SHARE PROGRAM LIBRARY AGENCY

PROGRAM NUMBER

05I022

University of Miami

1365 MEMORIAL DRIVE - CORAL GABLES, FLORIDA
(305) - 284-6257

SHARE PROGRAM LIBRARY SUBMITTAL FORM

SHARE PROGRAM LIBRARY AGENCY
Triangle Universities Computation Center
Post Office Box 12076
Research Triangle Park, North Carolina
27709 USA

SPLA CONTROL NUMBER: 183

This form should be completed and submitted with the program package to the SHARE Program Library Agency at the address shown above. Standards and instructions for submitting programs are in the "SHARE Reference Manual".

(1) Program Number (to be filled in by SPLA)..... 370D-05.1.022

(2) System Type (machine)..... S/370 Advanced Function

(3) Search Key..... VS1 HASP

(4) Programming Systems/Languages..... OS/VS Assembler and VS1 Rel 4.0

(5) Author's Name and Address..... Jim Allen
Computation Center
Duke University
Durham, N. C. 27706

(6) Direct Technical Inquiries to Name & Address (if different than Author)

(7) Title of Program..... VS1 HASP

(8) Submitter's Installation Membership Code..... DU

(9) Submitter's Own Program Identification and Suffix(Optional)..

(10) Primary Subject Code..... 03.4

(11) Minimum System Requirements Capability to Run VS1 Release 4.0

(12) New or Revision Code (if revision, show prior Program Number in Item 1) R

(13) Year Completed..... 1976

(14) Date of Submittal..... June 1, 1976

(15) Documentation (number of original pages submitted)..... 12

(16) Abstract (should contain sufficient information for a reader to determine the value of the program). Listed on the reverse side of this form are subjects which may serve as a guide for a descriptive abstract.

Revised 4/74

SHARE PROGRAM LIBRARY SUBMITTAL FORM

Subject Guide:

a. Purpose
b. Programming Language used
c. Version and modification level or release number
d. Field of application
e. Type of routine (main program, subroutine, etc.)
f. Specific description of machine requirements

ABSTRACT

VS1 HASP is a modification to HASP II V4.0 which provides the basic capability to run HASP on a VS1 Release 4.0 host system. The goal of the modification is to provide a VS1-HASP interface which is equivalent to the formal interface between VS2 Release 1 and HASP V4.0. Thus the HASP system operates as a job entry subsystem for VS1 with no loss of HASP function. The VS1-HASP interface is implemented by using SVC table intercepts and SMF exits so that the interface is independent of the VS1 host as much as possible. The two major features of the interface are an interface between HASP console services and VS1 Multiple Console Support (MCS), and the interface between HASP pseudo device I/O services and the VS1 Input/Output Supervisor. The bulk of the modifications are isolated into HASPINTF, a new assembly.

This modification includes fixes to all known bugs, the integration of PTF OY09762, and support for multiple concurrent Reader/Interpreters.

(Please attach additional pages if necessary).....Total pages attached____

Permission to Publish

"I hereby give the SHARE Program Library Agency permission to reprint, reproduce, and distribute this program."

(17) Signature of Submitter and Date Jim Allen June 1, 1976

(18) Signature of Installation Addressee

MAGNETIC TAPE KEY

This volume contains two files and three tape marks as follows:

File 1

Machine Readable Documentation
EBCDIC (120 graphics)
686 Logical records
RECFM: FBA
LRECL: 86
BLKSIZE: 1720
T/M

File 2

Source Deck
EBCDIC
2792 logical records
RECFM: FB
LRECL: 80
BLKSIZE: 1600
T/M
T/M

June 1, 1976 SPLA 370D-05.1.022

VS1 HASP

INTRODUCTION

VS1 HASP is a modification to HASP II V4.0 which provides the basic capability to run HASP on a VS1 Release 4.0 host system. The goal of the modification is to provide a VS1-HASP interface which is equivalent to the formal interface between VS2 Release 1 and HASP V4.0. Thus the HASP system operates as a job entry subsystem for VS1 with no loss of HASP function. The VS1-HASP interface is implemented by using SVC table intercepts and SMF exits so that the interface is independent of the VS1 host as much as possible. The two major features of the interface are an interface between HASP console services and VS1 Multiple Console Support (MCS), and the interface between HASP pseudo device I/O services and the VS1 Input/Output Supervisor. The bulk of the modifications are isolated into HASPINTF, a new assembly.

This modification is not guaranteed to be completely general. No formal tests have been conducted by the author to ascertain that the modification works with all VS1 options and all HASP options.

Installations who wish to install VS1 HASP must determine for themselves the viability of the modification. Users of the modification must assume total responsibility for maintenance. The modification should be a good base for installations which have a requirement to run HASP with VS1.

METHOD OF OPERATION

The documentation is going to be sketchy at best. Installations which plan to use this modification will have to study it in detail because they can expect no support from the author. The description that follows is an overview that should help in early attempts at understanding the code. As always, the code itself contains the detailed documentation in extended comments.

Initialization. During HASP initialization the HASPSVC is issued to return a handful of important addresses from the nucleus. This enables the HASPIOVB routine to locate the SVC table and to overlay the VS1 entries for SVCs which pertain to console service and IOS entry points. This code is similar to that provided in the earlier versions of HASP. The HASPSVC also stores the $HVT address into the CVTHJES field. This field is used by the SMF exits and R/I interface module to communicate with HASP.

HASPIOVB performs some miscellaneous traps. If the &AUTORDR option is selected, then the first twelve bytes of IEFUBAT1, the JES "hot

reader" attention appendage, is overlaid to give control to the HASP attention appendage. A word in IEECMWSV, the MCS WQE service routine, is overlaid to implant a call of the HASPSVC. HASPSVC will enter the $WTOSVC2 exit to edit the WQE. The entry point in the JES communication table of the JES job list manager is overlaid to point to the $JLMRT routine in HASPINTF. It will receive control at JES job enqueue time and pass control to the real job list manager after editing the job list manager parameter list.

HASP issues a START command for a JES reader which is used to pass JCL images across to VS1 for execution. Instead of ATTACHing the HASPWTR module, a JES writer is STARTed to a pseudo device in order to retrieve SMB output.

These traps and operator commands are reset at $PHASP so that HASP may be withdrawn from the system. In addition, a P INIT.PO command is issued to negate the effect of the S INIT.PC command which is assumed to be in the JCL for the HASP job. This causes HASP to be loaded at the same virtual storage address each time it is invoked.

MCS Console Support. The console support is one of the two major areas of development necessary for VS1 HASP. VS2 provides three formal exits into HASP for console support. The first is an exit in SVC 34 so that HASP may process console input. This routine is provided by VS1 HASP by trapping on the LINK/XCTL to IGC0403D. The second is the exit provided by WTO/WTL so that HASP may create its HASP JOB LOG. This interface is provided by trapping on the

occurrence of a WTO/WTL supervisor call. The last exit in VS2 is provided by a late load of WTO/WTL. This is an exit that is difficult to provide without a source change to WTO/WTL. The solution provided here is to overlay some code in IEECMWSV with a call to the HASPSVC. The HASPSVC then examines the communications task work queue and passes recently added WQEs to $WTCSVC2. The method for providing this exit is weak. However, it is not required for the system to function. It just time-stamps messages, and, when possible, provides a job number for the text. These features are attractive, so the exit is provided. $WTOSVC2 is entered under the TCB of the communication task; hence the job association subroutine rarely matches the message with a HASP controlled job. Users who find this unattractive are advised that IGC0003E (microfiche name IEEMFWTO) has a large patch area that may be used to provide a direct entry into $WTOSVC2.

Job Execution. The entry points to the Input/Output Supervisor (IOS) have been stolen by HASP to provide the basic spooling function. Dummy SMF routines provide additional exits so that HASP can monitor the progress of the job through various stages (job initiation, step initiation, etc.) of execution. The following is a description of a job which is executed under VS1.

When a job is selected for initiation, HASP passes the job's JCL to a JES reader in the same fashion as it does for an OS R/I. The SMF User Input Validation exit is entered in HASP where the SMF User Identification Field is set to the EBCDIC representation of the

JCTDSKEY so that all SMF records generated for the job can be ccalesced.

The $JLMRT routine gets control when JES readers and writers call the job list manager. It negates the effect of TYPRUN=HOLD and internally forces the job CLASS indicated in the HASP Partition Information Table. Note that TYPRUN=HOLD and CLASS are supported by HASP now. The interface is a bit too intimate with JES, but it should be reliable. If it is not, then TYPRUN can be negated by disguising the HOLD operand on the card image passed to JES (a la DD *) and correcting it when the image returns via the JES writer. This technique may not work too well with CLASS. It is possible to force a specific class (e.g. A) in order to direct the job to a set of initiators.

VS1 does not interpret the job until it is selected for initiation. At this time, the old R/I is "attached" as a subtask of the initiator and the job is interpreted. The HASP R/I appendage has been made re-entrant to support multiple R/Is. A major exit provided by VS1 HASP is the R/I appendage. The standard R/I appendage of MFT/MVT has been decommitted in VS1 (and even the code has been removed). VS1 HASP provides the exit via Linkage Editor CHANGE cards. The R/I JCL conversion module IEFVFA calls either IEFVJA (JCB card), IEFVEA (EXEC card), or IEFVDA (DD card) after converting the JCL to internal text. These three CSECTs have been renamed by Linkage Editor CHANGE cards to DUKVJA, DUKVEA, and DUKVDA, respectively. New routines are supplied

which enter the HASP XJCISCAN exit, then pass control to the original I/O routines. This is the only part of VS1 HASP which requires a modification to VS1. It is simple enough, however, to survive almost any PTF.

After the job gets into execution it is free to do regular pseudo I/O. The SVC table traps steal control from IOS and direct requests for pseudo devices to $EXCPSVC. The basic $EXCPSVC routine came through okay, and even works for ACB/RPL access methods (JES readers and writers). The TCBUSER field is used instead of the JSCB field provided by VS2.

When the job terminates, the SMB output is retrieved by using a JES writer which prints it on a pseudo device. The end of SMB output is signaled by $WTOSVC when it detects either an IEF049I ("job on device") or and IEF868I ("writer waiting for work") messages generated by the JES writer. This technique is used because IEFUJP, the SMF User Job Purge exit, is not entered if there are no SYSOUT datasets for the job.

Disabled Page Faults. Page faults which occur when the system is disabled from I/O and external interrupts are problematic in VS1. The system is forced to perform an I/O operation before the task can be dispatched again. VS1 programs disable the system for two reasons. The first is the case in which interrupts handlers are not re-entrant and can not tolerate a second interrupt while processing a first. This is the primary reason that it is possible to disable interrupts.

The second reason is to "allocate" some resource that is shared among a number of tasks. By disabling the system from interrupts, a task can maintain control of the CPU until updates are complete. Disabling the system for this reason is poor. It is routinely done because VS1 does not provide efficent "locking" facilities or the capability to pass messages between tasks.

Consider the consequences of suffering a disabled page fault in the second case. The system must perform a page-in, and perhaps a page-out. This means that the paging supervisor must be dispatchable. Any SVC that the paging task issues (EXCP, WAIT, POST, etc.) must take special precautions to handle page faults. Any task which runs enabled may be dispatched. If it attempts to disable the system, it must be forced to wait for the previous task to relinquish control before it is allowed to enter the disabled mode.

If programs disable the system by using the MODESET SVC, then the supervisor is aware of the attempt to disable the system, and will stack the SVC until the system is again enabled. The system is unaware of mode changes via the STOSM and STNSM instructions. Furthermore, it can not identify those tasks which will disable the system. VS1 allows each request block to specify how the supervisor should handle disabled page faults. Programs may request only enabled tasks may be dispatched ("supervisor lock") or that only the paging supervisor may be dispatched ("system lock"). VS1 HASP requests a system lock whenever a disabled page fault occurs. (This is the

standard request for VS1 modules which can get page faults).

There are many modifications spliced into HASP which are designed to reduce the frequency of incurring disabled page faults. It does not affect the logical accuracy of the program.

RESTRICTIONS

It seems that it would be worthwile for the author to comment on restrictions imposed by VS1 HASP. The items listed are just the ones which come to mind. Installations which plan to use VS1 HASP can expect to discover a few restrictions of their own. I am interested in receiving (preferably written) news about installations which have installed VS1 HASP, problems encountered, restrictions discovered, etc. I can not, however, provide maintenance, consultation, or whatever for the code.

The restrictions imposed by HASP documented in the HASP II V4.0 Systems Programmer's Guide (pages 164-166) still apply.

JES readers and writers will run concurrently with HASP. The JES "hot reader" facility, however, does not work when HASP is up.

HASP consumes the SMF exits to provide its interface with VS1. Users who desire to use SMF exits must provide exits into the HASP Vector Table for proper execution. Alternatively, HASP could be extended so that it would call the true user exits, like the HASPACCT task does for IEFUJP. The true user SMF exits would be linked with HASP. No formal tests have been conducted to determine if SMF records

have been adjusted properly to reflect the existence of HASP. Users of VS1 HASP should determine for themselves the accuracy of these records.

Installation Specified Selection Parameters (ISSP) is not supported. This is beyond the scope of the project.

HOLD=YES on a SYSOUT DD card is ignored.

TYPRUN=SCAN works, but its support could be improved. As it stands now, a job to be scanned is scheduled as a regular job. When it is selected for execution, the initiator bypasses execution in much the same way it would a job with a JCL error. A better way would be to assign a special class at HASPRDR time for TYPRUN=SCAN and to reserve a HASP initiator for these jobs.

Dynamic System Support (DSS) may not work properly with HASP (if it works properly at all). I don't know the reason, though I suspect there are some problems with the $TRACE service.

The Checkpoint/Restart Facility is not supported. The RESTART= keyword should not be used.

HASPGEN CONSIDERATIONS

The second file on the distribution tape contains the source of the modification. It is a HASPGEN deck, followed by five small source decks, and then some sample cataloged procedures and JCL. The source file contains about 2500 cards. The simplest thing to do is to punch the cards.

Installation of VS1 HASP requires an augmented VS1 SYSGEN and HASPGEN. The directions in the HASP II V4.0 Systems Programmer's Guide still apply to a VS1 HASP system, except that the two VS2 parameters &APG and &ISOSTCN are not pertinent. VS1 HASP adds the &INITSVC to the HASPGEN to specify the SVC number assigned to the HASPSVC. The VS1 SYSGEN requires the MCS and SMF options. The MCS option is required because the $WTOSVC2 exit is provided by overlaying a MCS routine. The SMF exits must be taken, but VS1 HASP does not require the SMF records to be written to the dataset. The additional requirements for a VS1 HASP generation follow below in roughly the order in which they should appear in the HASP Systems Programmer's Guide.

Included in the source materials are four trivial SMF source programs, that when installed into the VS1 system, provide the HASP-SMF interface. To install these programs into the VS1 system, assemble and link-edit according to the directions given in the "OS/VS System Management Facilities (SMF)" manual. The form number is GC35-0004-6. The assemblies require SYS1.HASPSRC as part of the SYSLIB concatenation, so the HASPGEN must be performed before this step.

A fifth source module is the HASP interpreter interface module. Installing this module is the closest thing to a VS1 modification for VS1 HASP. To link-edit this module into the interpreter requires, in general, modifying the STAGE II link-edit deck for the

Reader/Interpreter (member IEFIRC). CHANGE cards are required to change the name of the R/I modules so that the VS1 HASP routines receive control. The following sequence of CHANGE cards accomplish this:

```
CHANGE IEFVDA(DUKVDA)
INCLUDE ACSP3(IEFVDA)
CHANGE IEFVEA(DUKVEA)
INCLUDE ACSP3(IEFVEA)
CHANGE IEFVJA(DUKVJA)
INCLUDE AOSP3(IEFVJA)
```

The interpreter interface module should be assembled with SYS1.HASPSRC in the concatenation for SYSLIB and the object deck included in the R/I link-edit.

Since VS1 HASP uses the JES readers and writers, a small SYS1.SYSPOOL dataset is required. Its capacity is that required to spool JCL images and SMB output. The MCS hardcopy log is also written to SYS1.SYSPOOL.

The bulk of the code consists of the HASPGEN update cards. All PTFs through OY09762 are integrated into the deck. There is a new module HASPINTF which contains about half of the new code. It should be "ORDERed" after HASPNUC so that the page fix routine in

initialization will work properly. VS1 HASP cards are identified by the character string 'VS1 ' in columns 68-71 of the source images, except for the cards N2594000, S0154C0C-S9999900, and the entire HASPINTF module.

The VS1 HASP update deck contains a HASPGEN parameter &INITSVC that specifies the SVC number to be given the HASPSVC. The default value is 251.

The HASPWTR module is not used in the VS1 HASP system.

Jim Allen
Computation Center
Duke University
Durham, N. C. 27706