOS) Reference Manual (093-000075-08).

----------, 1984: RDOS, DOS and DG/RDOS Command Line Interpreter (069-400015-01, plus 5/85 Addendum 086-000098-00).

----------, 1985: RDOS System Reference (093-400027-01).

SCHEDULING BACKGROUND ACTIVITY

PART A: INFORMATION AND INSTALLATION

PROGRAM NAME: AEX

AAL ID:
REVISION NO.: 8.61

PURPOSE: AEX schedules background program activity using a schedule file. Replacing BGMON, it also emulates BGMON at the Dasher so the same CLI commands can be used. Programs may still be run manually at either the ADMs or the Dasher. Optionally, it can be set to reboot the system if it detects that AFOS is down (automatic system restart) or it detects a CTRL-B CTRL-B sequence at the Dasher (manual system restart). Or, it can be set to shut itself down and return to BGMON if it detects that AFOS is down.

PROGRAM INFORMATION:

Development Programmer:
Harold Opitz
Location: RFC CIN
Phone: (FTS) 684-2871
Language: DG FORTRAN IV/5.20
DG ASSEMBLER

Maintenance Programmer:
Harold Opitz
Location: RFC CIN
Phone: (FTS) 684-2871
Type: Multitasking

Save File Creation Dates:

Original Release/Version 8.1	08/03/87
Update/Version 8.61	01/08/88

Running Time: Runs continously

Disk Space:

Program	42 RDOS blocks
Data	36 RDOS blocks (AEX.CF)

PROGRAM REQUIREMENTS

Program Files:

Name	Disk Location	Comments
AEX.SV	APPL1*	
AEXSCHD.SV	APPL1*	Creates/maintains schedule file
RBOOT.SV	SYSZ*	Optional, for auto-reboot
BG.MC	SYSZ*	" " "
CLI.SV,.ER	SYSZ*	For level 2 execution

Data Files:

Name	Disk Location	R/W	Comments
AEX.CF	SYSZ*	R/W	Schedule file created by AEXSCHD
AEX.TM	SYSZ*	R/W	System time/date file for auto-reboot. AEX creates and updates.

*These locations apply to AFOS; for S/140's AEX.SV can exist in any directory. AEX.CF must reside in the directory where AEX executes, which should be the operating directory. The CLI files and, if used, BG.MC and RBOOT must be placed in the master directory. AEX.TM must remain in the master directory.

AFOS Products:

ID	Action	Comments
none		none required, may be used as schedule triggers

LOAD LINE

RLDR/P/N AEX.SV/S AEX.LS/L AEXMAIN AEX<0 AC 6 7 7A BT 1 2 3 4 5 8 B2 AM TG RT KL> <FMT BG UTIL FORTAEX SYSAEX>.LB (assumes AFOSE.LB linked to SYS.LB)

PROGRAM INSTALLATION

1. Move AEX.SV and AEXSCHD.SV to APPL1, link to SYSZ. Move the CLI.- files to SYSZ.

2. If you are going to use any auto-reboot feature, move RBOOT.SV to SYSZ. Create BG.MC in SYSZ with any commands or programs you want executed after a reboot (see RBOOT Parts A and B).

3. Create a schedule using AEXSCHD (see AEXSCHD documentation).

4. Add the following CLEAR command to the local SITESTOP (AFOS sites):

CLEAR CLI.<T S C><0 1 2 3> CLI.CM

This will clear the CLI's virtual buffers and the file used to pass command lines before AFOS comes up. If these files are left open by a system crash, AEX may not function properly.

5. If desired, edit the @AFOS@ indirect to start AEX after AFOS. You can also add a "POP" afterwards to ensure that AEX will start properly no matter what state it is in when AFOS is rebooted. For example, AFOS may hang up while AEX is suspended with CTRL-P (in CLI at level 2). Thinking that AEX is not running (because of the "R" prompt), someone may just CTRL-F and restart AFOS. Many error messages will result because some files will not clear properly and AEX will not execute when it is already running. But then @AFOS@ reaches the POP and AEX returns to level 1. The disadvantage is that

If AEX is terminated normally (CTRL-K), you'll get an error message (ATTEMPT TO RESTORE A NON-EXISTENT IMAGE) when the POP tries to execute at level 0. This does no harm.

The last two lines of @AFOS@ would be:

```
AEX
POP
```

SCHEDULING BACKGROUND ACTIVITY

PART B: EXECUTION AND ERROR CONDITIONS

PROGRAM NAME: AEX

AAL ID:
REVISION NO.: 8.61

PROGRAM EXECUTION:

AEX [/U]

1. Start AEX with RUN:AEX or AEX at the Dasher. (Make sure that any interground communication with AFOS is complete before starting - this means any product storage from a program or any SAVEs from AFOS ADMs.) AEX can be executed automatically by adding AEX as the last command in @AFOS@, etc.

2. To start AEX and update the schedule queue use AEX/U instead. (Useful if AEX has not run for a while, for example if a program hung while the system was unattended).

3. Terminate AEX using CTRL-K at the Dasher. CTRL-A will not stop AEX, only reset it. If a program is running, CTRL-C will stop both it and AEX in one shot. Effects of CTRL- commands:

CTRL-B CTRL-B	Reboots system if auto-reboot is enabled.
CTRL-I	Interrupt a CYCLE mode (i.e., "BUSY" at Dasher).
CTRL-K	Stop AEX and return to BGMON.
CTRL-P	Execute CLI at level 2; for prolonged Dasher use. AEX is suspended, POP to return.
CRTL-R	Reset the reference priority.
CTRL-U	Update the schedule to current date/time.
CTRL-X	Execute AEXSCHED to create or change schedule.
CTRL-A	Resets AEX. If a program is running, halts program and returns to AEX.

continued...

CTRL-C	Shuts down any program running and AEX with break files.

4. END-OF-YEAR action required: The dates of AFOS products are Julian dates (number of days since the beginning of the year) so there is no way to tell in what year a product orignated. When jobs are scheduled on the receipt of a new version of a product, AEX compares its date/time with the date/time of the version it last processed to determine if the product is new. A January date will not register as new if the previous date was in December. Therefore, CTRL-U as early as possible in the new year to update all the schedule entries to the current date and time.

ERROR CONDITIONS

Messages from ADM	Meaning
none	

Dasher Messages	Meaning
E=AEXCF	Can't access schedule control file. AEX.CF must reside in the directory where AEX executes. Linking is not permitted. Also, size must be 36 blocks.
U=AEXCF	The schedule control file is in use, and/or you tried to operate AEX on more than one system level.
E=AEXTM	The system date/time file is in use, and/or you tried to operate AEX on more than one system level.
E=SYS BLOCK UNDECLARED	The control file does not contain a system declaration block. Redo schedule. See AEXSCHD Part B.
E=AFOS DIR	No AFOS directory specified in AEX.CF. Redo schedule.
E=AEX DIR	The AEX directory specifier has been lost. Terminate AEX and restart it.

VI. Figures and Program Flow

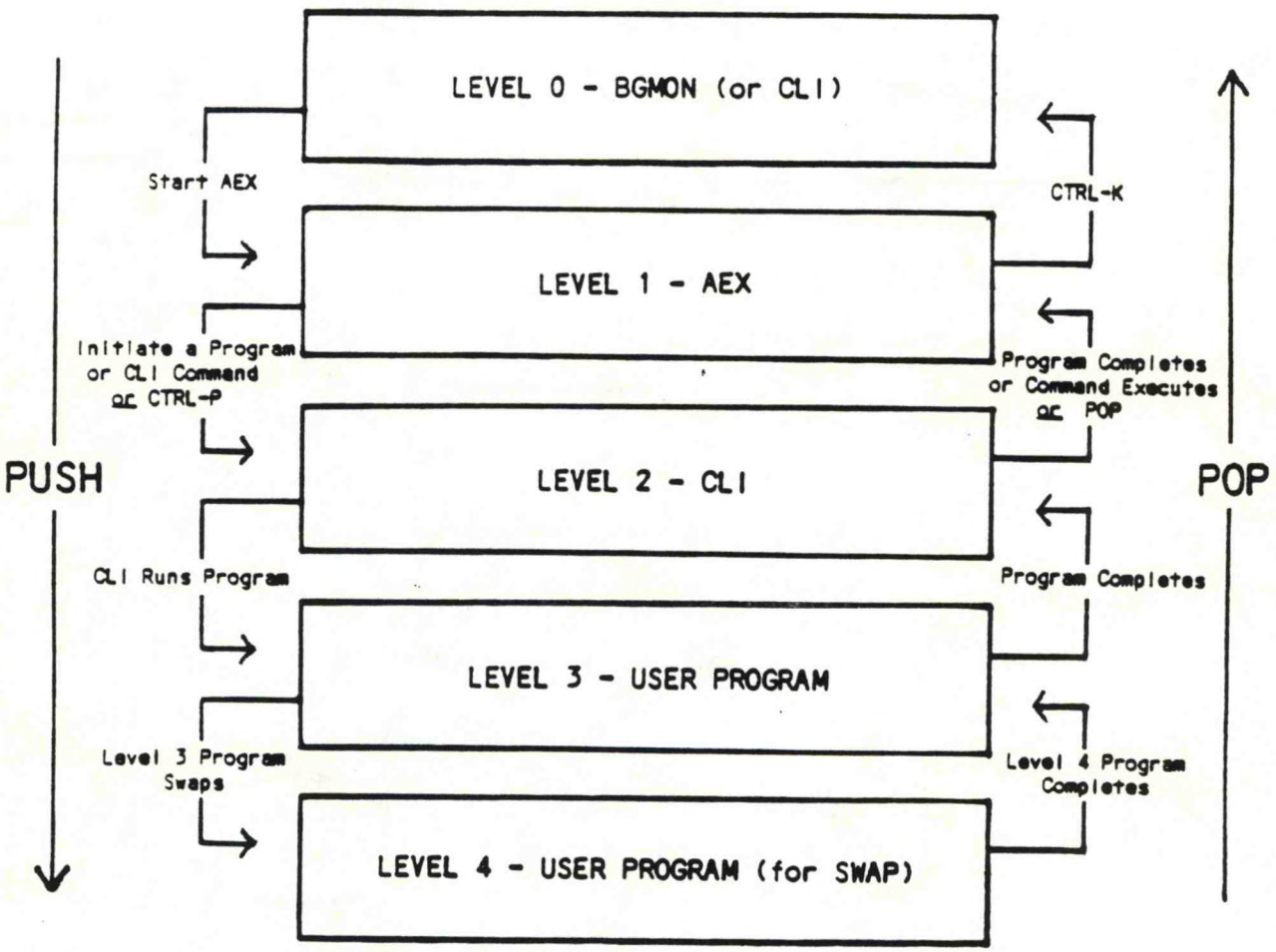

Figure 1

Levels of Execution

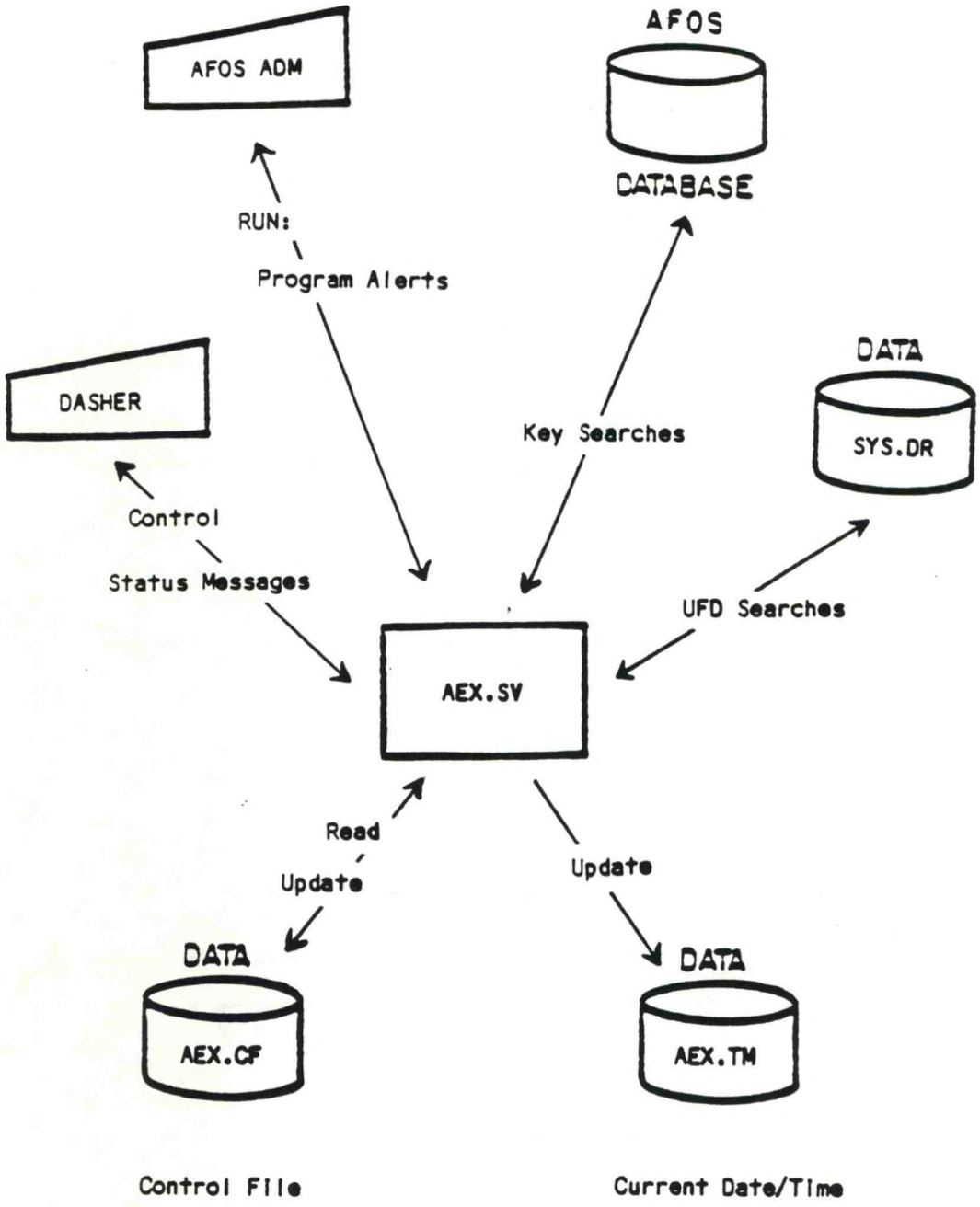

Figure 2

Relationships between AEX Files

Figure 3
Task program flow

All inter-task messages must pass through the Main task.

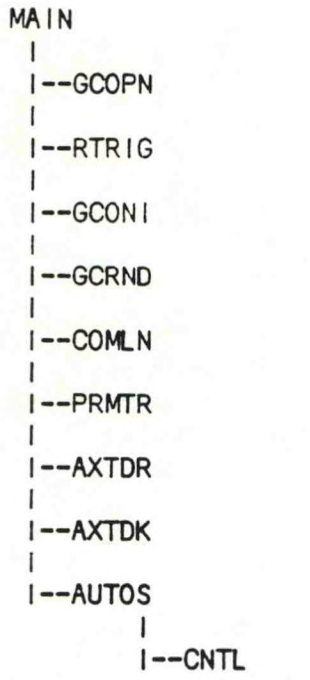

Figure 4
Task: MAIN program flow/support modules

MAIN provides program initialization and task management.

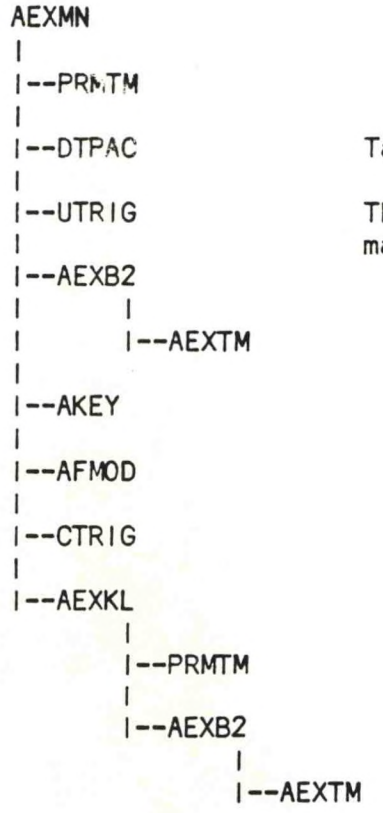

Figure 5
Task: AEXMN program flow/support modules.

This task provides CYCLE/system state management (schedule maintenance).

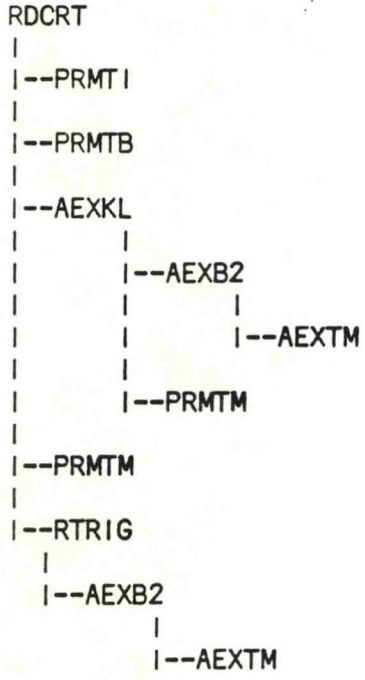

Figure 6
Task: RDCRT program flow/support modules

This task is the CLI emulation task. It also provides for AEX special command entry including the processing of the AEX control-I interrupt.

```
INTR
|
|--CNTL
```

Figure 7
Task: INTR program flow/support modules

INTR is the control-A/control-C interrupt handler.

AEXSCHD

I. Introduction

A. Purpose of Program

AEXSCHD.SV is an utility program used to maintain the AEX control file (AEX.CF).

B. Movtivation for Development

The binary AEX.CF file is essential to AEX operation and its integrity must be preserved, yet users need an easy way to create and edit this file. They also need to extract information from it in order to manage AEX.

C. Benefits to the User

Users can easily add/delete/change AEX.CF runtime parameters and schedule entries using the ASCII file AEX.IN. AEXSCHD then takes AEX.IN as input, checks it for errors and transfers the information (if correct) to AEX.CF. If errors are found, the program flags them on the Dasher and disregards the parameters and/or entries in question.

AEXSCHD also provides users with two kinds of optional external files for reference. The file AEX.HS displays historical runtime information stored inside the control file. The file AEX.OT is a user-readable reference/backup file that contains a duplicate ASCII image of the AEX.CF control file. Specialized versions of AEX.OT containing only those schedule entries of a specified priority or that are empty can also be created.

II. Methodology and Software Structure

A. Description of Program

AEXSCHD.SV is written in Data General FORTRAN IV and executes under a mapped RDOS system in either ground. 24K of memory are required.

When AEX is first installed, AEXSCHD is run once using global /D (for system declaration) to create AEX.CF and post the system declaration parameters to it. After installation, AEXSCHD has two distinct uses and program paths. One, the data input mode, allows the user to add, delete or change information in the control file. The other, data output mode, extracts information from the file for operator reference.

System Declaration

The first step in creating the AEX control file is to run AEXSCHD/D. This creates the 36-block random file AEX.CF in the current directory and posts the system declaration parameters to block 35 (last block) of the file.

Data INPUT Mode

The user creates/changes the schedule information in the ASCII file AEX.IN. This file can be created/edited using AFOS message composition, the AFOS file editor (M:/F) or any of the RDOS text editors. After creating/editing the file, the command AEXSCHD/I (Input) will write the information to AEX.CF. Samples of AEX.IN files appear in Figure 8 (AFOS) and Figure 9 (non-AFOS).

AEX.IN has three major parts: (1) the START line, which marks the start of the schedule and also contains the runtime parameters and system information, (2) the schedule information itself, which consists of schedule data lines and command data lines, and (3), the END line, which marks the end of the schedule. All data entries to AEX.IN are free format. Each entry requires a minimum of one space (octal 40) between commands or parameters with a maximum 78 characters total per line.

The START Line

The START line contains four user parameters and must always be present in the input file. This line marks the beginning of the schedule information. AEXSCHD will ignore any comments before this line.

.START par1 par2 par3 par4

.START The .START command must always begin in column 1. It determines the starting point of the schedule. Any comments before this command are ignored.

par1 Parameter 1 specifies the length of the AEX WAIT mode in seconds. WAIT mode starts when a carriage return or CTRL-I is typed at the Dasher. If no additional keystrokes (any key) occur before the par1 time elapses, AEX will return to CYCLE. Parameter 1 must be an integer value from 10 to 9999. The minimum value is 10 seconds.

par2 Parameter 2 contains 11 sub-parameters. These are user runtime options that can either be enabled (turned on) or disabled (turned off). Parameter 2 will contain a string of contiguous E's (for "E"nabled) or

D's (for "D"isabled) except for the last sub-parameter which is an integer value from 1-7. Each sub-parameter has the following function(s):

s-par1 AEX System Bootstrap. Enable this if you want to be able to reboot RDOS using CTRL-B CTRL-B and/or want the automatic RDOS bootstrap on detection of AFOS crash and/or want the Auto-Terminate function.

s-par2 AFOS Auto System Boostrap. If this is enabled AEX will reboot RDOS from any ground or program level when interground communications fail between AEX and AFOS. Sub-parameter 1 must also be enabled for this to work.

s-par3 Manual System Boostrap. If this is enabled, AEX will reboot RDOS from any ground/program level when the AEX control command CTRL-B CTRL-B is typed at the Dasher. Sub-parameter 1 must also be enabled.

s-par4 Enable/Disable CTRL-C. When applications trap, AEX interprets it as a CTRL-C and halts. If this happens frequently enough to be a problem, CTRL-C can be disabled.

s-par5 AFOS "RUN:". This must be enabled to have AEX handle AFOS ADM "RUN:" commands. Disable for non-AFOS use.

s-par6 Enable/Disable the AFOS KEY Triggers. Enable this to use the receipt/creation of AFOS products as job triggers in the schedule. If this is disabled, such triggers will be ignored. Enable it if AFOS is being used.

s-par7 Enable/Disable RDOS FILE Triggers. Enable this to use the receipt/creation of RDOS files as job triggers in the schedule. If disabled, such triggers will be ignored. Generally enabled.

s-par8 Enable/Disable INTERVAL Scheduling. Enable this to time schedule jobs. Such jobs (labeled "INTERVAL" in the schedule) will be ignored if this is disabled. Generally enabled.

s-par9 AEX Auto-Terminate. If this is enabled, AEX will terminate itself and return to level 0 BGMON or CLI if interground communications fail between AEX and AFOS. Saves the operator from having to issue a CTRL-K to stop AEX when AFOS has to be restarted (and manual bootstrap is

not being used). Similar to sub-parameter 2 except there is no RDOS reboot. Sub-parameter 1 must also be enabled for this to work.

s-par10 AEX Auto-Rerun. With this sub-parameter enabled, AEX will re-run a program that has been interrupted (by CTRL-A, CTRL-C or a crash other than a panic 6) after first running any ADM RUN:'s received after the interrupted program started. The re-run can be averted with CTRL-I (or by restarting AEX with the global I-switch) or aborted with CTRL-A (AEX will only re-run a program once).

s-par11 AEX/AFOS System (Boostrap/Auto-Terminate) Counter. This indicates the number of times AEX should recheck interground communications - after first detecting a problem - before initiating an automatic RDOS reboot (if sub-parameter 2 is enabled) or AEX automatic termination (if sub-parameter 9 is enabled). An integer value from 1-7 must be entered (even if both sub-parameters are disabled).

par3 This specifies the master disk directory/partition specifier for AFOS. (Currently this will be SYSZ.) The user must enter this specifier if any AFOS options are to be used. An abnormal program termination will occur if it is unspecified. For a non-AFOS setup, specify a value of -1.

par4 Parameter 4 is the last parameter. You do not need to enter anything because it is completed by the scheduler software. It displays the system's master directory specifier and is intended only for the operator's reference.

The Schedule (SCHEDULE DATA LINES AND COMMAND LINES)

The schedule consists of a maximum of 102 entries. Each consists of a scheduler data line that defines the job "trigger" and one or more command lines which define the job.

```
.par1 par2 par3 par4 par5 par6 par7 par8  <-- SCHEDULER DATA LINE
command line;command line;command line    <-- COMMAND LINE STRING
command line;command line;                <-- COMMAND LINE STRING
```

The scheduler data line has eight parameters:

par1 The first parameter must always begin in column 1 with a period followed by an integer value from 1-102. The integer value denotes an entry's sequential position in the schedule table.

This sequential position must be considered when specifying the execution priority of an entry (see par5, below). For example, if job number 7 and job number 12 are both priority 0 jobs and are both ready to queue, the lowest one in the schedule table will execute first. Additionally, by the time job 7 finishes, another priority 0 job lower than 12 may be ready to queue, and job 12 will have to wait some more. In general, the more important the program the lower its priority and table position should be.

par2 Parameter 2 contains a "label" consisting not more than 11 ASCII characters including any period. The label may be used to trigger jobs or just as an indicator on the Dasher (see par3 and par4).

Valid label examples:

NMCGPHP0A	9 total characters
PLOTP0A.MC	10 total characters
PLOTALLMAPS	11 total characters
BG.MC	5 total characters

Invalid label examples:

PLOTALLMAPS.	12 total characters
PLOT.ALLMAPS	12 total characters

par3 This parameter identifies the "type" of label found under parameter 2. There are only 3 "types" of label: KEY, FILE or INTERVAL. A fourth permitted entry, "ERASE", is used in editing the schedule.

KEY Specifies par2 is an AFOS key. Therefore the allowable maximum character count is 9.

FILE Specifies par2 is an RDOS filename. The maximum character count is 11 including any period. The filename must also follow the standard RDOS filename naming conventions.

INTERVAL Specifies a time scheduled event. In this case, parameter 2 is used solely as an identification label to the operator at the

output console. The maximum character count will be 11 including any period.

ERASE — This is actually a command. It is used to delete a current entry from the schedule. If used, no further scheduler line parameters are needed.

par4 — Parameter 4 specifies the actual trigger mechanism employed on the entries given in par2 and par3.

For par3=KEY the valid triggers are:

TIME — Queue the job if (1) DATAKEY0 contains a valid key record and (2) the creation time of the current product is later than that of the previous version. (The previous version time is recorded in the AEX control file, AEX.CF.)

STATUS — Queue the job if the product exists in the database. If the product is purged, then no queue will be established.

For par3=FILE the valid triggers are:

TIME — Queue the job if the file's UFD creation time in SYS.DR is later than the UFD creation time stored in the AEX control file. AEX maintains a record of the UFD file creation time.

STATUS — Queue the job if a valid UFD exists in SYS.DR for a particular filename. A job queue will not will occur if the UFD has been deleted from SYS.DR. (NOTE: No job queue will occur if the SYS.DR UFD flags a "FILE IN USE" and/or the file use-count is greater than 0.)

For par3=INTERVAL, the user will enter one of three time schedule designators. All times should be in whatever time is used for the system, since AEX checks schedule entries against the system date and time.

M**** — Minute intervals. This specifies a job will be queued every **** minutes. The intervals start from whenever AEX is started; there is no fixed starting point. The minute value can be an integer from 1 to 1440. M33 will queue a job every 33 minutes, M124 will queue a job every 124 minutes, etc.

H** Hour intervals. A job will be queued every ** minutes past the hour, i.e., H45 will queue a job every 45 minutes past the hour; 1:45, 2:45, 3:45, and so on.

D**** Daily intervals. Jobs may be queued once a day at a specified hour. D1230 will queue a job once a day at 1230Z; D2344 will queue at 2344Z, etc., assuming that the system runs on Z time.

par5 This parameter is the job queue priority and is an integer value from 0 to 7. 0 is the highest priority and 7 is the lowest. Higher priority jobs will execute before lower priority jobs. Within the same priority, schedule entry order determines execution (see par1, above).

par6 Day-of-Week specifier. AEX can be set up to queue a job every day or only on the specified day(s) of the week. Parameter 6 applies to all trigger types in parameter 3. Enter one of the following specifier types:

ALL Job will queued every day, Sunday through Saturday

DD Jobs may be queued only on certain specified days. days. To do this simply enter all the days a job is to be queued (using the day of week abbreviations) into a single, contiguous word. Days do not have to be in order. The abbreviations are:

SU for Sunday
MO for Monday
TU for Tuesday
WE for Wednesday
TH for Thursday
FR for Friday
SA for Saturday

Therefore, to have a job queued on Monday, Wednesday and Friday you must enter: MOWEFR. For Sunday, Monday, Tuesday and Saturday enter: SUMOTUSA.

par7 Par7 is an AFOS dependency flag. If "AFOS" appears here, AEX will not queue the job if AFOS is down. The job will be held indefinitely until AFOS is back up. (Any program accessing the database, for example, needs this flag since it will not run properly if AFOS is down.) If AFOS is not required, enter nothing for this parameter.

par8 The last parameter is an AFOS console ID. This is a value from 0 to 15 that specifies the AFOS ADM console to which any alert messages generated by the job will be sent. (To determine the value, use the GCID command at the desired ADM.) If you don't want any alerts returned, enter -1 for par8. Par7 must be labeled "AFOS" to use this option.

Command data lines contain the actual commands that AEX will execute if the conditions in the scheduler data line are met. The commands are executed in the order they appear in the command data lines. The commands may consist of CLI commands, macros, or programs - anything the CLI itself would accept. There are two restrictions on command lines:

(1) The total command line string may not be longer than 124 bytes. This includes all entered characters.

(2) The last, and only the last, command line in the entire command line string must absolutely end with an ASCII semi-colon (octal 73) and an ASCII carriage return (octal 15). Otherwise the entire schedule entry will be ignored. Semi-colons can be used within lines to separate commands, and carriage returns can end lines within the entire command string.

Examples of command data lines:

Example 1

```
PLOTP0A
SAODECODER.SV;                    <-- last line must end with ;
```

Example 2

```
PLOTP0A.MC;BG.MC;GTOD;DIR DP0     <-- NOT last line/entry, no ;
CLEAR/D;DIR %LDIR%;               <-- last line must end with ;
```

Example 3

```
DIR APPL1;CLEAR/D/A               <-- NOT last line/entry, no ;
INIT DZ0:APPL1                    <-- NOT last line/entry, no ;
WXR/R/D ADE/F NMCGPHCIN/P/F       <-- NOT last line/entry, no ;
DIR %MDIR%;GTOD;LIST/A PLOT;      <-- last line must end with ;
```

The END Line

AEX.IN must terminate with an ".END" command always starting with the period in column 1. AEXSCHD will stop reading at this point. Any characters/comments after this command will be ignored.

AEX.IN is also used to add, delete or change schedule entries in an existing AEX.CF. Only the START and END lines and the entries being added, deleted or changed need to be entered in this case. To add an entry, type it in using a currently empty entry number. To delete an entry, use "ERASE" for the label type (par3). To change an entry, just type in the new information. The runtime parameters may also be changed by editing the START line. Figure 3 shows an AEX.IN with changes to be made to the schedule in Figure 1. After the new AEX.IN is stored, implement the changes by rerunning AEXSHCD/I (or, if AEX is running, just type CTRL-X).

It is possible to set up two schedules and switch between them (normal versus severe weather, for instance). They should be set up carefully so that all the unwanted entries in one are erased or changed by the other. They can be linked to the alias AEX.IN for use. For example, to change to another schedule first stop AEX (CTRL-K) and execute the following:

```
UNLINK AEX.IN              (link to current schedule)
LINK AEX.IN APPL1:SCHD2    (the alternate schedule)
AEXSCHD/I                  (write new data to control file)
AEX                        (restart AEX)
```

It's also possible to just unlink and relink and then CTRL-X. Some manual input is always required, a safety feature to protect the local schedule(s).

Data OUTPUT Mode

AEXSCHD's data output mode produces two type of files. Both contain information from the AEX.CF control file translated into ASCII text.

The first type contains schedule information. The AEX.OT output file is a direct ASCII translation of the control file and is nearly identical in format to the AEX.IN input file. The purpose of the output file is two-fold. Its main purpose is to serve as a runtime reference file for the operator, displaying the actual current contents of the control file in ASCII text. In addition to the scheduler information, AEX.OT contains "last" runtime statistics of all the current entries in the comment section at the end of each entry line. These statistics are useful for determining the current status of any scheduled job. AEX.OT's format is also compatible with that of the input file

The command to create the AEX.OT file is:

AEXSCHD/O

The same type of file can be generated for specific priority or empty entries only. This is done by adding the priority value as a local argument to the command line. For example:

AEXSCHD/O 2

will output a file AEX.2 that contains all the priority 2 entries in AEX.CF. The command:

AEXSCHD/O E

will output a file AEX.E that lists all the entries which are empty (or "ERASED").

The AEXSCHD.SV program can also produce a history file named AEX.HS. This file contains the 6 latest runtime statistics for each of the schedule entries. The information can be useful for assessing the punctuality of all scheduled jobs by comparing the job's trigger to the times it actually was executed. This file is strictly informational in content and cannot be used as an AEX.CF backup since the command line strings are omitted. The command for assembling the AEX.HS history file is:

AEXSCHD/H

Both types of files may be assembled while AEX is running. Even though AEX.CF is exclusively opened by AEX, the RDOS system "FILE IN USE" and SYS.DR UFD use-count value returns do not affect execution of AEXSCHD.

B. Equations and Algorithms

AEXSCHD does not use any special equations or algorithms.

C. Relationships between Disk Files

See Figure 12.

D. Subroutine Functional Description (see also Figure 13)

SCHDI This module provides the logic path for data input to the AEX.CF control file and for the system declaration.

SCHDO This is the output path module. It posts ASCII text versions of the AEX.CF control file (AEX.OT, AEX.#, the history file AEX.HS

CVINT Converts numeric integer values to unpacked ASCII format.

DYMN Translates time to day-minutes. (0-1440 minutes)

CTRIG Supports the output path module by computing and loading the requested command block from the control file.

FILAS Posts a user-requested system declaration file block assignment to the control file. This block is hard-coded in the AEXSCHD.SV program. Determines the type of system AEX.SV is to operate on (i.e., S230+AFOS, S140+DATACOL, foreground, background partitions, etc).

SCHDM This module determines the system's master directory specifier.

SCHDA This module assembles the AFOS MODIFY filename specifiers as determined by the AFOS directory specifier.

DAWK Computes a numerical day-of-week value.

QCHK The code determines the current queue status of the control file.

III. Cautions and Restrictions

The AEXSCHD.SV scheduler utility program may reside in any initialized disk directory and/or partition. A program link will not affect program execution. Links to either the AEX.IN input file and AEX.OT/AEX.#/AEX.HS output files are permitted. However, the control file, AEX.CF, may not be accessed via link.

The AEX.CF control file should be established in the current operating directory. AEXSCHD.SV will create one if a control file does not exist. The initial run of the scheduler program must be a user SYSTEM DECLARATION (AEXSCHD/D) before any further accesses are allowed to the control file.

Errors found in the parameters for the .START line will void only that particular parameter. The current value(s) in the control file for those parameters in error will remain intact.

Errors found in the parameters for either the scheduler data line or command string line(s) will void the entire scheduler data line/command string line(s) for the particular entry. The current entry in the control file will be unaffected until the error is corrected and AEXSCHD is rerun.

AEX SCHEDULE MANAGER

PART A: INFORMATION AND INSTALLATION

PROGRAM NAME: AEXSCHD

AAL ID:
REVISION NO.: 8.60

PURPOSE: AEXSCHD is a utility used to create and maintain the schedule control file used by AEX.

PROGRAM INFORMATION:

Development Programmer:
Harold Opitz
Location: RFC CIN
Phone: (FTS) 684-2371
Language: DG FORTRAN IV/5.20

Maintenance Programmer:
Harold Opitz
Location: RFC CIN
Phone: (FTS) 684-2371
Type: Normal

Save File Creation Dates:

Original Release/Version 8.1	08/03/87
Update/Version 8.6	01/08/88

Running Time: variable, from 5 to 60 seconds

Disk Space:

Program	43 RDOS blocks
Data	36 RDOS blocks (AEX.CF only)

PROGRAM REQUIREMENTS

Program Files:

Name	Disk Location	Comments
AEXSCHD.SV	APPL1*	Link to SYSZ.

Data Files:

Name	Disk Location	R/W	Comments
AEX.CF	SYSZ*	R/W	Schedule control file, containing the triggers, schedule and history information for program execution.
AEX.SC	SYSZ*	W	Backup copy of current AEX.CF made by AEXSCHD before it makes new changes.

AEX.IN	SYSZ*	R	Input - ASCII schedule created via M:F/ or text editor.
AEX.OT	SYSZ*	W	ASCII replica of current AEX.CF. Also contains "last" runtime statistics, useful for determining state of queues. Can be used as a backup AEX.IN since format is similar.
AEX.#	SYSZ*	W	# is a number from 0 to 7. Lists only priority # entries.
AEX.E	SYSZ*	W	Lists only "ERASED" entries.
AEX.HS	SYSZ*	W	History file containing the six latest runtime statistics for each entry in schedule. Useful in comparing when jobs actually ran to when they were scheduled.

* These locations apply to AFOS; for S/140's AEXSCHD can exist in any directory. AEX.IN, AEX.OT and AEX.HS can be accessed through links. AEX.CF must reside in the same directory in which AEXSCHD executes, which should be the operating directory.

AFOS Products:

ID	Action	Comments
none		

LOAD LINE

```
RLDR/P/N AEXSCHD.LS/L AECSCHD AEXSCHDCOD SCHD<R C H I O P S M A Y Z>
<BG UTIL FORTAEX SYSAEX>.LB
```

PROGRAM INSTALLATION

1. Move AEXSCHD.SV to APPL1 and link to SYSZ.

AEX SCHEDULE MANAGER

PART B: EXECUTION AND ERROR CONDITIONS

PROGRAM NAME: AEXSCHD

AAL ID:
REVISION NO.: 8.60

PROGRAM EXECUTION:

AEXSCHD/[D,I,O,O *,H]

When installing AEX for the first time:

1. Create the schedule input file AEX.IN according to the instructions in ERCP #43.

2. Run AEXSCHD/D to create the control file AEX.CF with system information.

3. Then run AEXSCHD/I to enter all the schedule information.

To change the schedule:

1. Edit AEX.IN (E:F/ or text editor).

2. Rerun AEXSCHD/I or, if AEX is running, CTRL-X at the Dasher. CTRL-X will also produce an AEX.OT file. (As a safety feature, this AEX.OT is produced before the schedule triggers are updated. If disaster strikes and the schedule is destroyed, it can serve as a backup. But the "last run" section will not reflect the subsequent update.)

Utility options:

1. To create the output file AEX.OT (ASCII duplicate of the current AEX.CF), run AEXSCHD/O. You can also run AEXSCHD/O *, where * is a number from 0-7 or the letter E, to list only entries of priority 0-7 (output in file AEX.0-7) or only the "ERASED" entries (output in in file AEX.E).

2. To create the history file AEX.HS, run AEXSCHD/H.

All may be executed on the ADM (RUN:) or the Dasher, except CTRL-X which must be done on the Dasher. AEXSCHD requires one (and only one) switch.

ERROR CONDITIONS

Messages from ADM	Meaning
none	

Dasher Messages	Meaning
System Errors:	
E=UNSPECIFIED OPERATION	No switch used; rerun with appropriate global switch.
E=SYS BLOCK UNDECLARED	Incorrect system declaration. Check system info in AEX.IN
ERR=FILAS/W	Error writing the file assignment block for entering the system declaration word.
ERR=FILAS/R	Errror reading the file assignment block where the system declaration word is located.
EM=MDIR/R	Error reading file assignment block where the system master directory specifier is located.
EM=MDIR	AEXSCHD can't determine the system master directory specifier.
EM=MDIR/W	Error writing file assignment block for entering the system master directory specifier.
EA=MOD/R	Error reading the file assignment block where the AFOS directory specifier is located.
EA=MOD/W	Error writing the file assignment block where the AFOS directory specifier is located.
Input Errors:	
ERR= OPT DEFAULT	The runtime option parameters in the .START line (par2) are inconsistent or incorrect. The default values were loaded.
ERR= AFOS DIR	The entry in par3 (AFOS direc-

	tory) is invalid.
ERR= XXX	There is an error in schedule entry XXX (XXX ranges from 0 to 102). The command line for this entry was ignored, all others were processed.
ERR= XXX COM LIN	There is an error in the command line of schedule entry XXX (XXX ranges from 0 to 102). Make sure the command line ends with a semicolon. Other entries were processed.
ERR= ***	Numeric overflow. An illegal integer has been processed, check the parameter value.
ERR= CHAR	Illegal character in parameter.
ERR= IDLE	WAIT mode value is not between 0 and 9999. Lowest value may be -1 but the runtime default is 15 seconds.
ERR= TERM	The termination time in the .END line is invalid. Valid values range from -1 to 2359. Default is -1, continuous operation.
EI=AEXCF	AEX.CF (AEX control file) must reside in SYSZ. It cannot be accessed through a link.
EI=AEXCF SIZE	AEX.CF is not 36 blocks long or a system block was not declared on the initial run.
EI=AEXSC	AEXSCHD can't establish a backup scratch file.
EI=COPY ERROR	AEXSCHD can't copy some/all of the control file to the scratch file.
EI=AEXIN	Can't find AEX.IN (schedule input file).
EI=CON	Can't establish output console.

Output Errors:

E0=AEXCF	Can't read AEX.CF (control file).
E0=AEXOT	Can't create AEX.OT (output file) or AEX.HS (history file)

VI. Figures

```
.START 30 EEEEEEEEDD3 SYSZ                    <--  AFOS running in SYSZ; 30 sec WAIT
.1 cccSAOxxx KEY TIME 0 ALL AFOS 3            <--  Highest priority job, msgs to ADM 3
SAOCK;
.2 ROUNDUP INTERVAL H15 1 ALL AFOS -1         <--  RWR runs every hour at H+15, AFOS
RWR/E/H/B ALL/A;                                   dependent but no console alert
.5 NMCPLTPOA KEY TIME 1 ALL AFOS 2            <--  PLOTNA macro runs when a new plot-
PLOTNA;                                            file is received; msgs to ADM 2
.10 TTBBD INTERVAL D0210 1 ALL AFOS 1         <--  Runs TTBBD at 0210Z; msgs to ADM 1
TTBBD;
.11 TTBBD INTERVAL D1410 1 ALL AFOS 1         <--  Same as above but for 12Z run
TTBBD;
.15 FILNAM.TX FILE TIME 2 ALL AFOS 1          <--  Runs two programs whenever there
RP ALB/P ACY/P                                     is a new FILNAM.TX (from TTBBD)
THTE ALB/S ACY/S;                             <--  Last line must end with semi-colon
.20 cccFRHT61 KEY TIME 3 ALL AFOS 2           <--  Runs when new FRH received, lower
FRHPLOT LGA/N                                      priority job
.25 MEF INTERVAL D1145 3 ALL AFOS -1          <--  Runs TIMCHEK to check if MEF has
TIMCHEK cccMEFxxx/J 350/N ADM1/S;                  been filled out
.26 MEF INTERVAL D2040 3 ALL AFOS -1          <--  Same as above but for 12Z run
TIMCHEK cccMEFxxx/J 350/N ADM1/S;
.28 VERIFY INTERVAL D1240 3 ALL AFOS 2        <--  Runs VERIFY and backup macro daily
VERIFY
VERBU;
.29 VERIFY INTERVAL D0045 3 ALL AFOS 2        <--  Same as above but for 12Z run
VERIFY
VERBU;
.35 cccVERxxx KEY TIME 4 TUSA                 <--  Keys on VERIFY completion but will
VERDAT;                                            only run Tuesday and Saturday
.50 CLEANUP INTERVAL D2340 5 ALL              <--  Daily disk cleaning macro - low
CLEANDSK;                                          priority and not AFOS dependent
.END
```

Figure 8

Short example of an AFOS-related AEX.IN file using a variety of trigger types. All options are enabled except auto-terminate and auto-rerun. The counter is set to 3, so AEX will check interground communications three times after it detects a problem before initiating automatic reboot. (Some entries from William Gery and Steve Hentz.)

```
.START  30  EDEDDDEEDD7  -1                <-- No AFOS directory, 30 sec WAIT
.1   CLERR1CLE FILE TIME 2 ALL             <-- Runs CLE.MC on receipt of new
CLE.MC;                                      RR1 file every day
.17  CHIRR2CHI FILE TIME 2 ALL             <-- Similar to above, runs CHI2.MC
CHI2.MC;                                     when new RR2 file arrives
.40  GOES00 INTERVAL D0105 6 ALL           <-- Runs daily at 105Z at pri 6
L00.MC
@ALLGOES@;
.53  WEEKEND INTERVAL D1700 4 SUSA         <-- Only on Saturday and Sunday
NSTG/C/Y;                                    at 1700Z
.54  CRWWRKLKH   FILE STATUS 0 ALL         <-- Initiated by the existence of file
DELETE CRWRVFLKH                             The file is renamed so the job does
RENAME CRWWRKLKH LKH                         not run continuously.
NSTG/Y
XFER CRWRVFLKH $AFOS
DELETE LKH.SF
APPEND LKH.<SF S1 1> QUM3;
.83  DISK  INTERVAL D1005  0 ALL           <-- Runs everyday at 1005Z
DIR DZ0:PROD1
DELETE -.<5 6 8> <PROFJCL MAINJCL>.-/N;    <-- CLI command nesting is allowed
.95  AEXSCHD INTERVAL M0030 0 ALL          <-- Run daily every 30 minutes
AEXSCHD/0
AEXSCHD/H
INIT DZ0:ARCDATA;
.END
```

Figure 9

This is a non-AFOS example of an AEX.IN input file. Note that the START line contains -1 for an AFOS directory specifier. The manual bootstrap, RDOS file trigger and interval scheduling (time trigger) options are enabled, but CTRL-C, all the AFOS-related options, auto-terminate and auto-rerun are disabled. The counter is set to 7 (even though both options it relates to are disabled, a number must still be entered).

```
.START 40 EEEDEEEEDD1 SYSZ                <--  40 sec WAIT, CTRL-C off, counter=1
.2 ROUNDUP INTERVAL H12 1 ALL AFOS -1     <--  Change to run at H+12
RWR/E/H/B ALL/A;
.8 UAPLTS  INTERVAL D0225 1 ALL AFOS 1    <--  Add job to plot upper-air data
PLOTUA;
.9 UAPLTS  INTERVAL D1425 1 ALL AFOS 1    <--  Same as above but for 12Z run
PLOTUA;
.25 MEF    ERASE                          <--  Delete MEF check job for 00Z
.26 MEF    ERASE                          <--  Delete MEF check job for 12Z
.END
```

Figure 10

Example of an AEX.IN file containing changes that could be made to the schedule created from the AEX.IN input in Figure 8. These are: increasing WAIT to 40 seconds, disabling CTRL-C and reducing the counter to 1 (START line); adding two new jobs to plot the upper-air plotfiles (entries #8 and #9); and removing the MEF check jobs (entries #25 and #26). Entries not appearing in this file will not be changed.

```
.START  0120  EEEEEEEEDD3   SYSZ          SYSZ            ;RUN=03/23 20:42Z PRI=A

.1   cccSAOxxx   KEY       TIME   0 ALL          AFOS  3;RUN=03/23 19:52Z
.2   ROUNDUP     INTERVAL  H0015  1 ALL          AFOS -1;RUN=03/23 20:15Z
RWR/E/H/B ALL/A;
.3   ENTRY       ERASED                                ;RUN= NONE
.4   ENTRY       ERASED                                ;RUN= NONE
.5   NMCPLTPOA   KEY       TIME   1 ALL          AFOS  2;RUN=03/23 19:03Z
PLOTNA;
.6   ENTRY       ERASED                                ;RUN= NONE
.7   ENTRY       ERASED                                ;RUN= NONE
.8   ENTRY       ERASED                                ;RUN= NONE
.9   ENTRY       ERASED                                ;RUN= NONE
.10  TTBBD       INTERVAL  D0205  1 ALL          AFOS  1;RUN=03/23 02:05Z
TTBBD;
.11  TTBBD       INTERVAL  D1430  1 ALL          AFOS  1;RUN=03/23 14:30Z
TTBBD;
.12  ENTRY       ERASED                                ;RUN= NONE
.13  ENTRY       ERASED                                ;RUN= NONE
.14  ENTRY       ERASED                                ;RUN= NONE
.15  FILNAM.TX   FILE      TIME   2 ALL          AFOS  1;RUN=03/23 14:38Z
RP ALB/P/ ACY/P
THTE ALB/S ACY/S;
.16  ENTRY       ERASED                                ;RUN= NONE
.17  ENTRY       ERASED                                ;RUN= NONE
.18  ENTRY       ERASED                                ;RUN= NONE
.19  ENTRY       ERASED                                ;RUN= NONE
.20  cccFRHT61   KEY       TIME   3 ALL          AFOS  2;RUN=03/23 18:08Z
FRHPLOT LGA/N;
.21  ENTRY       ERASED                                ;RUN= NONE
.22  ENTRY       ERASED                                ;RUN= NONE
.23  ENTRY       ERASED                                ;RUN= NONE
.24  ENTRY       ERASED                                ;RUN= NONE
.25  MEF         INTERVAL  D1145  3 ALL          AFOS -1;RUN=03/23 11:45Z
TIMCHEK cccMEFxxx/J 350/N ADM1/S;
.26  MEF         INTERVAL  D2045  3 ALL          AFOS -1;RUN=03/23 00:00Z
TIMCHEK cccMEFxxx/J 350/N ADM1/S;
.28  VERIFY      INTERVAL  D1240  3 ALL          AFOS  2;RUN=03/23 12:41Z
VERIFY
VERBU;
.29  VERIFY      INTERVAL  D0040  3 ALL          AFOS  2;RUN=03/23 00:00Z
VERIFY
VERBU;
 ...
.35  cccVERxxx   KEY       TIME   4 TUSA               ;RUN=03/23 11:23Z SKIP
VERDAT
 ...
.50  CLEANUP     INTERVAL  D2340  5 ALL                ;RUN=03/23 00:00Z
CLEANDSK;
>...
.END
```

Figure 11

Sample AEX.OT (ASCII duplicate of the schedule created from the input in Figure 8). Second SYSZ (inserted by AEXSCHD) and AEXSCHD timestamp appear in START line. After the semicolon in each entry's schedule data line AEXSCHD writes the date and time the job last executed (except for daily jobs that have not yet executed: their times are entered as 00:00). If it's past the time for an daily job and it has not yet executed, the entry is marked "QUED". If the time passes into the next day and the job has still not run, it's marked "LATE". "SKIP" indicates that the trigger conditions were met but the job wasn't scheduled to run on the current day (entry #25).

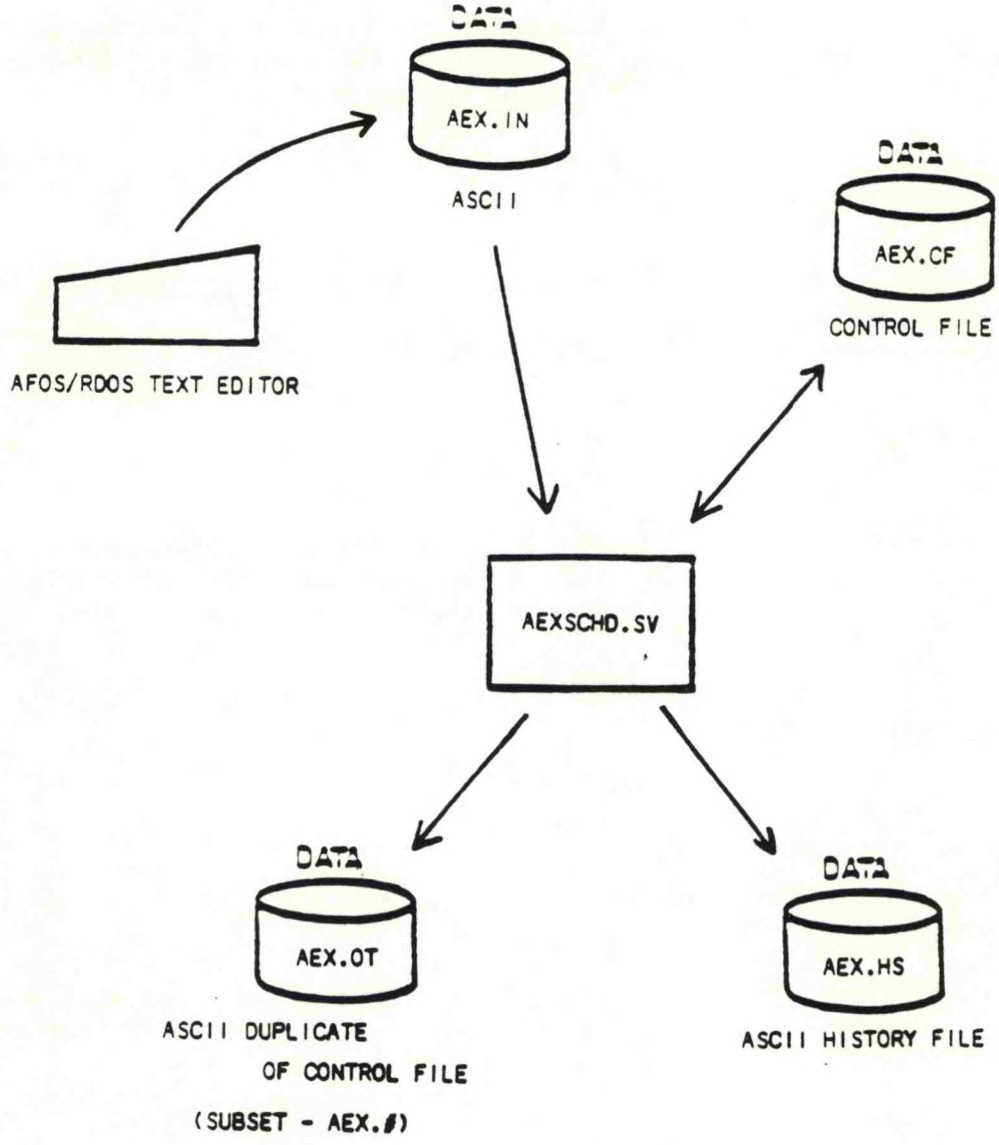

Figure 12

Relationships between AEXSCHD files

The MAIN logic will only pass through the INPUT or OUTPUT paths. The two paths function independently.

```
MAIN
 | |
 | |
 | |---------SCHDI        DATA INPUT MODE/SYSTEM DECLARATION paths.
 |
 |-----------SCHDO        DATA OUTPUT MODE path.


SCHDI
 |
 |--FILAS                The INPUT program path converts the
 |      |                AEX.IN text and posts the information
 |      |--SCHDM         to the AEX.CF control file.
 |
 |--SCHDA
 |
 |--CVINT
 |
 |--CTRIG


SCHDO
 |
 |--CVINT                The OUTPUT program path translates the
 |                       AEX.CF control file into ASCII text and
 |--DYMN                 posts it to AEX.OT or AEX.HS.
 |
 |--DAWK
 |
 |--QCHK
 |
 |--CTRIG
```

Figure 13

AEXSCHD logic flow

RBOOT

I. Introduction

A. Purpose of Program

RBOOT is an AEX utility program used to eliminate operator intervention (entering time and date at the console) during a system bootstrap (single-CPU sites only).

B. Movtivation for Development

At part-time sites, AFOS may hang or crash while unattended. This results in loss of incoming data and output from scheduled programs. AEX can detect that AFOS is non-operative and can initiate a system reboot, but the normal RDOS reboot requires that an operator enter the date and time at the console.

C. Benefits to the User

RBOOT.SV takes advantage of inherent RDOS features to bypass manual intervention in a system reboot. Thus AEX can fully restore AFOS operation unattended. RBOOT.SV is intended for this use only.

II. Methodology and Software Structure

A. Program Flow and Description

RBOOT.SV is an applications program that operates under the Data General mapped RDOS operating system. It is written in Data General Assembly language and will always execute in background. The program requires 2K of memory.

RBOOT is initiated (indirectly) by AEX. If the AFOS auto-reboot option has been selected, AEX periodically checks for foreground response. If it detects a problem (and determines that MODIFY is not running), it types "SYS AFOS 0" on the console and then retests foreground response once per cycle. The number of retests is determined by the number entered in parameter 2, sub-parameter 11 of the AEX.CF START line. On each unsuccessful try, AEX types "SYS AFOS #" on the console, where # is the number of the try. All schedule jobs requiring AFOS are held.

If the foreground does not respond within the allotted number of tries, AEX begins the reboot process. First it checks the CPU switches. If they are set to either 177777 or 100027, AEX removes the Permanent attribute from the existing RDOS file RESTART.SV, renames it ARESTART.SV and then links the alias RESTART.SV to RBOOT.SV. (If any errors are encountered, AEX restores these files to their original

state and terminates itself. Control returns to BGMON. The files are also restored to their original state should the rebooting process be interrupted at this point by CTRL-A.) If the switches are not set properly, AEX types "CPU" on the console and no renaming or relinking takes place, but the reboot continues.

At this point, AEX executes a .BOOT RDOS system call. .BOOT restarts the system and executes RESTART.SV.

If AEX's relinking was successful, RBOOT will execute via the link and reads AEX.TM for the date and time. (AEX maintains the current date/time in this file, updating it after every cycle throught the schedule and before and after any program is run.) RBOOT increments this date/time by one minute, and sets the system's date and time to the new values. If AEX.TM cannot be read or the system date/time cannot be reset, RBOOT will bring the system up on the default time of 00:00:00 on 1/1/68. RBOOT then unlinks RESTART.SV, renames ARESTART.SV back to RESTART.SV and replaces the Permanent attribute. Last, RBOOT starts BGMON/CLI and through it the macro BG.MC (if it exists). This locally-created optional macro may contain any combination of valid RDOS system commands (as specified by the Command Line Interpreter manual) necessary to fully restore the system after an RDOS reboot. (AFOS sites will want to restrart AFOS, for example.)

If the switches were not set properly and the original RESTART.SV is executed, the system will ask the operator for the date and time at the Dasher, and the reboot will end with the BGMON R-prompt. BG.MC will not execute.

The rebooting process can also be initiated manually by entering a CTRL-B, CTRL-B sequence at the Dasher (manual auto-reboot), if this option has been enabled. This makes manual rebooting simpler.

Figure 14 shows a sample Dasher printout of an unattended reboot.

B. Equations and Algorithms

There are no special equations or algorithms used by the RBOOT.SV program.

C. Relationship Among Disk Files (see also Figures 15 and 16)

RBOOT.SV requires the AEX.TM file for resetting the system date and time values. This file is updated constantly by AEX.SV on every CYCLE and before all manual or schedule initated program execution. The file contains the system date and time in the following format:

bits	0 1 2 3 4	5 6 7 8	9 10 11 12 13 14 15	word 1
	day	month	year	
bits	0 1 2 3 4	5 6 7 8 9 10	11 12 13 14 15	word 2
	hour	minute	not used	

AEX.TM is always 4 bytes long and will always be located in the system master directory.

D. Subroutine Functional Descriptions

RBOOT does not call any subroutines.

III. Cautions and Restrictions

RBOOT.SV must reside in the master directory. The program should not reside in any other sub-directory or sub-partition since these areas may not be properly initialized during reboot.

AEX can only initiate a reboot for "soft" crashes (foreground hung or down) on single-CPU systems. In a hard crash (no Dasher response, no CPU activity, panic) the system must be rebooted manually with the CPU switches. Since AEX will not have been able to rename or relink anything, rebooting will proceed as always. The operator enters the date and time and starts AFOS manually.

A system panic 13 will occur if, for whatever reason, RBOOT.SV takes an error return (.ERTN system call: return to the next higher level program). The panic occurs because CLI will not be the next higher level program (level 0) since RBOOT was started via a chain from RDOS. RBOOT itself starts CLI via a chain. The solution is to restart the system with the CPU data switches set to -1. RDOS will query the user with "FILENAME ?" in the usual manner and RBOOT will not be utilized.

The data switch register on a Data General S/140 computer is located in virtual memory cell 11A. One may enter a system "BREAK" keyboard character to access the cell via virtual console during runtime. Return to the system by entering "P <CR>" and the system will resume execution at the current interruption point. Caution should be exercised when performing this procedure.

National Weather Service AFOS management policy may restrict the use of RBOOT on AFOS to certain sites only.

IV. References

Data General Corporation, 1980: Eclipse S/140 Field Engineer's Maintenance Manual (015-000104-00).

----------, 1984: RDOS, DOS and DG/RDOS Command Line Interpreter (069-400015-01, plus 5/85 Addendum 086-000098-00).

----------, 1985: RDOS System Reference (093-4000027-01).

V.

ERCP #43
June 1988

RDOS RESTART

PART A: INFORMATION AND INSTALLATION

PROGRAM NAME: RBOOT

AAL ID:
REVISION NO.: 8.60

PURPOSE: RBOOT brings up RDOS without operator intervention. If the CPU switches are set to either 177777 or 100027, it bypasses "FILENAME?" and sets system date/time from the file AEX.TM (maintained by AEX). (100027 is the normal switch setting, 177777 has all the switches up.) RBOOT then chains to BGMON (CLI.SV on non-AFOS systems) and executes the file BG.MC, if it exists. (RBOOT.SV is not needed if you are not using any of the auto-reboot functions of AEX.)

PROGRAM INFORMATION:

Development Programmer:
Harold Opitz
Location: RFC CIN
Phone: (FTS) 684-2871
Language: DG Assembler

Maintenance Programmer:
Harold Opitz
Location: RFC CIN
Phone: (FTS) 684-2871
Type: Chain

Save File Creation Dates:

Original (test) Release/Version 8.1	08/03/87
Update/Revision 8.60	01/08/88

Running Time: .75 seconds

Disk Space:

Program	6 RDOS blocks
Data	1 RDOS block

PROGRAM REQUIREMENTS

Program Files:

Name	Disk Location	Comments
RBOOT.SV	SYSZ*	In SYSZ, AEX will rename the RDOS system file RESTART.SV and then link RESTART.SV to RBOOT.SV when executing any restart. RBOOT will undo these changes after RDOS comes up.
BG.MC	SYSZ*	Optional; executes after BGMON

Data Files:

Name	Disk Location	R/W	Comments
AEX.TM	SYSZ	R	Holds system date/time for reboot Created and maintained by AEX.

* Locations for AFOS sites; elsewhere, RBOOT.SV and BG.MC must exist in the master directory.

AFOS Products:

ID	Action	Comments
none		

LOAD LINE

```
RLDR/P/N RBOOT.SV/S RBOOT.LS/L RBOOTMAIN FORTAEX.LB SYSAEX.LB
```

PROGRAM INSTALLATION

1. Move RBOOT.SV to SYSZ. This file must reside in the master directory.

2. Create the macro BG.MC to execute whatever commands or programs you want on rebooting. AFOS sites can start AFOS. A very simple BG.MC would contain:

```
@AFOS@
```

to restart AFOS automatically when RDOS is rebooted. To prevent the CLI.T<0 1 2 3> and CLI.S<0 1 2 3> files from being left open after a reboot, CLEAR them in either BG.MC or @AFOS@ (before the foreground is started). You may also want to restart AEX.

4. If you are installing RBOOT on an S/140, you must change 10 locations in RBOOT.SV with the octal editor OEDIT:

loc	S/230	S/140
650	40506	41514
651	47523	44456
652	27123	41515
653	52000	00000
654	00000	00000
664	41107	41514
665	46517	44456
666	47056	51526
667	51526	00000
670	00000	00000

All values are in octal.

RDOS RESTART

PART B: EXECUTION AND ERROR CONDITIONS

PROGRAM NAME: RBOOT.SV

AAL ID:
REVISION NO.: 8.60

PROGRAM EXECUTION:

RBOOT is executed indirectly through AEX if either the AFOS auto-reboot or manual auto-reboot options is enabled. Before rebooting RDOS, AEX links the alias RESTART.SV to RBOOT.SV if the CPU switches are set to the normal position 100027 or 177777 (-1, all up). After the reboot, RDOS chains to RBOOT.SV via the link. RBOOT reads the date and time from the file AEX.TM, sets the system date and time (bypassing the Dasher questions) and then undoes the RESTART.SV link. RBOOT then chains to BGMON (or CLI on non-AFOS systems) and executes BG.MC.

CAUTION: If AEX.TM is not available for some reason, the default date/time of 00:00:00 on 01/01/68 is used. You will have to reset the date and time manually (SDAY/STOD on ADM; remember to disable archiving first).

ERROR CONDITIONS

Messages from ADM	Meaning
none	

Dasher Messages	Meaning
FILE DOES NOT EXIST	AEX.TM not available, date/time of 00:00:00 on 01/01/68 was used. Reset manually.
System Panic 13	Unknown system/RBOOT program error. Set CPU switches to anything but -1 and manually reboot.

VI. Figures and Program Flow

```
@ (@)
ERASE@@   T
MSG TEXT ERROR:     27
@ (@)
(IF @7:40:@@ E161:TIME PURGE IN

SYS AFOS @

SYS AFOS 1

SYS BOOTSTRAP

FG ACTIV
FG ACTIV
FG ACTIV

MASTER DEVICE RELEASED

MAPPED ECLIPSE (S/230) AFOS VERSION I / ROOS REV 5.19
WCS LOADED

@

STARTING ASYNCH WSO AFOS AT @7:45:02 06-19-87

CLEARED $TTI
CLEARED $TTO
CLEARED $TTP
CLEARED $TTP
CLEARED $LOM
CLEARED SYS.DR
STOP
CLEARED $TTI
CLEARED $TTO
CLEARED $TTP
CLEARED $TTP
CLEARED $LOM
CLEARED SYS.DR
CLEARED $TTI
CLEARED $TTO
CLEARED $TTP
CLEARED $TTP
CLEARED $LOM
CLEARED SYS.DR
CLEARED $TTI
CLEARED $TTO
CLEARED $TTP
CLEARED $TTP
CLEARED $LOM
CLEARED SYS.DR
CLEARED $TTI
CLEARED $TTO
CLEARED $TTP
```

Figure 14

Sample Dasher Output from AFOS Automatic Reboot

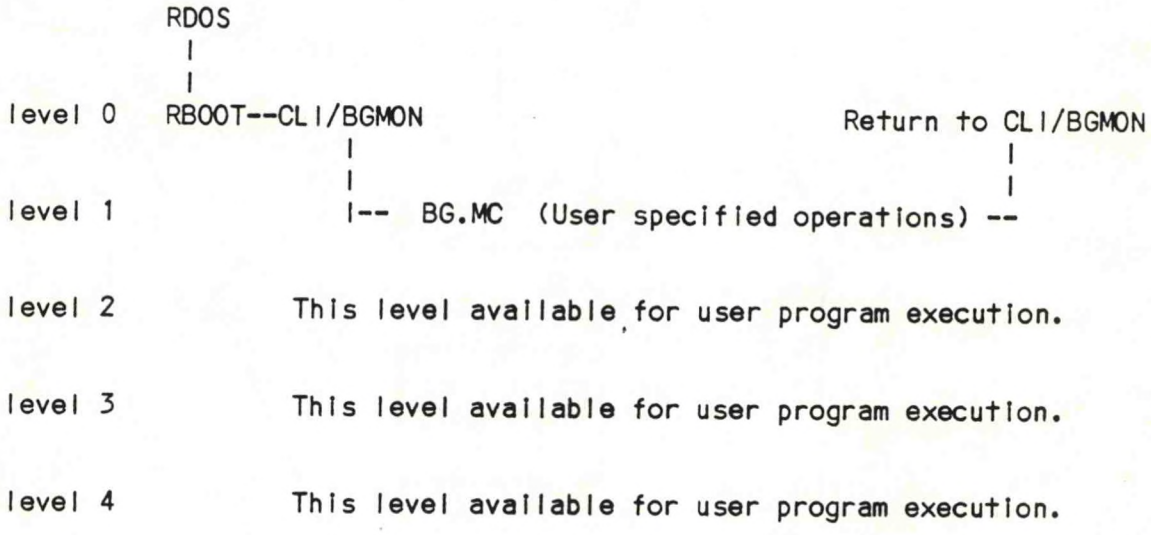

Figure 15

System Logic Macroview

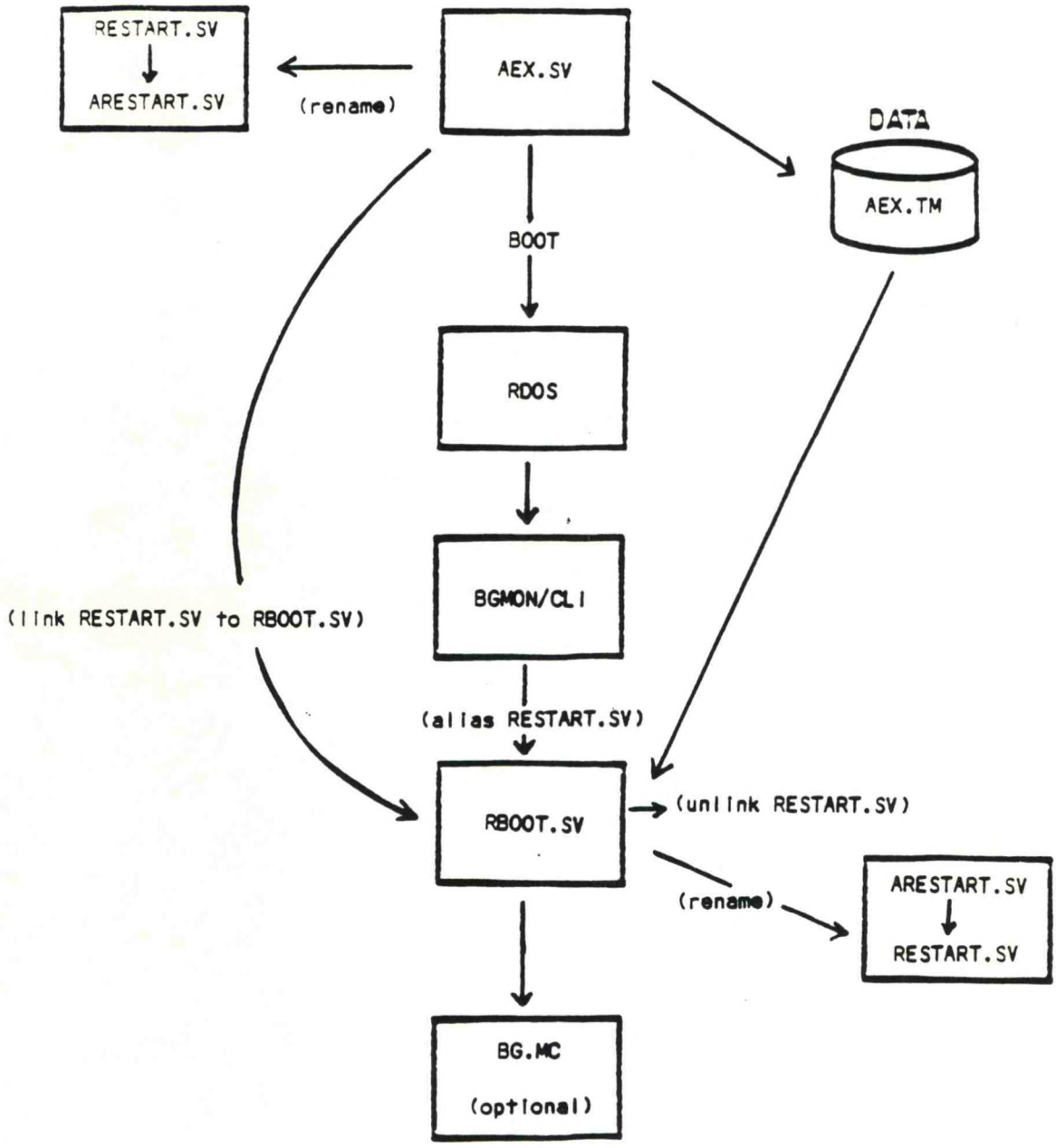

Figure 16

Relationship Among Files

NOTE

Due to the volume of documentation, source code has not been included in this ERCP. Requests for source code should be addressed to ERH SSD or to the author, Harold Opitz (OHRFC).

Eastern Region Computer Programs and Problems (Continued)

19 Verification of Asynchronous Transmissions. Lawrence Cedrone, March 1984. (PB84 189885)

20 AFOS Hurricane Plotter. Charles Little, May 1984. (PB84 199629)

21 WARN - A Warning Formatter. Gerald G. Rigdon, June 1984. (PB84 204551)

22 Plotting TDL Coastal Wind Forecasts. Paula Severe, June 1984 (Revised) (PB84 220789)

23 Severe Weather Statistics STADTS Decoder (SWX) and Plotter (SWY). Hugh M. Stone, June 1984. (PB84 213693)

24 WXR. Harold Opitz, August 1984. (PB84 23722) (Revised August 1985, PB84 100815/AS)

25 FTASUM: Aviation Forecast Summaries. Matthew Peroutka, August 1984. (PB85 112977)

26 SAOSUM: A Short Summary of Observations. Matthew Peroutka, October 1984. (PB85 120384)

27 TRAJ - Single Station Trajectory Plot. Tom Niziol, December 1984. (PB85 135002)

28 VIDTEX. Gerald G. Rigdon, February 1985. (PB85 175669/AS)

29 Isentropic Plotter. Charles D. Little, February 1985. (PB85 175651/AS)

30 CERR: An Aviation Verification Program. M. Peroutka, April 1985. (PB85 204824/AS)

31 Correlation and Regression Equation - REGRS. Hugh M. Stone, May 1985. (PB85 213353/AS)

32 Scatter Diagram and Histogram Program - SCATR. Hugh M. Stone, May 1985. (PB85 213346/AS)

33 TIMCHEK. Gerald G. Rigdon, June 1985. (PB85-221257/AS)

34 A MOS Temperature - PoP Forecast Plot. William C. Randel, October 1985. (PB86 120029/AS)

35 ROTODRAW. Thomas Niziol, November 1985 (PB86 131828/AS)

36 LAWEB: Data Processing for the Great Lakes. William C. Randel and Matthew R. Peroutka, March 1986. (PB86 176658/AS)

37 Convective Parameters & Hodograph Program - Convect. Hugh M. Stone, (Revised) January 1988. (PB88 167259/AS)

38 DWXR - SHEF Product Compression Program. Harold H. Opitz, September 1986.

39 CRASHQ: Listing Products Being Transmitted At the Time of a Crash. William C. Randel, January 1987 (PB87-151890/AS)

40 AVGPLOT and AVGCLIM. Alan Blackburn, March 1987 (PB87-180626/AS)

41 Severe Weather Potential (SPOT) Plotfile Generator. Ken LaPenta, July 1987. (PB87 217717/AS)

42 COARS Family of Programs. Lawrence Cedrone, November 1987 (PB88-131602)

NOAA SCIENTIFIC AND TECHNICAL PUBLICATIONS

The National Oceanic and Atmospheric Administration was established as part of the Department of Commerce on October 3, 1970. The mission responsibilities of NOAA are to assess the socioeconomic impact of natural and technological changes in the environment and to monitor and predict the state of the solid Earth, the oceans and their living resources, the atmosphere, and the space environment of the Earth.

The major components of NOAA regularly produce various types of scientific and technical information in the following kinds of publications:

PROFESSIONAL PAPERS—Important definitive research results, major techniques, and special investigations.

CONTRACT AND GRANT REPORTS—Reports prepared by contractors or grantees under NOAA sponsorship.

ATLAS—Presentation of analyzed data generally in the form of maps showing distribution of rainfall, chemical and physical conditions of oceans and atmosphere, distribution of fishes and marine mammals, ionospheric conditions, etc.

TECHNICAL SERVICE PUBLICATIONS—Reports containing data, observations, instructions, etc. A partial listing includes data serials; prediction and outlook periodicals; technical manuals, training papers, planning reports, and information serials; and miscellaneous technical publications.

TECHNICAL REPORTS—Journal quality with extensive details, mathematical developments, or data listings.

TECHNICAL MEMORANDUMS—Reports of preliminary, partial, or negative research or technology results, interim instructions, and the like.

Information on availability of NOAA publications can be obtained from:

NATIONAL TECHNICAL INFORMATION SERVICE
U. S. DEPARTMENT OF COMMERCE
5285 PORT ROYAL ROAD
SPRINGFIELD, VA 22161