North American Response Centers

HP 3000 APPLICATION NOTE #28

The Startup State Configurator

HEWLETT PACKARD

May 15, 1987
Document P/N 5958-5824R2720

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

HP 3000 APPLICATION NOTES *are published by the North American Response Centers twice a month and distributed with the Software Status Bulletin. These notes address topics, where the volume of calls received at the Centers indicates a need for addition to or consolidation of information available through HP support services. You may obtain previous notes (single copies only, please) by returning the attached Reader Comment Sheet listing their numbers. Notes #1 and #2 have been superseded by #4 and #13 respectively.*

Note #	*Published*	*Topic*
3	4/01/86	*HP 3000 Plotter Configuration Guide*
4	4/15/86	*HP 3000 Printer Configuration Guide - Revised*
5	5/01/86	*MPE System Logfile Record Formats*
6	5/15/86	*HP 3000 Stack Operation*
7	6/01/86	*COBOL II/3000 Programs: Tracing Illegal Data*
8	6/15/86	*KSAM Topics: COBOL's Index I/O; File Data Integrity*
9	7/01/86	*Port Failures, Terminal Hangs, TERMDSM*
10	7/15/86	*Serial Printers - Configuration, Cabling, Muxes*
11	8/01/86	*System Configuration or System Table Related Errors*
12	8/15/86	*Pascal/3000 - Using Dynamic Variables*
13	9/01/86	*Terminal Types for HP 3000 HPIB Computers - Revised*
14	9/15/86	*Laser Printers - A Software and Hardware Overview*
15	10/01/86	*FORTRAN Language Considerations - A Guide to Common Problems*
16	10/15/86	*IMAGE: Updating to TurboIMAGE & Improving Data Base Loads*
17	11/01/86	*Optimizing VPLUS Utilization*
18	11/15/86	*The Case of the Suspect Track for 792X Disc Drives*
19	12/01/86	*Stack Overflows: Causes & Cures for COBOL II Programs*
20	1/01/87	*Output Spooling*
21	1/15/87	*COBOLII and MPE Intrinsics*
22	2/15/87	*Asynchronous Modems*
23	3/01/87	*VFC Files*
24	3/15/87	*Private Volumes*
25	4/01/87	*TurboIMAGE: Transaction Logging*
26	4/15/87	*HP 2680A, 2688A Error Trailers*
27	5/01/87	*HPTrend: An Installation and Problem Solving Guide*

The Startup State Configurator

The Startup State Configurator is a feature (introduced with MPE version G.01.00, "T-MIT") which allows a System Manager to specify exactly what should happen every time the system starts up.

Prior to the Startup State Configurator, when MPE was started up, default values were automatically assigned for various system parameters, such as jobfence, joblimit, or outfence. Many users wanted to change these parameters to fit their individual needs so they assigned a logon UDC to OPERATOR.SYS. However, this UDC had to execute each time OPERATOR.SYS logged on allowing undesirable activities to possibly take place before the system was properly configured. The Startup State Configurator alleviates these problems.

To use the Startup State Configurator, the System Manager places commands corresponding to each type of startup desired into a special file named SYSSTART.PUB.SYS. The sections of SYSSTART.PUB.SYS begin with a header line so that they will be executed only for the type of startup being performed. In addition, you can include commands which execute every time the system is started up, regardless of the type of startup being performed.

We receive many calls at the Response Center on the format of, and security issues with, the file used for the Startup State Configurator. When the Configurator was first introduced an article was published in *The Communicator* (Volume 2, Issue 1). This note is an updated version of that article with additional information on file security.

This note describes the format, creation and installation of SYSSTART.PUB.SYS and operation of the Startup State Configurator, gives an example of a SYSSTART.PUB.SYS file, and outlines special considerations.

FORMAT OF SYSSTART.PUB.SYS

The SYSSTART file can contain up to 6 sections. A section consists of a header, a body and a section terminator. None of the sections is specifically required, but, if present, each must be in the proper format as discussed below. The sections can be in any order and the order of the sections in the file is not related to the order in which the sections are processed.

The SYSSTART Section Header

The SYSSTART section header is one line long and can contain one of the following six keywords:

```
STARTUP
WARMSTART
COOLSTART
UPDATE
COLDSTART
RELOAD
```

The SYSSTART Section Body

The body of the SYSSTART file section contains MPE commands to be executed, one per line. Continuation records within the file are allowed, and are produced by terminating the record to be continued with an '&'. A subset of MPE commands are available for use within the Startup State Configurator.

COMMANDS ALLOWED.

:ALLOW	:HEADON	:STARTCACHE
:ALTLOG	:IMFCONTROL	:STARTSESS
:AUTOALLOCATE	:JOBFENCE	:STARTSPOOL
:CACHECONTROL	:JOBPRI	:STOPCACHE
:COMMENT	:JOBSECURITY	:STOPSPOOL
:CONSOLE	:LDISMOUNT	:STREAM
:DEALLOCATE	:LIMIT	:STREAMS
:DISALLOW	:LMOUNT	:SUSPENDSPOOL
:DISCRPS	:LOG	:TAKE
:DOWN	:MPLINE	:TELL
:DOWNLOAD	:MRJECONTROL	:TELLOP
:DSCONTROL	:NETCONTROL	:TUNE
:FOREIGN	:NSCONTROL	:UP
:GIVE	:OUTFENCE	:VMOUNT
:HEADOFF	:REFUSE	

NOTE

The commands which can be used in the SYSSTART file must be able to be executed via the COMMAND intrinsic, must not prevent the system from starting up and must not compromise system security in any way. Any password required for execution of the commands in SYSSTART.PUB.SYS must be included in the SYSSTART file.

The SYSSTART Section Terminator

A section of the SYSSTART file is terminated with an uncontinued record starting with an '*', with a keyword indicating the start of another section of the file, or by reaching the end of the file.

CREATION AND INSTALLATION OF SYSSTART.PUB.SYS

The SYSSTART.PUB.SYS file can be produced by most HP 3000 editors including EDITOR and TDP/3000. The file must be created by the MANAGER.SYS, must be in PUB.SYS, must have fixed length ASCII records, and may be numbered or unnumbered. A lockword on the file is not allowed. An example of a file appears below in the section entitled *SAMPLE SYSSTART.PUB.SYS FILE*. The information in the SYSSTART.PUB.SYS file is processed after the date and time prompts appear, but before a session is

signed on to the console. If you wish to prevent the use of the SYSSTART file, you can :RENAME or :PURGE the SYSSTART.PUB.SYS file before the system is shut down.

If no SYSSTART.PUB.SYS file can be found, no processing occurs. If a SYSSTART.PUB.SYS file is found, but an error is encountered, an error message is sent to the console and the system continues to come up.

If a valid SYSSTART.PUB.SYS file is found, the following steps will occur:

1. The type of requested startup is determined (that is, WARMSTART, UPDATE, etc.).

2. The SYSSTART.PUB.SYS file is searched for a section headed "STARTUP" and for a section whose header corresponds to the type of startup determined in step 1.

3. The commands in the found section(s), if any, are executed. If a STARTUP section exists, it is processed first. Then the section corresponding to the startup type, if it exists, is processed. Messages indicating the beginning of the sections are sent to the console.

When a section of the SYSSTART.PUB.SYS file is processed, the following steps are executed. The steps are repeated until the end of a section of the SYSSTART file is encountered:

1. Read a command from the SYSSTART file.

2. Remove any passwords or lockwords from this command. Retain the unedited command line for later use.

3. Echo the edited command to the console.

4. Check that the command is in the above list of allowed commands. If the command is not allowed, print a message on the console and return to step 1.

5. Pass the unedited command to the COMMAND intrinsic for execution.

6. If an error occurs, print the appropriate error message on the console. Any error in the file can be corrected after startup is complete.

7. Return to step 1.

NOTE

It is important to remember that the commands in SYSSTART are executed as if MANAGER.SYS had typed them from a session. For this reason, SYSSTART.PUB.SYS must be created by MANAGER.SYS and passwords are not echoed. It is important to understand file system security before implementing the Startup State Configurator as describe here. If the SYSSTART file contains sensitive information such as passwords, the :ALTSEC command should be used to alter the security on the SYSSTART file so that other users cannot read the contents. For example,

```
:ALTSEC SYSSTART.PUB.SYS;(R,W,A,L,X:CR)
```

This limits access to the creator only, in this case, MANAGER.SYS.

SAMPLE SYSSTART.PUB.SYS FILE

The following is an example of a valid Startup State file:

```
1   STARTUP
2   STREAMS 10
3   ALLOW @.@;COMMANDS=REPLY
4   ALLOCATE COBOLII.PUB.SYS
5   LIMIT 4,16
6   JOBFENCE 4
7   OUTFENCE 5
8   OUTFENCE 12;LDEV=14
9   COMMENT A continuation record follows:
10  DSCONTROL 90;OPEN;RETRY=10;&
11  TRACE,ON,ALL,,60,WRAP
12  ************************************************
13  WARMSTART
14  STARTSESS 20;MGR.ACCT;HIPRI;NOWAIT
15  STARTSESS 21;USER.ACCT
16  STARTSESS 22;USER.ACCT
17  STARTSESS 23;USER.ACCT
18  ************************************************
19  COOLSTART
20  STARTSESS 20;MGR.ACCT;HIPRI;NOWAIT
21  STARTSESS 21;USER.ACCT
22  STARTSESS 22;USER.ACCT
23  STARTSESS 23;USER.ACCT
24  ************************************************
```

This example defines a STARTUP section to be invoked every time the system is started. In addition, a WARMSTART or a COOLSTART will cause MGR.ACCT to log on to logical device 20 (presumably the console) and will create sessions for USER.ACCT on three other terminals. These users may have LOGON UDCs that invoke an application program. As a result, when the system is started from the disc, four sessions will be created and could enter application programs.

Note that in this example, there are no definitions for UPDATE, RELOAD, or COLDSTART. If the system comes up with any of these types of starts, only the commands listed in the STARTUP section will be executed.

SPECIAL CONSIDERATIONS

- Since the SYSSTART file must be present when the system is started, the NULL and ACCOUNTS options of a RELOAD do not work. The SYSSTART file is not present on the system during these two options.
- The startup processing cannot be stopped by the operator at any time. The [BREAK] key is ignored.

- Commands are not checked in advance for correct syntax. You should be sure that the syntax of each command is correct when the SYSSTART file is created. If the syntax of a command is incorrect, or if the command is not allowed (see *Commands Allowed* above), a run-time error message will appear on the console and execution of the SYSSTART file will continue.

- Since a startup file can contain STARTSESS commands, it is possible to start a session on the console. If this is requested in the SYSSTART file, OPERATOR.SYS will not automatically log on at the console.

- The default STARTSESS command will wait for a [RETURN] before it completes the log on process. Since having an unexpected read presented at the console just after the system has started up could be confusing, the NOWAIT option of :STARTSESS should be used for any STARTSESS which is to be directed to the console from the SYSSTART file. The console terminal should be at the speed specified in the I/O configuration for correct operation in this case.

READER COMMENT SHEET

North American Response Centers
HP 3000 Application Note #28: SYSSTART FILE
RC Questions & Answers (May 15, 1987)

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

	AppNote	RC Q&A
Is this publication applicable to your site?	☐ Yes ☐ No	☐ Yes ☐ No
Are the concepts and wording easy to understand?	☐ Yes ☐ No	☐ Yes ☐ No
Would you like to see additional Notes on this subject?	☐ Yes ☐ No	☐ Yes ☐ No

Comments and/or suggestions for future Application Notes:

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

FROM: **Date** _____

Name _____

Company _____

Address _____

_____

_____

FOLD FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1710, SANTA CLARA, CA

POSTAGE WILL BE PAID BY ADDRESSEE

Application Note / RC Q & A Comments
Hewlett-Packard Western Response Center
3300 Scott Boulevard
Santa Clara, CA 95054

FOLD FOLD

HEWLETT PACKARD RESPONSE CENTER QUESTIONS & ANSWERS

HP 3000 Questions Commonly Received by the North American Response Centers

Q. How can I obtain information on what TELESUP utilities are on my system?

A. There is a group in the TELESUP account call DOC. Within this group there is an explanation of each TELESUP utility. To access this information, enter the following commands:

```
:HELLO [sessionname,] MGR.TELESUP,DOC
:RUN LIST.PUB.TELESUP

     >>LIST @.DOC.TELESUP
```

To send the documentation to a device other than your terminal (the terminal is the default):

```
>>OUTPUT n   (where 'n' is the ldev of a printer)
```

To send documentation to your terminal:

```
>>OUTPUT TERM
```

For an explanation of the LIST command:

```
>>HELP LIST
```

Q. How can you match a character followed by blanks in QUERY?

A. If a blank follows a character in a MATCH statement, it will be ignored by QUERY and only the character will be matched. For example:

```
MATCH "XY"
```

This will find any entry beginning with "XY". To selectively find an entry with blanks in the third and fourth positions, use a small letter 'b' in the place of the blanks:

```
MATCH "XYbb"
```

Note that the 'b' can be used to find blanks anywhere within a pattern to be matched (not just blanks at the end of a character string).

more...

Q. In Query, how can I count control breaks? That is, how can I count the number of groups my data was sorted into?

A. There is a way to do this in Query, although it is one of those "tricks" that is far from intuitive and requires a lot of effort to figure out. The following is an example of how to count control breaks:

```
S1,<sort item>              << sort on an item >>
   ...
R2,L,"1"
R2,S,R1                     << these four statements   >>
R3,A,R2                     << do the actual work      >>
R1,L,"1"
   ...
T1,R1                       << reset R1 at the control break >>
   ...
TF,"TOTAL GROUPS:",20       << print the results >>
TF,R3,35
   ...
etc.
```

Here is a step by step description of what happens:

Action	R1	R2	R2	Comments
Initially	0	0	0	
R2,L,"1"	0	1	0	Record 1
R2,S,R1	0	1	0	
R3,A,R2	0	1	1	
R1,L,"1"	1	1	1	
R2,L,"1"	1	1	1	Record 2
R2,S,R1	1	0	1	
R3,A,R2	1	0	1	
R1,L,"1"	1	0	1	
T1,R1	0	0	1	Control break 1 !!!
R2,L,"1"	0	1	1	Record 1
R2,S,R1	0	1	1	
R3,A,R2	0	1	2	
R1,L,"1"	1	1	2	
R2,L,"1"	1	1	2	Record 2
R2,S,R1	1	0	2	
R3,A,R2	1	0	2	
R1,L,"1"	1	0	2	
TF,R3,35	1	0	2	Control break 2 !!! (when we do final totals)

The key to the algorithm is subtracting R1 from R2. R1 is usually one, *except* when we've just initialized, or after a control break, and then R1 is zero. In this case, the difference of R2-R1 is one, and this gets added to R3, our accumulator.