A Practical Guide to the SYSMAN Startup Utility

November 17, 1989

This manual is a supplement to Digital's documentation for the STARTUP command set of the VMS SYSMAN Utility.

Operating System and Version: VMS Version 5.2

Richard J. Faust
Staff Specialist
E. I. du Pont de Nemours & Co. (Inc.)
Chemicals and Pigments Department
Chambers Works - Jackson Laboratory
Deepwater, NJ 08023

Contents

PREFACE v

CHAPTER 1 STARTUP COMPONENTS 1–1

1.1 LOGICAL NAMES 1–1

1.2 DATA FILES 1–2
1.2.1 STARTUP$PHASES 1–2
1.2.2 STARTUP$STARTUP_VMS 1–3
1.2.3 STARTUP$STARTUP_LAYERED 1–3
1.2.3.1 Record Format • 1–4
1.2.3.2 Controlling Order of Execution Using SORT • 1–4

1.3 COMMAND PROCEDURES 1–6
1.3.1 Debugging STARTUP.COM 1–6
1.3.2 BATCH Mode Command Files 1–7
1.3.3 Notes on SYS$BATCH 1–8
1.3.4 Phase Boundaries and Synchronization: Streamlining Execution 1–9
1.3.5 Interactive Login Limit 1–10

1.4 USER INTERFACE 1–10
1.4.1 Limitations and Undocumented Features 1–10
1.4.2 Using Variable Parameters 1–11

CHAPTER 2 NOTES ON IMPLEMENTATION 2–1

2.1 VMS$LAYERED.DAT 2–1

2.2 REDEFINE_SYS$STARTUP.COM 2–5

2.3 STARTUP_SHELL.COM 2–7

Contents

2.4 OA_STARTUP.COM 2–9

2.5 STARTUP$INTERACTIVE_LOGINS.COM 2–10

EXAMPLES

1–1 SORT_VMS$LAYERED.COM 1–5
1–2 VMS$LAYERED.SRT 1–6
1–3 Starting SYS$BATCH 1–8
1–4 Variable Parameter Syntax 1–12
2–1 VMS$LAYERED.DAT 2–2
2–2 REDEFINE_SYS$STARTUP.COM 2–5
2–3 STARTUP_SHELL.COM 2–7
2–4 RS1_STARTUP_SHELL.COM 2–9
2–5 OA_STARTUP.COM 2–9
2–6 STARTUP$INTERACTIVE_LOGINS.COM 2–10

TABLES

1–1 Logical Names 1–1
1–2 Site-Independent Startup Files 1–3
1–3 STARTUP.COM Debugging Flag Variables 1–7

Preface

Intended Audience

This document is intended for VMS system managers who are responsible for the establishment and maintenance of system startup procedures. Familiarity with the VMS operating system and management of VMS Version 5.n systems is presumed. Novice systems managers should seek the guidance of a more experienced colleague before attempting to implement a startup database.

Associated Documents

The reader of this document should be familiar with:

- *VMS SYSMAN Utility Manual*, Order Number AA-LA26A-TE
- *Guide to Setting Up a VMS System*, Order Number AA-LA25A-TE

General Comments

The System Management Utility (SYSMAN) is Digital's first effort at a bundled tool which centralizes the management of VMS nodes and clusters. The STARTUP command set of SYSMAN provides a standard interface to configure a data-driven system startup mechanism. Advantages, limitations, undocumented features, and suggestions for improving the functionality of the STARTUP portion of the SYSMAN utility are addressed in this document, which is based on experience gained from a full implementation of the SYSMAN startup database on a large production cluster.

The SYSMAN utility is a useful tool. In particular, organizing and standardizing the startup of VMS nodes and clusters is a valid concept. SYSMAN's STARTUP command set suffers from incomplete functionality and documentation, a problem common to the first release of most software packages.

The purpose of this document is to provide information which will improve the usefulness of the STARTUP command set of the SYSMAN utility, hereafter referred to as STARTUP. This goal will be pursued by augmenting the documentation supplied by Digital and by providing sample procedures which extend the functionality of the product. Modification of procedures supplied by Digital is not required to implement the enhancements which are described.

This document presumes the reader possesses a fair amount of knowledge of VMS and system management. A considerable amount of planning, development, and testing should precede any attempt to implement system startup using a SYSMAN STARTUP database on a production system or cluster. Novice system managers should proceed with caution.

1 STARTUP Components

STARTUP is comprised of three major components:

- Data Files
- Command Procedures
- User Interface

These components and the logical names which are used to reference them will be discussed in detail.

1.1 Logical Names

Table 1–1 summarizes important logical names which reference some of the various components. Note that several of the logical names are search lists which translate to node-specific and cluster-common directories. Procedures and commands which use these logical names should be composed carefully to avoid operations which place a common file in a specific directory or a specific file in a common directory.

Each of the logical names in Table 1–1 is defined in SYS$SYSTEM:STARTUP.COM, [1] the default site-independent system startup command procedure. STARTUP.COM uses logical names only to open files; hard-coded file specifications are never employed. Occasions where it may be desirable to redefine any of these logical names are discussed in later sections. Modification of these logical names may be included in SYLOGICALS.COM since all the logical names in Table 1–1 are defined prior to the execution of SYLOGICALS.

Table 1–1 Logical Names

Logical Name	Equivalence Name(s)
SYS$STARTUP	SYS$SYSROOT:[SYS$STARTUP]
	SYS$MANAGER
STARTUP$PHASES	SYS$STARTUP:VMS$PHASES.DAT
STARTUP$STARTUP_LIST	STARTUP$STARTUP_VMS
	STARTUP$STARTUP_LAYERED
STARTUP$STARTUP_VMS	SYS$STARTUP:VMS$VMS.DAT
STARTUP$STARTUP_LAYERED	SYS$STARTUP:VMS$LAYERED.DAT

[1] STARTUP.COM refers to this procedure. The term STARTUP refers to the SYSMAN STARTUP command set and related files.

1.2 Data Files

Three data files are referenced by logical names in STARTUP.COM to complete system startup (see Table 1–1):

- STARTUP$PHASES
- STARTUP$STARTUP_VMS
- STARTUP$STARTUP_LAYERED

The default file specification for each of these files begins with the logical name SYS$STARTUP, a logical search list which translates to four individual directories. This situation offers the advantage of allowing custom data files in node-specific directories for either development or production purposes.

The disadvantage lies in the possibility of making a modification to a backup or test copy of a data file (primarily STARTUP$STARTUP_ LAYERED since modification of STARTUP$PHASES and STARTUP$STARTUP_VMS is strongly discouraged by Digital) and copying that file to SYS$STARTUP. The result may be a current cluster-common data file in a node-specific directory. Cautious interactive manipulation of these files and well written command procedures are required to avoid this problem.

SYSTEM is the default owner of each of these files. The default file protection is S:RWED,O:RWED,G:RWED,W:RE. System managers may want to remove world access privileges for security reasons.

1.2.1 STARTUP$PHASES

There are eight individual startup phases, the relative order of which should not be changed. The first four are reserved for Digital and are site-independent. In order of execution they are INITIAL, CONFIG, SYSFILES, and BASEENVIRON.

The last four phases are site-dependent and are used to install layered products (hence the LP in the phase names) and configure the local software environment. These phases are called LPBEGIN, LPMAIN, LPBETA, and END.

There are no controls which prevent system managers from adding phases to this file. Any additions should be made *after* LPBEGIN since all procedures configured by Digital are executed by the time this phase is finished. Results would be unpredictable for local software started prior to the completion of the first four phases since the operating system environment would be incomplete.

Subsequent discussion may eliminate the initial tendency to want more phases in order to better control the flow of system startup.

1.2.2 STARTUP$STARTUP_VMS

As described in DEC's documentation, STARTUP$STARTUP_VMS is available for reference only and should not be modified. Files in this database which are executed during a full startup are listed in Table 1–2. The contents of the site-independent database may be listed with the following commands:

```
$ MCR SYSMAN
SYSMAN> STARTUP SET DATABASE STARTUP$STARTUP_VMS
SYSMAN> STARTUP SHOW FILE /OUTPUT=VMS$VMS.LIS
```

The /OUTPUT qualifier is optional. An undocumented mode named CALLED appears in the output generated by the above commands. This mode is associated with the valid P1 parameters which may be specified to STARTUP.COM. P1 may be supplied either in interactive mode or at system startup time by the SYSGEN parameter STARTUP_P1. File names which are CALLED are in the following form:

```
VMS$<phase name>-050_<valid P1 parameter>.COM
```

This mechanism prevents attempts to start multiple system processes such as OPCOM and ERRFMT. Parameters which are currently valid are CACHE_SERVER, CONFIGURE, CSP, ERRFMT, FULL, JOBCTL, MINIMUM, OPCOM, SMISERVER, and UPGRADE.

Note also that files normally associated with system startup such as SYLOGICALS.COM and SYCONFIG.COM are not executed directly by STARTUP but are called from more general procedures.

Table 1–2 Site-Independent Startup Files

Phase	File(s)†	Significant Calls
INITIAL	VMS$INITIAL-050_VMS.COM	SYPAGSWPFILES.COM
	VMS$INITIAL-050_LIB.COM	SYLOGICALS.COM
CONFIG	VMS$CONFIG-050_VMS.COM	
	VMS$CONFIG-050_LMF.COM	
SYSFILES	VMS$SYSFILES-050_VMS.COM	SYCONFIG.COM
	LICENSE_CHECK.EXE	
BASEENVIRON	VMS$BASEENVIRON-050_VMS.COM	
	VMS$BASEENVIRON-050_LIB.COM	
LPBEGIN	VMS$LPBEGIN-050_STARTUP.COM	SYSTARTUP_V5.COM

†All files are executed in DIRECT mode in the order listed.

1.2.3 STARTUP$STARTUP_LAYERED

The default file specification for STARTUP$STARTUP_LAYERED is SYS$STARTUP:VMS$LAYERED.DAT. This file contains site-dependent startup files and is the default target database referenced by SYSMAN's STARTUP command set.

1.2.3.1 Record Format

STARTUP$STARTUP_LAYERED is an indexed file with two keys. The primary key is the phase name and is 12 characters in length starting at offset 0 of each record. The secondary key is the name of the file to execute and is 79 characters starting at offset 13. The thirteenth character of each record–the character at offset 12–is a single character 'key' representing the mode in which the file should be executed. Valid modes are ANY, BATCH, CALLED, DIRECT, and SPAWNED; the mode character will be either A, B, C, D, or S.

The remainder of each record in STARTUP$STARTUP_LAYERED contains optional parameters P1 through P8, a flag which indicates whether or not node inclusions or exclusions are in effect, and an optional node list. Individual Pn parameters are terminated with a linefeed character. The Pn parameter list is terminated with a carriage return. The record is variable length format of 433 bytes maximum. The significance of the maximum record length is unknown.

1.2.3.2 Controlling Order of Execution Using SORT

An advantage of STARTUP is the mechanism it provides for parallel processing of system startup tasks. Multiple startup job streams are especially desirable on the multiprocessor systems which constitute an increasing fraction of DEC's product line. The maximum benefit of this parallel processing mechanism can be achieved only if the execution mode (BATCH, SPAWN, or DIRECT) can be used to control the order in which files in the startup database are run.

System managers had complete control over the order of execution of startup files prior to VMS Version 5. Commands were executed sequentially using SYSTARTUP.COM. STARTUP$STARTUP_LAYERED is read by STARTUP.COM using DCL indexed READ statements which specify the phase name as the key of reference. The four site-dependent STARTUP phases provide a course grain of control over the order in which files in the database are executed. Control within phases is limited to alphanumeric order since the phase and file name are the only indexed keys.

A brief example serves to demonstrate the benefit of using the mode to control the order of execution. Suppose in phase LPBEGIN you want to execute 10 procedures in direct mode while a spawned subprocess is starting DECnet. Assume further that the time required to start either DECnet or all 10 procedures is 1 minute. The time required to complete all 11 procedures depends on when STARTNET.COM is spawned: total elapsed time is about 1 minute if STARTNET is the first job in the phase to execute; elapsed time increases to about 2 minutes if STARTNET is the last to execute.

The VMS Sort utility can be used to control the relative order of execution of jobs within a particular phase based on mode. Example 1–1 and Example 1–2 demonstrate a method to guarantee that BATCH and SPAWN mode jobs are always executed before DIRECT mode jobs within any given phase. SORT_VMS$LAYERED.COM should be executed any time new files are added or the phase of a file is changed. The basic

concept of these examples is to sort the layered product startup database in order of phase, mode, and file name.

The STARTUP utility would be more functional and flexible if a third indexed key were added to the startup database record structure, PRIORITY. STARTUP.COM would have to be modified to take advantage of of this key since only one key can be referenced by a DCL indexed read statement. A PRIORITY key could be used in combination with the SORT utility to give system managers absolute control over the order of execution of startup files.

Example 1–1 SORT_VMS$LAYERED.COM

```
$! SORT_VMS$LAYERED.COM    Richard J. Faust    17-MAR-1989
$!
$! Sort VMS$LAYERED.DAT to produce a more predictable startup order.
$!
$! Revision history:
$! 17-MAR-1989 RJF Version 1.0
$! 12-APR-1989 RJF add use specification file
$!
$!============================================================================
$!
$ set noon
$ temp_file = "sys$common:[sys$startup]vms$layered.sorted"
$ fdl_spec = f$parse("startup$startup_layered",,,"device",) + -
             f$parse("startup$startup_layered",,,"directory") + -
             f$parse("startup$startup_layered",,,"name") + -
             ".FDL"
$ fdl_spec = "sys$common:[sys$startup]vms$layered.fdl"
$ specification_file = "sys$common:[sys$startup]vms$layered.srt"
$!
$ purge /nolog 'fdl_spec'
$ analyze_ /rms  /fdl     /out='fdl_spec' startup$startup_layered
$ edit_    /fdl  /nointer /ana='fdl_spec' 'fdl_spec'
$ create_  /fdl='fdl_spec 'temp_file
$ sort_ startup$startup_layered 'temp_file/overlay -
        /key=(pos:1,siz:12)  - ! Phase, primary indexed key
        /key=(pos:13,siz:1)  - ! Mode, not an indexed key
        /key=(pos:14,siz:79) - ! File, secondary indexed key
        /specification='specification_file -
        /statistics
$ rename /log 'temp_file startup$startup_layered
```

Example 1–2 VMS$LAYERED.SRT

```
! VMS$LAYERED.SRT    Richard J. Faust    12-APR-1989
!
! Specify the collating sequence to make sure that spawned subprocesses
! are created before direct calls are made during system startup.
!
! Revision history:
! 12-APR-1989 RJF Version 1.0
!
!============================================================================
!
/collating_sequence=(sequence=ascii,modification=("S"<"D"))
```

1.3 Command Procedures

A number of command procedures are supplied by Digital to complete system startup (see Table 1–2), the most important of these being STARTUP.COM. Other procedures are not of particular interest here and will not be discussed.

STARTUP.COM was completely rewritten for VMS Version 5 and has very functional capabilities. Actual startup tasks are completed by calling other procedures and programs (a notable exception is STARTUP* logical name definitions). Mechanisms are employed to allow parallel processing of startup jobs in any of three different modes, and steps are taken to guarantee that all jobs within a given phase are completed before proceeding to the next phase. A major advantage of STARTUP.COM with VMS Version 5 is its built-in debugging capability. This feature unfortunately is not documented by Digital.

1.3.1 Debugging STARTUP.COM

STARTUP.COM has two excellent debugging features which are controlled by the symbols STDRV$VERBOSE and STDRV$EXECUT. From STARTUP.COM:

```
$stdrv$false = 0
$stdrv$true = -1
   .
   .
   .
$stdrv$execut = stdrv$true
$stdrv$verbose = stdrv$false
```

STARTUP.COM normally is debugged by setting SYSGEN parameter STARTUP_P2 as TRUE.[1] STDRV$VERBOSE is a useful compromise between a full verify and nothing at all since procedure output is limited to informative messages regarding current phase and file execution if STDRV$VERBOSE is TRUE.

[1] P2=TRUE has no effect if STARTUP.COM is executed in interactive mode.

Files actually are executed only if STDRV$EXECUT is TRUE. Interactive debugging can be accomplished effectively if STDRV$VERBOSE is TRUE and STDRV$EXECUT is false. Debugging an actual reboot can be performed if both STDRV$VERBOSE and STDRV$EXECUT are TRUE. There is no value in setting both flag variables to false since no files are executed and no messages are displayed.

Caution should be employed in modifying STARTUP.COM to perform debugging. Digital may provide a new veresion of the file with future upgrades, thus overriding local modifications. An alternative method is to copy STARTUP.COM as supplied by Digital to other files and set the flag variables as shown in Table 1–3. STARTUP_DEBUG.COM should be used for interactive debugging, STARTUP_VERBOSE.COM for debugging during reboot.

Table 1–3 STARTUP.COM Debugging Flag Variables

File Name	STDRV$VERBOSE	STDRV$EXECUT
STARTUP.COM	FALSE	TRUE
STARTUP_VERBOSE.COM	TRUE	TRUE
STARTUP_DEBUG.COM	TRUE	FALSE

An advantage of using separate files for the various debugging functions is that any of the files can be selected at boot time using a conversational boot procedure. Use the following command after the processor-specific conversational boot procedure has been executed:

```
SYSBOOT> SET/STARTUP SYS$SYSTEM:STARTUP_VERBOSE.COM
SYSBOOT> CONTINUE
```

Note that the file specification in the above example is limited to 31 characters. Be careful to place the files either in SYS$COMMON:[SYSEXE] or SYS$SPECIFIC:[SYSEXE] as conditions warrant.

1.3.2 BATCH Mode Command Files

All BATCH mode command files are submitted to SYS$BATCH by STARTUP.COM. Log files are created for each separate batch job and written to SYS$STARTUP. The file name specification for the log files is STARTUP$n.LOG, where n is an integer (1,2,3...) corresponding to the number of the batch job *in the current phase*. There will be four log files named STARTUP$1.LOG in SYS$SPECIFIC:[SYS$STARTUP] after each reboot if there is at least one BATCH mode file in each of the layered product phases. This log file naming convention can be confusing, and an accumulation of startup log files presents a directory maintenance problem after several system restarts.

1.3.3 Notes on SYS$BATCH

The batch queue used by STARTUP.COM does not have to be named SYS$BATCH. The logical name SYS$BATCH may be defined as a *process* logical name, allowing the startup process to submit jobs to a queue such as SITE_STARTUP while other processes continue to SUBMIT jobs to the queue named SYS$BATCH or to the queue defined by the group or system logical name SYS$BATCH.

The queue manager and SYS$BATCH must be started in the first phase in which startup jobs are executed in BATCH mode. A hung startup would be the result since the SYNCHRONIZE command will continue to wait for completion of a job which will never execute. SYSTARTUP_V5.COM will complete execution before any jobs are submitted in the layered products phases and thus is a good place to start the queue manager and SYS$BATCH.

BATCH mode startup jobs can cause additional problems if a system should crash while a reboot is in progress and there are jobs still in the queue. A way to avoid this problem is to delete and initialize SYS$BATCH during each reboot. This measure is strong incentive to define SYS$BATCH as a process logical such as SITE_STARTUP or <node>_STARTUP to avoid deleting production jobs in the queue named SYS$BATCH. Example 1–3 demonstrates how starting SYS$BATCH might be coded.

Example 1–3 Starting SYS$BATCH

```
$ set noon
$ this_node = f$getsyi("nodename")
$!
$! Make sure the startup queue is empty before starting it.
$!
$ start /queue /manager site_data:jbcsysque.dat
$ delete /queue 'this_node'_startup
$ initialize 'this_node'_startup -
        /queue                   -
        /batch                   -
        /start                   -
        /job_limit=16            -
        /base_priority=4         -
        /protection=(g:rw,w)
$!
$! Define SYS$BATCH to make sure that SYSMAN has a running queue to which
$! startup jobs can be submitted.
$!
$ define sys$batch 'this_node'_startup
```

It may be important to note that global symbols defined in SYSTARTUP_V5.COM are available to SPAWN mode files but not BATCH mode files. Group or system logical names and DCL parameters P1-P8 can be used to pass data to command files executed in BATCH mode.

1.3.4 Phase Boundaries and Synchronization: Streamlining Execution

An assumed goal of the parallel processing techniques of STARTUP.COM is that systems should reboot as quickly as possible. The existence of four layered product startup phases addresses the conflict between executing all startup jobs in parallel and product dependencies. The elapsed time required for startup is minimized by maximizing parallel processing *within each phase* under the constraint of product dependencies. The fact that DECnet should be started before LAT and Message Router is an example of a product dependency which might be resolved by exection of the startup files in different phases.

STARTUP.COM uses the SYNCHRONIZE command to ensure that all BATCH jobs in a given phase are completed before proceeding to the next phase. The PRCCNT (process count) item of the F$GETJPI lexical function is used to determine that all subprocesses are completed. All BATCH mode jobs are completed before testing for subprocess completion begins.

The SYNCHRONIZE command can actually *increase* the amount of time required for startup since processing does not necessarily continue immediately after all jobs being synchronized are finished. SYNCHRONIZE evidently uses a polling mechanism at intervals which increase with the elapsed time of the jobs being polled. This delay generally will not be significant and is a necessary evil. A 2 second delay is coded in the loop which verfies that all subprocesses are completed.

STARTUP_VERBOSE.COM, discussed in Section 1.3.1, can be used to maximize parallel processing within phases. Phase synchronization occurs at the end of each phase after all files in the current phase have been spawned, submitted, or executed. A SYNCHRONIZE command and message is issued for each batch job in the phase and a message will be displayed until all subprocesses are completed if STDRV$VERBOSE is TRUE:

```
   .
   .
   .
%STDRV-I-STRTPHAS, Starting phase LPBEGIN
%STDRV-I-INDXFILE, using indexed file STARTUP$STARTUP_VMS
%STDRV-I-DIRECT, @SYS$STARTUP:VMS$LPBEGIN-050_STARTUP.COM
%STDRV-I-INDXFILE, using indexed file STARTUP$STARTUP_LAYERED
%STDRV-I-SPAWN, Spawn/nowait @SYS$STARTUP:STARTNET.COM
%STDRV-I-SPAWN, Spawn/nowait @SYS$STARTUP:COPY_SYSDUMP.COM
%STDRV-I-BATCH, Submit/name=Startup$1 SYS$STARTUP:CDDSTRTUP.COM
%STDRV-I-BATCH, Submit/name=Startup$2 SYS$STARTUP:VAXSIMPLUS_STARTUP.COM
%STDRV-I-DIRECT, @SYS$STARTUP:2780_3780_STARTUP_SHELL.COM
%STDRV-I-SYNCH, synchronizing with job STARTUP$2
%STDRV-I-SYNCH, synchronizing with job STARTUP$1
%STDRV-I-WAITSUB, waiting for subprocess completion(s)...
%STDRV-I-WAITSUB, waiting for subprocess completion(s)...
%STDRV-I-WAITSUB, waiting for subprocess completion(s)...
%STDRV-I-WAITSUB, waiting for subprocess completion(s)...

%STDRV-I-STRTPHAS, Starting phase LPMAIN
   .
   .
   .
```

The timing and number of the SYNCH and WAITSUB messages shown above can be used to determine whether elapsed time for DIRECT mode files is less than or greater than elapsed time for BATCH and SPAWN mode files for the current phase. The distribution of startup files between phases can then be modified to maximize overlap, again within the constraint of any dependencies. Startup files which have no dependencies, such as those for language compilers, can be used as 'filler' in phases where BATCH and SPAWN processing requires more time to execute than DIRECT mode files.

It may be counterproductive to submit or spawn some startup files since the time and overhead required to initiate the job may be greater than the resources required to actually run the job. Short command procedures should be run in DIRECT mode.

1.3.5 Interactive Login Limit

STARTUP.COM calls a procedure which sets the interactive login limit shortly after SYSTARTUP_V5.COM has completed execution and before any files in phase LPBEGIN are executed. The number of logins which are enabled is controlled by the global symbol STARTUP$INTERACTIVE_LOGINS, the default value of which is 64. Enabling logins at this point of the system startup is premature at sites where the software environment is not functional until many of the files in the layered product database have been executed.

The timing of enabling interactive logins can be controlled by redefining STARTUP$INTERACTIVE_LOGINS as 0 in SYSTARTUP_V5.COM. A procedure then can be created to enable interactive logins and called from the startup database at a more appropriate time. See Section 2.5.

1.4 User Interface

1.4.1 Limitations and Undocumented Features

The STARTUP command set of the VMS SYSMAN Utility provides the capability to read, modify, and display the startup databases. Although the documentation for this interface in generally adequate there are some practical characteristics and other limitations which constrict the effectiveness of the utility which are not covered. These 'features', some of which have been discussed previously, are summarized below:

- Parameters may be only 30 characters in length (probably a quirk of SYSMAN.EXE).
- All startup files must be located in the directories defined by the logical search list SYS$STARTUP.
- A given file may be called only once during system startup.
- All parameters must be reentered if an entry in a database is modified.

- Command procedures which are called from the startup database have default values for P1-P8 which are derived from SYSGEN parameters STARTUP_Pn. Parameters must explicitly be set to null values to avoid conflict. In particular, STARTNET.COM should be entered in the database with null parameters to avoid conflict:

```
$ MCR SYSMAN
SYSMAN> STARTUP ADD FILE STARTNET.COM /PAR=(P1,P2,P3,P4,P6,P6,P7,P8)
```

- BATCH mode files can be submitted only to SYS$BATCH.
- There is no explicit control over the order in which files are executed in a given phase.
- The STARTUP$STARTUP_VMS and STARTUP$STARTUP_LAYERED database files can not be read while another system in a cluster is booting.

A number of the limitations listed above result from the method used to store parameters P1-P8 in STARTUP$STARTUP_LAYERED (see Section 1.2.3.1). Terminating parameter fields with a linefeed character is a data compression technique which unnecessarily limits the flexibility of parameter entry and modification. Higher priority should be given to functionality and flexibility than to disk space requirements for this trivially small file.

1.4.2 Using Variable Parameters

There are occasions when it may be desirable to use STARTUP to specify a DCL parameter which is a variable rather than a constant. Lexical functions, logical names, and global symbols can be used to achieve this flexibility if careful attention is given to syntax. This flexibility is possible because of the way STARTUP.COM reads parameters from the startup database. Each file in the startup database is unique and described by a single record. DCL statements are used to parse each record and assign values to local symbols of the form STDRV$Xn. These STDRV$Xn symbols correspond to symbols P1-P8 when the file is actually executed. It may be important to repeat that parameters assume default values defined by SYSGEN parameters STARTUP_Pn if not explicitly set to null values.

If STDRV$X1 is a valid text string which represents a global symbol it will be evaluated before the procedure or program is executed or submitted by STARTUP.COM. The following code example simplifies how parameters from the database are treated by STARTUP.COM. Assume that the file to be executed is a command procedure and that the text string for P1 in the startup database is ''F$TRNLNM("RJE_NODE")'.

```
$ p1 = f$extract(stdrv$pos,stdrv$pos2,stdrv$param) !parse database record
$ show sym p1
  P1 = "''F$TRNLNM("RJE_NODE")'"
$ parameters = " """+p1+""""
$ show sym parameters
  PARAMETERS = " "''F$TRNLNM("RJE_NODE")'""
$ if mode .eqs. "DIRECT" then @'file_to_execute' 'parameters'
$ if mode .eqs. "BATCH" then submit 'file_to_execute' /par=("''p1'")
```

A similar mechanism is used to run programs. Any logical name is accessible via the F$TRNLNM lexical function. Any lexical function may be used. Also accessible are global symbols that are defined in procedures such as SYSTARTUP_V5.COM which are executed before any files in the layered product database are called. Example 1–4 demonstrates the command syntax required to specify variable parameters–parameters which are parameters.

Example 1–4 Variable Parameter Syntax

```
$ MCR SYSMAN
SYSMAN> STARTUP MODIFY FILE START_RJE.COM -
SYSMAN>_/PAR=(P1:"''F$TRNLNM(""RJE_NODE"")'",P2,P3,P4,P5,P6,P7,P8)
SYSMAN> STARTUP SHOW FILE START_RJE.COM /FU
%SYSMAN-I-COMFIL, contents of component database on node MYNODE
Phase        Mode   File
------------ ------ --------------------------------
LPMAIN       DIRECT START_RJE.COM
      Enabled on All Nodes
      P1:''F$TRNLNM("RJE_NODE")' P2: P3: P4: P5: P6: P7: P8:
```

2 Notes on Implementation

The advent of VMS Version 5 and the STARTUP utility presents opportunity and challenge. The overall architecture of system startup must be evaluated and many of the details previously discussed taken into consideration. Planning and testing should be careful and thorough.

This document is based on the full implementation of STARTUP on a production cluster with 8 nodes and a user base of more than 3000 accounts. Several examples for fully utilizing or extending the functionality of the STARTUP command set of the VMS SYSMAN Utility already have been discussed. Additional examples are presented in this chapter. The following principles guided the implementation of STARTUP on the production cluster:

- Startup procedures will be maintained in STARTUP$STARTUP_ LAYERED, in all possible cases. SYSTARTUP_V5.COM will contain very few commands as a result.
- No new startup phases or databases will be created.
- Interactive logins will not be enabled until after ALL-IN-1 is started.
- System startup should complete as quickly as possible.
- Tools will be implemented to extend the functionality of STARTUP as necessary.

This section is incomplete in that some explanatory text can be added to the comments which already are contained within each procedure. Inclusion of the examples as is hopefully will be of some value.

2.1 VMS$LAYERED.DAT

Example 2–1 is an example of a startup database for a production cluster. Full listings are provided for entries of particular interest. The relative order of important files such as STARTNET.COM and OA_ STARTUP.COM should be noted. Procedures are included in this database so as to maximize parallel processing while taking chronological product dependencies into consideration. Several actual reboots probably will be needed to fine tune the database.

Example 2–1 VMS$LAYERED.DAT

```
%SYSMAN-I-COMFIL, contents of component database on node MYVAX
Phase        Mode   File
------------ ------ --------------------------------
LPBEGIN      BATCH  CDDSTRTUP.COM
LPBEGIN      BATCH  COLLECT_SYSFIL_DATA.COM
LPBEGIN      BATCH  VAXSIMPLUS_STARTUP.COM
LPBEGIN      SPAWN  COPY_SYSDUMP.COM
LPBEGIN      SPAWN  MOUNT_DISKS.COM
LPBEGIN      SPAWN  STARTNET.COM
LPBEGIN      DIRECT 2780_3780_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:RJELOAD.COM P2:RJE_NODE P3:''F$TRNLNM("RJE_NODE")' P4: P5: P6: P7: P8:

LPMAIN       BATCH  CSOC_PCLINK_STARTUP.COM
LPMAIN       BATCH  OA_STARTUP.COM
      Enabled on All Nodes
      P1: P2: P3: P4: P5: P6: P7: P8:
LPMAIN       DIRECT BASIS_STARTUP.COM
LPMAIN       DIRECT CIMS_STARTUP.COM
LPMAIN       DIRECT DBMS_STARTUP_SHELL.COM
LPMAIN       DIRECT DTRSTUP.COM
LPMAIN       DIRECT DTRSTUPA1.COM
LPMAIN       DIRECT DTRSTUPT1.COM
LPMAIN       DIRECT FMSTARTUP.COM
LPMAIN       DIRECT INSTALL_IMAGES.COM
LPMAIN       DIRECT LAT_STARTUP.COM
LPMAIN       DIRECT LTM$SETUP.COM
LPMAIN       DIRECT MGR_STARTUP.COM
LPMAIN       DIRECT MSDS_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:MSDS_STARTUP.COM P2:MSDS_NODES P3:SUBMIT P4:/user=cortex_msds P5: P6: P7: P8:
LPMAIN       DIRECT NMS_RBMS$STARTUP.COM
LPMAIN       DIRECT PD_STARTUP_SHELL.COM
LPMAIN       DIRECT RSX$STARTUP.COM
LPMAIN       DIRECT S2020_STARTUP.COM
LPMAIN       DIRECT SITETERM.COM
LPMAIN       DIRECT SPM$STARTUP.COM
LPMAIN       DIRECT SSU_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:SSU_STARTUP.COM P2:CORTEX_DEV_NODE1 P3: P4: P5: P6: P7: P8:
LPMAIN       DIRECT STARTNWVMS.COM
LPMAIN       DIRECT START_ARSAP.COM
LPMAIN       DIRECT START_RJE.COM
      Enabled on All Nodes
      P1:''F$TRNLNM("RJE_NODE")' P2: P3: P4: P5: P6: P7: P8:
LPMAIN       DIRECT TDMSTRTUP.COM
LPMAIN       DIRECT VTXSTRTUP.COM
LPMAIN       DIRECT VTX_TPAT_STARTUP.COM
      Enabled on MYVAX1,MYVAX2,MYVAX3
      P1: P2: P3: P4: P5: P6: P7: P8:
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) VMS$LAYERED.DAT

```
LPBETA       BATCH  START_BATCH_QUEUES.COM
LPBETA       DIRECT DECW$STARTUP.COM
      Enabled on MYVAX1,MYVAX2
      P1: P2: P3: P4: P5: P6: P7: P8:
LPBETA       DIRECT DEFINE_FORMS.COM
LPBETA       DIRECT FLUSH_STARTUP.COM
LPBETA       DIRECT MANMAN_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:SYS$STARTUP:MMSTARTUP.COM P2:MANMAN_NODES P3: P4: P5: P6: P7: P8:
LPBETA       DIRECT SHOWCUSER_SETUP.COM
LPBETA       DIRECT SPM_COLLECT_TUNE.COM
LPBETA       DIRECT STARTUP$INTERACTIVE_LOGINS.COM
      Enabled on All Nodes
      P1: P2: P3: P4: P5: P6: P7: P8:
LPBETA       DIRECT START_ECD.COM

END          DIRECT ORACLE_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:ORACLE_STARTUP.COM P2:RS1_ORACLE_NODES P3:SUBMIT P4:/USER=ORACLE P5: P6: P7: P8:
END          BATCH  SMARTSTART_ALTERNATE.COM
END          BATCH  SQL$STARTUP.COM
END          BATCH  START_DDES.COM
END          BATCH  START_PRINT_QUEUES.COM
      Enabled on All Nodes
      P1:ALL P2: P3: P4: P5: P6: P7: P8:
END          BATCH  TEST_FOR_PROCESS.COM
END          BATCH  TSMSTARTUP.COM
END          DIRECT CALC$START.COM
END          DIRECT CATTS_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:CATTS_STARTUP.COM P2:CATTS_NODES P3: P4: P5: P6: P7: P8:
END          DIRECT CMSSTARTUP.COM
END          DIRECT CORTEX_DEV_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:CORTEX_DEV_STARTUP.COM P2:CORTEX_DEVELOPMENT_NODES P3: P4: P5: P6: P7: P8:
END          DIRECT CRS_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:CRS_STARTUP.COM P2:CRS_NODES P3:SUBMIT P4:/user=cortex_crs P5: P6: P7: P8:
END          DIRECT DISABLE_SEC_MSG.COM
END          DIRECT DQS$STARTUP.COM
END          DIRECT HELP_LIBRARY_STARTUP.COM
END          DIRECT IDLE_STARTUP.COM
END          DIRECT KEYPAK_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:KPAK_START.COM P2:KEYPAK_NODES P3: P4: P5: P6: P7: P8:
END          DIRECT LPS$LN03R$STARTUP.COM
END          DIRECT MTBSTART.COM
END          DIRECT NETWARE_STARTUP.COM
END          DIRECT NOTES$STARTUP.COM
END          DIRECT PATORD_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:PATORD_STARTUP.COM P2:PATORD_NODES P3:SUBMIT P4:/user=cortex_pat P5: P6: P7: P8:
END          DIRECT PQM_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:SYS_AUX1:[PQM_EF.COM]START.COM P2:PQM_NODES P3: P4: P5: P6: P7: P8:
END          DIRECT RS1_STARTUP_SHELL.COM
      Enabled on All Nodes
      P1:SYS_APP0:[RS1R3]RS1INS.COM P2:RS1_ORACLE_NODES P3: P4: P5: P6: P7: P8:
END          DIRECT SNALU62$STARTUP.COM
END          DIRECT SNATE$STARTUP.COM
END          DIRECT SNA_GWY.COM
END          DIRECT START_BIOS.COM
END          DIRECT START_CRAYSTN.COM
END          DIRECT TAM_STARTUP_SHELL.COM
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) VMS$LAYERED.DAT

```
      Enabled on All Nodes
      P1:START_TAM.COM P2:PD_NODES P3: P4: P5: P6: P7: P8:
END           DIRECT VPA$STARTUP.COM
END           BATCH  CRCC_AUDIT
END           BATCH  CRCC_AUDIT.COM
END           DIRECT STARTATALK.COM
```

2.2 REDEFINE_SYS$STARTUP.COM

Example 2–2 REDEFINE_SYS$STARTUP.COM

```
$! REDEFINE_SYS$STARTUP.COM    Richard J. Faust     3-JUL-1989
$!
$!      The logical name SYS$STARTUP is defined at system startup time
$!      by SYS$SYSTEM:STARTUP.COM.  Files which are included in the
$!      startup database must reside in a directory included in the
$!      SYS$STARTUP search list.  This procedure is used to expand the
$!      SYS$STARTUP search list and increase flexibility for locating
$!      system startup files.
$!
$!      A particular advantage of redefining SYS$STARTUP is that files
$!      will not have to be relocated to the default SYS$STARTUP
$!      directories at sites which are implementing the STARTUP utility
$!      for the first time.  Further, maintenance of startup files may
$!      be simplified on clusters which have multiple system disks.
$!
$!      This procedure is written to redefine SYS$STARTUP using the same
$!      qualifiers as STARTUP.COM.  This procedure will not have to be
$!      modified if DEC changes the characteristics of the SYS$STARTUP
$!      logical name.
$!
$!      REQUIRED PRIVILEGES:  SYSNAM, CMEXEC
$!
$! Revision history:
$!  3-JUL-1989 Rich Faust
$!      Initial version.
$! 27-JUL-1989 Rich Faust
$!      Put SITE_STARTUP at the end of the search list rather than the
$!      beginning.
$!
$!==============================================================================
$!
$!      Perform setup
$!
$ write sys$output ""
$ facility           = f$parse(f$environment("procedure"),,,"name")
$ say               := "write sys$output ""%''facility', """","
$ say "Starting at ''f$time()'"
$ required_privs = "SYSNAM,CMEXEC"
$!
$!      Check for privileges.
$!
$ priv_states = f$setprv(required_privs)
$ if .not. f$priv(required_privs)
$ then
$       say "''required_privs' required"
$       goto finished
$ endif
$!
$!      Determine logical name characteristics.
$!
$ access_mode = f$trnlnm("sys$startup","lnm$system_table",,,,"access_mode")
$ concealed   = f$trnlnm("sys$startup","lnm$system_table",,,,"concealed")
$ terminal    = f$trnlnm("sys$startup","lnm$system_table",,,,"terminal")
$ confine     = f$trnlnm("sys$startup","lnm$system_table",,,,"confine")
$ no_alias    = f$trnlnm("sys$startup","lnm$system_table",,,,"no_alias")
$ translation_attributes = ""
$ name_attributes = ""
$!
$ if confine then name_attributes = ",confine"
$ if no_alias then name_attributes = name_attributes + ",no_alias"
```

Example 2–2 Cont'd. on next page

Example 2–2 (Cont.) REDEFINE_SYS$STARTUP.COM

```
$ if name_attributes .nes. "" then -
    name_attributes = "/name_attributes=(" + name_attributes + ")" - ","
$!
$ if concealed then translation_attributes = ",concealed"
$ if terminal then translation_attributes = translation_attributes + ",terminal"
$ if translation_attributes .nes. "" then -
    translation_attributes = "/trans=(" + translation_attributes + ")" - ","
$!
$!      Build the current search list
$!
$ max_index = f$trnlnm("sys$startup","lnm$system_table",,,,"max_index")
$ index = 0
$ search_list = ""
$!
$next_index:
$ equivalence_name = f$trnlnm("sys$startup","lnm$system_table",index)
$ if equivalence_name .nes. "SITE_STARTUP"
$ then
$       search_list = search_list + equivalence_name + ","
$ endif
$ index = index + 1
$ if index .le. max_index then goto next_index
$!
$ search_list = search_list + "SITE_STARTUP"
$ define_ /system /'access_mode 'translation_attributes 'name_attributes -
        sys$startup 'search_list
$ show logical /full sys$startup
$!
$!      Reset privileges and verify state.
$!
$finished:
$ x = f$setprv(priv_states)
$ say f$fao("Finished at !AS!/",f$time())
$ exit
```

2.3 STARTUP_SHELL.COM

Example 2–3 STARTUP_SHELL.COM

```
$! STARTUP_SHELL.COM    Richard J. Faust    14-MAR-1989
$!
$! This file is a workaround.  The SYSMAN startup database utility does
$! not allow a variable to be used to specify nodes on which to execute
$! procedures and programs.  'product'_startup_shell should be in the
$! SYSMAN startup database and call this file using the following
$! parameters:
$!
$! P1 = filename of command procedure to execute
$! P2 = list of nodes which should execute (may be a symbol, logical name,
$!      or comma-separated list)
$! P3 = SUBMIT if the file specified by P1 should be submitted as a batch
$!      job.
$! P4-P8 are available as parameters for batch jobs.
$!
$! The mode for 'product'_startup_shell should be DIRECT or SPAWN if
$! P2 is a symbol; the symbol would be undefined in batch mode.
$!
$! Revision history:
$! 18-MAR-1989 RJF Version 1.0
$! 21-SEP-1989 RJF add some comments
$!
$!==============================================================================
$!
$ set noon
$ facility   = f$parse(f$environment("procedure"),,,"name")
$ say       := "write sys$output ""%''facility'-"","
$ this_node := 'f$getsyi("nodename")
$ delimiter  = ","
$ node_list  = ""
$ started_product = 0
$ saved_default = f$environment("default")
$ set default sys$startup
$!
$!==============================================================================
$!
$! Make sure the file to execute exists and is a command procedure.
$!
$ if f$search("''p1'") .eqs. ""
$ then
$       if p1 .eqs. "" then p1 = "NONAME"
$       say "F-FILNOTFOU, ''p1' not executed: file not found"
$       goto finished
$ endif
$ if f$parse(p1,,,"type") .nes. ".COM"
$ then
$       say "F-NOTCOMPRO, ''p1' is not a command procedure"
$       goto finished
$ endif
$ product = f$parse(p1,,,"name") - "_STARTUP_SHELL"
$!
$!==============================================================================
$!
$! Define the node list
$!
$ node_list = f$trnlnm(p2)
$ if node_list .eqs. ""
$ then
$    if f$type('p2) .eqs. "STRING" then node_list = 'p2
$    if node_list .eqs. ""
```

Example 2–3 Cont'd. on next page

Example 2–3 (Cont.) STARTUP_SHELL.COM

```
$       then
$           node_list = p2
$       endif
$ endif
$ if node_list .eqs. ""
$ then
$    say "F-NONODES, P2/node list undefined"
$    goto finished
$ endif
$ node_list = f$edit(node_list,"upcase")
$!
$!==============================================================================
$!
$! Determine if this node should execute the file.
$!
$ write sys$output ""
$ say "I-ATTSTART, attempting startup of ''product'"
$ element = 0
$next_node:
$ node = f$element(element,delimiter,node_list)
$ if node .eqs. delimiter then goto finished
$ if this_node .eqs. node
$ then
$    started_product = 1
$    if f$edit(p3,"upcase") .eqs. "SUBMIT"
$    then
$       say "I-SUBMIT, submit ''product'..."
$       submit 'p1 'p4 'p5 'p6 'p7 'p8
$       goto finished
$    else
$       say "I-EXECUTE, execute ''product'..."
$       @'p1 "''p3'" "''p4'" "''p5'" "''p6'" "''p7'" "''p8'"
$       goto finished
$    endif
$ else
$    if .not. f$getsyi("cluster_member",node) then -
$       say "W-NOTCLUMEM, node ''node' is not a cluster member"
$    element = element + 1
$    goto next_node
$ endif
$finished:
$ set def 'saved_default
$ if started_product
$ then
$    say f$fao("I-STARTED, !AS started on !AS!/",product,this_node)
$ else
$    say f$fao("I-NOSTART, !AS not started on !AS!/",product,this_node)
$ endif
$ exit
```

Example 2–4 RS1_STARTUP_SHELL.COM

```
$! RS1_STARTUP_SHELL.COM    Richard J. Faust    15-MAR-1989
$!
$! Call STARTUP_SHELL to start RS1.
$!
$! Revision history:
$! 15-MAR-1989 RJF Version 1.0
$!
$!==============================================================================
$!
$ @sys$startup:startup_shell - !
        "''p1'" - ! name of startup file
        "''p2'" - ! node list (symbol, logical name, comm-separated list)
        "''p3'" - ! p1 for startup file
        "''p4'" - ! p2 for startup file
        "''p5'" - ! p3 for startup file
        "''p6'" - ! p4 for startup file
        "''p7'" - ! p5 for startup file
        "''p8'"   ! p6 for startup file
```

2.4 OA_STARTUP.COM

Example 2–5 OA_STARTUP.COM

```
$! OA_STARTUP.COM    Richard J. Faust    14-MAR-1989
$!
$! This file guarantees that startup files critical to the ALL-IN-1
$! environment are executed in the proper order at system startup time.
$! DECnet should already be started.  Batch queues should not be started
$! until the OA environment is available since there may be jobs which
$! are dependent on that environment.
$!
$! Revision history:
$! 16-MAR-1989 RJF Version 1.0
$!
$!==============================================================================
$!
$ set noon
$ @sys$manager:start_mailbus
$ @sys$manager:start_a1
$ @oa_site:a1_site_startup
$ @sys$startup:startup$interactive_logins
```

2.5 STARTUP$INTERACTIVE_LOGINS.COM

The following line is included in SYS$MANAGER:SYSTARTUP_V5.COM to prevent interactive logins by users who do not hold OPER privilege:

```
$ startup$interactive_logins == 0
```

Section 2.5 is called from OA_STARTUP.COM and also listed as a separate entry in the startup database; there may be nodes or situations in which OA_STARTUP is not executed. The multiple presence of this procedure guarantees that interactive logins will be enabled as soon as the ALL-IN-1 environment is available or during phase LPBETA, whichever comes first.

Example 2–6 STARTUP$INTERACTIVE_LOGINS.COM

```
$! STARTUP$INTERACTIVE_LOGINS.COM    Richard J. Faust    14-MAR-1989
$!
$! Enable interactive logins and notify users that the system is up.
$!
$! Revision history:
$! 18-MAR-1989 RJF Version 1.0
$!
$!==============================================================================
$!
$ set noon
$ this_node = f$getsyi("nodename")
$!
$ if this_node .eqs. "MYVAX1"
$ then
$    startup$interactive_logins = 10
$ else
$    startup$interactive_logins = 256
$ endif
$!
$ set logins /interactive='startup$interactive_logins
$ reply /all /bell /node='this_node' -
"''startup$interactive_logins' logins enabled on ''this_node' at ''f$time()'..."
```