Section 3: Requirements

The Simulation Supervisor (SS) shall be a set of computer programs, written in assembly language for the IBM System 360 Model 44 (360/44) computer. SS shall provide the capabilities to test, debug, and integrate DCSG programs within PSCS/LPSS. Facilities shall be incorporated to provide the necessary interface between 44PS, SS, DCSG programs, and specialized user provided routines. SS shall monitor DCSG program execution and output debugging information in the form of traces and dumps.

3.1 CPCEI Characteristics

3.1.1 Functional Allocation of Simulation Supervisor

SS shall consist of:

- o A modified version of the 360/44 Programming System (44PS)
- o An Initialization Phase
- o A Processing Phase
- o Specialized User Provided Routines

3.1.1.1 Modified 44PS

44PS Modifications shall include:

- o A set of routines that become active only when SS gains control. These routines shall intercept SVC, Machine Check, External and Program Interruptions.
- o Two communication areas shall be maintained: one for exclusive use of SS, and the second for DCSG Executive Control System (ECS) and SS use.

3.1.1.2 Initialization Phase

This phase shall be brought into core storage by the Job Control Processor (JCP) after the recognition of the execute SS control statement. The functions of this phase shall be to:

- o Read in the seven types of SS control statements
- o Build the queues and tables corresponding to the control statements
- o Setup the link between the constructed queues and the processing phase
- o Bring the processing phase into core storage and give it control

3.1.1.3 Processing Phase

This phase shall contain the routines necessary to interface and insure proper DCSG execution.

The following is a list of the interface groupings of SS:

- o Program Interruption Handler
- o External Interruption Handler
- o Machine Check Interruption Handler
- o SVC Interruption Handler
- o Specialized User Provided Routines

3.1.1.3.1 Program Interruption Handler

The Program Interruption Handler (PIFLIH) shall field all program interruptions. Program interruptions shall result from:

- o Operation Exceptions
- o Privileged Operations
- o Execute Exceptions
- o Addressing Exceptions
- o Specification Exceptions
- o Data
- o Fixed Point Overflow
- o Fixed Point Divide
- o Decimal Divide
- o Decimal Overflow
- o Exponent Overflow
- o Exponent Underflow
- o Significance
- o Floating Point Divide

3.1.1.3.2 External Interruption Handler

Since a timer interrupt shall be the only External Interruption that can be caused by the DCSG Programs, the External Interruption Handler (EXTFLIH) shall be enabled to process this particular interruption.

An external interruption shall be caused by the interval timer turning negative (the elapsed time for a Program Segment has expired).

3.1.3.3.3 Machine Check Interruption Handler

A machine check interruption is caused by a machine malfunction and shall be fielded by the Machine Check Interruption Handler. If the interruption is caused by a DCSG program, SS shall allow the DCSG ECS Machine Check Handler to attempt recovery. Otherwise, a message shall be output and the simulation cancelled.

3.1.1.3.4 SVC Interruption Handler

SVCs shall be segregated into two groups; DCSG and SS (including 44PS) SVCs. All DCSG SVCs shall have an interruption code starting with 100_{10}. SS SVCs shall have interruption codes below 100_{10}.

There shall be two SVCs belonging to the DCSG that shall concern SS; the SEGEND and PUEND SVC. Upon recognition of either SVC, trace and debugging options shall be acted upon before returning control to the DCSG ECS. A special SVC for the SS PDUMP routine shall also be made available to the DCSG programs.

3.1.1.3.5 Specialized User Provided Routines

To allow for a flexible simulation environment, SS shall have the capability of incorporating user provided routines that simulate data manipulation and any other function peculiar to the on-orbit computer. By incorporating these routines into SS, the actual DCSG programs shall not have to be modified substantially to handle I/O or other specialized requests. As envisioned, there shall be a routine to handle each I/O device being accessed by DCSG programs. SS shall pass along all information needed in a communication region (COMREG) and the user routines shall be able to pass their output parameters (PSW settings, etc.) for the DCSG ECS in this same communication region.

SS shall call the following user provided routines (the mnemonics of which are used for clarity only) for each of the following circumstances:

- SSSIO - After encountering a SIO instruction
- SSTIO - after encountering a TIO instruction
- SSHIO - after encountering a HIO instruction
- SSTCH - after encountering a TCH instruction
- SSCKIO - entered to generate an I/O interruption
- SSAMUIN - to initialize the DCSG ECS
- SSPROP - after encountering other privileged instructions not handled by SS

To add a more realistic aspect to I/O simulation, SSCKIO shall be entered to generate an I/O interruption. The interruption should be initiated for one of the following conditions:

- to pass information corresponding to a previous I/O operation
- to generate an I/O interruption corresponding to an entry in the I/O Events Table.

After completing a requested I/O operation for the DCSG programs, a "stack table" should be constructed containing the simulated status for the operation. This table should be interrogated by SSCKIO and if there are entries in it, the first one should be used for generating the interruption. The I/O Events Table should be interrogated each time SSCKIO is entered. If the time corresponding to the first (next) event is greater than or equal to the Simulation Time, an interruption should be generated using the CSW status contained in the Events table. Again, COMREG shall be used to pass the required parameters.

Simulation of the AMU shall be an integral part of the routines provided by the user of SS. It is suggested here that an intermediate step be taken to reformat the output of PPP to look like a real AMU expected by the DCSG ECS. Through control statements provided in SS, the simulated AMU may be equated to a 44 PS I/O device. SSAMUIN shall be entered at the start of the simulation only. Its only function shall be to simulate an Initial Program Loading of the DCSG ECS and pass control back to SS. If the IPL procedure is unsuccessful, an error flag shall be set before returning.

SSPROP shall be entered to process privileged operations peculiar to the 4PI computer and those not handled by SS.

A detailed description of inputs, restrictions, and outputs expected for those user provided routines shall be found within paragraph 3.2.

User routines shall be expected to reside on the phase library under the name of SSUSRTNS. The address of SSAMUIN should be specified as the main entry to the phase during linkage editing. During setup in the processing phase, SSUSRTNS shall be read into core storage. The code at the beginning of SSUSRTNS must be as follows:

```
START          32000
DC             AL4(SIO Routine)
DC             AL4(TIO Routine)
DC             AL4(HIO Routine)
DC             AL4(TCH Routine)
DC          AL4 (CHECK IO Routine)
DC          AL4 (PRIV.OP Routine)
USING       *,BASEREG
```

By starting at 32000, user routines shall have approximately 33000 bytes of available core storage.

DCSG routines shall have 64000 bytes (16000 words) of storage which corresponds to the 4Pi computer being simulated.

3.1.1.3.5.1 External Inputs for I/O Processing

SS shall contain provisions to accept three types of control statements for I/O processing. Two of these statements shall contain the information necessary to equate DCSG devices to a PSCS/LPSS device for data input or output. The third type shall be used to generate special I/O interruptions to the DCSG ECS. These interruptions may be used to simulate special conditions that occur outside of and independent of normal DCSG operations. An example of this would be an uplink command.

Along with the Events Table which contains the interruption status, a file should be maintained on tape containing the data corresponding to the interruption. It is assumed that the ECS shall respond to this type of interruption by issuing a SIO to collect or output the required data. This file may be formatted one record for each interruption causing data input.

3.1.3 Simulation Supervisor Timing and Sequencing

3.1.3.1 Simulation Supervisor Initialization Phase

The sequencing of operations among the computer program components of the

Simulation Supervisor Initialization Phase (SSINIT) shall be dependent upon the order in which the SS control statements are introduced in the input stream.

Once SSINIT is loaded into core storage by the JCP, the sequence of operations among the main SSINIT computer program components is:

- o SSINIT Control Routine
- o SSINIT Get Control Card Routine
- o SSINIT Card Type Determination Routine
- o SSINIT Check Card Routine
- o SSINIT Control Card Processing Routine

For any pass through the SSINIT Card Type Determination Routine, depending upon the current SS control statement being processed, one of the following SS card-type processing routines shall be invoked:

- o SSEQUIVI
- o SSEQUIVO
- o SSIOEVNT
- o SSDEB
- o SSQUE

The card type processing routines shall call the following CPC's:

- o SSBRKOUT
- o CNVTDA
- o CNVRT
- o CNVRTD

When all of the SS control statements have been processed, the SSINIT Control Routine shall invoke the SSINIT Load Simulation Supervisor Routine (SSLDSS). This routine shall load the SS Processing Phase and relinquish control to it.

3.1.3.2 Simulation Supervisor Processing Phase

Initially, the processing routines gain control from SSINIT. After SSINIT concludes processing, it shall pass control to the SS Processing Phase Set Up Routine (CLKINIT). CLKINIT shall bring in the user routines and pass control to SSAMUIN to initialize the DCSG ECS. Once control is given to the DCSG ECS, by generating an I/O interruption, control shall be regained only through one of the following types of interruptions:

- o SVC
- o Machine Check
- o External (Interval Timer elapsing)
- o Program

Depending upon which of the above types caused the interrupt, the first level interrupt handler, which shall be resident in the modified 44PS Section, shall give control to SSIRPT. SSIRPT shall then enter UPDTCLK to update the DCSG running times and set the next interval timer value. SSIRPT shall then determine the type of interruption and pass control to one of the following interruption processors:

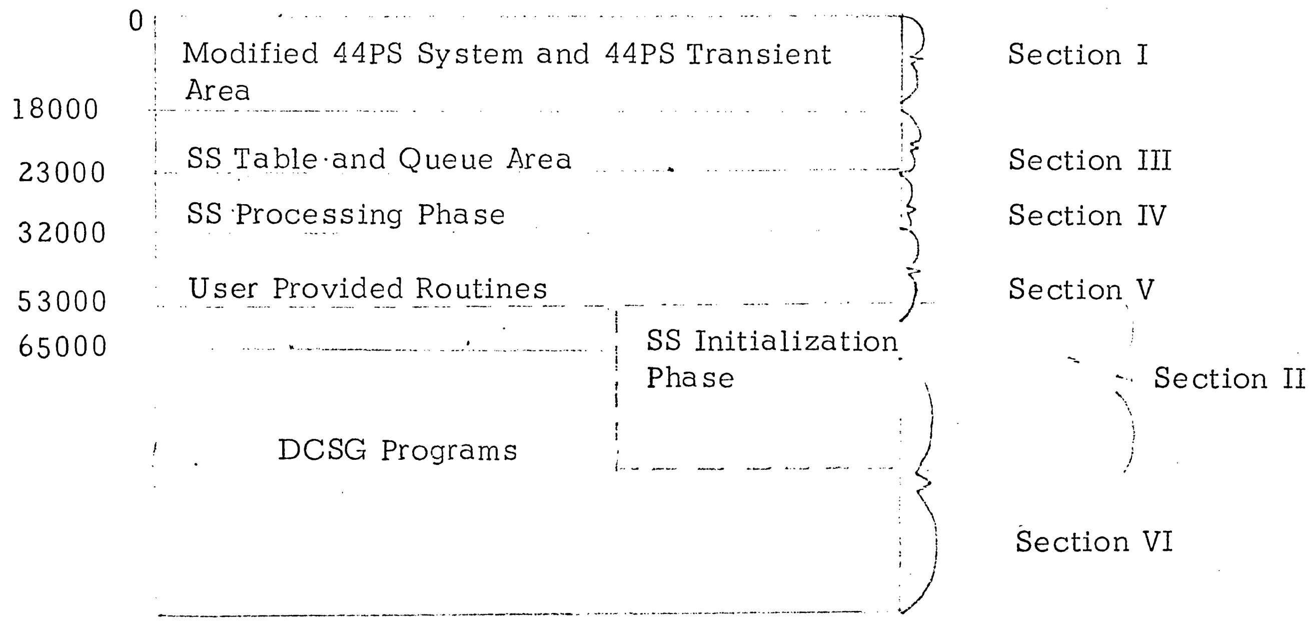
- o SVCH
- o EXTH
- o PIH
- o MCKH

At the conclusion of interruption processing, SSRTN shall be entered to reset status and return control to the DCSG.

If for any reason, the simulation is to be terminated, SSEXIT shall be entered by issuing a SVC 98 to reset 44PS status and normal 44PS operations shall continue. Also, if a core dump is wanted by a DCSG program, an SVC may be issued with a SVC interruption code of 99_{10} (see 3.2.33 for parameter settings).

3.1.4 Simulation Supervisor Storage Allocation

The following diagram depicts the overall allocation of storage when SS gains control.



The following is a description of each section pertaining to SS pictured above.

3.1.4.1 Section I (approximately 18000 bytes)

Section I of core storage shall contain all existing programs of the 44PS system (IBM Form #Y28-6812-0, IBM System/360 Model 44 Programming System, Supervisor and Job Control). This section shall be modified to include the following SS dependent routines and table areas.

- o SS Communication Region (CREGION) (3.1.5.7)
- o Communication Area for the DCSG ECS (ECSCOMRG) (3.1.5.9)
- o Address Constants for SS Initialization Phase. These constants contain the addresses of SS first level interrupt handlers (STAT44) (3.1.5.10)
- o Machine Check First Level Interrupt Handler (MCKFLIH) (3.2.18)
- o Program Interruption First Level Interrupt Handler (PIFLIH) (3.2.19)

- o Input Equivalence Queue List (IQUEUE)
- o Input Equivalence Table (ITBL)
- o Output Equivalence Queue List (OQUEUE)
- o Output Equivalence Table (OTBL)
- o Debugging Aids Option Queue (DEBQUE)
- o Debugging Aids Option List (DEBLST)
- o Queue Dump Queue List (QDQUE)
- o Queue Dump Table (QDTBL)
- o I/O Events Table (EVNTAB)

The queues shall be constructed from low to high core and the tables corresponding to the queues shall go from high to low core (within allocated table space area). These queues and tables shall vary in length from one simulation to another due to differences in the number and type of control statements processed.

3.1.4.4 Section IV (approximately 9000 bytes)

This section shall contain all the SS processing routines and table COMREG.

The names and CPC numbers of the routines that shall be contained within this section are:

- o SS Set Up Routine (CLKINIT) (3.2.24)
- o SS Interrupt Determination Routine (SSIRPT) (3.2.25)
- o Update Interval Timer (UPDTCLK) (3.2.26)
- o Program Interruption Handler (PIH) (3.2.27)
- o SVC Interruption Handler (SVCH) (3.2.28)
- o External Interruption Handler (EXTH) (3.2.29)
- o Machine Check Interruption Handler (MCKH) (3.2.30)
- o DCSG Return (SSRTN) (3.2.31)
- o Set System Mask Handler (SSMH) (3.2.32)
- o Load PSW Handler (LPSWH) (3.2.33)
- o Debugging Aids Routine (SSDBUG) (3.2.34)

- SVC First Level Interrupt Handler (SVCFLIH) (3.2.20)
- External First Level Interrupt Handler (EXTFLIH) (3.2.21)
- Exit Routine (SSEXIT) (3.2.22)
- Save Status Routine (SSSAVER) (3.2.23)

3.1.4.2 Section II (approximately 8000 bytes)

SS shall use Section II for the SS Initialization Phase. This Phase shall be resident here until its functions are completed. The area shall then be overlayed by user routines and DCSG programs. The SS Initialization routines that shall occupy this section, along with their CPC numbers, follows:

- Control Routine (SSINIT) (3.2.1)
- Read Control Card Routine (SSGETCC) (3.2.2)
- Control Card Type Determination Routine (SSTYPDET) (3.2.3)
- Construct Input Equivalence Queue and Table (SSEQUIVI)(3.2.4)
- Construct Output Equivalence Queue and Table (SSEQUIVO) (3.2.5)
- Construct Trace and Debugging Queue and Table (SSDEB) (3.2.6)
- Construct Queue Dump Queue and Table (SSQUE) (3.2.7)
- Event Data Descriptor Routine (EVNTDATA) (3.2.8)
- Construct I/O Event Table (SSIOEVNT) (3.2.9)
- Load SS Routine (SSLDSS) (3.2.10)
- Error Check Control Card Routine (SSCHKCRD) (3.2.11)
- Parameter Check and Move Routine (CHKMOVE) (3.2.12)
- Continuation Card Processing Routine (SSCONTIN) (3.2.13)
- Parameter Break-Out Routine (SSBRKOUT) (3.2.14)
- Convert from Hexadecimal EBCDIC to Binary (CNVRT) (3.2.15)
- Convert to SYSUNI Index (CNVTDA) (3.2.16)
- Convert from Decimal EBCDIC to Binary (CNVRTD) (3.2.17)

3.1.4.3 Section III (approximately 5000 bytes)

This section shall be constructed during the SS Initialization Phase: It shall contain the following queues and tables:

- Input Equivalence Queue List (IQUEUE)
- Input Equivalence Table (ITBL)
- Output Equivalence Queue List (OQUEUE)
- Output Equivalence Table (OTBL)
- Debugging Aids Option Queue (DEBQUE)
- Debugging Aids Option List (DEBLST)
- Queue Dump Queue List (QDQUE)
- Queue Dump Table (QDTBL)
- I/O Events Table (EVNTAB)

The queues constructed shall be built from low to high core and the tables shall go from high to low core (within allocated table space area). These queues and tables shall vary in length from one simulation to another due to differences in the number and type of control statements processed.

3.1.4.4 Section IV (approximately 9000 bytes)

This section shall contain all the SS processing routines and table COMREG.

The names and CPC numbers of the routines that shall be contained within this section are:

- SS Set Up Routine (CLKINIT) (3.2.24)
- SS Interrupt Determination Routine (SSIRPT) (3.2.25)
- Update Interval Timer (UPDTCLK) (3.2.26)
- Program Interruption Handler (PIH) (3.2.27)
- SVC Interruption Handler (SVCH) (3.2.28)
- External Interruption Handler (EXTH) (3.2.29)
- Machine Check Interruption Handler (MCKH) (3.2.30)
- DCSG Return (SSRTN) (3.2.31)
- Set System Mask Handler (SSMH) (3.2.32)
- Load PSW Handler (LPSWH) (3.2.33)
- Debugging Aids Routine (SSDBUG) (3.2.34)

- o Core Dump Routine (SSPDUMP) (3.2.35)
- o Print out Routine (SSDPUT) (3.2.36)
- o Data Look up Routine (DATALOOK) (3.2.37)
- o Determine Program Unit Name (DETPRUN (3.2.38)
- o Convert Binary to Hexadecimal (BI2HEX (3.2.39)
- o Determine Program Unit Pointers (GETPTRS) (3.2.40)
- o Address Check and Determination (ADDCK) (3.2.41)
- o User and SS Communication Region (COMREG) (3.1.5.8)

3.1.4.5 Section V

Section V shall contain all user provided routines. SS shall allow approximately 33000 bytes for these routines.

3.1.4.6 Section VI

This section shall contain the DCSG programs. Approximately 64000 bytes of core storage shall be available.

3.1.5 Simulation Supervisor Data Base Characteristics

SS shall construct and maintain tables and queues that comprise the data base for simulation. A detailed definition of the contents of these is described in the following paragraphs. The actual locations of each may vary from one run to another depending upon the order SS control cards appear. However, it should be noted that whenever a queue is contructed for accessing a table , the queues start in low core of the SS Table Section and proceed in an ascending manner through core. The tables corresponding to the queues are constructed in a high-to-low core manner within the SS Table Section. The addresses of the queues shall be kept in CREGION.

A list of the queues and tables that shall be incorporated in the Simulation Supervisor is:

o	IQUEUE	-	Input Equivalence Queue (3.1.5.1.1)
o	ITBL	-	Input Equivalence Table (3.1.5.1.2)

o	OQUEUE	-	Output Equivalence Queue (3.1.5.2.1)
o	OTBL	-	Output Equivalence Table (3.1.5.2.2)
o	DEBQUE	-	Debugging Aids Option Queue (3.1.5.3.1)
o	DEBLST	-	Debugging Aids Option List (3.1.5.3.2)
o	QDQUE	-	Queue Dump Queue List (3.1.5.4.1)
o	QDTBL	-	Queue Dump Table (3.1.5.4.2)
o	EVNTAB	-	I/O Events Table (3.1.5.5)
o	CREGION	-	SS Communication Region (3.1.5.7)
o	COMREG	-	User Communication Region (3.1.5.8)
o	ECSCOMRG	-	DCSG ECS Communication Region (3.1.5.9)
o	STAT44	-	Saved status (PSW's) of 44PS (3.1.5.10)

3.1.5.1 Input Equivalence Queue and Table

The Input Equivalence Queue and Table shall be constructed from information contained on the SS DDI Control Statements. The formats of the DDI cards are:

o DDI statement when input is from a real (PSCS/LPSS) device:

```
ccl      6
  *DDI   prog,yyy(xxx,nnn)(xxx,nnn) ---etc.
```

where:

*DDI	-	identifies this control card as an SS Input Device Equivalence Control Card
prog	-	is the program unit name (upto 4 characters)
yyy	-	is the 44PS real input tape address. The last three symbolic characters that would be used in the allocation of this device on the 44PS ALLOC or ACCESS control cards replace yyy.
xxx	-	is the DCSG simulated input device address (hex value)
nnn	-	is the file number of the data

- o DDI card used when input is from core storage:

ccl	6
*DDI	prog,CRE(xxx,1111)(xxx,1111),---etc.

where:

*DDI	-	identifies this control card as an SS Input Device Equivalence Control Card
prog	-	is the program unit name (upto 4 characters)
CRE	-	is the keyword which identifies the 44PS real input device address as being core storage
xxx	-	is the DCSG simulated input device address (hex value)
1111	-	is the symbol within the program unit where the xxx data is located. From one through four character symbols are allowed

Figure 3.1.5-1 graphically portrays the relationship between the queue and the table.

<u>3.1.5.1.1 Input Equivalence Queue</u>

The Input Equivalence and Data Definition Routine (SSEQUIVI), which interpret DDI control cards, shall construct the Input Equivalence Queue.

The program unit name shall be extracted from the DDI statement and placed in the queue. Following each program unit name there shall be a four-byte field. When a DDI card is encountered for a particular program unit, a 4-byte address shall point to the entry in the Input Equivalence Table.

<u>3.1.5.1.2 Input Equivalence Table</u>

The Input Equivalence Table shall consist of information used to equate real PSCS/LPSS I/O devices to DCSG I/O devices being simulated.

The following entries are included for input from PSCS/LPSS devices:

- o One-byte DCSG simulated device address
- o Two-byte SYSUNI index of corresponding PSCS/LPSS device
- o One-byte file number at which data resides
- o Two-byte record position field

The following entries are included for input from core storage:

- o One-byte DCSG simulated device address
- o Two-byte field containing the characters 'CR' indicating data is in core
- o Two-byte record position field
- o Four-byte label indicating location of data within program unit

3.1.5.2 Output Equivalence Queue and Table

The Output Equivalence Queue and Table shall be constructed from information contained on the SS DDO Control Statement.

```
        ccl     6
         *DDO   prog,yyy(xxx,xxx,---,xxx)
```

where:

*DDO	-	identifies this control card as an SS Output Device Equivalence Control Card
prog	-	is the program unit name (upto 4 characters)
yyy	-	is the 44PS real output device address. The last three symbolic characters which would be used in the allocation of this device on the 44PS ALLOC or ACCESS control cards replace yyy.
xxx	-	is the DCSG simulated output device address (hex value)

Figure 3.1.5-2 graphically portrays the relationship between the queue and the table.

3.1.5.2.1 Output Equivalence Queue

The Output Equivalence and Data Definition Routine (SSEQUIVO), which shall interpret DDO control statements, shall construct the Output Equivalence Queue. As a DDO statement is processed, the program unit name shall be extracted and placed into the queue. Following the four byte program unit name, a pointer to the table entry shall be generated and placed into the next four bytes of the queue. An end of queue flag shall be inserted after processing the last DDO control statement.

3.1.5.2.2 Output Equivalence Table

The Output Equivalence Table shall contain the address of the device being simulated for the program unit followed by the PSCS/LPSS real device SYSUNI index. The SYSUNI index shall be obtained by a table lookup using the yyy value of the control statement. Each entry of this table shall be one byte.

3.1.5.3 Debug Queue and Option List Table

The Debugging Aids Queue and Option List Table shall be constructed by using the information contained on the SS DEB control card. The format of the SS DEB control card, along with a description of each item on the card is:

ccl 6

$$\text{*DEB}\quad \text{prog,yyy}\ [\text{,UNC}]\ [\text{,PCB}]\ [\text{,PVC}]\ \left[\left(\text{t, ssss,}\begin{Bmatrix}\text{eeee}\\ \text{\$\$\$\$}\end{Bmatrix}\right)\right]\left[\left(\text{t, ssss,}\begin{Bmatrix}\text{eeee}\\ \text{\$\$\$\$}\end{Bmatrix}\right)\right]\text{---etc}$$

where:

*DEB	-	identifies this control card as an SS Debugging Aids Control Card
prog	-	is the program unit name (upto 4 characters) or the keyword DCSG.
yyy	-	is the 44PS output device for debugging aids. The last three symbolic characters that would be used in the allocation of this device on the

		44PS ALLOC or ACCESS control cards replace yyy.
UNC	-	Universal Common of the DCSG programs is to be dumped (optional)
PCB	-	Program Control Block of the program unit is to be dumped (optional)
PVC	-	Private Common of the program unit is to be dumped (optional)
t	-	type of dump limits for the following pair: S - for symbolic; H - for hexadecimal
ssss	-	starting address from which a core dump is to be taken (symbolic or hexadecimal according to t)
eeee	-	ending address of core dump (symbolic or hexadecimal according to t)
$$$$	-	core dump is to be taken to end of core storage

NOTE:
1. UNC, PCB, and PVC may appear on the card in any order
2. ssss and eeee must be either symbols or hexadecimal displacements from the start of the program unit. Symbols and displacements can be mixed on a control card, but not mixed within parentheses. A maximum of 4 pairs may be specified for a program unit.

Figure 3.1.5-3 graphically portrays the relationship of the queue and option list table.

3.1.5.3.1 Debug Queue

The Debugging Aids Queue shall be constructed for ease of indexing into the Debugging Aids Option List Table. The size of the queue shall be dependent upon the number of SS DEB control cards included as input.

The Debug Queue contains the following items which shall be repeated for each program unit specified on the control statements.

- o A four-byte field containing the program unit name.
- o A one-byte field containing the SYSUNI index of the device on which traces and dumps shall be output for this program unit.
- o A flag byte to indicate which keywords have been specified.
- o A one-byte field containing the number of dump limits specified.
- o A flag byte indicating whether dump limits are specified symbolically or in hexadecimal.
- o A four-byte field containing the address of the corresponding entry in the Debug Option List Table for this program unit (set to zero if no dumps are specified.)

One entry in the queue, therefore, shall occupy twelve bytes. Following the last entry in the queue shall be a four-byte end-of-queue flag.

<u>3.1.5.3.2 Debug Option List Table</u>

The Debug Option List Table consists of a series of entries corresponding to each program unit requesting a dump. The entries are the lower and upper dump limits. When a dump limit is specified as a hexadecimal displacement, it appears in the option list in a fixed length three-byte field. If the limit is specified symbolically, the symbol itself shall appear in the list, preceded by a one-byte field containing the length of the symbol.

<u>3.1.5.4 Queue Dump Queue and Table</u>

The Queue Dump Queue and Table shall be constructed by using the information contained on the SS QUE control statement. It is constructed for the purpose of allowing the ECS to dump selective queues after the completion of specific program units.

The format of the SS QUE control statement, along with a description of each item on the card, is:

```
ccl      6
  *QUE   prog,yyy(que1,que2,que3,---)
```

where:

*QUE	-	identifies this control card as an SS Queue Dump Control Card.
prog	-	is the program unit name (upto 4 characters)
yyy	-	44PS output device address yyy is equivalent to the last 3 characters of a symbolic device address as would be specified on the 44PS ALLOC or ACCESS control card.
que1, que2,....	-	are the DCSG ECS Queue names (upto 4 characters) scheduled for dumping at the execution-completion of the program unit specified by "prog".

Figure 3.1.5-4 graphically portrays the relationship of the queue and the table.

3.1.5.4.1 Queue Dump Queue

The Queue Dump Queue shall be constructed for ease of indexing into the Queue Dump Table. The size of the Queue shall be dependent upon the number of SS QUE control statements included. The following items are contained in the queue:

- o A four-byte field containing the program unit name
- o A one-byte field containing the SYSUNI index of the device on which the queues shall be output for this program unit.
- o A three-byte field containing the address of the corresponding entry in the Queue Dump Table for this program unit.

One entry in the queue, therefore, occupies eight bytes. A four-byte end- of queue flag follows the last entry in the queue.

3.1.5.4.2 Queue Dump Table

The Queue Dump Table contains the following items which shall be repeated for each program unit specified on the control statements.

- o A one-byte field containing the number of queue names specified in the variable length queue name field
- o A variable length queue name field - each queue name occupying four bytes.

3.1.5.5 I/O Events Table

This table shall be constructed to permit various interruptions (events) to occur after a spedific time in the simulation. This table shall be constructed in an ascending order and shall be time ordered. The size of this table shall be dependent upon the number of SS IOE control statements input. The table shall contain a header consisting of:

- o A one-byte field containing the file number of the event data
- o A one-byte SYSUNI index field
- o A two-byte field for record position
- o A four-byte field to contain the address of the next event to be acted upon

The following fields shall be repeated for each control statement:

- o A four-byte field containing the event time
- o A one-byte field denoting whether input is expected with this interruption
- o A one-byte field containing the simulated device address
- o A two-byte field containing the simulated CSW status

A four-byte flag shall signify the end of the table.

Figure 3.1.5-5 is a graphic description of the table.

3.1.5.5.1 I/O Events Control Statements

The format of the control statements for constructing the I/O Events Table is:

- Event data descriptor - This card should appear only once during a simulation. Its function is to describe the location of the event data.

```
ccl      6
 *EDD    yyy,ff
```

where:

*EDD	-	identifies this card as the event data descriptor
yyy	-	44PS input device address containing the event data. yyy is equivalent to the last 3 characters of a symbolic device address as would be specified on the 44PS ALLOC or ACCESS control card.
ff	-	is the file number on the yyy where the data is located

- Time dependent event data

```
ccl      6
 *IOE    (zxx,tttt,cccc) (zxx,tttt,cccc)....etc.
```

where:

*IOE	-	identifies this card as an I/O event data card
z	-	if non-zero, data corresponding to this interrupt shall be found on the IOE file. If zero, no data corresponding to this interrupt shall be found on the IOE file. (one hex digit)
xx	-	is the DCSG simulated device address that shall be used to generate the interruption (hex value)
tttt	-	time in milliseconds at which interrupt shall be

		initiated. Up to 7 decimal digits may be used.
cccc	-	is the CSW status to be used in generating the interruption (hex value)

This card must appear immediately after the EDD card. If more interruptions are desired than those contained on continuation cards, the *IOE may be used as many times as necessary (table space shall be the only limitation on the number of entries).

3.1.5.6 Conventions for SS Job Control Statements

3.1.5.6.1 Identifier

SS job control statements are identified by an asterisk (*) in column 1.

3.1.5.6.2 Type Field

The Type Field contains 3 characters which identify the type of control statement being specified. The type field is always in columns 2,3, and 4 and is followed by one blank.

3.1.5.6.3 Operand Field

The operand field contains the statement parameters and begins in column 6 and may extend through column 71. The operand field is recognized as being complete if column 72 is blank and the last character is either immediately followed by a blank or is in column 71.

3.1.5.6.4 Comments Field

Comments may be included on SS control cards. When comments are included, the comments field must be separated from the operand field by one or more blanks and may extend through column 71.

3.1.5.6.5 Continuation Cards

An SS job control statement may not extend beyond column 71. If necessary, up to three continuation cards may be used for each control statement. The rules for continuation are:

1. All parameters up to the first left parenthesis must appear on the first card. Thereafter, the statement may be interrupted immediately before any left parenthesis.
2. In addition, the QUE and DDO control statements may be interrupted after any comma within parentheses.
3. An interrupted statement may be followed with one or more blanks and comments, if desired, up to column 71.
4. A nonblank character must appear in column 72.
5. An asterisk must appear in column 1 of the next card image. Columns 2 through 15 must be blank.
6. The continuation of the interrupted statement must begin in column 16.
7. A comment alone may not be continued. However, a comment may appear on a continuation card if there is at least one operand present.

3.1.5.7 Table CREGION

CREGION shall be the communication region used by SS programs. Space shall be allocated within the 44PS section to contain this table. The SS Initialization Phase shall store queue addresses into the positions alloted for them as the tables and queues are constructed. The following is the format of CREGION:

Bytes	Description	Mnemonic
0-63	The sixteen general registers saved from DCSG	GRS
64-67	Interval Timer Value	ITVAL
68-71	Address of SSIRPT	ADSSIRPT
72-75	Pointer to I/O Events Table	AEVNTAB
76-79	Pointer to Input Equivalence Queue	PTIEQ
80-83	Pointer to Output Equivalence Queue	PTOEQ
84-87	Pointer to Debugging Queue	PTDQ
88-91	Pointer to Queue Dump Queue	PTDQL

Bytes	Description	Mnemonic
92-95	Pointer to Saved 44PS Status	STAT44A
96-99	Pointer to List Containing SS Status	STATSSA
100-103	Address of DCSG ECS Communication Region	ADECSCR
104-107	DCSG Total Running Time	DCSGRUN
108-111	Segment Running Time	SEGRUN
112-115	Timer Value at Last Exit	DCCCHK
116-123	Dummy PSW	DUMPSW
124-131	Old PSW from Interruption	INPSW

3.1.5.8 Table COMREG

COMREG shall be maintained for communication between SS and user-provided routines. The table shall be resident within the processing phase of SS. The following is the format of COMREG.

Bytes	Description	Mnemonic
0-3	Name of Program Unit in Control	PUNAME
4-11	Old PSW of Interruption	OPSW
12-15	Channel Address Word	CAWL
16-23	Channel Status Word	CSWL
24-27	Starting Address of DCSG EXEC	STDCSG
28-31	Pointer to Input Equivalence Queue	IEQ
32-35	Pointer to Output Equivalence Queue	OEW
36-39	Pointer to I/O Events Table	EVENTS
40-43	Instruction Causing Interruption	INSTR
44-47	Total simulation running time	SIMTIM
48-51	Address of SSDBUG	DBUGAR
52-55	Address of SSPDUMP	PDMPAD
56-59	Address of SSRTN	RTMAD
60-63	Address of DATALOOK	ADDATLK

Bytes	Description			Mnemonic
64	Return Address Flag			RETAD
	RETAD settings are:	0-	return to point of interruption	
		88-	return to DCSG External interruption handler	
		96-	return to DCSG SVC interruption handler	
		104-	return to DCSG Program interruption handler	
		112-	return to DCSG Machine Check interruption handler	
		120-	return to DCSG I/O interruption handler	

3.1.5.9 DCSG ECS Communication Region

The DCSG ECS shall maintain a communication region within 44PS. SS shall make use of the following items within this region. These items are:

SMCORE	-	Base of managed core
UVCOM	-	Location of Universal Common
SLPCB	-	Location of the PCB for the program unit last selected.
SLRPL	-	Location of the start of the DCSG ECS Resident Program List (RPL)

The total length of this region shall be 400 bytes.

In addition to this area, a Permanent Storage Assignment Area shall be maintained. This area shall contain the ECS status (PSW's, Timer location, etc.) and be located in front of the ECS communication region. SS shall reserve 200 bytes for this area.

3.1.5.10 STAT44 Table

This table shall be constructed by the SSLDSS routine of the Initialization Phase. It shall contain the contents of the new PSW's used by 44PS. Before SS terminates a simulation, the contents of this table shall be placed into the new PSWs for normal 44PS operations.

3.1.5.11 Input Data Other Than Control Statements

To aid in the use of the DDI control statement the following procedures are discussed. The following options are available for each program unit that accesses a simulated input device:

- o Pre-formatted tape input data
- o Core resident data

3.1.5.11.1 Pre-Formatted Tape Input Data

The following input techniques may be used for those users of SS who must have large amounts of data for simulating their devices. Within a program unit and for each device that input data is to be read, a file is generated that contains a record for each time the device is to be accessed. A two-byte area shall be set aside within the I/O Equivalence Table for each device within a program unit for the user to keep track of record positions.

3.1.5.11.1.1 IOE Events File

This file should be constructed so as to contain all the input records specified as being present on the IOE control statement. The records should be present corresponding to the order of the event's occurrence.

3.1.5.11.2 Core Resident Data

SS shall allow data to be resident in core as well as on tape. The user may assemble data into his particular program unit as a separate segment. In order for SS to locate this data, a 'SYMBDICT' control statement must have been input to the Program Preparation Processor (PPP) to allow PPP to place the symbol into the program unit's symbol dictionary. As many sets of data as desired may be generated in this manner.

3.1.5.12 Simulating DCSG Output Devices

The DDO queue and table are constructed for the purpose of equating a real (44PS) device to a DCSG device. The SYSUNI index value should be used in conjunction with the standard 44PS SVC WRITE service routine.

3.1.5.13 Relationship of Simulation Supervisor Components to Data Base

The relationship of the Simulation Supervisor components to the various tables and items within the data base is discussed in the following subparagraphs. Each subparagraph is identified by the Simulation Supervisor component tag (symbolic code) and its component paragraph number.

The identity of each table that is constructed or used by the component is portrayed in each subparagraph. The specific items, within each table that the component uses (U) or sets (S), are also portrayed. Each table is identified by a tag (symbolic code) and its paragraph number.

SSINIT (3.2.1)

Computes the starting and ending addresses of the queue and table area (3.1.4.3)

SSEQUIVI (3.2.4)

IQUEUE (3.1.5.1.1)
Program Unit Name (S)
Location of table entry (S)

ITBL (3.1.5.1.2)
Simulated Device Address (S)
SYSUNI index value (S)
data position (file or core) (S)

CREGION (3.1.5.7)
Location of IQUEUE (S)

SSEQUIVO (3.2.5)

OQUEUE (3.1.5.2.1)
Program Unit Name (S)
Location of table entry (S)

QTBL (3.1.5.2.2)

Simulated Device Address (S)

SYSUNI index value (S)

CREGION (3.1.5.7)

Location of OQUEUE (S)

SSDEB (3.2.6)

DEBQUE (3.1.5.3.1)

Program Unit Name (S)

SYSUNI index value (S)

Keyword flags (S)

Number of dumps (S)

Dump limit designations (S)

Location of Option List entry (S)

DEBLST (3.1.5.3.2)

Starting location of dump (S)

Ending location of dump (S)

CREGION (3.1.5.7)

Location of DEBQUE (S)

SSQUE (3.2.7)

QDQUE (3.1.5.4.1)

Program Unit Name (S)

SYSUNI Index Value (S)

Location of Queue Dump Table entry (S)

QDTBL (3.1.5.4.2)

Number of entries for the Program Unit (S)

Queue names (S)

CREGION (3.1.5.7)

Location of QDQUE (S)

<u>SSIOEVNT (3.2.8)</u>

<u>EVNTAB (3.1.5.5)</u>

File number of events (S)

SYSUNI index (S)

Address of next event (S)

Time of event (S)

DCSG device address (S)

CSW status (S)

<u>CREGION (3.1.5.7)</u>

Address of EVNTAB (S)

<u>SSLDSS (3.2.9)</u>

<u>CREGION (3.1.5.7)</u>

Location of SS PSW's (U)

Location to save 44PS PSW's (U)

<u>SVCFLIH (3.2.20)</u>

<u>CREGION (3.1.5.7)</u>

Address of SSIRPT routine (U)

INPSW (S)

<u>SSEXIT (3.2.22)</u>

<u>CREGION (3.1.5.7)</u>

Address of saved 44PS PSW's (U)

<u>SSSAVER (3.2.23)</u>

<u>CREGION (3.1.5.7)</u>

DCSG general registers (S)

Interval Timer value (S)

<u>CLKINIT (3.2.24)</u>

<u>CREGION (3.1.5.7)</u>

Address of SSIRPT routine (S)

Interval Timer Value (S)

Location of IQUEUE (U)

Location of OQUEUE (U)

Location of EVNTAB (U)

Location of DCSG ECS Communication Region (U)

Dummy PSW (S)

COMREG (3.1.5.8)

Location of IQUEUE (S)

Location of OQUEUE (S)

Location of EVNTAB (S)

Location of SSPDUMP routine (S)

Location of SSDBUG routine (S)

Location of SSRTN routine (S)

Return address flag (S)

ECSCOMRG (3.1.5.9)

Sets all items by picking up first 150 words of DCSG ECS

SSIRPT (3.2.25)

CREGION (3.1.5.7)

Dummy PSW (U)

PSW from interruption (U)

COMREG (3.1.5.8)

PSW from interruption (S)

Program Unit Name (S)

UPDTCLK (3.2.26)

CREGION (3.1.5.7)

Segment running time (U), (S)

Total DCSG running time (U), (S)

Last set DCSG time (U), (S)

COMREG (3.1.5.8)

Total DCSG running time

<u>PIH (3.2.27)</u>

<u>COMREG (3.1.5.8)</u>

PSW from interruption (U)

Program Unit Name (S)

Return Address flag (S)

Channel Address Word (S)

<u>COMREG (3.1.5.8)</u>

Total DCSG Running time (S)

<u>ECSCOMRG (3.1.5.9)</u>

Interval timer location (U), (S)

<u>SVCH (3.2.28)</u>

<u>CREGION (3.1.5.7)</u>

DCSG general registers (U)

<u>COMREG (3.1.5.8)</u>

PSW from interruption (U), (S)

Return address flag (S)

<u>EXTH (3.2.29)</u>

<u>CREGION (3.1.5.7)</u>

Interval timer value (S)

Segment running time (U), (S)

Total DCSG running time (U), (S)

Last set time (U), (S)

<u>COMREG (3.1.5.8)</u>

Return address flag (S)

<u>ECSCOMRG (3.1.5.9)</u>

Interval timer location (S)

<u>MCKH (3.2.30)</u>

<u>COMREG (3.1.5.8)</u>

Return address flag

SSRTN (3.2.31)

CREGION (3.1.5.7)

PSW from interruption (U), (S)

Interval timer value (U)

Dummy PSW (S)

DCSG general registers (U)

COMREG (3.1.5.8)

Return address flag (U)

PSW from interruption (U)

Channel Status Word (U)

ECSCOMRG (3.1.5.9)

Interval time location (S)

Old PSW locations (S)

CSW location (S)

New PSW locations (U)

SSMH (3.2.32)

CREGION (3.1.5.7)

Dummys PSW (S)

COMREG (3.1.5.8)

PSW from interruption (U)

Program unit name (S)

Return address flag (S)

Instruction causing interruption (U)

ECSCOMRG (3.1.5.9)

Address of Program Control Block (U)

LPSWH (3.2.33)

CREGION (3.1.5.7)

Dummy PSW (S)

COMREG (3.1.5.8)

PSW from interruption (U)

Program unit name (S)

Return address flag (S)

Instruction causing interruption (U)

<u>ECSCOMRG (3.1.5.9)</u>

Address of Program Control Block (U)

<u>SSDBUG (3.2.34)</u>

<u>CREGION (3.1.5.7)</u>

GRS (U)

PTDQ (U)

PTDQL (U)

DCSGRUN (U)

SEGRUN (U)

ADESCSCR (U)

<u>ECSCOMRG (3.1.5.9)</u>

UVCOM (U)

SLPCB (U)

<u>COMREG (3.1.5.8)</u>

PU NAME (U)

OPSW (U)

<u>DEBQUE (3.1.5.3.1)</u>

Program Unit Names (U)

SYSUNI Index values (U)

Keyword flags (U)

Number of dumps (U)

Dump limit designations (U)

Pointer to entry in list (U)

<u>DEBLST (3.1.5.3.2)</u>

uses all items in this table

<u>QDQUE (3.1.5.4.1)</u>

Program Unit Names (U)

Real Device address (U)

Pointer to dump table (U)

QDTBL (3.1.5.4.2)

uses all items in this table

SSPDUMP (3.2.35)

CREGION (3.1.5.7)

DCSG general registers (U)

DETPRUN (3.2.38)

COMREG (3.1.5.8)

Program Unit Name (S)

PSW from interruption (U)

ECSCOMRG (3.1.5.9)

Location of current PCB (U)

ECS ending address (U)

GETPTRS (3.2.40)

CREGION (3.1.5.7)

Address of ECS Communication Region (U)

DCSG general registers (U)

COMREG (3.1.5.8)

Program Unit Name (U)

ECSCOMRG (3.1.5.9)

Location of current PCB (U)

Location of RPL (U)

ADDCK (3.2.41)

CREGION (3.1.5.7)

DCSG registers (U)

COMREG (3.1.5.8)

Instruction causing interruption (U)

Start location of ECS (U)

3.2.1 SSINIT Control Routine

The Simulation Supervisor Initialization Phase shall be loaded into core storage as a result of the Job Control Processor's (JCP) recognition of the EXEC Simulation Supervisor control card.

The entry point to the SS Initialization Phase is defined as SSINIT within the SSINIT Control Routine. The functions performed are:

- Establish starting and ending addresses for the table and queue area
- Clear out pointers to tables and queues
- Control the processing flow within the SS Initialization Phase
- Sort the entries in the IOE Table
- Transfer control to the SS Processing Phase at initialization end

3.2.1.1 SSINIT Control Routine Description

At 44PS JCP completion the Simulation Supervisor Initialization Phase (SSINIT) shall be loaded into core storage. SSINIT shall be entered to begin the processing of the phase.

SSINIT shall control the processing flow throughout the Initialization Phase. The only exit from this routine shall be at the completion of initialization. Once the Processing Phase routines are loaded into core storage, control shall be passed to the SS start up routine (CLKINIT).

3.2.1.3 SSINIT Control Routine Interfaces

The entry point to the Initialization Phase shall be defined as SSINIT. This phase shall be loaded into core storage as a result of JCP recognition of the EXEC Simulation Supervisor control card (//EXEC SIMSUP).

The subroutines called by the SSINIT routine shall be:

o	SSGETCC	-	Get Control Card Routine (3.2.2)
o	SSTYPDET	-	Card Type Determination Routine (3.2.3)
o	SSLDSS	-	Load SS Processing Phase (3.2.10)

After all SS control cards are processed, and the appropriate SS interface tables and queues are constructed, control shall be passed to SSLDSS (3.2.10).

3.2.1.4 SSINIT Control Routine Data Organization

The entire table and queue area of approximately 5000 bytes shall be established at the end of the 44PS transient area. The table and queue area shall then be used by the appropriate SS control card processing routines for constructing their tables and queues for SS interface.

3.2.1.5 SSINIT Control Routine Limitations

There are no known or anticipated limitations of the SSINIT Control Routine.

3.2.2 SSINIT Get Control Card Routine (SSGETCC)

This routine shall obtain SS Control Cards, one at a time, from the card reader and shall determine the type of control statement.

3.2.2.1 SSGETCC Description

This subroutine shall be used by SSINIT to obtain a control card. It shall also be used by the SS Continuation Card Routine (SSCONTIN) to obtain a continuation card.

If the control card obtained is not an SS control card, the simulation job shall be cancelled. If the card is an SS control card, control shall return to the calling routine.

3.2.2.3 SSGETCC Interfaces

SSGETCC shall issue a SVC READ to obtain a control card. All input data to SSGETCC shall come from the card reader and shall consist of SS control cards. The format of the SS control cards is described below.

The card-type processing routine for the SS DDI control card is the SSINIT Input Equivalence and Data Definition Routine (SSEQUIVI). The format of the SS DDI card, for data input from tape, is:

```
        ccl     6
       *DDI     prog,yyy(xxx,nnn)(xxx,nnn)---etc.
```

where	*DDI	-	identifies this as an SS Input Device Equivalence Control Card
	prog	-	is the program unit name (up to 4 characters)
	yyy	-	is the 44PS real input tape address. The last three symbolic characters that would be used in the allocation of this device on the 44PS ACCESS or ALLOC control cards replace yyy.
	xxx	-	is the DCSG simulated input device address (hex value)
	nnn	-	is the file number of the data

The format of the SS DDI card, for data input from core storage, is:

```
        ccl     6
       *DDI     prog,CRE(xxx,1111)(xxx,1111)---etc.
```

where:	*DDI	-	identifies this as an SS Input Device Equivalence Control Card
	prog	-	is the program unit name (up to 4 characters)

CRE	-	is the keyword which identifies the 44PS real input device address as being core storage
xxx	-	is the DCSG simulated input device address (hex value)
llll	-	is the symbol within the program unit where the xxx data is located. From one through four character symbols are allowed.

The card-type processing for the SS DDO control card is the SSINIT Output Equivalence and Data Definition Routine (SSEQUIVO). This control card is used to define output device equivalence. The format of the SS DDO card is:

```
        ccl        6
         *DDO      prog,yyy(xxx,xxx,xxx,---)
```

where:	*DDO	-	identifies this as an SS Output Device Equivalence Control Card
	prog	-	is the program unit name (up to 4 characters)
	yyy	-	is the 44PS real output device address. The last three symbolic characters that would be used in the allocation of the device on the 44PS ACCESS or ALLOC control cards replace yyy.
	xxx	-	is the DCSG simulated output device address (hex value)

The card-type processing routine for the SS DEB control card is the SSINIT Debugging Aids Routine. The format of the SS DEB card is:

```
        ccl        6
                                                        eeee          eeee
         *DEB      prog,yyy  ,UNC  ,PCB  ,PVC   (t,ssss,$$$$  (t,sss,$$$$
                                                                   ---etc.
```

where:	*DEB	-	identifies this as an SS Debugging Aids Control Card
	prog	-	is the program unit name (up to 4 characters)

		or the keyword DCSG
yyy	-	is the 44PS output device for debugging aids. The last three symbolic characters that would be used in the allocation of this device on the 44PS ACCESS or ALLOC control cards replace yyy.
UNC	-	Universal Common of the DCSG programs is to be dumped (optional).
PCB	-	Program Control Block of the specified program unit is to be dumped (optional).
PVC	-	Private common of the specified program unit is to be dumped (optional)
t	-	type of dump limits for the following pair: S - for symbolic H - for hexadecimal
ssss	-	starting address from which a core dump is to be taken (symbolic or hexadecimal according to t)
eeee	-	ending address of core dump (symbolic or hexadecimal according to t)
$$$$	-	core dump is to be taken to end of core storage

NOTE: 1. UNC, PCB and PVC may appear on the card in any order.

2. ssss and eeee must be (1) symbols or (2) hexadecimal displacements from the start of the program unit. Symbols and displacements can be mixed on a control card, but not mixed within parentheses. A maximum of four pairs of dump limits may be specified for a program unit.

The card-type processing routine for the SS QUE control card is the SSINIT DCSG Executive Queue Dump Routine. The format of the SS QUE card is:

```
        ccl     6
         *QUE   prog,yyy(que1,que2,que3,-----)
```

where:	*QUE	-	identifies this as an SS Queue Dump Control Card
	prog	-	is the program unit name (up to 4 characters)
	yyy	-	44PS output device address. yyy is equivalent to the last three characters of a symbolic device address as would be specified on the 44PS ACCESS or ALLOC control card.
	que1,que2,..		are the DCSG ECS Queue names (up to 4 characters) scheduled for dumping at the execution-completion of the program unit specified by "prog".

The card-type processing routine for the SS EDD control card is the SSINIT Event Data Routine. The format of the SS EDD card is:

```
        ccl     6
         *EDD   yyy,ff
```

where:	*EDD	-	identifies this card as the Event Data Descriptor
	yyy	-	PSCS/LPSS input device containing the event data. yyy is equivalent to the last 3 characters of the symbolic unit name as would be specified on a 44PS ALLOC or ACCESS control card.
	ff	-	file number in which the data is located on yyy

The card-type processing routine for the SS IOE control card is the SSINIT I/O Events Routine. The format of the SS IOE card is:

```
          ccl       6
            *IOE    (zxx,tttt,cccc)(zxx,tttt,cccc)...etc.
```

where:	*IOE	-	identifies this an an I/O Event Data card
	z	-	if non-zero, data corresponding to this interrupt shall be found on the IOE file. If zero, no data corresponding to this interrupt shall be found on the IOE file. (one hex digit)
	xxx	-	is the DCSG simulated device address that shall be used to generate the interruption
	tttt	-	time in milliseconds at which interrupt shall be initiated. Up to 7 digits may be used.
	cccc	-	CSW status to be stored when interrupt is generated.

The last SS control card shall be used to denote the end of all SS control cards. The format of the SS END card is:

```
          ccl
            *END
```

where:	*END	-	identifies to SSINIT the end of all SS control cards.

A list of the subroutines called by the SSGETCC routine is:

- o SSCHKCRD
- o 44PS SVC READ, WRITE and CANCEL services.

SSINIT shall be the only routine to call SSGETCC.

3.2.2.4 SSGETCC Data Organization

SSGETCC shall be resident within the SS Initialization Phase area.

3.2.2.5 SSGETCC Limitations

The Simulation Job shall be canceled if SSGETCC obtains a card from the card reader that is not an SS control card. The use of the *END control card denotes the endof all SS control cards.

3.2.3 SSINIT Card Type Determination Subroutine (SSTYPDET)

This subroutine shall be used by SSINIT. The functions this subroutine performs are:

- o Recognize a specific type of SS control card, and upon recognition, pass control to the appropriate card type processing routine
- o Ensure that all of a particular type of SS control card is grouped together

3.2.3.1 SSTYPDET Description

The card type, determined by SSGETCC, shall be checked to determine which control statement shall be processed. If it is found that the control card is out of sequence, an error message shall be output and the statement ignored. If the statement is in sequence, the processing routine corresponding to the statement shall be entered.

3.2.3.3 SSTYPDET Interfaces

This subroutine shall use as input the card type flag set in SSGETCC. Upon recognition of *END, the end flag shall be set to halt further control card processing.

A list of the subroutines called by the SSTYPDET routine is:

- SSEQUIVI
- SSEQUIVO.
- SSDEB
- SSQUE
- SSIOEVNT
- EVNTDATA
- Supervisor Call Write services

The only computer program component that shall call SSTYPDET is SSINIT. (3.2.1).

3.2.3.4 SSTYPDET Data Organization

The SSINIT Card Type Determination Routine shall be resident within the SS Initialization Phase area.

3.2.3.5 SSTYPDET Limitations

If a control statement is found to be out of sequence, SSTYPDET shall ignore it.

3.2.4 SSINIT Input Equivalence and Data Definition Subroutine (SSEQUIVI)

This subroutine shall be called by SSTYPDET. The functions this routine performs are:

- o Completes the building of the Input Equivalence Queue
- o Builds an Input Equivalence Table

3.2.4.1 SSEQUIVI Description

This subroutine shall receive control from SSTYPDET upon recognition of the *DDI SS Control Card.

SSEQUIVI shall break out the parameters on the control card and construct the queue and table entries. The real input device specified on the control card may specify core storage or a tape drive. If it is core storage, the symbol shall be placed into the table without converting. Otherwise the symbolic unit specified for the real device shall be converted to its SYSUNI index value. The simulated device address shall be converted to its hexadecimal value. These values shall then be placed into the table.

3.2.4.3 SSEQUIVI Interfaces

This subroutine shall be called by SSTYPDET to process the SS *DDI control card.

The only source of input data to this routine shall be the current SS *DDI control card which is in the input card buffer created by SSCHKCRD.

The output of SSEQUIVI consists of the Input Equivalence Queue List (IQUEUE) and the Input Equivalence Table (ITBL). The format and the interrelationships of the Equivalence Queue and the Equivalence Table are shown in Table 3.2.4-1. The destination of the output is to the table and queue area within core storage.

A list of the subroutines called by SSEQUIVI is:

- SSBRKOUT
- CNVTDA
- CNVRTD
- CNVRT

The only computer program component that shall call SEQUIVI is SSTYPDET (3.2.3).

3.2.4.4 SSEQUIVI Data Organization

SSEQUIVI shall be resident within the SS Initialization Phase area.

3.2.4.5 SSEQUIVI Limitations

There are no known or anticipated limitations.

3.2.5 SSINIT Output Equivalence and Data Definition Subroutine (SSEQUIVO)

This subroutine shall be called by SSTYPDET to process a *DDO control card. The functions performed are:

- o Constructs the Output Equivalence Queue
- o Constructs the Output Equivalence Table

3.2.5.1 SSEQUIVO Description

This subroutine shall receive control from SSTYPDET upon recognition of the *DDO Control Card. SSEQUIVO shall break out the control card parameters and construct the queue and table entries. The 44PS output device specified may be any of the available devices associated with the system. The real and simulated devices shall be converted and placed in the table. The program unit name and the address of the table entry shall be placed into the queue.

3.2.5.3 SSEQUIVO Interfaces

Input to this routine shall consist of the current SS *DDO control card contained in the card buffer created by SSCHKCRD.

The output of SSEQUIVO shall be the Output Equivalence Queue (OQUEUE) and Table (OTBL). The format and the interrelationships of the Equivalence Queue and the Equivalence Table are shown in Table 3.2.5-1.

The subroutines called by SSEQUIVO are:

- o SSBRKOUT
- o CNVTDA
- o CNVRT

SSTYPDET shall be the only routine that calls SSEQUIVO.

3.2.5.4 SSEQUIVO Data Organization

SSEQUIVO shall be resident within the SS Initialization Phase area.

3.2.5.5 SSEQUIVO Limitations

There are no known or anticipated limitations.

3.2.6 SSINIT Debugging Aids Subroutine (SSDEB)

This subroutine shall be used by the SSINIT Card Type Determination Routine to process the SS Debugging Aids control card (SS *DEB card).

SSDEB uses the parameters from the SS *DEB control card to build the tables used by the Simulation Supervisor Debugging Aids Routines for producing program traces and dumps. Traces and dumps are produced automatically if certain execution errors or timer interruptions occur. However, to receive a trace or dump at segment and program unit ends, the user must make a request through the SS *DEB control card - one card per program unit name.

The functions that SSDEB performs are:

- the building of a Debug Queue Table
- the building of a Debug Option List

3.2.6.1 SSDEB Description

SSDEB receives control from SSTYPDET upon recognition of the *DEB SS Control Card. This routine shall break out the parameters, convert the device addresses and construct the queue and table entries.

3.2.6.3 SSDEB Interfaces

This subroutine shall be called by SSTYPDET to process the SS *DEB control card.

The only source of input data to the SSDEB routine shall be the current SS *DEB control card which is in the input card buffer established by SSCHKCRD. The format of the SS *DEB control card is shown in paragraph 3.2.2.3.

The output of the SSDEB routine consists of (1) the Debug Queue Table (DEBQUE) and (2) the Debug Option List (DEBLST). The format and the interrelationships of the Queue Table and the Option List are shown in Table 3.2.6-1. The destination of the output is to the table and queue area within core storage.

A list of subroutines called by the SSDEB routine is:

- SSBRKOUT
- CNVRT
- CNVTDA

The only computer program component that calls SSDEB is SSTYPDET (3.2.3).

3.2.6.4 SSDEB Data Organization

This subroutine shall be resident within the SS Initialization Phase area.

3.2.6.5 SSDEB Limitations

There are no known or anticipated limitations.

3.2.7 SSINIT DCSG ECS Queue Dump Subroutine (SSQUE)

A special debugging aid shall be made available to the system programmers during the development of the DCSG ECS. Specifically, this debugging aid provides the capability to dump the contents of any queue in a contiguous format at the execution-completion of a particular program unit.

In order for this capability to be provided, this routine shall perform the following functions:

- o Build a Queue Dump Table containing the names of the queues to be dumped
- o Set up the interface between the Queue Dump Table and the SS Debugging Aids Routine by building a Queue Dump Queue List

3.2.7.1 SSQUE Description

This subroutine shall be invoked by SSTYPDET. The only exit from this subroutine shall be to the caller. The parameters of the *QUE control card shall be stored in the Queue List along with the count of queues for a program unit. The SYSUNI index and the output device address shall be converted before being placed into the table.

3.2.7.3 SSQUE Interfaces

This subroutine shall be called by SSTYPDET to process the SS *QUE control card.

The only source of input data to SSQUE is the current SS *QUE control card which is in the input buffer established by SSCHKCRD. The format of the SS *QUE control card is shown in paragraph 3.2.2.3.

The output of the SSQUE routine consists of (1) the Queue Dump Queue List (QDQUE) and (2) the Queue Dump Table (QDTBL). The format and the interrelationship of the Dump Queue and the Dump Table are shown in Table 3.2.7-1. The destination of the output is to the table and queue area within core storage.

A list of the subroutines called by the SSQUE routine is:

- o SSBRKOUT
- o CNVTDA

The only computer program component that shall call SSQUE is SSTYPDET (3.2.3).

3.2.7.4 SSQUE Data Organization

This subroutine shall be resident within the SS Initialization Phase area.

3.2.7.5 SSQUE Limitations

There are no known or anticipated limitations.

3.2.8 SSINIT Event Data Descriptor Subroutine (EVNTDATA)

This subroutine shall be called by SSTYPDET to process the SS *EDD control card.

3.2.8.1 EVNTDATA Description

This subroutine shall receive control from SSTYPDET upon recognition of the *EDD control card.

EVNTDATA shall break out the parameters on the *EDD card and construct the header for the I/O Event Table. The file number shall be converted from EBCDIC decimal to hexadecimal and stored in the first byte of the table. The symbolic unit specified for the real device shall be converted to its SYSUNI index value and stored in the second byte of the table. The two-byte record poistion field shall be initialized to zero, and the four-byte pointer field shall be made to point to the first entry in the table.

3.2.8.3 EVNTDATA Interfaces

This subroutine shall be called by SSTYPDET to process the SS *EDD control card.

The input data to this subroutine shall consist of the *EDD control card parameters which have been placed in the input buffer by SSCHKCRD.

The output of EVNTDATA shall consist of the header of the I/O Events Table (EVNTAB). The format of the header is shown in Table 3.1.5-5.

The subroutines which shall be called by EVNTDATA are:

- o SSBRKOUT
- o CNVTDA
- o CNVRTD

The only computer program component that shall call EVNTDATA is SSTYPDET.

3.2.8.4 EVNTDATA Data Organization

EVNTDATA shall be resident within the SS Initialization Phase area.

3.2.8.5 EVNTDATA Limitations

There are no known or anticipated limitations.

3.2.9 SSINIT I/O Events Subroutine (SSIOEVNT)

This subroutine shall be called by SSTYPDET to build the I/O Events Table from the SS *IOE Control Card.

3.2.9.1 SSIOEVNT Description

This subroutine shall receive control from SSTYPDET upon recognition of the *IOE control card.

SSIOEVNT shall break out the parameters on the *IOE card and construct the I/O Event Table entries. For each I/O event the time shall be converted from EBCDIC decimal to hexadecimal and stored in a four-byte field; the simulated device address shall be converted from EBCDIC hexadecimal to hexadecimal and stored in a two-byte field; the CSW status shall be converted from EBCDIC hexadecimal to hexadecimal and stored in a two-byte field. A four-byte end-of-table flag shall be stored after the last entry. This flag shall be overlayed if another *IOE card is read in.

3.2.9.3 SSIOEVNT Interfaces

This subroutine shall be called by SSTYPDET to process the SS *IOE control card.

The input data to this subroutine shall consist of the *IOE control card parameters which have been placed in the input buffer by SSCHKCRD.

The output of SSIOEVNT shall consist of the I/O Events Table (EVNTAB) shown in 3.1.5-5.

The subroutines which shall be called by SSIOEVNT are:

- o SSBRKOUT
- o CNVTDA
- o CNVRTD
- o CNVRT

The only computer program component that shall call SSIOEVNT is SSTYPDET.

3.2.9.4 SSIOEVNT Data Organization

SSIOEVNT shall be resident within the SS Initialization Phase area.

3.2.9.5 SSIOEVNT Limitations

There are no known or anticipated limitations.

3.2.10 SSINIT Load Simulation Supervisor Routine (SSLDSS)

Control shall be passed to SSLDSS by the SSINIT Control Routine after all of the Simulation Supervisor control cards are read, analyzed and appropriate initialization action performed.

The functions that SSLDSS performs are:

- o saving of the 44PS SVC, Machine Check, Program and External new PSW first-level--handler addresses
- o insertion of the Simulation Supervisor's SVC, Machine Check, Program, and External new PSW first-level-handler addresses
- o loading of the Simulation Supervisor Routines and passing of control to the SS set up routine

3.2.10.1 SSLDSS Description

This routine shall be invoked by the SSINIT Control Routine after the processing of all the SS control cards has been completed for the Initialization Phase. SSLDSS shall save the 44PS new PSWs and replace them with those of SS. The processing phase shall then be loaded and control passed to CLKINIT.

3.2.10.3 SSLDSS Interfaces

This routine shall receive control from the SSINIT Control Routine after all of the SS control cards are read and processed.

The output of SSLDSS shall be the loading of the SS Processing Phase routines. These routines shall be in the same format as they were when placed on the Phase Library as output of the 44PS Linkage Editor. The destination of the output is core storage starting at the end of the SS table and queue area.

3.2.10.4 SSLDSS Data Organization

This routine shall be resident within the SS Initialization Phase area.

3.2.8.5 SSLDSS Limitations

There are no known or anticipated limitations of the SSLDSS routine.

3.2.11 SSINIT Check Control Card Routine (SSCHKCRD)

This routine shall check the formats of all SS control cards and list each on the printer. It shall be called by SSGETCC each time an SS control card is read in.

3.2.11.1 SSCHKCRD Description

SSCHKCRD shall scan control cards and check for

- illegal characters embedded in parameters
- parameters of improper length
- illegal delimiters
- characters in improper columns

If any of the above error conditions are found, the card in error and all of its continuations shall be flushed from the input stream and an error message shall be output. Control shall be returned to SSGETCC with an error indication.

If a control card which has no errors contains a non-blank character in column 72, a call shall be made to SSCONTIN to get the continuation card. When control is returned to SSCHKCRD, the continuation card shall also be checked for errors. This process shall continue until all continuation cards have been checked. Control shall than be returned to SSGETCC.

If more than the three allowable continuation cards are present, the parameters on the extra cards shall be ignored and a warning message shall be output, but processing shall continue.

3.2.11.3 SSCHKCRD Interfaces

The only routine which shall call SSCHKCRD is SSGETCC.

SSCHKCRD shall call the following routines:

- o CHKMOVE
- o SSCONTIN

The following conventions shall be observed for communication with other routines:

- o The type code set up by SSGETCC shall indicate the type of control card currently being processed.

  0=DDI
  1=DDO
  2=DEB
  3=QUE
  4=EDD
  5=IOE
  6=END

- o Before entry to CHKMOVE, the character code shall be set to indicate the type of characters which are valid for the parameter being checked.

  0=a symbol which must begin with a letter
  1=a symbol which can begin with a letter or a digit
  2=a hex digit
  3=a decimal digit

- o Before return from CHKMOVE, the character code shall be set to indicate the type of delimiter encountered.

0=comma

1=left paren

2=right paren

3=blank

3.2.114 SSCHKCRD Data Organization

SSCHKCRD shall be resident within the SS Initialization Phase area.

3.2.11.5 SSCHKCRD Limitations

If any error conditions are found on a card, SSCHKCRD shall branch to SSGETCC to flush the card and its continuations from the input stream.

3.2.12 SSINIT Check and Move Routine (CHKMOVE)

This routine shall be called by SSCHKCRD to check and move one parameter to CRDTBL.

3.2.12.1 CHKMOVE Description

This routine shall receive, upon entry, a pointer to a parameter and a character code which indicates the type of characters valid for the parameter. The parameter shall be scanned until a delimiter is encountered. As each character is checked for validity, it shall be moved to an area called CRDTBL. When the entire parameter has been moved to CRDTBL, a comma shall be stored after the last character and the pointer to CRDTBL shall be updated to the location immediately following the comma. The arrangement of parameters in this manner shall facilitate the picking up of parameters by the SSBRKOUT routine.

Upon completion of the scan, the length of the parameter shall be indicated and the character code shall be set to indicate the type of delimiter encountered. If the parameter is found to have an invalid character, to be too long or to end with an invalid delimiter, an error message shall be output and control returned to SSCHKCRD with an error indication.

3.2.12.3 CHKMOVE Interfaces

The only routine which shall call CHKMOVE is SSCHKCRD.

No routines shall be called by CHKMOVE.

The following conventions shall be observed for communication with SSGETCC:

- Before entry to CHKMOVE, the character code shall be set to indicate the type of characters which are valid for the parameter being checked.

  0 = a symbol which must begin with a letter

  1 = a symbol which may begin with a letter or a digit

  2 = a hex digit

  3 = a decimal digit

- Before return from CHKMOVE, the character code shall be set to indicate the type of delimiter encountered.

  0 = comma

  1 = left paren

  2 = right paren

  3 = blank

3.2.12.4 CHKMOVE Data Organization

The CHKMOVE routine shall be resident within the SS Initialization Phase area.

3.2.12.5 CHKMOVE Limitations

There are no known or anticipated limitations.

3.2.13 SSINIT Continuation Card Routine (SSCONTIN)

The SSCONTIN routine shall be called by SSCHKCRD to get continuation cards.

3.2.13.1 SSCONTIN Description

SSCONTIN shall branch and link to SSGETCC to obtain a continuation card. Upon return, the card shall be printed and checked for the following errors:

- No asterisk in column one
- Columns 2-15 non-blank
- Column 16 blank

If no errors are detected, a pointer to column 16 shall be passed to SSCHKCRD upon return. If errors are present, an error indication shall be passed.

3.2.13.3 SSCONTIN Interfaces

The only routine which shall call SSCONTIN is SSCHKCRD.

SSCONTIN shall call the SSGETCC routine.

Output from SSCONTIN shall consist of a pointer to column 16 of the continuation card.

3.2.13.4 SSCONTIN Data Organization

SSCONTIN shall be resident within the SS Initialization Phase area.

3.2.13.5 SSCONTIN Limitations

If more than three continuation cards are encountered, the extra cards shall be read in, but not checked for validity. An indication of the extra cards shall be passed to SSCHKCRD.

3.2.14 SSINIT Parameter Breakout Routine (SSBRKOUT)

The SSBRKOUT routine shall pass control card parameters to the SSEQUIVI, SSEQUIVO, SSDEB, SSQUE, EVNTDATA, and SSIOEVNT routines.

3.2.14.1 SSBRKOUT Description

SSBRKOUT shall pick up the pointer to the next parameter in the CRDTBL area. All characters up to the next comma shall be loaded into registers 0 and 1 left-justified and padded with blanks. The pointer to CRDTBL shall be updated to the position immediately following the comma and control returned to the caller.

If the SSBRKOUT routine is entered and the character pointed to is a blank, an "end of parameters" indication is passed to the caller and the pointer is restored to the beginning of CRDTBL.

3.2.14.3 SSBRKOUT Interfaces

The CRDTBL area used by SSBRKOUT shall be built by the SSCHKCRD and CHKMOVE routines. The table shall consist of all parameters which appeared on a control card and its continuations. The parameters shall be separated by commas, and the last parameter shall be followed by a comma and a blank.

Output from SSBRKOUT shall consist of a paramter passed to the calling routine in registers 0 and 1.

The following routines shall call SSBRKOUT to break out parameters:

- o SSEQUIVI
- o SSEQUIVO
- o SSDEB
- o SSQUE
- o SSIOEVNT
- o EVNTDATA

No routines shall be called by the SSBRKCUT routine.

3.2.14.4 SSBRKOUT Data Organization

SSBRKOUT shall be resident within the SS Initialization Phase area.

3.2.14.5 SSBRKOUT Limitations

There are no known or anticipated limitations.

3.2.15 Convert from EBCDIC to Hexadecimal (CNVRT)

This routine shall convert a four character hexadecimal EBCDIC string to a hexadecimal value.

3.2.15.1 CNVRT Description

Upon entry to this routine, register 0 shall contain the four character EBCDIC string. The high order portion of the character shall be cleared, shifted into position and added to construct the hexadecimal value. If the character is an EBCDIC 10-15(A-F), the high order portion will be cleared and a hexadecimal 10 shall be added to the low order portion of the character before it is shifted and added to construct the hexadecimal value.

3.2.15.3 CNVRT Interfaces

Input to CNVRT:

- o Register 0 shall contain 4 character EBCDIC value to be converted to hexadecimal.

Output from CNVRT:

- o Register 0 shall contain the hexadecimal value.

CNVRT shall not call any other subroutine.

Routines calling CNVRT:

- o SSEQUIVI
- o SSEQUIVO
- o SSDEB
- o SSIOEVNT

3.2.15.4 CNVRT Data Organization

This routine shall be resident within the SS Initialization phase area.

3.2.15.5 CNVRT Limitations

There are no known or anticipated limitations.

3.2.16 Convert to SYSUNI Index Number (CNVTDA)

This routine shall convert a three character code to a SYSUNI index value useable by 44PS I/O service routines.

3.2.16.1 CNVTDA Description

This routine shall use a table to look up the SYSUNI index value. The table shall contain a list of all possible 3 character codes as explained in the SS control statement descriptions.

3.2.16.3 CNVTDA Interfaces

Input to CNVTDA:

- Register 0 shall contain the three character code, right justified.

Output from CNVTDA:

- Register 0 shall contain the SYSUNI index value.

CNVTDA shall not call any subroutines.

Subroutines calling CNVTDA are:

- SSEQUIVI
- SSEQUIVO
- SSDEB
- SSQUE
- SSIOEVNT

3.2.16.4 CNVTDA Data Organization

This routine shall reside within the SS Initialization Phase area. The table used by CNVTDA shall contain the last three characters contained in Table 8, "Symbolic Unit (SYSUNI) Index Values", under Symbolic Unit. This table is found in IBM System/360 Model 44 Programming System - Guide to System Use Form #C28-6812.

3.2.16.5 CNVTDA Limitations

There are no known or anticipated limitations.

3.2.17 Convert from EBCDIC Decimal to Hexadecimal (CNVRTD)

This routine shall convert a decimal EBCDIC string up to 7 characters long to a hexadecimal value.

3.2.17.1 CNVRTD Description

Upon entry to this routine registers 0 and 1 shall contain the right-justified EBCDIC string, and register 2 shall contain a count of the number of characters to be converted. Each byte, starting with the low-order digit, shall be picked up and ANDed with 0F to clear out the zones. It shall then be multiplied by the appropriate power of ten and added to the hexadecimal value being constructed. The total shall be returned to the calling routine in a register.

3.2.17.3 CNVRTD Interfaces

Input to CNVRTD:

- o Registers 0 and 1 shall contain the right-justified EBCDIC string which is to be converted.
- o Register 2 shall contain a count of the number of characters in the string.

Output from CNVRTD:

- o The hexadecimal value shall be returned in a register.

Routines calling CNVRTD:

- o SSEQUIVI
- o EVNTDATA
- o SSIOEVNT

3.2.17.4 CNVRTD Data Organization

This routine shall be resident within the SS Initialization phase area.

3.2.17.5 CNVRTD Limitations

There are no known or anticipated limitations.

3.2.18 Machine Check First Level Handler (MCKFLIH)

MCKFLIH shall be added to the standard System 360/44 Programming System (44PS). Its function shall be to interrogate all machine check interruptions encountered during the simulation.

3.2.18.1 MCKFLIH Description

This routine shall place the simulation in "time-out" mode. It then determines if SS or DCSG had control. If SS had control, a message shall be output and control given to SSEXIT. If DCSG had control, program status shall be saved. The interrupt flag, register 12, shall be set to 12 and register 1 shall be set to the machine check old PSW address. Next, control shall be given to SVCFLIH2.

3.2.18.3 MCKFLIH Interfaces

MCKFLIH Inputs are:

- Interval Timer
- Machine Check old PSW
- The 16 general registers

Outputs are:

- Interrupt flag in register 12
- Old machine check PSW address in register 1

Called routines are:

- SSAVER
- SSEXIT
- SVCFLIH2
- 44PS SVC WRITE service routine

Calling routines are:

None. Control shall be received when a machine check interruption occurs.

Tables and constants referenced:

- Machine Check old PSW

3.2.18.4 MCKFLIH Data Organization

This routine shall reside within the 44PS area.

3.2.18.5 MCKFLIH Limitations

There are no known or anticipated limitations

3.2.19 Program Interruption First Level Handler (PIFLIH)

PIFLIH shall be added to standard 44PS. Its function shall be interrogation of all program interruptions encountered during simulation.

3.2.19.1 PIFLIH Description

This routine shall place simulation in "time-out" mode. It then determines if interruption occurred during SS or DCSG processing. If SS was in control, a message shall be output and control given to SSEXIT to terminate the simulation. If DCSG was in control, status shall be saved. The interrupt flag, register 12, shall be set to 0 and register 1 shall be set to the address of the Program old PSW. Next, control shall be given to SVCFLIH2.

3.2.19.3 PIFLIH Interfaces

PIFLIH Inputs are:

- o Interval Timer
- o Program Interruption old PSW
- o The 16 general registers

Outputs are:

- o Interrupt flag in register 12
- o Old Program PSW address in register 1

Called routines are:

- o SSCNTMCI
- o SVCFLIH2
- o SSSAVER

Calling routines are:

None. Control shall be received when a program interruption occurrs.

Tables and constants used:

- o Program old PSW

3.2.19.4 PIFLIH Data Organization

This routine shall reside within the 44PS area of core storage

3.2.19.5 PIFLIH Limitations

There are no known or anticipated limitations.

3.2.20 SVC First Level Handler (SVCFLIH)

SVCFLIH shall be added to the standard 44PS. Its function shall be interrogation of all SVC interruptions during simulation.

3.2.20.1 SVCFLIH Description

This routine shall place the simulation in "time-out" mode. It shall then determine if the SVC was issued by SS or a DCSG program. If issued by SS the SVC shall be tested as a simulation terminator. SSEXIT shall be entered when termination is requested. If the SVC was issued for a 44PS service routine, the SVC handler of 44PS shall be entered.

If the SVC was issued by a DCSG program status shall be saved. Register 12, the interrupt flag, shall be set to 4, and the address of the SVC old PSW shall be placed into register 1.

SVCFLIH2 shall containue processing by saving the old PSW that corresponds to the interrupt type and giving control to SSIRPT.

3.2.20 SVC First Level Handler (SVCFLIH)

SVCFLIH shall be added to the standard 44PS. Its function shall be interrogation of all SVC interruptions during simulation.

3.2.20.1 SVCFLIH Description

This routine shall place the simulation in "time-out" mode. It shall then determine if the SVC was issued by SS or a DCSG program. If issued by SS the SVC shall be tested as a simulation terminator. SSEXIT shall be entered when termination is requested.

If the SVC was issued by a DCSG program status shall be saved. Register 12, the interrupt flag, shall be set to 4, and the address of the SVC old PSW shall be placed into register 1.

SVCFLIH2 shall continue processing by saving the old PSW that corresponds to the interrupt type and giving control to SSIRPT.

3.2.20.3 SVCFLIH Interfaces

SVCFLIH Inputs are:

- o Interval Timer
- o SVC old PSW
- o The 16 general registers
- o Location of the old PSW corresponding to the interruption type

Outputs are:

- o Contents of an old PSW stored in CREGION (INPSW)
- o Interrupt flag in register 12

Called routines are:

- o SSSAVER
- o SSIRPT
- o 44PS SVC Handler

Calling routines are:

- o PIFLIH
- o EXTFLIH
- o MCKFLIH
- o SVC interruption occurrence

Tables and constants used:

- o CREGION - ADSSIRPT, INPSW

3.2.20.4 SVCFLIH Data Organization

This routine shall reside within the 44PS core storage area

3.2.20.5 SVCFLIH Limitations

General Register 10 shall be destroyed when any SS or User routine issues an SVC referencing a 44PS SVC service routine.

3.2.21 External First Level Interrupt Handler (EXTFLIH)

EXTFLIH shall be added to the standard 44PS. Its function shall be interrogation of all interval timer (external) interruptions.

3.2.21.1 EXTFLIH Description

An external interrupt occurs when the interval timer elapses. EXTFLIH shall receive control at this point and the simulation shall be placed in "time-out" mode. A check shall then be made to determine if SS or DCSG was in control. If SS was in control, the interval timer shall be reset to 0 and control returned to the point of interruption. Otherwise, status shall be saved. The interrupt flag, register 12, shall be set to 8, register 1 shall be set to the address of the External Old PSW, and control given to SVCFLIH2.

3.2.21.3 EXTFLIH Interfaces

EXTFLIH Inputs are:

- o External old PSW
- o The 16 general registers
- o Interval Timer

Outputs are:

- o Resetting of the interval timer
- o Interrupt flag in register 12
- o Address of External old PSW in register 1

Called routines are:

- o SVCFLIH2

Calling routines are:

None. Control shall be received by the occurrence of an external interruption.

Tables and constants used:

- o Interval Timer
- o External old PSW

3.2.21.4 EXTFLIH Data Organization

This routine shall reside within the 44PS core storage area

3.2.21.5 EXTFLIH Limitations

There are no known or anticipated limitations.

3.2.22 SS EXIT Routine (SSEXIT)

This routine shall reset the 44PS PSWs. The address pointing to the PSWs shall be saved by the Initialization Phase.

3.2.22.1 SSEXIT Description

PSW addresses shall be picked up from CREGION. SSEXIT shall reset the four new PSWs saved by the Initialization Phase and issue an SVC CANCEL to terminate the simulation.

3.2.22.3 SSEXIT Interfaces

SSEXIT Inputs shall be:

- o The address of the saved PSW to be reset

Outputs are:

- o Resetting 44PS SVC new PSW
- o Resetting 44PS External new PSW
- o Resetting 44PS Program new PSW
- o Resetting 44PS Machine Check new PSW

An SVC 15 (CANCEL) shall be issued to terminate the simulation.

Tables and constants used:

- o CREGION - STAT44A
- o STAT44 - All items

3.2.22.4 SSEXIT Data Organization

This routine shall reside within the modified 44PS section.

3.2.22.5 SSEXIT Limitations

There are no known or anticipated limitations.

3.2.23 Status Saving Routine (SSSAVER)

This routine shall save the contents of the 16 general registers and the interval timer.

3.2.23.1 SSSAVER Description

This routine shall store the contents of the registers directly, except for register 1 whose contents shall be found in a temporary save area. The value of the interval timer shall also be found in the temporary save area, and it shall be picked up and placed in CREGION along with the registers.

3.2.23.3 SSSAVER Interfaces

SSSAVER Inputs are:

- o The contents of the 16 general registers
- o The value of the interval timer

Outputs are:

- o The contents of the 16 general registers placed in CREGION
- o The interval timer value placed in CREGION

This routine shall not call any other routines.

This routine shall be called by:

- o SVCFLIH
- o MCKFLIH
- o EXTFLIH
- o PIFLIH

Tables and constants referenced:

- o CREGION - GRS, ITVAL
- o SSTEMPSV (contents of register 1 and the interval timer shall be found here)

3.2.23.4 SSSAVER Data Organization

This routine shall reside within the 44PS core storage area.

3.2.23.5 SSSAVER Limitations

There are no known or anticipated limitations.

3.2.24 SS Set Up Routine (CLKINIT)

This routine shall be entered at the conclusion of SS Initialization Phase processing. The functions it performs are:

- o Initialize the interval timer.
- o Load the user provided routines and give control to SSAMUIN to initialize the DCSG ECS
- o Give control to the DCSG ECS

3.2.24.1 CLKINIT Description

Upon entry, CLINIT shall set up addressibility for itself and the rest of SS. The address of the interrupt processor (SSIRPT) shall be stored in the communication region. The address of CREGION shall be entered to initialize the DCSG ECS. An initial value shall then be set into the DCSG ECS interval timer location, CREGION, and the interval timer. Next, control shall be given to the DCSG ECS address specified in the ECS IPL PSW.

If the user routines are not found in the phase library, the simulation job shall be cancelled.

3.2.24.3 CLKINIT Interfaces

CLKINIT shall be invoked at the conclusion of the Initialization Phase.

The address of CREGION shall be obtained by issuing an EXTRACT SVC. DCCHK and ADSSIRPT shall be set during CLKINIT processing.

Subroutine SSAMUIN shall be a user provided routine that performs Initial Program Loading of the DCSG. SSAMUIN can use any of the routines available to SS, since SS base registers (3, 4, and 5) shall be set. After bringing in the ECS, SSAMUIN shall return to CLKINIT with an indicator in register 0. This indicator shall be set to 0 if the IPL procedure was normal, and non-zero if an error occurred.

Restrictions placed upon SSAMUIN are:

- o The contents of registers 3, 4, and 5 may not be destroyed
- o Return to CLKINIT shall be made by branching on register 14
- o Base registers must be set upon entry to SSAMUIN

SSAMUIN Inputs consist of:

- o Register 12 pointing to COMREG
- o Starting address of where to load the ECS shall be contained in COMREG

The outputs SS shall expect from SSAMUIN are:

- o Loading of the DCSG ECS.
- o Indicator in register 0

Routines called by CLKINIT are:

- o SSAMUIN
- o 44PS SVC service routines
- o DCSG ECS

Routines calling CLKINIT are:

- o SSLDSS

3.2.24.4 CLKINIT Data Organization

This routine shall reside within the SS Processing Phase area of core storage.

3.2.24.5 CLKINIT Limitations

There are no known or anticipated limitations.

3.2.25 Interrupt Determination Routine (SSIRPT)

SSIRPT shall receive control from the first level interrupt handlers (PIFLIH, EXTFLIH, MCKFLIH, SVCFLIH). It shall determine the type of interruption and pass control to the proper processor.

3.2.25.1 SSIRPT Description

This routine shall create an old PSW in COMREG which is like the one the DCSG programs expect. This is done so SS may monitor the items in the PSW and regulate the flow of events. After creating the PSW, SSIRPT determines if an external interruption caused entry. If it did, EXTH recieves control. If not, segment running time and DCSG total running time are incremented with a call to UPDTCLK. After regaining control, SSIRPT shall relinquish control to one of the specialized interrupt handlers (PIH, SVCH, MCKH).

3.2.25.3 SSIRPT Interfaces

SSIRPT Inputs shall be:

- o Interrupt flag

Outputs are:

- o Creation of DCSG PSW for SS interrogation
- o Base register settings for the remainder of SS

Called routines are:

- o EXTH
- o PIH
- o UPDTCLK
- o SVCH
- o MCKH

Tables and constants used:

- o CREGION - DUMPSW, INPSW
- o COMREG - OPSW

3.2.25.4 SSIRPT Data Organization

This routine shall be resident within the SS processing phase area.

3.2.25.5 SSIRPT Limitations

There are no known or anticipated limitations.

3.2.26 Update Interval Timer (UPDTCLK)

This routine shall increment the DCSG and Segment running times and reset the DCSG ECS interval timer location.

3.2.26.1 UPDTCLK Description

UPDTCLK shall increment total DCSG and segment running times. This shall be accomplished by subtracting the current interval timer value from the timer value at which SS last had control. This time, in micro seconds, shall be converted to milliseconds and added to the DCSG and segment running times. Next, the DCSG timer location shall be examined. If the current value is unchanged from the value at last SS control, the DCSG timer location is reduced by the Δt computed from the interval timer. If the current value is changed, the DCSG timer location is converted and stored in the interval timer location. This conversion is effected by multiplying the DCSG timer value by $77\left(\frac{10^3 \text{ micro seconds}}{\text{milliseconds}} \times \frac{1}{13}\right)$. Before exit from UPDTCLK the current times are saved for the next entry.

3.2.26.3 UPDTCLK

This routine requires that register 13 point to CREGION.

Output from this routine shall consist of:

- o Resetting the ECS interval timer location
- o Updating DCCHK, ITVAL, and DCSGLAST

UPDTCLK shall not call any other routine.

SSIRPT shall be the only calling routine.

Tables and constants used are:

- o CREGION - ITVAL, SEGRUN, DCCHK
- o COMREG - SIMTIM

3.2.26.4 UPDTCLK Data Organization

This routine shall reside within the SS processing phase area.

3.2.26.5 UPDTCLK Limitations

This routine shall assume the use of a high resolution timer.

As DCSG timer values increase, SS timing error shall increase. This results from the conversion factor between the 4Pi timer and the PSCS/LPSS timer. The conversion factor could be carried at greater accuracy but this would not be commensurate with other timing considerations.

Timing accuracy may be effected if the DCSG timer location is changed more than once between SS control points.

3.2.27 Program Interruption Handler (PIH)

This routine shall determine the cause of a Program interruption. There shall be four groups of instructions determined:

- o I/O
- o Status - switching
- o Privileged, other than above
- o Other exceptions (operation, addressing, etc).

3.2.27.1 PIH Description

PIH shall determine if an "other exception" (error) caused the interruption. If it did, SSDBUG shall be entered to output debugging information. If it was not an exception, PIH shall determine if the interruption was caused by the ECS executing one of the I/O instructions. If this is found to be true, a user provided routine shall be enterd. If the instruction was found to be a LPSW or SSM, routines LPSWH and SSMH shall be entered. If none of the above operations caused the interruption SSPROP, a user provided routine, shall be entered.

3.2.27.1.1 Requirements for user provided routines

3.2.27.1.1.1 SSSIO

This routine shall be called by PIH when a SIO instruction is encountered. SSSIO shall simulate the required I/O operation.

3.2.27.1.1.1.1 Inputs to SSSIO

The I/O Equivalence Queues and tables, along with table COMREG, shall contain all inputs necessary to insure proper simulation. General register 12 shall contain the address of COMREG. The input items within COMREG shall be:

- o Program unit name initiating the I/O
- o Old Program Interruption PSW
- o Channel Address Word
- o Starting address of DCSG ECS
- o Pointer to I/O Equivalence Queues
- o Actual instruction causing interrupt
- o Address of Datalook Subroutine

See 3.1. for a complete description of the table and items.

Register 13 shall point to the contents of the DCSG registers which may be used in forming the DCSG device address.the registers are stored from 0-15.

3.2.27.1.1.1.2 Outputs expected from SSSIO

In addition to simulating the requested I/O, SS expects the following items (if applicable during simulation) to be set in COMREG:

- o A condition code setting, that may be tested by the DCSG ECS, placed in OPSW
- o A CSW that may be interrogated by the DCSG program in CSWL
- o A return address setting in RETAD (see 3.1.5.6)

3.2.27.1.1.1.3 SSSIO Restrictions

The following restrictions and conventions must be adhered to:

- o Registers 3 and 4 may be used if reset before calling or returning to any SS routine.

- SVC conventions as stated in IBM System/360 Model 44 Programming System Guide to System Use (Form C28-6812)
- A base register must be provided
- References to low core should not be made
- Exit from SSSIO shall be to the address specified in item RTNAD of COMREG
- SS subroutine linkage

3.2.27.1.1.2 SSHIO

This routine shall be called by PIH when a HIO instruction is encountered. The function of this routine shall be to simulate the HIO request.

3.2.27.1.1.2.1 Inputs to SSHIO

SS shall make available all inputs specified in 3.2.27.1.1.1.1.

3.2.27.1.1.2.2 Outputs from SSHIO

The outputs from SSHIO should be the same as 3.2.27.1.1.1.2.

3.2.27.1.1.2.3 SSHIO Restrictions

SS shall expect SSHIO to conform to the restrictions and conventions specified in 3.2.27.1.1.1.3.

3.2.27.1.1.3 SSTIO

This routine shall be called by PIH upon recognition of a TIO instruction. SSTIO shall be expected to simulate the request.

3.2.27.1.1.3.1 Inputs to SSTIO

The inputs to SSTIO shall be the same as 3.2.27.1.1.1.1.

3.2.27.1.1.3.2 Outputs from SSTIO

The outputs from SSTIO should be the same as 3.2.27.1.1.1.2.

3.2.27.1.1.3.3 SSTIO Restrictions

SS shall expect SSTIO to conform to the restrictions and conventions specified in 3.2.27.1.1.1.3.

3.2.27.1.1.4 SSTCH

This routine shall be called by PIH upon recognition of a TCH instruction. SSTCH shall be expected to simulate the request.

3.2.27.1.1.4.1 Inputs to SSTCH

The inputs to SSTCH shall be the same as 3.2.21.1.1.1.1.

3.2.27.1.1.4.2 Outputs from SSTCH

The outputs from SSTCH should be the same as 3.2.27.1.1.1.2

3.2.27.1.1.4.3 SSTCH Restrictions

SS shall expect SSTCH to conform to the restrictions and conventions specified in 3.2.27.1.1.1.3.

3.2.27.1.1.5 SSPROP

This routine shall be called by PIH upon encountering any privileged operation not already mentioned in this section. SS shall expect SSPROP to recognize the operation and take appropriate action.

3.2.27.1.1.5.1 Inputs to SSPROP

All items in COMREG shall be available. Register 12 shall contain the address of COMREG. Register 13 shall point to the contents of the DCSG registers.

3.2.27.1.1.5.2 Outputs from SSPROP

The return flag must be set upon return to SS.

3.2.27.1.1.5.3 SSPROP Restrictions

SS shall expect SSPROP to conform to the restrictions and conventions specified in 3.2.27.1.1.1.3.

3.2.27.1.1.6 Terminating the Simulation

If, for any reason, the simulation is to be terminated, an SVC 98_{10} should be issued. This shall reinstate 44PS status and cancel the remainder of the job. An appropriate message may be output by using the SS Debugging Aids Routines.

3.2.27.1.1.7 SS Resident Routines

This section shall reference SS routines that may be of value to the user.

3.2.27.1.1.7.1 SSDBUG

This routine may be entered for the purpose of outputting trace information of the program unit that had control at the time of interruption.
(see 3.2.34)

3.2.27.1.1.7.2 SSPDUMP

This routine may be entered to output selective dumps of core. It may also be used, if desired, to output DCSG outputs after encountering a SIO. Section 3.2.35 contains the SSPDUMP description.

3.2.27.1.1.7.3 DATALOOK

This routine shall provide the core location of the inputs specified in the equivalence tables for data located in core (see 3.2.37).

3.2.27.3 PIH Interfaces

PIH shall expect register 13 to contain the address of CREGION. If the operation was a SIO, the contents of DCSG register 0 must contain the Program Index Value (PIX) for use by SS to index into the DCSG ECS Resident Program List (RPL) to extract the program unit name that initated the request.

Outputs from PIH are:

- o Return flag shall be set
- o Register 12 shall be set to start of COMREG

Routines called:

SSPROP
SSSIO
SSTIO
SSHIO
SSTCH
LPSWH
SSMH
SSDBUG
SSRTN

SSIRPT shall be the only calling routine.

Tables and constant used:

- o CREGION - INPSW, DCSGST, ADECSCR
- o COMREG - PUNAME, RETAD, INSTR, OPSW, CAWL
- o ECSCOMRG - SLRPL

3.2.27.4 PIH Data Organization

This routine shall reside within the SS processing phase area.

3.3.27.5 PIH Limitations

There are no known or anticipated limitations.

3.2.28 SVC Interruption Handler (SVCH)

This routine shall determine if the SVC issued should cause SS intervention. The SVCs that shall cause SS activity are:

- o PUEND
- o SEGEND
- o PDUMP
- o 44PS service routines

3.2.28.1 SVCH Description

SS shall determine if an SVC is of the above type. With PDUMP SVCs register 1 shall be reset from CREGION and PDUMP shall be called. Upon return from PDUMP, the return flag shall be set to 0 and SSRTN shall be entered. If an SVC is a SEGEND or PUEND type, SSDBUG shall be entered to output trace and debugging information. SSRTN shall be called after returning from SSDBUG with the return flag set to 96. If an SVC is issued for a 44PS service routine, all saved registers shall be reset and the SVC reissued. Upon return, the return flag shall be set to 0, and control shall be given to SSRTN. If an SVC is not one of the above, a special user provided routine, SSCKIO, shall be entered. If an I/O interrupt is to be generated, SVCH shall reset the interruption location to point to the SVC instruction. This allows the DCSG ECS to think the I/O interruption occurred before the SVC was issued. The return flag shall then be set to 120 and SSRTN shall be entered.

3.2.28.1.1 SSCKIO

This routine should be inserted to initiate an I/O interruption corresponding to a previous SIO or to an I/O event. If included, SSCKIO will create a more realistic environment for the I/O simulation.

3.2.28.1.1.1 Inputs available to SSCKIO

SS shall make available all inputs specified in 3.2.27.1.1.1. including the address of the I/O events table in COMREG.

3.2.28.1.1.2 Outputs from SSCKIO

The following items in COMREG shall be expected to be set:

- o A condition code setting in OPSW
- o A CSW to be passed to DCSG programs
- o A return address setting in RETAD of hexadecimal 'FF' if an I/O interruption is to be generated; a zero setting if an interrupt is not to be generated.

3.2.28.1.1.3 SSCKIO Restrictions

SS shall expect SSCKIO to conform to the restrictions and conventions specified in 3.2.27.1.1.1.3. Return from SSCKIO shall be made by branching on register 14.

3.2.28.3 SVCH Interfaces

SVCH shall expect register 13 to point to the start of CREGION.

SVCH shall output a return flag setting in RETAD.

Routines called are:

- o SSRTN
- o SSCKIO
- o SSDBUG
- o 44PS SVC service routines

Tables and constants used:

CREGION - GRS, SEGRUN
COMREG - RETAD, OPSW

3.2.28.4 SVCH Data Organization

This routine shall reside within the SS processing phase area.

3.2.28.5 SVCH Limitations

There are no known or anticipated limitations

3.2.29 External Interruption Handler (EXTH)

This routine shall process external interruptions and update DCSG total and segment running times.

3.2.29.1 EXTH Description

The last value set into the DCSG ECS interval timer location DCSGLAST shall be checked against the current value of the DCSG ECS interval timer location. If they are not equal, DCSGLAST shall be added to segment and DCSG total running times. The current value of the DCSG ECS interval timer location shall be converted and set into DCCHK and ITVAL, the return flag shall be set to 0, and SSRTN shall be called. If the values were the same, the running times shall be incremented; a negative one set into the DCSG interval timer location; a value large enough to allow the DCSG ECS to recover shall be set into DCCHK and ITVAL; SSDBUG shall be entered to output debugging information and SSRTN shall be called with the return flag, RETAD, set to 88.

3.2.29.3 EXTH Interfaces

EXTH shall expect register 13 to be set to the start of CREGION.

EXTH shall update running times and set RETAD.

Routines called by EXTH are:

- o SSDBUG
- o SSRTN

Tables and constants used:

- o CREGION - ITVAL, DCSGRUN, SEGRUN, ADECSCR
- o COMREG - RETAD, SIMTIM
- o ECSCOMRG - DCSG ECS interval timer location

3. 2.29.4 EXTH Data Organization

This routine shall reside within the SS processing phase area.

3.2.29.5 EXTH Limitations

There are no known or anticipated limitations.

3.2.30 Machine Check Interruption Handler (MCKH)

This routine shall determine whether the DCSG PSW has its machine check mask on.

3.2.30.1 MCKH Description

Upon entry, MCKH shall set the dump flag to indicate a special dump. A test shall then be made to determine if the DCSG machine check bit is on. It it is, a message shall be output through SSDBUG and the return flag shall be set to 112. If the machine check bit is off a message shall be output and the run terminated.

3.2.30.3 MCKH Interfaces

Register 13 shall be set to CREGION before entry to this routine.

Outputs shall consist of special messages written through SSDBUG and setting of the return flag.

Routines called by MCKH shall be:

- o SSDBUG
- o SSEXIT (through an SVC)

Tables and Constants to be used:

- o CREGION - DUMPSW
- o COMREG - RETAD

3.2.30.4 MCKH Data Organization

This routine shall reside within the SS processing phase area.

3.2.30.5 MCKH Limitations

There are no known or anticipated limitations.

3.2.31 DCSG Return Routine (SSRTN)

This routine shall reset status before issuing a LPSW to return to the DCSG program. SSRTN shall determine where the return is to be made by interrogating the return flag (RETAD).

3.2.31.1 SSRTN Description

The DCSG interval timer location shall be updated with a new value computed in UPDTCLK or EXTH. A CSW value shall be placed in the DCSG CSW location. RETAD shall then be checked to determine where the return point shall be. The return may be to the point of interruption, DCSG external interruption handler, DCSG I/O interruption handler, DCSG SVC interruption handler, DCSG program interruption handler, or DCSG Machine Check interruption handler. If the return is to one of the interruption handlers, the dummy PSW (DUMPSW) shall be reset from the DCSG ECS new PSW location corresponding to the interruption and an old PSW containing the DCSG status previously shall be set.

3.2.31.3 SSRTN Interfaces

Inputs to SSRTN:

- o A flag setting in RETAD
- o An updated old PSW in OPSW

Outputs from SSRTN shall consist of:

- o Resetting a DCSG old PSW location corresponding to the type of interruption to be generated
- o Resetting DCSG general registers

SSRTN shall not call any routines.

Tables and constants used:

- o CREGION - OPSW, GRS, ITVAL, INPSW, DUMPSW
- o COMREG - OPSW, CSW, RETAD

3.2.31.4 SSRTN Data Organization

This routine shall reside within the SS processing phase

3.3.31.5 SSRTN Limitations

There are no known or anticipated limitations

3.2.32 Set System Mask Handler (SSMH)

This routine shall process a SSM instruction issued by the DCSG ECS. If an error is found while processing the instruction, a message shall be output and the error indicated to the DCSG ECS.

3.3.32.1 SSMH Description

This routine shall call ADDCK to compute the address of the new system mask and to determine if an addressing exception would result by the instruction. If it would, debugging information shall be output and OPSW shall be reset to reflect the error. RETAD shall then be set to 104 and control passed to SSRTN. Otherwise, the system mask in DUMPSW is reset. Next, SSCKIO shall be entered to determine if an I/O interruption may be forced on the ECS. If it may, RETAD is set to 120 and control is given to SSRTN. If no I/O interruption is to be generated, control is given to SSRTN with RETAD set to 0.

3.2.32.3 SSMH Interfaces

Inputs to SSMH consists of:

- o Register 13 set to the start of CREGION
- o The SSM instruction in INSTR

Outputs consist of:

- o resetting OPSW
- o resetting DUMPSW
- o setting RETAD

PIH shall be the only routine calling SSMH.

Called routines are:

- o SSRTN
- o ADDCK
- o SSCKIO

Tables and constants used:

- o CREGION - DUMPSW
- o COMREG - RETAD, OPSW

3.2.32.4 SSMH Data Organization

This routine shall reside within the SS processing phase area.

3.2.32.5 SSMH Limitations

There are no known or anticipated limitations.

3.2.33 Load PSW Handler (LPSWH)

This routine shall process a LPSW instruction issued by the DCSG ECS. If the instruction is found to be in error, a message shall be output and the error indicated to the DCSG ECS.

3.2.33.1 LPSWH Description

LPSWH shall call ADDCK to compute the address of the double word specified in the instruction and check if an addressing exception would occur. If an addressing exception would occur, the interruption code in OPSW is set to reflect this, RETAD is set to 104, and control given to SSRTN. If this condition is not found a check is made to determine if a specification error would arise. If it would the interruption code in OPSW is set to reflect this, RETAD is set to 104 and control given to SSRTN. Otherwise DUMPSW +OPSW shall be reset from the computed address. Next SSCKIO shall be entered to determine if an I/O interruption may be forced upon the ECS. If it can, RETAD is set to 120 and control given to SSRTN. Otherwise, SSRTN is entered with RETAD set to 0.

3.2.33.3 LPSWH Interfaces

Inputs to LPSWH consists of:

- o Register 13 set to the start of CREGION
- o The LPSW instruction in INSTR

Outputs consist of:

- o resetting OPSW
- o resetting DUMPSW
- o setting RETAD

PIH shall be the only routine calling SSMH.

Called routines are:

- o SSRTN
- o ADDCK
- o SSCKIO

Tables and constants used:

- o CREGION - DUMPSW
- o COMREG - RETAD, OPSW

3.2.33.4 LPSWH Data Organization

This routine shall reside within the SS processing phase area.

3.2.33.5 LPSWH Limitations

There are no known or anticipated limitations.

3.2.34 Debugging Aids Routine (SSDBUG)

SSDBUG shall provide the debugging support for SS.

The routine shall provide the following services:

- o Traces at segment ends and program unit ends if requested
- o Automatic traces if timer interrupts, program interrupts, or special program errors occur
- o Dump of program unit if program interrupts or timer interrupts occur
- o Snapshots and Dumps of Core with each trace for a program unit, if requested
- o Dumps of selected DCSG Queues at unit end for particular program units if requested

3.2.34.1 Description

The entry point to the Debugging Aids Routine shall be SSDBUG. The calling sequence for the routine shall be: Branch and link to SSDBUG using register 14 as the return register. Register 0 must have a hexadecimal value of 96, 72, 2D, CA, or D9 depending if the trace is a SEGEND, PUEND, Timer Interrupt, Program Interrupt, or Special Trace. If it is a Special Trace and a message is to be output with the trace, register 1 must point to the message buffer. If there is no message, register 1 must contain zeros. The first byte of the message must contain the message length.

Upon entry to SSDBUG, the registers shall be stored and the current program unit name shall be picked up. If the trace originated in the DCSG ECS, DCSG shall be used as the program unit name.

The first word of each entry in the Debug Queue shall be checked to see if it matches the program unit name. If a match is found, the SYSUNI value associated with the real output device shall be picked up from the entry in the table and placed in the I/O Request Control Block (RCB). When no match is found before the end of table mark is encountered and the trace type is a timer, Program Interrupt or Special, the SYSUNI value associated with SYSLST shall be placed in the I/O RCB and a trace shall be given.

Depending on the trace type, traces containing different items shall be output by SSDBUG. However, many items shall be standard for all traces.

The program unit name, contents of DCSG general registers, trace type label, the interrupt location, and timer information shall be standard for all traces. The current segment entry point shall be a trace item except when the trace is a Special trace for the DCSG ECS.

The Interruption Code shall be a trace item for Program Interrupt traces; the next segment entry point shall be given with a SEGEND trace; and program unit entry points shall be given with PUEND traces. Messages, if provided, shall be trace items with Special traces.

SSDBUG shall dump the program unit if the trace is a timer trace or a program interruption trace. If a program interruption occurs in the DCSG ECS, SSDBUG shall dump the DCSG ECS.

SSDBUG shall determine if any dumps were requested for the particular program unit. If dumps were requested, the keyword flags and dump limits shall be used as dump parameters.

When the UNC flag is on, the starting and ending locations of the DCSG universal common shall be set as dump limits for the SSPDUMP routine. SSPDUMP shall be called to dump the area of main storage defined by these dump limits. If the PVC flag is on, the address and size of the program unit's private common shall be picked up from the PCB. The dump limits shall be determined and SSPDUMP shall be called to give a snapshot of that core area.

If the PCB flag is on, the PCB's size shall be found in the PCB and dump limits shall be determined. These limits shall be set as dump parameters and SSPDUMP shall be called again to give a snapshot of the area specified.

The last word of the Debug Queue entry shall be checked to see if it is non-zero. If it is zero, no dump limits were specified for this program unit so the trace type shall be inspected to determine what information is to be output next. However when it is non-zero, it shall point to an entry in the option list.

The pointer shall be loaded into a register and the table entry checked to see if the limits were given as hex displacements or symbols. When they are given as symbols, the symbol length and symbol shall be picked up from the Option List. If they are specified as displacements, the displacements shall be used to determine the effective dump addresses.

The address of the start of the program unit shall be picked up from the PCB. The dump limits shall be picked up and added to the text address to form the dump addresses. These addresses shall be set as parameters for SSPDUMP which shall be called to dump this segment of core.

When the dump limits are determined to be symbols, the length of each symbol shall be loaded and the symbols picked up. The symbol dictionary shall be searched to find the core address of each symbol. When both addresses are determined, they shall be set as dump parameters for the PDUMP routine. If either symbol remains unresolved, no dump shall be given.

A check shall be made to see if any more dumps have been requested. If not, a list shall be made to see if the trace is a PUEND trace. When this test proves positive, a search shall be made of the Queue Dump Queue. If the queue dumps have been requested for this program unit, they shall be output at this time.

Switches that have been set shall be cleared and registers that have been saved restored. Exit shall be made to the calling program.

3.2.34.3 Interfaces

The Debugging Aids Routine SSDBUG shall have the following interfaces:

a. Input

Inputs to SSDBUG shall be a pointer to the external message buffer when a special trace is to be generated.

The Debug Queue and Option List shall contain information about which program units shall require debugging aids and which areas of core shall be dumped with a program trace. Figure 3.1.5.6-1 shows the format and content of Debug Queue and Option List.

The Queue Dump Queue and Queue Dump Table shall contain information about which DCSG queues shall be dumped at the end of which program units. Figure 3.1.5.7-1 shows the format and content of the list and table.

b. Output

Outputs from SSDBUG shall be trace information placed in the PRNTLN buffer area for SSDPUT, the pointers to an external message buffer for SSDPUT, and dump limits which shall be parameters for SSPDUMP. Figure 3.2.34.3-1 is a comparison of the different traces. Figures 3.2.34.3-2 - 3.2.34-6 are schematic comparisons of formats and contents of SEGEND, PUEND, Timer Interrupt, Program Interrupt or Special Trace.

c. Subroutines called:

- SSDPUT (3.6.36)
- SSPDUMP (3.2.35)
- DATALOOK (3.2.37)
- BI2HEX (3.2.39)
- GETPTRS (3.2.40)

Subroutines Calling SSDBUG

The other Computer Program Components which shall call SSDBUG are:

- PIH (3.2.26)
- SVCH (3.2.28)
- EXTH (3.2.29)
- MCKH (3.2.30)
- LPSWH (3.2.33)
- SSMH (3.2.32)

The external tables and buffers which shall be used by SSDBUG are:

- PRNTLN buffer
- DEBUG Queue
- DEBUG Option List
- Queue Dump Queue
- Queue Dump Table
- CREGION

3.2.34.4 SSDBUG Data Organization

There shall be no unique tables for SSDBUG. The SSDBUG routine shall be resident within the SS Processing Phase.

3.2.34.5 Limitations

Messages to be output with Special Traces shall be restricted to 131 bytes.

3.2.34.-1 Debugging Aids Comparison Chart

TRACE	SEGEND	PUEND	TIMER	PROGRAM	SPECIAL
Program Unit Name	X	X	X	X	X[ab]
Trace Type Label	X	X	X	X	X
Location of interrupt (OLD PSW)	X	X	X	X	X
Current Segment Entry Point	X	X	X	X	X[b]
Interruption Code				X	
Next Segment Entry Point	X				
Timer Information	X	X	X	X	X
Unit Entry Point		X			
Contents of Registers	X	X	X	X	X
Message					X
DUMP					
Dump of Program Unit			X	X	
Optional Dumps of Core	X	X	X	X	X[b]
Queue Dumps		X			

a DCSG is listed when interrupt occurred in the DCSG ECS.

b These items are output with a special trace if the trace originated in a program unit.

Figure 3.2.34-2

A Segment End trace consists of the following:

- o the Program Unit Name
- o a label noting segment end
- o the location of segment end
  (PSW at interruption and interrupt point)
- o the current segment entry point
- o the next segment entry point
- o timer information
- o the contents of the general registers
- o optional core dumps

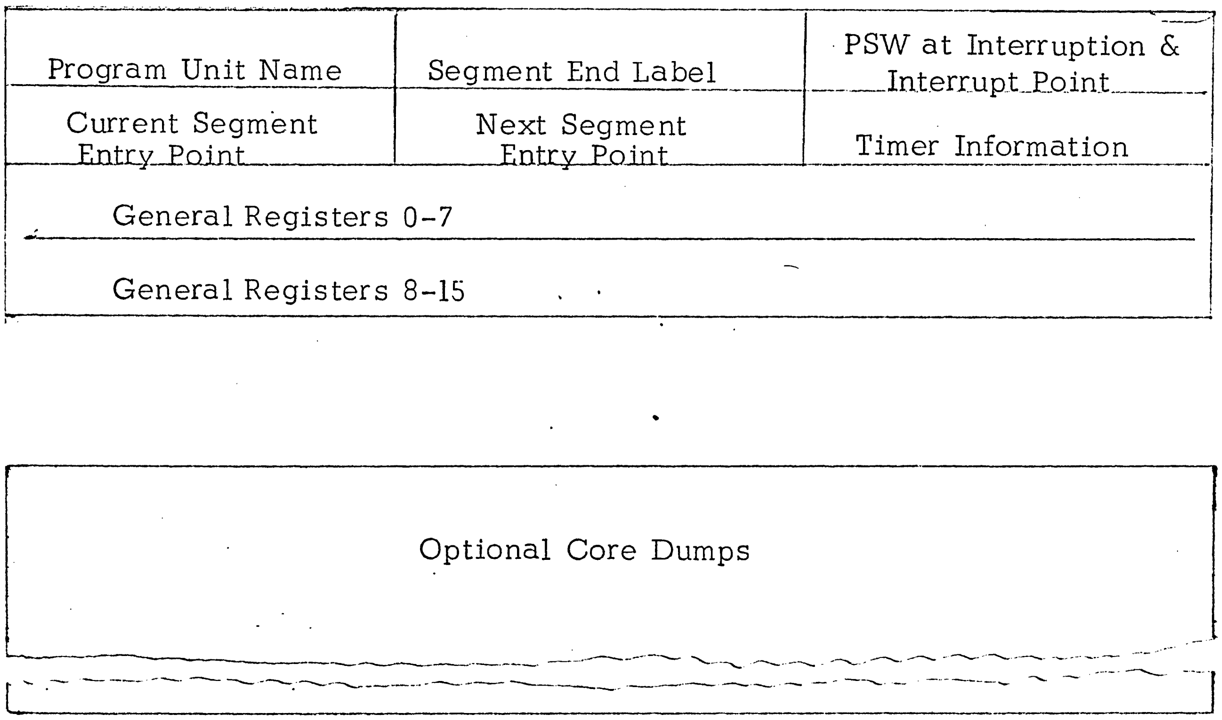


Figure 3.2.34-3

A Unit End trace consists of the following:

- o the Program Unit Name
- o a label noting program unit end
- o the location of program unit end (PSW at interruption of interrupt point)
- o the current segment entry point
- o unit entry point
- o timer information
- o the contents of the general registers
- o optional core dumps
- o queue dumps

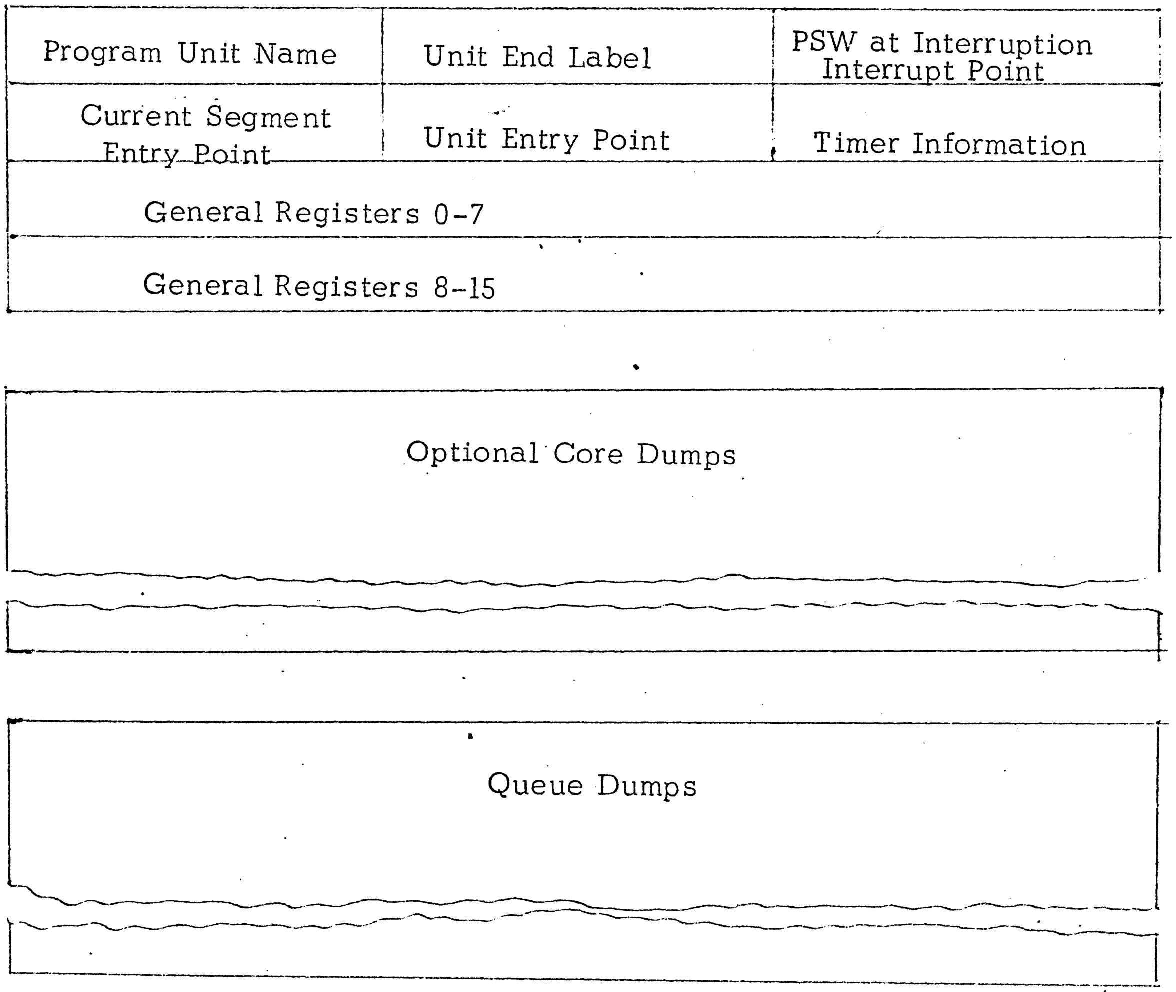


Figure 3.2.34-4

A Timer Interrupt Trace consists of the following

- o the Program Unit Name
- o a label noting timer interrupt trace
- o the interrupt location
  (PSW at interruption & interrupt point)
- o the current segment entry point
- o timer information
- o the contents of the general registers
- o dump of program unit
- o optional core dumps

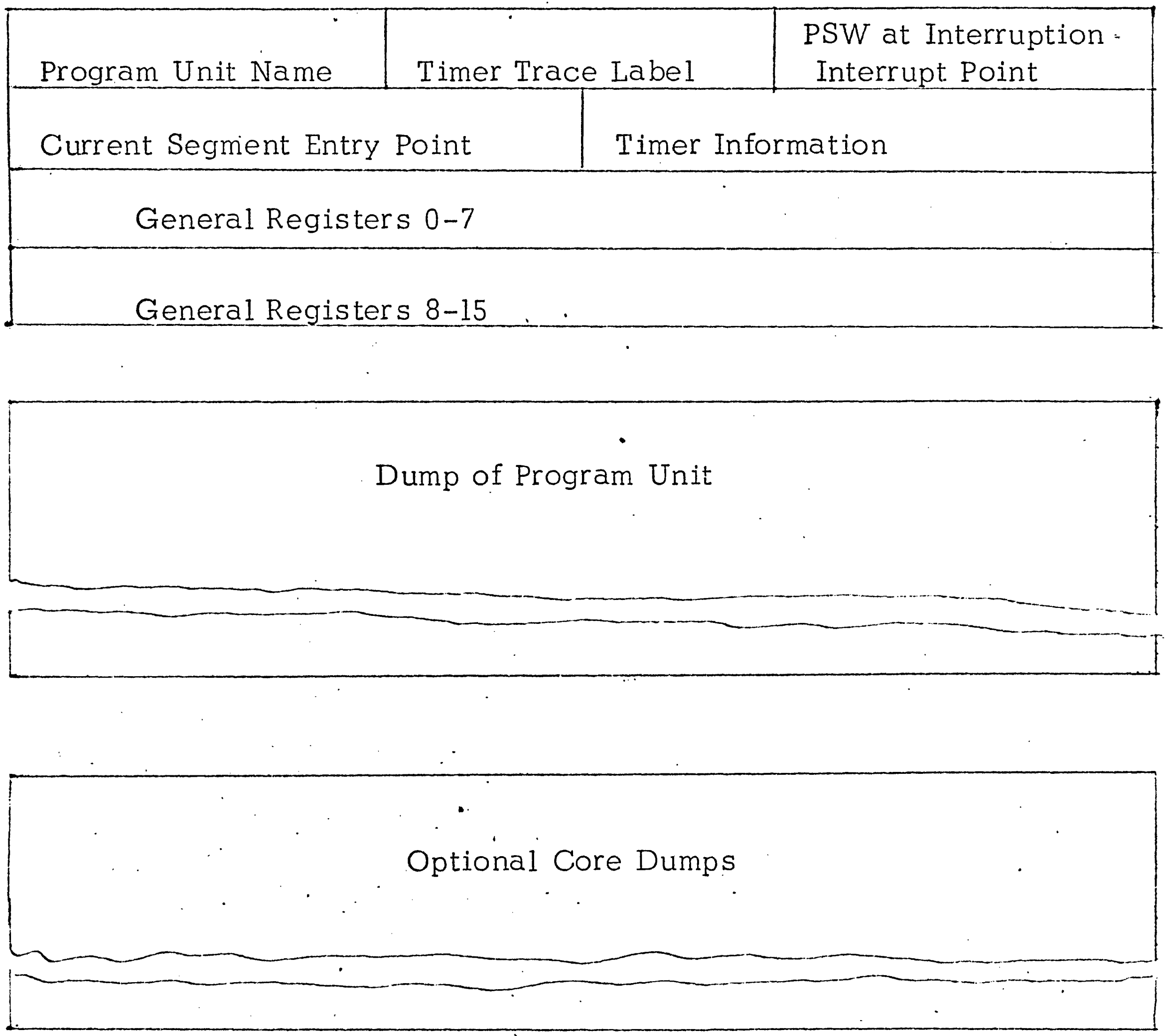


Figure 3.2.34-5

A Program Interrupt Trace consists of the following:

- o the Program Unit Name
- o a label noting program interruption
- o the interrupt location
  (PSW at interruption & interrupt point)
- o the current segment entry point
- o the interruption code
- o timer information
- o the contents of the general registers
- o dump of program unit
- o optional core dumps

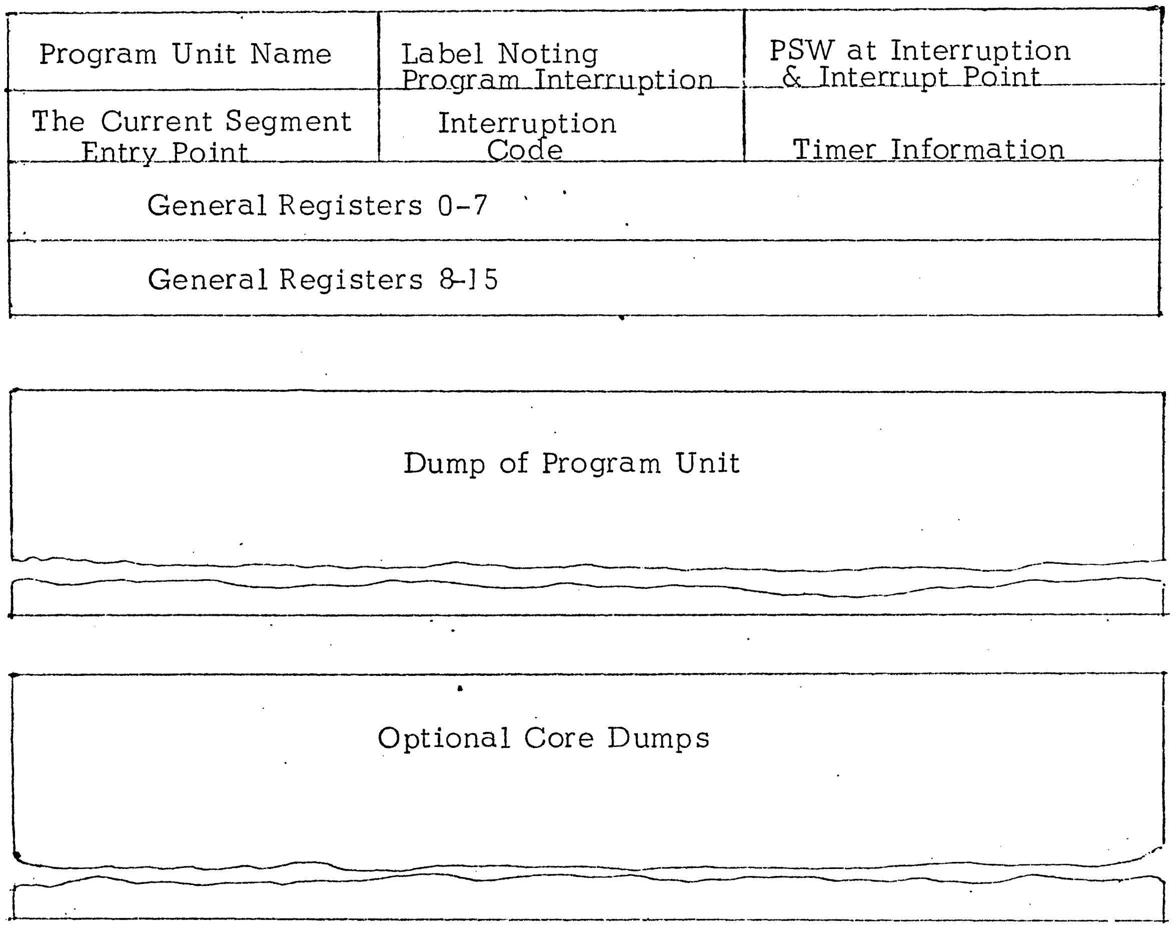
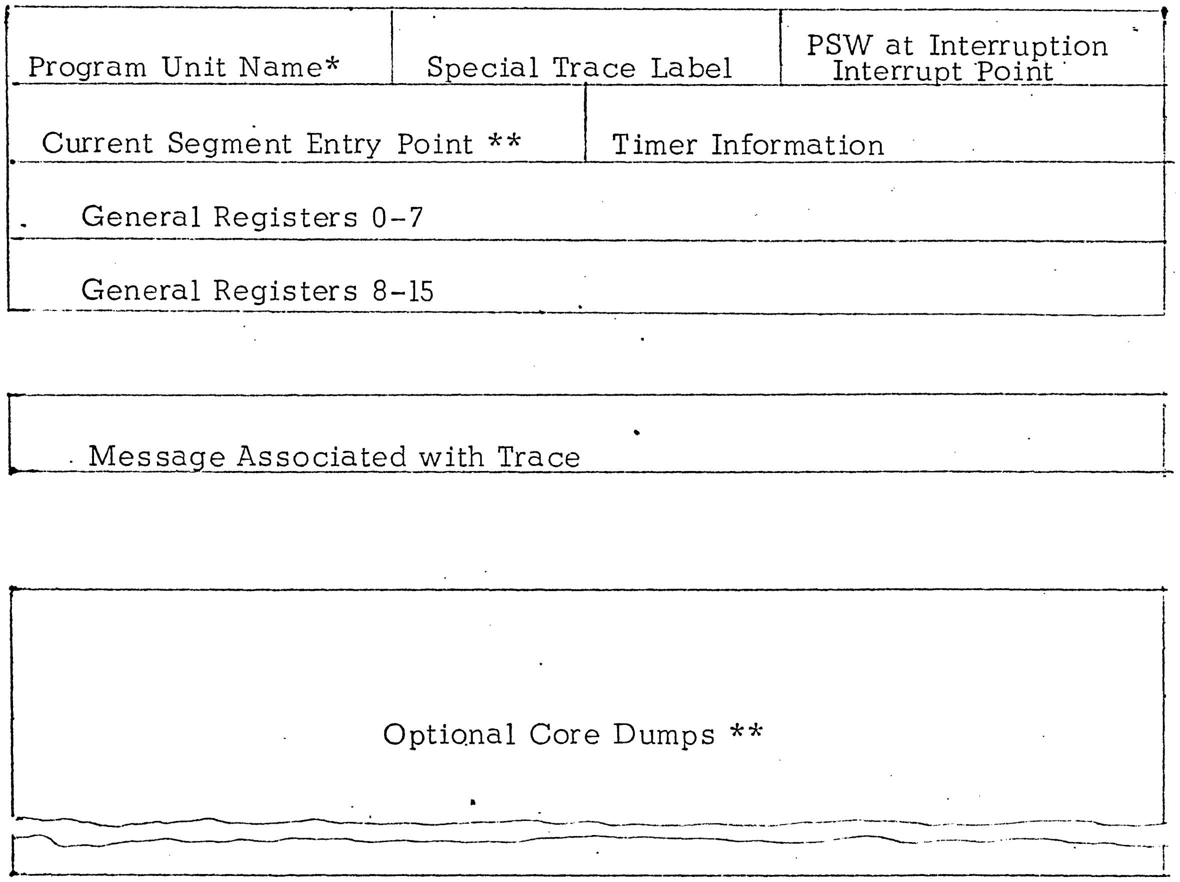


Figure 3.2. 34-6

A Special Trace consists of the following:

- o the Program Unit Name *
- o a label noting a special
- o interrupt location
  (PSW at interrupt & interrupt point
- o the current segment entry point **
- o timer information
- o the contents of the general registers
- o message
- o optional core dumps **



*This item listed as DCSG when Trace originates in executive program.

**These items given when interrupt is in a program unit.

3.2.35 Core Dump Routine (SSPDUMP)

This routine shall provide selective dumps of core storage.

3.2.35.1 Description

The entry point to the SS PDUMP routine shall be SSPDUMP. The calling sequence for PDUMP shall be: Branch and link on register 15. Register one shall point to a parameter list. This list shall be 3 words long, aligned on full-word boundaries. Word one shall contain the starting address of the dump. The first byte of word two shall contain a flag indicating if the contents of the DCSG general registers are wanted with the dump. The byte must be non-zero if the registers are wanted. The rest of word two shall contain the ending address of the dump. The first byte of word three shall contain the SYSUNI index value of the output device and the rest of word three shall point to a message buffer or shall be zeros.

Upon entry to SSPDUMP, initialization for the dump shall be carried out. The line count shall be zeroed for carriage control and the print buffers blanked out. The first dump limit shall be picked up from the parameter list and rounded to the next lower full word boundary unless it is already a full word boundary. The second dump limit shall be rounded to the next higher full word boundary unless it is a full word boundary already.

The second limit shall be compared with the end address of core. If it is larger than the end address of core, the second limit shall be discarded and the end address of core shall become the effective dump limit.

The SYSUNI index shall be picked up from the parameter list and put in the I/O request control block of SSDPUT routine. The last three bytes of the parameter list shall be tested for zero. If they are zero, a blank header shall be set up to cause a skip to a new page. If they are not zero, they shall point to a header message. The address of this message shall be put in the I/O parameter list and SSDPUT shall be called upon to output the message as the header to the dump.

Before each word shall be converted from binary to hexadecimal and put into the print line, a test shall be made to see if the end of the dump has been reached. If it has, the line shall be completed with blanks. Otherwise the current word shall be converted to hex, placed in the buffer, and the next word picked up.

When a line is completed, it shall be compared with the previous line. If it is equal, a flag shall be set on and a message prepared noting that identical lines have been foundd. The next line shall then be processed.

When a comparison proves unequal, a test shall be made to determine if the equal flag has been turned on. If it has, the message shall be output followed by the dump line. However, if the flag is off, the dump line shall be output directly.

A count of the lines shall be maintained. A skip to a new page carriage control command shall be issued each time a page has become full.

<u>3.2.35.3 Interfaces</u>

The following shall be the PDUMP interfaces:

a. <u>Inputs</u>
Inputs to PDUMP shall be the parameters in the three word parameter list. Register one shall point to this list upon entry to PDUMP. The list shall contain the first and second dump limits, the SYSUNI index of the output device, and a pointer to a header message or zeros if no message is provided.

b. <u>Outputs</u>
The outputs from PDUMP shall be hexadecimal dumps of core. The output shall go to the output device specified by the SYSUNI index parameter.

c. <u>Subroutines called</u>

- o SSDPUT (3.2.36)
- o BI2HEX (3.2.39)

d. <u>Subroutines calling SSPDUMP</u>

SSDBUG and user provided routines (if desired) shall be the only subroutines calling SSPDUMP.

e. <u>External Areas</u>

The external tables or buffers used by PDUMP shall be PRNTLN and PRNTLN2, output buffers of SSDPUT.

3.2.35.4 PDUMP Data Organization

PDUMP shall be resident within the SS Processing Phase.

3.2.35.5 Limitations

Dumps shall be given in hexadecimal format only.

3.2.36 Output Service Routine (SSDPUT)

SSDPUT shall provide I/O services for SSDBUG and SSPDUMP.

3.2.36.1 SSDPUT Description

The entry point of the Output Service Routine shall be SSDPUT. Linkage shall be effected by a branch and link instruction using register 2 as the return register.

Upon entry to this routine, register one shall be loaded with the address of a parameter list. A SVC5 (write) shall be given followed by a SVC6 (check). Control shall be returned to the caller.

3.2.36.3 Interfaces

SSDPUT shall have the following interfaces:

a. Input

Inputs to SSDPUT shall be the content of PRNTLN buffer if called by SSDBUG, the contents of PRNTLN and PRNTLN2 if called by SSPDUMP and pointers to message buffers. The contents of PRNTLN and PRNTLN2 shall be changed dynamically depending on the trace or dump being given.

PRNTLN and PRNTLN2 shall be 132 character long buffers. The message buffers shall be of variable length with the first byte of each buffer giving its length.

b. Output

Output from SSDPUT shall be the trace information and dump lines passed to it in PRNTLN and PRNTLN2 and the contents of the external message buffers pointed to by the parameters passed to SSDBUG and SSPDUMP.

The output for a program unit shall go to the device specified on the *DEB card for the program unit.

c. Subroutines called

SSDPUT calls no other subroutines.

d. <u>Subroutines calling SSDPUT</u>

- o SSDBUG
- o SSPDUMP

e. <u>External Tables</u>

The only external tables, buffers, constants, or control registers used by SSDPUT shall be the external message buffers.

3.2.36.4 Data Organization

SSDPUT shall be resident within the SS Processing Phase. PRNTLN and PRNTLN2 buffers shall be unique to SSDPUT although accessed by SSDBUG and SSPDUMP

3.2.36.5 Limitations

SSDPUT shall do no analysis of the supervisor return code in register 15. Any data that does not transmit properly shall be lost. However, safeguards such as incorrect length suppression shall be programmed into the I/O parameter list.

3.2.37 Data Look Up Routine (DATALOOK)

DATALOOK shall provide a symbol dictionary look up for symbols that have been included in a program unit's symbol dictionary.

3.2.37.1 Description

The entry point to the Data Look Up Routine shall be DATALOOK. The calling sequence for the routine shall be: Branch and link to DATALOOK using register 14 as the return register. Register 1 must point to the symbol to be looked up. The symbol area must be two words aligned on a full word boundary. The symbol must be left justified followed by blank padding.

Upon entry to DATALOOK, registers shall be stored and GETPTRS routine called to set up the pointers to the program unit's symbol dictionary.

Upon return from GETPTRS, the dictionary shall be searched for the symbol. If a match is not found, register zero shall be set to zeros and control returned to the calling program. However, when a match is found, the address of the symbol shall be placed in register zero before exit is made.

3.2.37.3 Interfaces

DATALOOK shall have the following interfaces. Input to DATALOOK shall be a pointer in register one to the symbol whose address is desired. Output from DATALOOK shall be the address of the symbol in register zero. If the symbol cannot be found, register zero shall contain zeros. DATALOOK shall call GETPTRS (3.2.40). SSDBUG (3.2.34) and user supplied routine (if desired) shall call DATALOOK. DATALOOK shall use no external areas.

3.2.37.4 Data Organization

DATALOOK shall be resident within the SS Processing Phase.

3.2.37.5 Limitations

There are no known or anticipated limitations to the DATALOOK routine.

3.2.38 Determine Program Unit Name (DETPRUN)

This routine shall pick up the name of the DCSG program unit that had control before the interruption occurred. If the ECS was in control, 'DCSG' shall be used as the program unit name.

3.2.28.1 DETPRUN Description

The problem state mask in OPSW shall be checked to see if it is on. If it is, the location of the PCB for the program unit is obtained from SLPCB of ECSCOMRG and the program unit name is extraced from the PCB and placed into PUNAME. If the mask is off, the location of the interrupt is tested against the end of the DCSG ECS areas. If the interrupt location is greater, it shall be assumed the interruption was caused by a program unit. The program unit name shall then be extraced from its PCB. If the DCSG ECS was in control, 'DCSG' shall be placed into PUNAME.

3.2.38.3 DETPRUN Interfaces

Inputs shall consist of:

- OPSW in COMREG
- Address of current PCB
- Ending address of DCSG ECS

Outputs shall be the program unit name or 'DCSG' placed in PUNAME.
This routine shall be called by SSIRPT.
DETPRUN shall not call any subroutines.
Tables and constants used:

o	COMREG	-	OPSW, PUNAME
o	ECSCOMRG	-	SLPCB, SMCORE

3.2.38.4 DETPRUN Data Organization

This routine shall be resident within the SS processing phase area.

3.2.38.5 DETPRUN Limitations

Since program units are allowed to operate in the supervisor state, it may be impossible to determine if the interruption occurred in the DCSG ECS or a program unit.

3.2.39 Convert Binary to Hex (BI2HEX)

BI2HEX shall convert binary numbers to hexadecimal numbers.

3.2.39.1 Description

The entry point for this routine shall be BI2HEX. The calling sequence shall be: Branch and link to BI2HEX with register 2 containing the return address.

Register usage for this routine shall be the following: Register 7 must contain the data to be converted, left adjusted. Register 8 must contain the count of half-bytes to be converted. Register 5 must point to the area in core where the converted data shall be stored.

Upon entry to the routine, the data shall be converted and Register 5, upon exit shall point to the byte following the last byte converted.

3.2.39.3 Interfaces

Input to BI2HEX shall be the contents of registers 5,7, and 8. Output shall be the hexadecimal equivalent of the data at the location specified by register 5.

BI2HEX shall call no other subroutines.

The subroutines which shall call BI2HEX shall be:

- o SSDBUG (3.2.34)
- o SSPDUMP (3.2.35)

BI2HEX shall use no external areas.

3.2.39.4 Data Organization

BI2HEX shall be a SS resident routine.

3.2.39.5 Limitations

BI2HEX can convert a maximum of 8 half-bytes at a time.

3.2.40 Determine Program Unit Pointers (GETPTRS)

GETPTRS shall determine certain program unit pointers from the current program unit's PCB.

3.2.40.1 Description

The entry point for this routine shall be GETPTRS. The calling sequence shall be: Branch and link to GETPTRS with register 2 containing the return address.

Upon entry to GETPRTS, registers shall be stored and the program unit name shall be checked to see if it is DCSG. When it is, exit shall be made to the calling program. If it is not DCSG, it shall be checked against the first word of the current PCB. If this matches, the program unit pointers shall be determined. When the program unit names do not match, the DCSG RPL shall be checked to determine if the proper program unit can be located. If it cannot, exit is made to the calling program. If it can, program unit pointers shall be determined.

The address of the program unit shall be extraced from the PCB. Next, the address of the program unit text, the text end address, and program unit entry point shall be determined from information in the PCB.

A check shall be made to see if there is a symbol dictionary for this program unit. If the is, the symbol dictionary address shall be determined. If no dictionary exists, the address shall be set to zero.

Registers shall be restored and exit made to the calling program.

3.2.40.3 Interfaces

Input to GETPTRS shall be the pointer to the current PCB from SLPCB and the PCB of the current program unit. Output from GETPTRS shall be the pointers extracted from the current PCB. GETPTRS shall call no other subroutines.

Other subroutines which shall call GETPTRS shall be

- o SSDBUG (3.2.34)
- o DATALOOK (3.2.37)

External areas used by GETPTRS are:

- o COMREG
- o CREGION
- o ECSCOMRG
- o Current PCB

3.2.40.4 Data Organization

GETPTRS shall be a SS resident Routine

3.2.40.5 Limitations

For interrupts caused by an SIO, DCSG register zero must contain a PIX value to be used by GETPTRS to find the correct entry in the RPL.

3.2.41 Address Determination and Check Routine (ADDCK)

This routine shall compute an absolute address using the contents of the specified base register and displacement found in a LPSW or SSM instruction. ADDCK shall also determine if the computed address shall cause an addressing exception.

3.2.41.1 ADDCK Description

ADDCK shall separate the Base register and the displacement from the LPSW or SSM instruction issued by a DCSG program. The contents of the base register shall then be added to the displacement to form the address. Tests shall then be made to determine if the address lies within the DCSG program area. If it does not, an error exit is taken back to the calling routine.

3.2.41.3 ADDCK Interfaces

Input to ADDCK:

- The LPSW or SSM instruction found in COMREG
- The 16 saved registers found in CREGION
- The starting address of the DCSG programs found in COMREG

Output from ADDCK:

- The absolute address shall be placed in register 5

Calling Sequence:

```
BAL, LNKI, ADDCK
B       ERROR
    .
    .
    .
```

Routines calling ADDCK:

- LPSWH
- SSMH

ADDCK shall not call any routine

ADDCK Data organization

This routine shall reside within the processing phase of SS.

ADDCK Limitations

There are no known or anticipated limitations.

INPUT EQUIVALENCE QUEUE

0				4			
P	N	M	1	A	D	D	1
P	N	M	2	A	D	D	2
P	N	m	3	A	D	D	3
P	N	m	4	A	D	D	4
*	*	*	*				
END OF QUEUE FLAG							

INPUT EQUIVALENCE TABLE

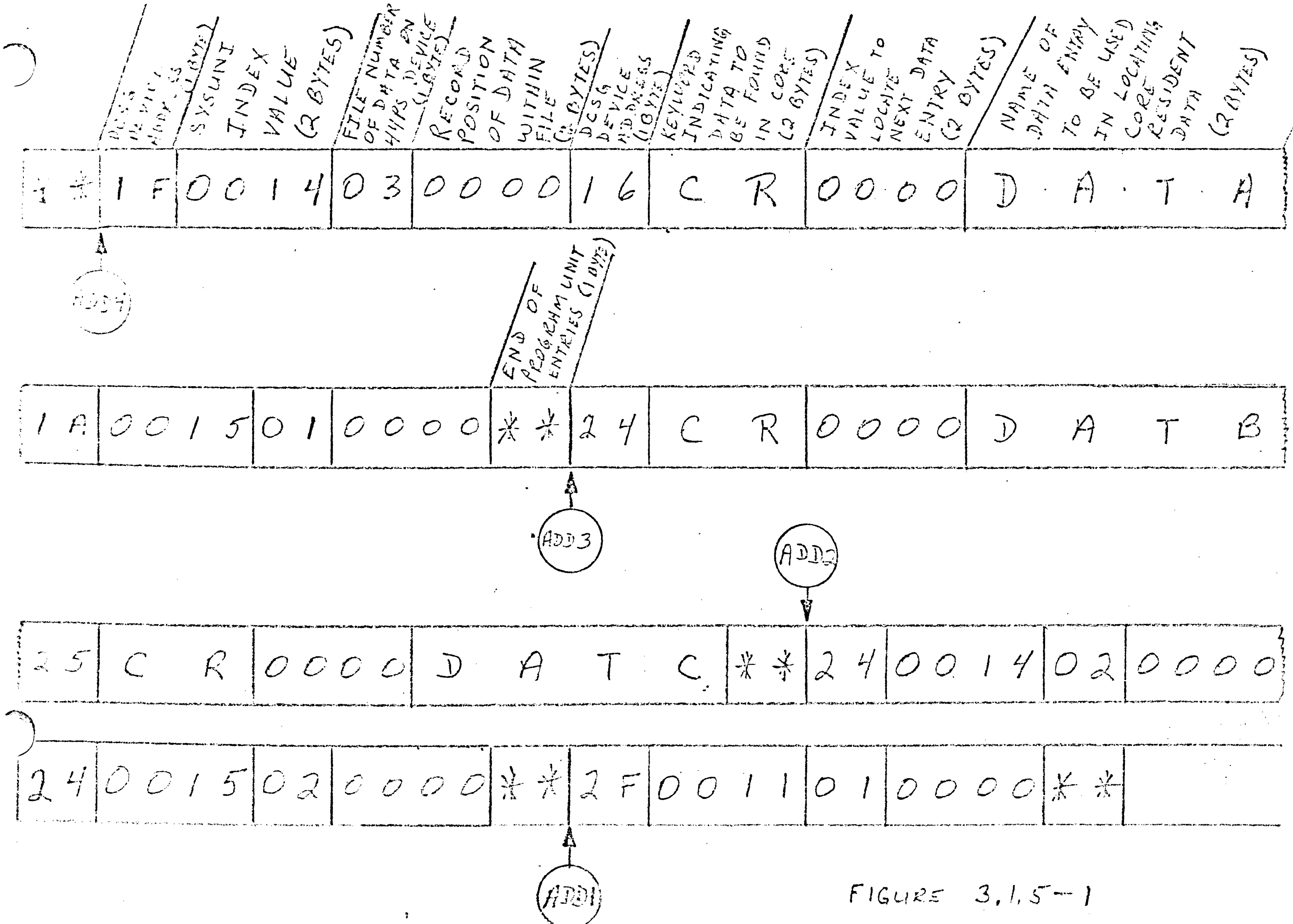
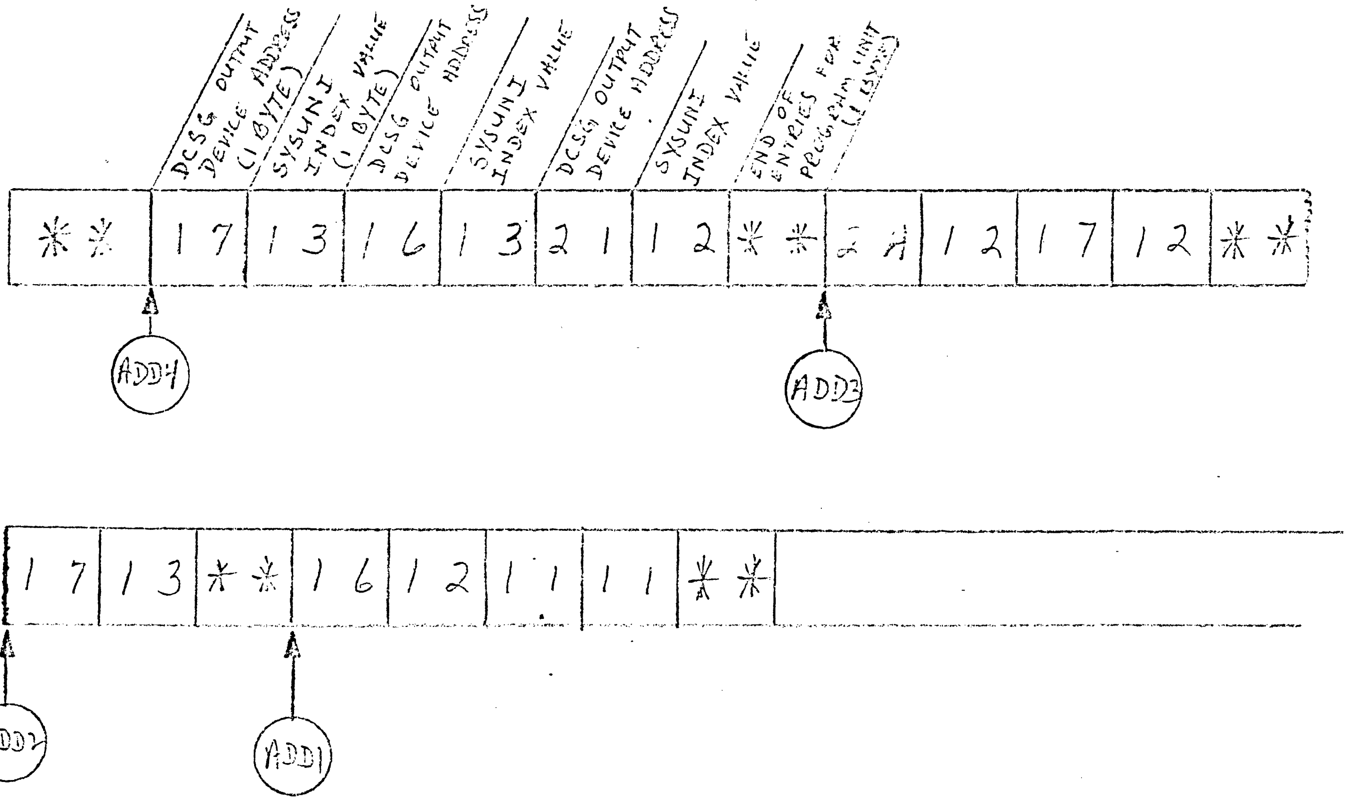

DCSG DEVICE ADDRESS (1 BYTE)
SYSUNIT INDEX VALUE (2 BYTES)
FILE NUMBER OF DATA ON 44PS DEVICE (1 BYTE)
RECORD POSITION OF DATA WITHIN FILE (2 BYTES)
DCSG DEVICE ADDRESS (1 BYTE)
KEYWORD INDICATING DATA TO BE FOUND IN CORE (2 BYTES)
INDEX VALUE TO LOCATE NEXT DATA ENTRY (2 BYTES)
NAME OF DATA ENTRY TO BE USED IN LOCATING CORE RESIDENT DATA (2 BYTES)
* * | 1 F | 0 0 1 4 | 0 3 | 0 0 0 0 | 1 6 | C R | 0 0 0 0 | D A T A
ADD4
END OF PROGRAM UNIT ENTRIES (1 BYTE)
1 A | 0 0 1 5 | 0 1 | 0 0 0 0 | * * | 2 4 | C R | 0 0 0 0 | D A T B
ADD3
ADD2
2 5 | C R | 0 0 0 0 | D A T C | * * | 2 4 | 0 0 1 4 | 0 2 | 0 0 0 0
2 4 | 0 0 1 5 | 0 2 | 0 0 0 0 | * * | 2 F | 0 0 1 1 | 0 1 | 0 0 0 0 | * *
ADD1

FIGURE 3.1.5-1

OUTPUT EQUIVALENCE QUEUE

P	N	M	2	A	D	D	1
P	N	M	6	A	D	D	2
P	N	M	5	A	D	D	3
P	N	M	1	A	D	D	4
* END	* OF	* QUEUE	* FLAG				

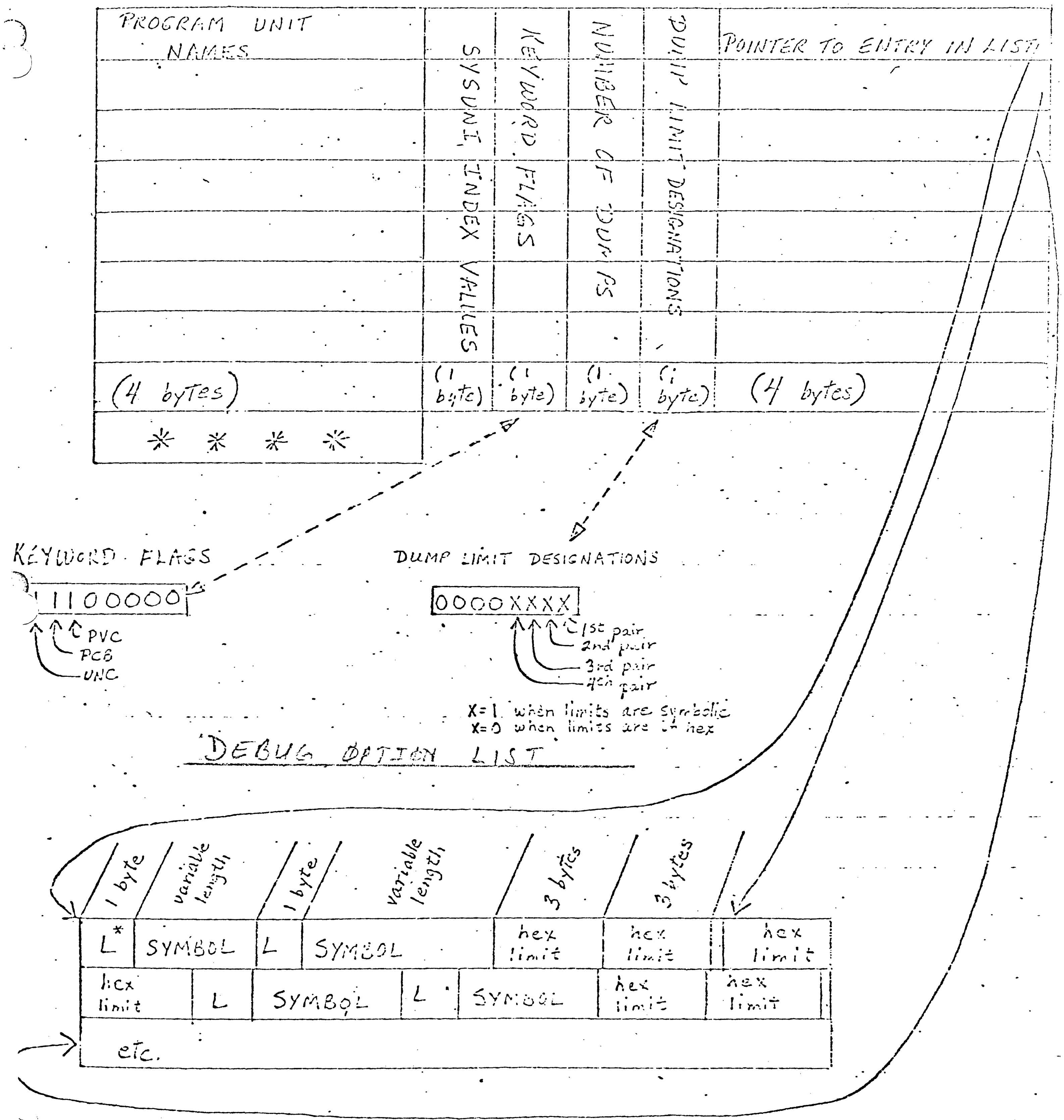
OUTPUT EQUIVALENCE TABLE



DCSG OUTPUT DEVICE ADDRESS (1 BYTE)
SYSUNI INDEX VALUE (1 BYTE)
DCSG OUTPUT DEVICE ADDRESS
SYSUNI INDEX VALUE
DCSG OUTPUT DEVICE ADDRESS
SYSUNI INDEX VALUE
END OF ENTRIES FOR PROGRAM UNIT (1 BYTE)
* * | 1 7 | 1 3 | 1 6 | 1 3 | 2 1 | 1 2 | * * | 2 A | 1 2 | 1 7 | 1 2 | * *
ADD4
ADD3
1 7 | 1 3 | * * | 1 6 | 1 2 | 1 1 | 1 1 | * *
ADD2
ADD1

FIGURE 3.1.5-2

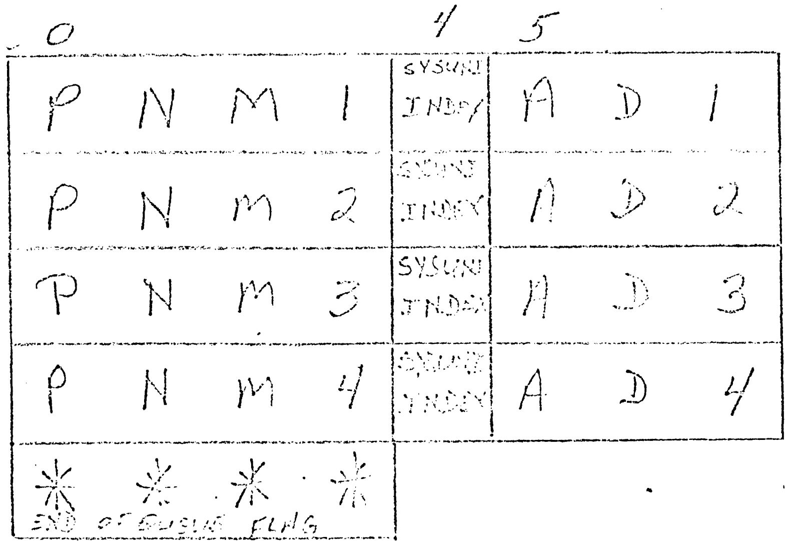
DEBUG QUEUE



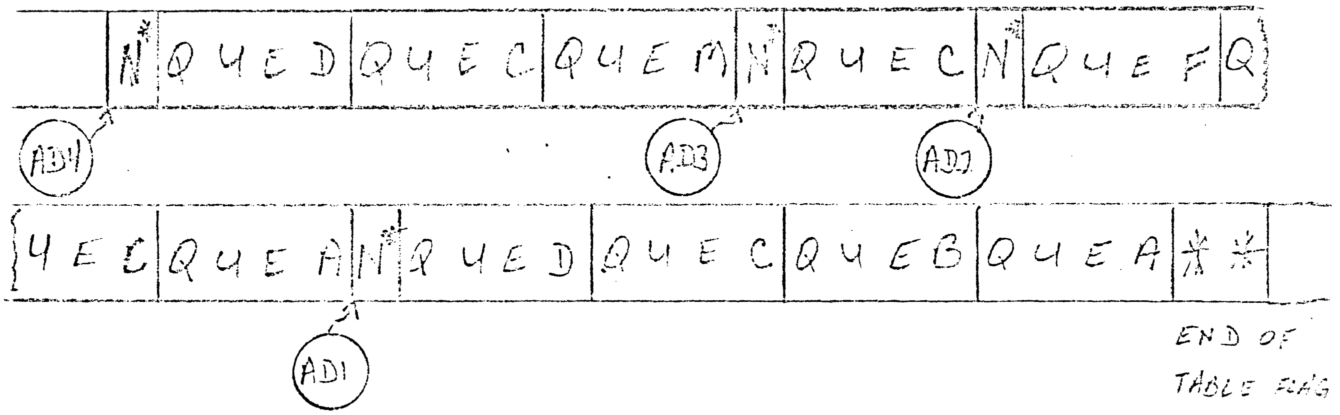
*L is length of the symbol

FIGURE 3.1.5-3

QUEUE DUMP QUEUE



QUEUE DUMP TABLE



N* = 1 byte value containing number of queues in the table for the particular program unit

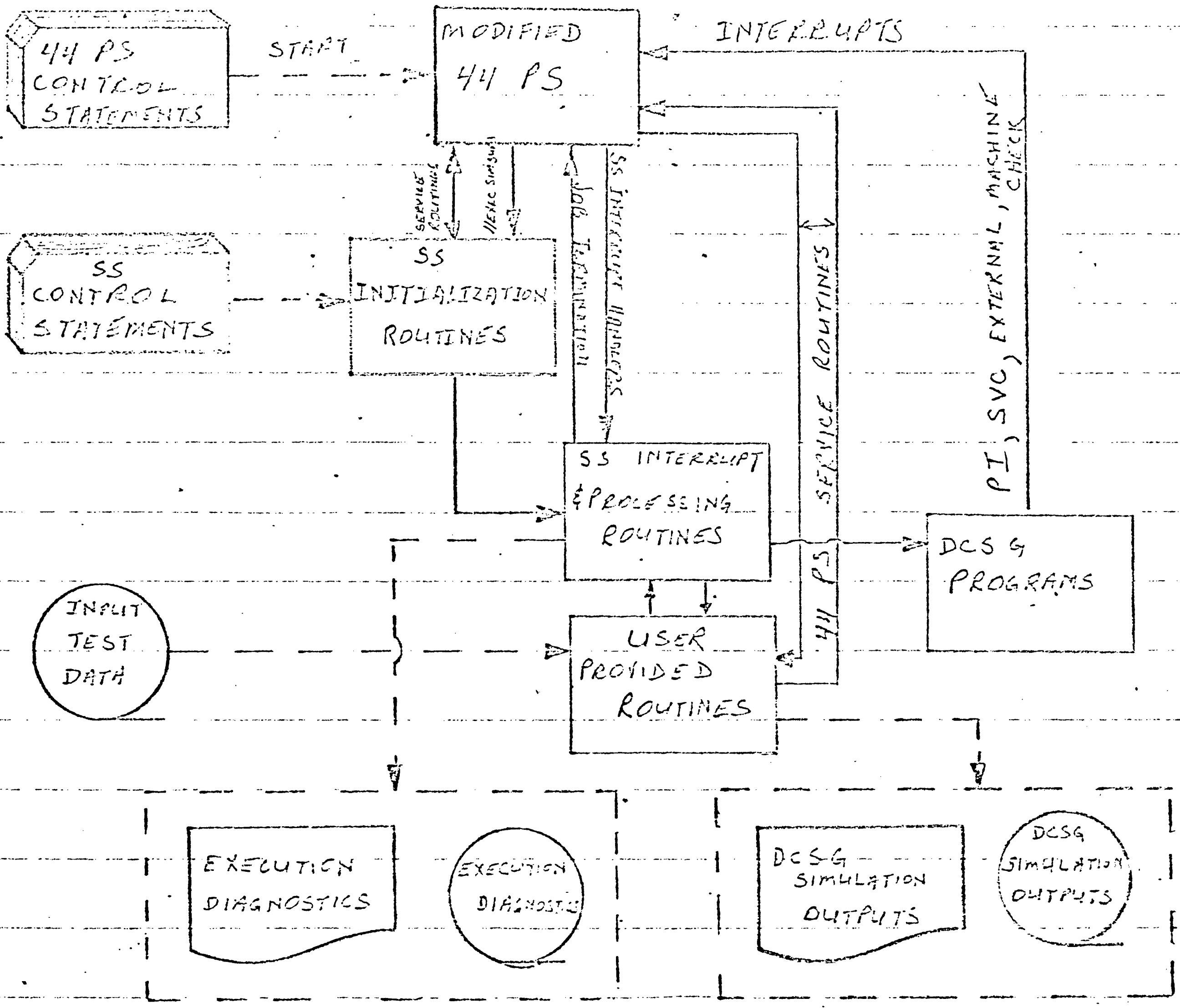
Figure 3.1.5-1

I/Ø EVENT TABLE

Byte							
0	FILE NO. OF I/Ø EVENTS	SYSUNIT INDEX	RECORD POSITION OF NEXT I/Ø EVENT	ADDRESS OF NEXT EVENT IN TABLE (INITIALLY SET TO FIRST EVENT)			} HEADER WORDS
8	TIME OF EVENT			INPUT DATA FLAG	SIMULATED DEVICE ADDRESS	SIMULATED SW STATUS	
16	TIME OF EVENT			INPUT DATA FLAG	SIMULATED DEVICE ADDRESS	SIMULATED CSW STATUS	
	TIME OF EVENT			INPUT DATA FLAG	SIMULATED DEVICE ADDRESS	SIMULATED CSW STATUS	
	* * * * END OF TABLE FLAG						

Figure 3.1.5-5

3.1.2 SS JOB FLOW


44 PS CONTROL STATEMENTS
START
MODIFIED 44 PS
INTERRUPTS
SERVICE ROUTINES
INITIAL SIMSYS
JOB TERMINATION
SS INTERRUPT HANDLERS
SS CONTROL STATEMENTS
SS INITIALIZATION ROUTINES
SERVICE ROUTINES
PI, SVC, EXTERNAL, MACHINE CHECK
SS INTERRUPT & PROCESSING ROUTINES
DCSG PROGRAMS
44 PS
INPUT TEST DATA
USER PROVIDED ROUTINES
EXECUTION DIAGNOSTICS
EXECUTION DIAGNOSTICS
DCSG SIMULATION OUTPUTS
DCSG SIMULATION OUTPUTS