SC23-0336-0
File No. S370-35

Program Offering

IBM 3090 Vector Facility Simulator Program Description/ Operations Manual

Program Number 5798-DWF

Release Number 1.0

IBM

PROGRAM SERVICES

Central Service will be provided until otherwise notified. Users will be given a minimum of six months notice prior to the discontinuation of Central Service.

During the Central Service period, IBM through the program sponsor(s) will, without additional charge, respond to an error in the current release of the program by issuing known error correction information to the customer reporting the problem and/or issuing corrected code or notice of availability of corrected code. However, IBM does not guarantee service results or represent or warrant that all errors will be corrected.

Any on-site program service or assistance will be provided at a charge.

WARRANTY

EACH LICENSED PROGRAM IS DISTRIBUTED ON AN 'AS IS' BASIS WITHOUT WARRANTY OF ANY KIND EITHER EXPRESSED OR IMPLIED.

Central Service Location: IBM Corporation
Department 41RA, Mail Station 920
Neighborhood Road
Kingston, New York 12401

Attention: IBM 3090 Vector Facility Simulator

Telephone: All customers within the 48 contiguous United States and Puerto Rico should call the toll free number 800-648-9248. Customers located in Alaska and Hawaii should call 914-385-1495 collect.

First Edition (February 1986)

This edition applies to Release 1 of the IBM 3090 Vector Facility Simulator, Program Number 5798-DWF. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 and 4300 Processors Bibliography*, GC20-0001, to find out which editions are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

Publications are not stocked at the address given below. Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department 52Q, Neighborhood Road, Kingston, New York, U.S.A. 12401. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. You may, of course, continue to use the information you supply.

Preface

In this manual, the IBM 3090 Vector Facility Simulator is referred to simply as "the simulator".

This book describes the procedures for:

- Setting up the simulator environment and using the simulator to execute programs that contain vector instructions.
- Installing the simulator on System/370 and System/370 XA processors with or without the 3090 Vector Facility.

The book is directed to application programmers developing, testing, and debugging vector applications.

Organization of This Book

Figure 1 shows the chapters in this book and who should read each chapter.

Title	Page	Who Should Read
Chapter 1. Simulator Overview	1	New users
Chapter 2. Using the Simulator	5	New users who want to learn how to test vector applications under the simulator
Chapter 3. Technical Reference Information	25	Experienced users needing detailed information about the simulator
Chapter 4. Simulator Messages	33	Anyone needing information about simulator messages
Chapter 5. Installing the Simulator	39	Anyone installing the simulator on VM/CMS

Figure 1. Book Map

Operating Systems Supporting the Simulator

The IBM 3090 Vector Facility Simulator requires CMS Release 3 or later, and any release of VM/SP (with or without HPO) that supports the release of CMS that you are using. Although the simulator can run under Extended Architecture (XA) systems, it cannot process application programs that access storage above 16 megabytes.

Hardware Requirements

The simulator runs on any processor supported by the operating systems listed above.

Storage Requirements

The simulator occupies approximately 76K bytes of CMS nucleus storage.

Required Publication

- *IBM System/370 Vector Operations*, SA22-7125

Related Publications

- Assembler H Version 2
  - *Assembler H Version 2: Application Programming: Guide*, SC26-4036
  - *Assembler H Version 2: Application Programming: Language Reference*, GC26-4037
- VS FORTRAN Version 2
  - *VS FORTRAN Version 2 Programming Guide*, SC26-4222
  - *VS FORTRAN Version 2 Language and Library Reference*, SC26-4221
- For Vector FORTRAN Programmers using VM:
  - *VM/SP Application Development Guide*, SC24-5247
- Virtual Machine/System Product
  - *VM/SP CMS Command and Macro Reference*, SC19-6209
  - *VM/SP CMS User's Guide*, SC19-6210
  - *VM/SP CP Command Reference for General Users*, SC19-6211

Contents

Figures

Chapter 1. Simulator Overview

The 3090 Vector Facility is an extension of the IBM 370 architecture. The Vector Facility implements 171 new machine instructions, plus:

- Sixteen vector registers, each containing 128 32-bit elements.
- Various other registers used by the Vector Facility, such as the vector status register (VSR).

The central processor uses these vector instructions and registers to speed up the execution of a program by operating on vectors (ordered sets of scalars).

With the IBM 3090 Vector Facility Simulator, you can execute programs that contain vector instructions and use vector registers, whether or not your processor has the 3090 Vector Facility. Your processor does not have to be a 3090. The execution of each vector instruction is simulated in software by the 3090 Vector Facility Simulator. The vector registers are also simulated. The simulated vector registers are represented in storage locations, where you can easily check their contents during program execution.

When you run your vector program under the simulator, you get:

- The same computational results you would get if you ran the program on a 3090 processor with the Vector Facility.
- Information about vector instructions in error that would cause an interrupt when executed on the vector hardware.
- Access to the simulated vector hardware, such as vector registers.
- Useful statistics about the vector portion of your program. These statistics can show you how effectively your program would make use of the 3090 Vector Facility. Your application program can be an object deck from any assembler or compiler.

The simulator is an addition to the CMS environment that is almost completely transparent to the user. You can carry on a normal CMS session while the simulator is activated. The simulator does not interfere with the use of your favorite editors, compilers, or other programs.

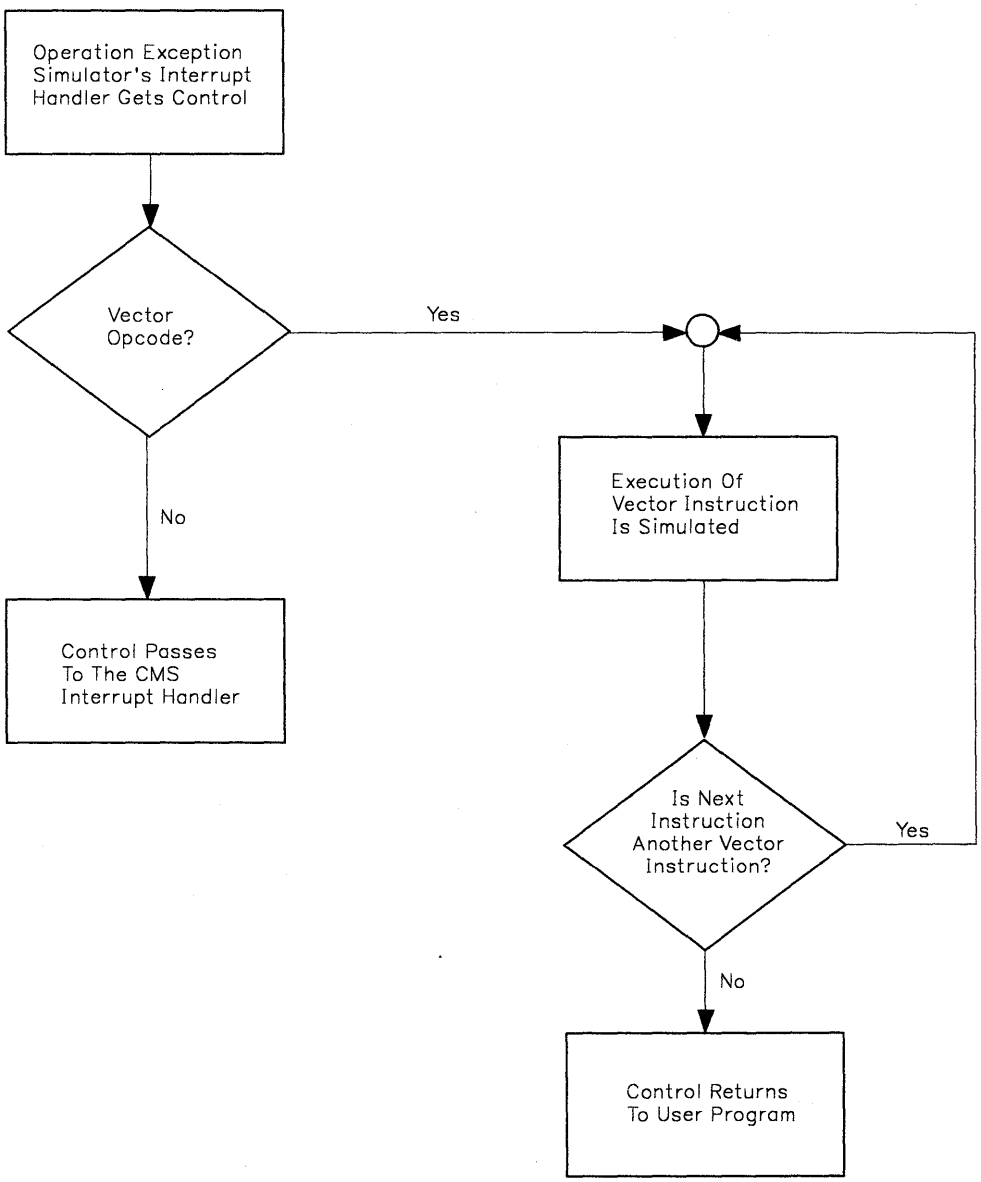


Figure 2. How Vector Instructions are Simulated

How Vector Machine Instructions Are Simulated

The 3090 Vector Facility Simulator uses the interrupt process to trap vector instructions and simulate them. This is an overview of how the execution of a vector instruction is simulated in software:

Assume that your vector program is executing under the simulator environment on a processor that does not have the Vector Facility. When the processor encounters a vector instruction, either an **operation exception** or a **vector operation exception** will occur, because the processor will not recognize the vector opcode. Under the simulator environment, the simulator's interrupt handler receives control instead of the normal CMS interrupt handler:

If the interrupt was caused by the attempted execution of a vector instruction, the results of that instruction are simulated.

- If the vector instruction is valid and would execute correctly on the vector hardware, execution of your program continues.

- If the vector instruction is in error and would cause an interrupt on the vector hardware, an error message gives you the details about the condition, and then control passes to the CMS interrupt handler.

The simulator continues to simulate consecutive vector instructions or Load Address instructions. When a scalar instruction other than Load Address is encountered, control is returned to the user program and the scalar instruction is executed directly by the hardware.

For a graphic depiction of how vector instructions are simulated, see Figure 2 on page 2.

Information Provided by the Simulator

The simulator provides two basic types of information useful in testing vector applications:

1. Storage Map of Simulated Vector Components

   When debugging your programs, you may need to know the contents of the vector registers and other vector components. The simulator allocates storage locations for the contents of the simulated vector components. In order to see the contents of the simulated registers, you need to know their addresses in storage. The VECTOR MAP file lists the addresses of these simulated components:

   - Vector registers 0-15
   - Vector mask mode
   - Vector count
   - Vector interruption index
   - Vector in use bits
   - Vector change bits

- Vector mask register
- Vector activity count

2. Vector Statistics

   The simulator can collect statistics useful in determining how efficiently your program uses the Vector Facility. You can get statistics for entire programs or sets of programs, or you can get statistics about specific portions of a program. The statistics are collected in a file called VECTOR STAT. The statistics collected are:

   - Total number of vector instructions, distributed by format. The instances of the following formats of vector instructions are counted by the simulator:

     ```
     QST     RSE     VS
     QV      S       VST
     RRE     VR      VV
     ```

   - Distribution of vector lengths for VLVCU vector loops.
   - Distribution of vector strides.

You have to instruct the simulator to create the VECTOR MAP and VECTOR STAT files before you can use them. This is explained in Chapter 2.

Chapter 2. Using the Simulator

This chapter provides the basic information you need to use the simulator, including the steps you go through when testing your vector programs. If you require detailed technical information about the simulator's design and operation, refer to "Chapter 3. Technical Reference Information" on page 25.

Overview of Using the Simulator

You activate the simulator environment by executing the VSIM command. Once the simulator environment is activated, you can continue with a normal CMS session.

While the simulator environment is activated, you can write, compile, load and test your vector programs. You can use the information the simulator provides to help debug your programs.

The VSIM Command

You must execute the VSIM command at least once to set up the simulator environment. You can issue subsequent VSIM commands to change the settings for options, or to perform services such as PRTSTAT.

Note: If the IMPEX setting for your virtual machine is off, or if you invoke VSIM from an assembler program, you must include EXEC in front of the command name (EXEC VSIM).

Activating the Simulator with the First VSIM Command

The first VSIM command activates the simulator by replacing the normal CMS interrupt handler with the simulator's interrupt handler. You can provide options on the first command to override defaults for simulator settings. Or you can issue the VSIM command without any options to activate it with the default settings.

Note: Executing the VSIM command with the HELP or "?" operand **will not** *activate the simulator.*

VSIM Command Syntax

VSIM [a] | b [,b...]

You can use one "a" option, or you can use one or more "b" options.

A maximum of 15 operands may be coded on any VSIM command. Operands are processed from left to right. If duplicate operands are coded, each will be executed.

"a" can be one of:

- DROP
- HELP
- ?

"b" can be one or more of the following options:

- No Operand
- CLEAR
- CLRSTAT
- LOOK or NOLOOK
- MAP
- MSG or NOMSG
- PRTSTAT
- QUERY
- STAT or NOSTAT

VSIM Command Options

"a" Options

Each "a" option must be entered as the only operand on the VSIM command.

DROP
VSIM DROP deactivates the simulator by restoring the normal CMS environment, including the normal CMS interrupt handler. The simulator code is removed from your virtual machine's storage.

HELP
Causes a help message to be displayed describing the simulator. You can also get help through the CMS HELP facility, by typing HELP VSIM.

?
Executing VSIM ? is exactly the same as executing VSIM HELP.

"b" Options

"b" options can be entered with other "b" options on the VSIM command.

(no operand)

If you invoke the VSIM command with no specified options, and the simulator is not yet activated, the simulator will be activated with the default settings for simulator options:

- LOOK
- MSG
- NOSTAT

Note: If the simulator has already been activated, executing VSIM again will have no effect. Re-executing the VSIM command will not restore the default settings for simulator options.

CLEAR

Clears simulated vector registers, vector mask register, vector status register, and vector activity count.

After you run a program under the simulator, you should use the CLEAR option to reset the registers to zero. This prevents the register contents of the previous program from affecting the execution of the next program.

CLRSTAT

Clears all statistics data being maintained by the simulator. No statistics are collected by the simulator unless you instruct it to do so by executing VSIM STAT. When you want to see the statistics that have been collected, instruct the simulator to create the VECTOR STAT file by using the PRTSTAT option.

LOOK|NOLOOK

- LOOK specifies that groups of consecutive vector instructions or Load Address instructions will be simulated without passing through the interrupt process. When the processor encounters a vector instruction, an interrupt occurs and the simulator receives control. After the vector instruction is simulated, the simulator will look ahead to the following instruction to determine if it is also a vector instruction, or a Load Address instruction. If so, the simulator retains control of the processor and simulates the instruction immediately. This avoids the interrupt and related processing required to give control to the simulator. Thus the simulator works faster when LOOK is in effect. LOOK is the default setting.

- NOLOOK specifies that, after a vector instruction is simulated, the simulator will not look ahead to the next instruction to see if it should be simulated. Instead, control returns to the application program. All vector and scalar instructions are presented directly to the processor for execution. You should use the NOLOOK option when you are doing certain types of debugging, such as instruction tracing with the CP PER command.

For more information about using LOOK|NOLOOK while debugging, see "Using VM Debugging Facilities with the Simulator" on page 12.

Note: The computational outcome of your program will be the same, regardless of the setting of LOOK|NOLOOK.

MAP

MAP causes a disk file to be created showing the addresses of simulated components of the vector architecture. The addresses are written to a file called VECTOR MAP. You should use the MAP option when you first install the simulator, so you will be able to find out where the simulated vector components are located.

The first record of the VECTOR MAP file contains a title and the date and time the map was produced. The following records show the storage locations of the:

- Vector registers 0-15
- Vector mask mode
- Vector count
- Vector interruption index
- Vector in use bits
- Vector change bits
- Vector mask register
- Vector activity count

For more information about the vector storage map, see "Using the VECTOR MAP File to Access Vector Components" on page 10.

When the VECTOR MAP file is produced, any existing VECTOR MAP file is overwritten.

MSG|NOMSG

- MSG - specifies that the simulator message relating to exceptions (message DWF020) will be displayed at the terminal. This is the default setting.

  Other messages will always be displayed regardless of the setting of the MSG option. They are messages relating to:

  - Errors when submitting the VSIM command, such as the use of incorrect operands.
  - I/O errors when writing storage map and statistics files.
  - Overflow in a field of the statistics table. If an overflow occurs, data for that field will be invalid.

- NOMSG - specifies that messages relating to exceptions will not be displayed. NOMSG provides a screen environment identical to the vector hardware. When a program is running under the simulator with NOMSG in effect, the screen messages will be identical to the messages displayed when the program runs on a processor with the real Vector Facility.

PRTSTAT

PRTSTAT writes the statistics collected so far to the VECTOR STAT file. To enable statistics collection by the simulator, use the STAT option of the VSIM command.

PRTSTAT does not clear the statistics kept by the simulator. Use the CLRSTAT option to reset the statistics (begin with zeros in each field). Figure 6 on page 17 shows the format of the VECTOR STAT file.

Any existing VECTOR STAT file will be overwritten when you use PRTSTAT.

QUERY

VSIM QUERY prints the current setting of these simulator options at your terminal:

- LOOK|NOLOOK
- MSG|NOMSG
- STAT|NOSTAT

STAT|NOSTAT

- STAT - specifies that the following statistics be collected for all programs running under the simulator, until NOSTAT is entered:

  1. Total number of vector instructions by format (QST, QV, RRE, RSE, S, VR, VS, VST, VV).

  2. Distribution of vector lengths for VLVCU vector loops.

  3. Distribution of vector strides.

  For information about using simulator statistics, see "Using Statistics Provided by the Simulator" on page 14.

- NOSTAT - specifies that no statistics be collected. This is the default setting when you invoke VSIM.

The STAT, CLRSTAT and PRTSTAT options are designed to generate statistics about entire programs or sets of programs. To generate statistics about specific portions of a program, use the selective statistics routines described in "Selective Statistics Gathering" on page 18.

VSIM Command Usage Examples

You can combine operands on the VSIM command to perform more than one action. Operands are processed from left to right. The following invocation will activate the simulator with statistics collection on, and create the VECTOR MAP file at the same time.

```
VSIM STAT MAP
```

For the next example, assume that you have already tested a program under the simulator. You can then have the statistics written to the VECTOR STAT file, clear the statistics, and clear the vector registers and components in preparation for the next program. The command to do all this would be:

```
VSIM PRTSTAT CLRSTAT CLEAR
```

Program Development under the Simulator

Use of the simulator is transparent to the CMS user. You can compile or assemble, load, run and debug your program after the simulator environment is activated. You can run your vector or scalar programs without alteration.

Using VSIM CLEAR to Reset Registers

If the previous vector program caused an interrupt which left a non zero value in the vector interruption index (VIX), the first vector instruction of the next program would be processed incorrectly; processing would not begin with the first element. You can avoid this situation by executing VSIM CLEAR between vector programs.

Using the VECTOR MAP File to Access Vector Components

During debugging, you may need to see the contents of the simulated vector registers and other Vector Facility components. The simulated components of the Vector Facility exist as data areas in storage. You need to know their locations in order to see their contents and debug your program effectively. You can easily find out the locations of vector components by instructing the simulator to produce the VECTOR MAP file.

To create the VECTOR MAP file, execute the VSIM command with the MAP option:

```
VSIM MAP
```

An A-disk file will be produced, with a fixed record length of 80. Each record has an identifier for the simulated component in column 2, followed by the corresponding hexadecimal address in column 8.

When the VECTOR MAP file is produced, any existing VECTOR MAP file is overwritten. Figure 3 on page 11 shows the format of the VECTOR MAP file.

```
*-VECTOR MAP-*  3090 VECTOR FACILITY SIMULATOR  V1M0  mm/dd/yy  hh:mm:ss

VR0      xxxxxxxx       address of vector register 0
VR1      xxxxxxxx       address of vector register 1
.
.
.
VR15     xxxxxxxx       address of vector register 15
VMM      xxxxxxxx       address of vector mask mode
VCT      xxxxxxxx       address of vector count
VIX      xxxxxxxx       address of vector interruption index
VIU      xxxxxxxx       address of vector in use bits
VCH      xxxxxxxx       address of vector change bits
VMR      xxxxxxxx       address of vector mask register
VAC      xxxxxxxx       address of vector activity count
```

Figure 3. VECTOR MAP Format

Figure 4 is an example of a VECTOR MAP file produced by executing VSIM MAP. The figure shows that simulated vector register 4 is contained in the four bytes starting at location hex 221DC8. The simulated vector interruption index (VIX) is contained in the four bytes starting at location hex 21E3C0.

```
*-VECTOR MAP-*    3090 VECTOR FACILITY SIMULATOR  V1M0   09/05/86  13:59:13

VR0    0021FDC8
VR1    002205C8
VR2    00220DC8
VR3    002215C8
VR4    00221DC8
VR5    002225C8
VR6    00222DC8
VR7    002235C8
VR8    00223DC8
VR9    002245C8
VR10   00224DC8
VR11   002255C8
VR12   00225DC8
VR13   002265C8
VR14   00226DC8
VR15   002275C8
VMM    0021E3B0
VCT    0021E3BC
VIX    0021E3C0
VIU    0021E3C4
VCH    0021E3C8
VMR    0021E3CC
VAC    0021E3A8
```

Figure 4. Example of a VECTOR MAP File

Implementation of Vector Components by the Simulator

This section describes how the simulator represents the simulated vector components in storage. Refer to *IBM System/370 Vector Operations* for a complete description of the Vector Facility components.

- Each vector register is 128 consecutive fullwords.
- The vector status register is not represented as specified in *IBM System/370 Vector Operations*; each of its constituent parts (VMM, VIX, VCT, VIU and VCH) is represented as a fullword.
  - The VMM is represented as a fullword. The rightmost bit of the VMM fullword contains the mask mode.
  - The VIX and VCT are represented as fullword integers.
  - The rightmost byte of the VIU and the VCH fullwords contain the vector-in-use bits and vector-change bits.
- The vector mask register is represented as 128 consecutive bits starting at the address shown in VECTOR MAP.
- The vector activity count is represented as an 8-byte value.

Using VM Debugging Facilities with the Simulator

Both CP and CMS provide debugging facilities you can use while the simulator is activated. Before you use these facilities, you will need a listing of the VECTOR MAP file produced by the **VSIM MAP** command.

Since the components of the 3090 Vector Facility are simulated in storage, the simulated components can be inspected or set with VM commands that inspect and/or set storage.

Note: CP Vector commands that inspect or set real Vector Facility components will not provide meaningful results. See "Restrictions" on page 30.

Setting Breakpoints

You can set breakpoints with the CP ADSTOP command, the CP PER facility, or with CMS DEBUG.

Note: Do not set breakpoints within the simulator code. When you set breakpoints, ensure that they are within the address range occupied by your application program, as shown in the LOAD MAP produced when you load your program.

PER Breakpoints under the Simulator: For a PER breakpoint, the break normally occurs after the instruction is completed, and you can then examine the results of the instruction. However, when you specify a simulated vector instruction as the target of a PER breakpoint, the break will occur **before** the vector instruction is simulated. If you examine the results of a vector instruction at this point, the only result you will see is that the instruction caused an exception. The simulator must execute several actual machine instructions in order to simulate one vector instruction. To properly use the CP PER facility under the simulator, you should set the PER breakpoint at the **next instruction after the one you wish to examine**. This allows the simulator to execute the set of real machine instructions required in order to simulate the vector instruction. When the breakpoint is encountered at the next instruction, you can then examine the results of the simulated vector instruction.

Instruction Tracing with the Simulator

You can execute your program in a step by step fashion using the CP TRACE or CP PER facilities. Two very important considerations must be noted when tracing under the simulator:

1. Restrict tracing to the address range occupied by your program.

   Vector instructions in your program cause interrupts that transfer control to the simulator. If you activate a general trace without restricting it to your program's address range, you will be forced to step through the internal code of the simulator as it simulates vector instructions. You can easily step through your program by activating a trace for the address range shown in the LOAD MAP after you load the program.

2. Use the NOLOOK option of the VSIM command.

   Before attempting to step through your program, ensure that the NOLOOK option of the VSIM command is in effect. You can execute VSIM QUERY to see the setting for LOOK|NOLOOK.

   If LOOK is in effect, the simulator will simulate groups of consecutive vector instructions or Load Address instructions without returning control to your application program. This means that a trace on these instructions will not work because they are not presented to the processor for execution. To ensure that all vector instructions are presented directly to the processor, thus allowing the instructions to be traced, use the NOLOOK option.

Obtaining Storage Dumps of Your Virtual Machine

You can obtain a dump of your virtual machine storage using CP DUMP, CP VMDUMP, or (CMS DEBUG) DUMP. The simulated components of the Vector Facility will be included in the dumps, at the location specified in the VECTOR MAP file.

Using Simulator Messages to Help Debug Your Program

If the MSG option is in effect, a message is issued when the simulator encounters a vector instruction that would cause an exception if executed on the real vector machine. The message indicates:

- The type of exception that was encountered
- The address of the instruction that was interrupted
- The vector element on which the interrupt occurred.

The syntax of the message relating to vector operation exceptions detected by the simulator is:

DWF020 *exception-type* **at address** *address* **while processing vector element** *element-no*

You can use the information in the message to help correct your program and find the cause of the interrupt.

Figure 5 shows an example of the message issued when a vector instruction is used to attempt division by zero:

```
DWF020: Floating point divide exception at address 0202AE
        while processing vector element   03
```

Figure 5. Example of Vector Operation Exception Message

Using Statistics Provided by the Simulator

The simulator collects statistics to help you determine how effectively your program would make use of the Vector Facility. You must instruct the simulator to begin collecting statistics, and then have the statistics written to the VECTOR STAT file. You can use the VSIM command to get statistics for entire programs or sets of programs, or you can get statistics about specific portions of a program by using the routines described in "Selective Statistics Gathering" on page 17. Both the VSIM command options and the selective statistics routines use the same internal switches to control whether or not statistics are collected. This section describes how to get statistics using the VSIM command.

- To instruct the simulator to collect statistics, invoke the VSIM command with the STAT option:

  ```
  VSIM STAT
  ```

- To instruct the simulator to print the statistics collected for all vector programs executed since the statistics were last cleared, execute the VSIM command with the PRTSTAT option:

  ```
  VSIM PRTSTAT
  ```

  This creates a disk file, VECTOR STAT.

- To clear the statistics and start with zeros, execute the VSIM command with the CLRSTAT option:

  ```
  VSIM CLRSTAT
  ```

  You may want to use CLRSTAT before you run each vector program to clear the statistics from the previous vector program.

Types of Statistics Collected by the Simulator

The statistics contained in the VECTOR STAT file can be grouped into three main types:

1. Number of vector instructions executed, distributed by format. These types of vector instructions are counted:

QST	RSE	VS
QV	S	VST
RRE	VR	VV

   By checking the distribution of the different formats of vector instructions, you can determine what types of instructions are being generated by your compiler.

2. Distribution of vector lengths for VLVCU vector loops.

   A vector instruction typically performs a similar operation (such as addition or multiplication) on a series of elements. Several vector instructions are grouped together into a vector loop. On each iteration of the vector loop, the vector count contains a value indicating the number of elements to be processed by each vector instruction in the loop. On every iteration except the last one, the vector count contains the section size. On the last iteration, the vector count indicates the number of elements remaining to be processed.

   Vector length is defined as the accumulated length through all iterations of the vector loop. It is incremented once for each iteration. For example, if a FORTRAN Do-loop beginning with the following statement were vectorized, the vector length would be N.

   ```
   DO 10 I=1,N
   ```

   The VECTOR STAT file provides a breakdown of vector lengths in your program. You may want to monitor how many loops have low vector lengths. Programs with loops having low vector lengths do not efficiently utilize the Vector Facility.

The simulator collects the vector length statistic by incrementing a counter each time a VLVCU instruction is executed. When the VLVCU instruction gives a condition code of 3, the size of the vector loop is known, allowing the simulator to determine the appropriate range.

There can be more than one vector instruction within a vector loop. Vector length is independent of the number of operations within the vector loop.

Note: The simulator only collects statistics for vector loops controlled by the VLVCU instruction. The vector length statistic will not be valid for programs that do not use the VLVCU instruction to control vector loops. The VS FORTRAN Version 2 compiler uses the VLVCU instruction to generate vector loops.

3. Distribution of vector strides.

   A vector is contained in storage in a linear fashion, with the elements of the vector appearing contiguously. But your program may not access the elements contiguously. For instance if your program accessed every other element, the **stride** would be 2. A stride of 1 indicates that the elements are accessed contiguously. A stride of 0 indicates that the same element will processed over and over. A negative stride means that the vector is processed in reverse order.

   For every instruction that references a vector in storage (QST and VST instructions), the simulator determines the stride and increments the appropriate counter (see Figure 6 on page 17).

   You may want to monitor the number of vector instructions that use high absolute stride values. The 3090 Vector Facility achieves the greatest efficiency when processing vectors using low strides. If possible, you should rework an algorithm which results in vectors with high strides (negative or positive).

Format of VECTOR STAT File

Figure 6 on page 17 shows the format of the VECTOR STAT file.

Figure 7 on page 17 is an example of a statistics file created by executing VSIM PRTSTAT.

```
*-RUN STATISTICS-* 3090 VECTOR FACILITY SIMULATOR  V1M0  mm/dd/yy hh:mm:ss

# QST instructions
# QV  instructions
# RRE instructions
# RSE instructions
# S   instructions
# VR  instructions
# VS  instructions
# VST instructions
# VV  instructions
Vector Lengths   (0,10)
                (11,20)
                (21,50)
               (51,128)
              (129,256)
              (257,512)
             (513,1024)
              >1024
Stride(4 bytes)     0,1
                      2
        (3,32),(-1,-32)
               >32,<-32
Stride(8 bytes)     0,1
        (2,16),(-1,-16)
               >16,<-16
```

Figure 6. Format of the VECTOR STAT File

```
 *-RUN STATISTICS-*  3090 VECTOR FACILITY SIMULATOR  V1M0  09/05/86  13:48:06

 # QST instructions.....................................0
 # QV  instructions.....................................80000
 # RRE instructions.....................................6
 # RSE instructions.....................................0
 # S   instructions.....................................2
 # VR  instructions.....................................800
 # VS  instructions.....................................0
 # VST instructions.....................................160400
 # VV  instructions.....................................800
 Vector Lengths   (0,10).....................................0
                 (11,20).....................................0
                 (21,50).....................................0
                (51,128).....................................0
               (129,256).....................................2
               (257,512).....................................0
              (513,1024).....................................0
               >1024.........................................0
 Strides (4 bytes)   0,1.....................................0
                       2.....................................0
          (3,32),(-1,-32).....................................0
                 >32,<-32.....................................160400
 Strides (8 bytes)   0,1.....................................0
          (2,16),(-1,-16).....................................0
                 >16,<-16.....................................0
```

Figure 7. Example of a VECTOR STAT File

Selective Statistics Gathering

The simulator provides routines that allow you to gather information about particular areas of your program, instead of the entire program. You can call the routines from a FORTRAN or an assembler program. With these routines, you can turn statistics gathering on and off, clear the statistics, and have the statistics written to a file. When using the selective statistics routines, you must call the routine to initialize the VECTOR STAT file before enabling statistics collection. You must also call the routine to terminate statistics collection before you can actually see the statistics.

Your program can call these routines while the simulator is activated. If any of the routines are called when the simulator is not active, a message is displayed and the call is ignored.

Some of the routines perform functions similar to VSIM command options. The relationship between VSIM command options and selective statistics routines is described in "Interaction of VSIM Options and Selective Statistics Routines" on page 22.

The routines are in DWFSTAT TXTLIB and will be linked to your application program if you have identified them to the system with the GLOBAL TXTLIB command.

Format of VECTOR STAT File Created by Statistics Routines

When you use the VSIM PRTSTAT command to write the statistics to the VECTOR STAT file, an existing VECTOR STAT file will be overwritten. However the selective statistics routines treat the VECTOR STAT file differently. When your program calls the selective statistics routine to write the statistics, the statistics are **appended** to the VECTOR STAT file, instead of overwriting it. This means you can capture the statistics many times without losing any of the data. Since an existing file is not overwritten, you must call the routine to initialize the file when you want to start anew, before gathering any statistics about a program. The routines accomplish their functions through normal VSIM commands.

During statistics collection (after a DWFIN call but before a DWFTM call), a temporary file, $VECTOR STAT, is used to hold the statistics. The statistics will not be in the permanent VECTOR STAT file until a call is made to DWFTM (the routine that terminates statistics collection).

Figure 8 on page 19 shows the format of the VECTOR STAT file as created by the selective statistics gathering routines.

```
*-RUN STATISTICS-* 3090 VECTOR FACILITY SIMULATOR  V1M0  mm/dd/yy hh:mm:ss

< user label from first DWFPS call >
# QST instructions
# QV  instructions
# RRE instructions
# RSE instructions
# S   instructions
# VR  instructions
# VS  instructions
# VST instructions
# VV  instructions
Vector Lengths   (0,10)
                (11,20)
                (21,50)
               (51,128)
              (129,256)
              (257,512)
             (513,1024)
              >1024
Stride(4 bytes)     0,1
                      2
        (3,32),(-1,-32)
               >32,<-32
Stride(8 bytes)     0,1
        (2,16),(-1,-16)
               >16,<-16

< user label from second DWFPS call >
# QST instructions
.
.
.
Stride(8 bytes)     0,1
        (2,16),(-1,-16)
               >16,<-16

< user label from third DWFPS call >
# QST instructions
.
.
.
```

Figure 8. Format of VECTOR STAT File Created by Selective Statistics Routines

Descriptions of the Selective Statistics Routines

DWFIN

Initializes statistics gathering. DWFIN should be called only once per program, before any calls to any of the other statistics routines. DWFIN erases the VECTOR STAT file and initializes the $VECTOR STAT file so that any statistics collected afterward will be for the current program only. A call to DWFIN is not comparable to any VSIM command option.

DWFST

Turns on statistics gathering. The effect of a call to this routine is as if VSIM STAT were issued at that point during the execution of the program. This is not the same as **initializing** statistics gathering. Initialization should be done only once per program, but you can turn statistics gathering on and off many times.

DWFNS

Turns off statistics gathering. The effect of a call to this routine is as if VSIM NOSTAT were issued at that point during the execution of the program.

Note: This is not the same as **terminating** *statistics gathering.*

DWFCS

Clears the statistics. The effect of a call to this routine is as if VSIM CLRSTAT were issued at that point during the execution of the program.

DWFPS [*label*]

Appends the current values of the statistics to the $VECTOR STAT file. A call to this routine is similar to VSIM PRTSTAT, except that the statistics are **appended** to the $VECTOR STAT file, instead of overwriting it. DWFPS accepts a character string as a parameter to allow you to label the set of statistics being appended. The label parameter is optional. If you supply the label, it is printed prior to the statistics to distinguish between sets of data if more than one call to DWFPS is made in a program. The label is limited to 80 characters; if a longer parameter is supplied, it is truncated.

If you supply an invalid parameter on the invocation, the statistics are still written to the file. In the spot where the label would be, the following line is inserted:

```
*****  INVALID PARAMETER TO DWFPS  *****
```

Invalid parameters include:

- An invalid address for the character string
- A negative length for the character string
- More than one argument.

DWFTM

Terminates statistics gathering, and creates the permanent VECTOR STAT file from the temporary $VECTOR STAT file. DWFTM should only be called once per program. No calls to any other statistics routines should be made after DWFTM has been invoked. A call to DWFTM is not comparable to any other VSIM command option.

FORTRAN Calling Syntax

Figure 9 shows how to collect statistics about particular loops in a FORTRAN program.

```
      PROGRAM EXMPLE

.
.
.
      CALL DWFIN

.
.
.
      CALL DWFST
      CALL DWFCS
      DO 10 I = 1,N

.
.
.
10    CONTINUE
      CALL DWFPS('LOOP 10 STATISTICS')
      CALL DWFNS

.
.
.
      CALL DWFST
      CALL DWFCS
      DO 20 I = 1,N

.
.
.
20    CONTINUE
      CALL DWFPS('LOOP 20 STATISTICS')
      CALL DWFNS

.
.
.
      CALL DWFTM

.
.
.
      END
```

Figure 9. FORTRAN Calling Syntax for Selective Statistics Gathering

In Figure 9, the calls to DWFNS and DWFST between loop 10 and loop 20 are not necessary since the call to DWFCS before the start of loop 20 will clear any statistics accumulated prior to loop 20. If there is a sizable portion of code between the two loops however, these two calls will allow the program to execute faster by avoiding any unnecessary statistics overhead.

Assembler Language Calling Syntax

Figure 10 shows the calling syntax for all routines without parameters:

```
.
.
.
        LA     13,SAVEAREA       ADDRESS OF 18 WORD SAVE AREA
        LA     1,0               ZERO SINCE NO PARAMETERS
        L      15,=V(DWFxx)      ADDRESS ROUTINE(xx = NS, ST, CS, IN, or TM)
        BALR   14,15             CALL ROUTINE
```

Figure 10. Calling Syntax for Routines Without Parameters

Figure 11 shows the calling syntax for DWFPS:

```
.
.
.
        LA     13,SAVEAREA       ADDRESS OF 18 WORD SAVE AREA
        LA     1,PLIST           ADDRESS OF PARAMETER LIST
        OI     PLIST,X'80'       MAKE HIGH BIT 1 BECAUSE LAST PARAMETER
        L      15,=V(DWFPS)      ADDRESS ROUTINE
        BALR   14,15             CALL ROUTINE

.
.
.
PLIST   DC     A(STRING)
        DC     A(LENGTH)
STRING  DC     C'LABEL FOR THE STATISTICS'
LENGTH  DC     F'24'
```

Figure 11. Calling Syntax for DWFPS

To call DWFPS without the character string parameter use the above syntax for routines without parameters.

Interaction of VSIM Options and Selective Statistics Routines

This section demonstrates the interaction between the statistics routines and the VSIM commands. In Figure 12 on page 23, the statistics routines are called from a program named VECTOR. The chart shows the following data for various points before, during, and after program execution:

- STAT - This column contains the current setting of the STAT option (ON for STAT, OFF for NOSTAT).
- Values - This column contains the current values of the statistics being collected within the simulator. An X represents unknown values which depend on the code being run.

- VECTOR STAT file - This column lists the current state of or action done to the VECTOR STAT file.
- $VECTOR STAT file - This column lists the current state of or action done to the $VECTOR STAT file.

	STAT	values	VECTOR STAT file	$VECTOR STAT file
R;	OFF	X	may exist	shouldn't exist
VSIM STAT	ON	X	"	"
R;				
LOAD VECTOR	ON	X	"	"
R;				
START VECTOR	ON	X	"	"
inside program:				
.				
.				
CALL DWFIN	ON	X	erased	initialized with a header line
.				
.				
CALL DWFST	ON	X	no change	no change
CALL DWFCS	ON	zeroes	"	"
.				
<program code>				
.				
CALL DWFNS	OFF	X	"	"
CALL DWFPS(LBL)	OFF	X	updated then erased	appended to with the statistics collected since DWFCS call
.				
.				
CALL DWFST	ON	X	no change	no change
CALL DWFCS	ON	zeroes	"	"
.				
<program code>				
.				
CALL DWFNS	OFF	X	"	"
CALL DWFPS(LBL)	OFF	X	updated then erased	appended to with the statistics collected since DWFCS call
CALL DWFCS	OFF	zeroes	no change	no change
CALL DWFTM	OFF	zeroes	$VECTOR STAT is renamed to VECTOR STAT	
.				
END				
R;	OFF	zeroes	no change	shouldn't exist

Figure 12. Interaction of VSIM Options and Selective Statistics Routines

Note that even though statistics gathering was turned on before the program was run, it is now off due to the DWFNS call. Note also that statistics need not have been turned on with the VSIM STAT command before program execution because the call to DWFST does this within the code.

Performance

Scalar programs will run as fast as normal under the simulator. However a program with vector instructions will take longer to run under the simulator than a corresponding scalar program. This is because each vector instruction must pass through the interrupt process and then be simulated in software.

If you have a large program to test under the simulator, you should consider testing a few subroutines at a time, rather than testing the entire program at once. If your program is very large and takes a long time to run ordinarily, it may be impractical to test it all at once under the simulator.

Chapter 3. Technical Reference Information

This chapter contains technical reference information about the 3090 Vector Facility Simulator. The information is not necessarily required by new users in order to use the simulator. It is provided for experienced users who want to know more about the simulator's operation.

Simulator Design

VSIM EXEC and the Nucleus Extension

The 3090 Vector Facility Simulator exists in two main parts:

1. VSIM EXEC

   The EXEC installs the simulator and serves as the main command interface.

2. Nucleus Extension

   The object code of the simulator is implemented as a CMS nucleus extension. The nucleus extension is called by the EXEC to perform simulator services, and receives control after a program interrupt.

   Nucleus extensions are programs that do not have to be loaded each time they are called. Once the simulator is activated (loaded as a nucleus extension), it will be retained until it is unloaded explicitly. This means that you don't have to load the simulator into virtual storage every time you test a vector program. You can load and execute your vector programs with no special procedures for simulation.

   The nucleus extension part of the simulator is divided into two functional parts:

   a. Options handler

      The options portion of the nucleus extension is called by the VSIM EXEC to process certain options.

   b. Main simulator code, including the simulator's interrupt handler

      This portion of the nucleus extension contains the simulator's interrupt handler. It also contains the code that simulates the execution of vector instructions.

Simulator Interfaces

Initialization-Termination Interface

Initialization: You execute the VSIM command to initialize the simulator environment. This is an overview of actions taking place as the simulator is activated:

1. The simulator is loaded as a nucleus extension.

2. The **program new PSW** at hex 68 is replaced by a PSW with an address pointing to the simulator's interrupt handler. This allows the simulator to receive control after program interrupts.

3. The section size (128) is written to the halfword beginning at CVT + hex 4C, to maintain a compatible interface for FORTRAN programs. The next halfword field in the CVT is the partial sum number. This halfword is initialized to 4.

4. The Vector Facility for your virtual machine is disabled by turning off bit 14 of control register 0. If you are running on a processor without the Vector Facility, this has no effect.

Termination: When you terminate the simulator environment (VSIM DROP), a NUCXDROP command is issued from the VSIM EXEC to remove the simulator as a nucleus extension. Also, the program new PSW, the CVT, and control register 0 are restored to their original values.

Command Interface

The VSIM EXEC serves as the main command interface. The HELP and DROP options are processed entirely by the EXEC; control is not passed to the nucleus extension. For other options, the VSIM EXEC invokes the nucleus extension which processes the options from left to right.

Interrupt Interface

The simulator receives control through program interrupts. If the interrupt is an operation exception or a vector operation exception caused by the attempted execution of a vector instruction, the instruction is simulated. On all other interrupts, control is passed to the interrupt handler that was replaced by the first invocation of the simulator, normally the standard CMS interrupt handler.

If an interrupt occurs while simulating and the interrupt code is between hex 04 and hex 12 (inclusive), the program interrupt old PSW is modified and control is passed to the interrupt handler replaced by the simulator. The program old PSW is modified such that when the interrupt is reflected to the original interrupt handler, it will appear as if an actual vector interrupt occurred.

After simulation is completed for an instruction, if LOOK is on, the next instruction is analyzed to see if it is also a vector instruction or a Load

Address instruction. If so, it is simulated. This avoids the program check and related processing which is required to give the simulator control when the instruction is executed. If the next instruction is not a vector instruction or a Load Address instruction, the simulator gives up control and the scalar instruction is executed.

Simulated Interrupts: Certain interrupt conditions are detected by the simulator without an actual machine interrupt occurring. These "simulated" interrupts are:

1. Specification exceptions on vector instructions (including misalignment)

2. Unnormalized operand

3. Floating-point divide.

The first two must be detected by the simulator since they would not be detected by the hardware. Floating-point divide is detected by the simulator for performance reasons.

CMS Operating System Interface

The simulator is installed as a nucleus extension with the NUCXLOAD command, using the SERVICE and SYSTEM attributes. The SYSTEM attribute specifies that the simulator will remain as a nucleus extension after abnormal termination of a user program. The SERVICE attribute specifies that the nucleus extension will be invoked by CMS when a user program abnormally terminates and when the NUCXDROP command is executed. The nucleus extension facility is described in the *VM/SP CMS Command and Macro Reference.*

As described in the *CMS Command and Macro Reference* manual, nucleus extensions with the SERVICE attribute are called by CMS for abnormal termination (abends, HX command, etc.) and when the NUCXDROP command is issued. The NUCXDROP command is issued by the VSIM EXEC when you execute VSIM DROP. Therefore, the VSIM command interface supports two additional options that are not part of the user interface. When called by CMS, the high order byte of register 1 is set to hex FF. The simulator requires that these options be invoked with the high-order byte of register 1 set to hex FF.

Figure 13 shows the parameter list for abnormal termination. In this case, all simulator state information is cleared.

```
DS    0F
DC    CL8'VSIM'
DC    CL8'PURGE'
DC    8X'FF'
```

Figure 13. Parameter List for Abnormal Termination

Figure 14 shows the parameter list for NUCXDROP. In this case, the PSW that was replaced by the original invocation of the simulator is restored at hex 68. Also the CVT and control register 0 bit 14 are restored to their original values.

```
DS    0F
DC    CL8'VSIM'
DC    CL8'RESET'
DC    8X'FF'
```

Figure 14. Parameter List for NUCXDROP

3090 Vector Facility Components Not Supported

The following components of the 3090 Vector Facility are not supported under the simulator:

- Vector Activity Count (VAC)

  A VAC is maintained in storage so that the instructions that manipulate this field (SAVE VAC and RESTORE VAC) can be supported by the simulator. However the VAC is not updated by the simulator and does not contain relevant information.

- Vector Facility bits of the machine check interrupt code.

- Asynchronous interrupts during vector instructions.

- Vector control bit - when the simulator is activated, this bit is always off, even on a processor with the Vector Facility.

- Vector-Change and Vector-in-Use Bits (VCH and VIU)

  On the real Vector Facility, the vector-change bits and the vector-in-use bits for a vector register pair are set whenever either register in the pair is modified. Under the simulator the VCH and VIU bits for a vector register pair are set whenever either register in the pair is the target of a vector instruction, whether or not the register is actually modified. This does not affect the results of your program. The only effect is that a vector register may be saved or restored even though it was not actually modified.

Simulator's Interaction with CMS

Control Blocks

For a description of CMS control blocks, see *VM/SP CMS System Programmers Guide*, SC19-6203, and *VM/SP CMS Control Blocks, Vols. 1 and 2*, LY24-5220 and LY24-5221.

- NUCON + hex 00

  The value in register 12 is stored into this location when the simulator is activated. This location is also used temporarily for internal data transfers by the simulator.

- NUCON + hex 28

  This location contains the program check old PSW. It is inspected when the simulator receives control on interrupts, to determine the interrupt code. The interrupt code is not cleared by the simulator. The simulator alters the value in this location when a vector interrupt is generated, and when the simulator returns control to the user program.

- NUCON + hex 68

  When the simulator is activated, the program check new PSW at hex 68 is replaced by a PSW with the address pointing to the simulator's program interrupt interface. The previous value at hex 68 is saved locally, for future restoration by the simulator. You should be aware of this if you use other tools that alter the PSW. See "Using the Simulator with Other Tools" on page 30.

- CVT

  In scalar CMS, the storage at location CVT + hex 4C is not used. In vector CMS, CVT + hex 4C contains the section size and partial sum number in consecutive halfwords. The simulator sets these halfwords independently of the underlying hardware and operating system.

Vector Control Bit of Control Register 0

The simulator turns off bit 14 of control register 0, which disables the Vector Facility for your virtual machine. This has no effect on a scalar system. On a processor equipped with the Vector Facility, no benefits of the vector hardware will be realized when this bit is off.

Storage Protection

You cannot get a fetch protection exception when running under the simulator. The simulator normally runs with the system storage key. During store operations instigated by an application program running under the simulator, the user's storage key is used. This results in a storage protection exception if the application program attempts a store to an invalid location.

Unnormalized Operand Interrupt

This interrupt is newly defined for the vector architecture. CMS systems that do not support vectors will not recognize this condition. However the simulator will issue a message informing you of the error. After the unnormalized operand message is issued from the simulator, the next message from CMS will not be relevant, since the CMS system does not support vectors. The message issued from the simulator should be enough to inform you of the error.

Restrictions

Using the Simulator with Other Tools

Because the simulator replaces the program new PSW with a PSW pointing to the simulator's interrupt handler, all tools that alter the program new PSW must be installed and removed from the system in "last-in-first-out" order. If tools are not installed and removed from the system in the proper order, control may pass to an invalid location following an interrupt. Since there is no external program to manage this discipline, the user is responsible for maintaining the integrity of the system:

1. A tool should only be removed from the system after all tools that have been installed after it have been removed.

2. When a tool is removed from the system, it should restore the PSW that it replaced to location hex 68.

3. When fielding an interrupt that it does not need to process, the tool should pass control to the interrupt handler that it replaced. It should pass control after it has restored all registers and the transfer should be done via an LPSW instruction.

4. A tool should not replace the program new PSW more than once.

Vector CP Functions Not Supported under the Simulator

The simulator turns off the vector control bit and thus disables the Vector Facility. Therefore certain vector CP functions relating to the real Vector Facility will not be available or will not provide relevant information:

- CP vector commands to display real vector registers and other vector components.

  These commands will not be provide relevant information while the Vector Facility is disabled by the simulator.

- Vector fields in the Monitor Performance Class Record and User Performance Class Record

  These facilities will not provide relevant information under the simulator.

Additionally, if you have executed a vector program prior to activating the simulator, then any vector save areas associated with your virtual machine will remain. To reclaim storage, you may release these areas manually by issuing one of these commands prior to activating the simulator:

- IPL CLEAR
- SYSTEM CLEAR
- LOGOFF

Chapter 4. Simulator Messages

This chapter contains the messages issued by the simulator.

3090 Vector Facility Simulator Component ID

The component ID for the simulator is **DWF**. Therefore the simulator error message numbers all start with DWF. Only error messages have numbers. Informational simulator messages do not have numbers.

Message Descriptions

Informational Messages

3090 Vector Facility Simulator installed as nucleus extension.

Explanation: The 3090 Vector Facility Simulator environment has been successfully initialized. The 3090 Vector Facility Simulator is now ready for use.

System Action: As in explanation.

User Action: None required.

3090 Vector Facility Simulator removed as nucleus extension.

Explanation: The 3090 Vector Facility Simulator environment has been terminated. No further vector simulation should be attempted until the simulator environment is reinitialized.

System Action: As in explanation.

User Action: None required.

Error Messages

DWF001 3090 Vector Facility Simulator installation failed.

Explanation: The 3090 Vector Facility Simulator environment could not be initialized because the NUCXLOAD command, which is used to make the simulator a CMS nucleus extension, failed.

System Action: No initialization is performed.

User Action: Determine why the NUCXLOAD failed (see *VM/SP CMS Command and Macro Reference*). Correct the problem. Specify VSIM again.

DWF002 3090 Vector Facility Simulator deinstallation failed.

Explanation: The 3090 Vector Facility Simulator environment could not be terminated because the NUCXDROP command, which is used to remove the simulator as a CMS nucleus extension, failed.

System Action: No termination processing is performed.

User Action: Determine why the NUCXDROP failed (see *VM/SP CMS Command and Macro Reference*). Correct the problem. Specify VSIM DROP again.

DWF003 Error *nn* from FSWRITE macro writing VECTOR STAT file.

Explanation: An error was encountered in attempting to write the vector statistics file. *nn* is the error code returned by the FSWRITE macro.

System Action: Incomplete output is generated.

User Action: Determine why the FSWRITE failed (see *VM/SP CMS Command and Macro Reference*). Correct the problem. Specify VSIM PRTSTAT again.

DWF004 Error *nn* from FSWRITE macro writing VECTOR MAP file.

Explanation: An error was encountered in attempting to write the vector map file. *nn* is the error code returned by the FSWRITE macro.

System Action: Incomplete output is generated.

User Action: Determine why the FSWRITE failed (see *VM/SP CMS Command and Macro Reference*). Correct the problem. Specify VSIM MAP again.

DWF005 Invalid option to VSIM: *xxxxxxxx*.

Explanation: *xxxxxxxx* was specified as an option to the VSIM command. It is not a valid option.

System Action: None.

User Action: Respecify the option intended.

DWF006 3090 Vector Facility Simulator not installed.

Explanation: A call was made to a selective statistics gathering routine before the simulator was installed.

System Action: The call is ignored.

User Action: Install the simulator and call the routine.

DWF007 Error *nn* from FSREAD macro reading VECTOR STAT file.

Explanation: An error was encountered in attempting to read the vector statistics file. *nn* is the error code returned by the FSREAD macro.

System Action: Incomplete output is generated by the selective statistics gathering routines.

User Action: Determine why the FSREAD failed (see *VM/SP CMS Command and Macro Reference*). Correct the problem. Call the routine again.

DWF008 Error *nn* from FSWRITE macro writing $VECTOR STAT file.

Explanation: An error was encountered in attempting to write the temporary vector statistics file. *nn* is the error code returned by the FSWRITE macro.

System Action: Incomplete output is generated by the selective statistics gathering routine.

User Action: Determine why the FSWRITE failed (see *VM/SP CMS Command and Macro Reference*). Correct the problem. Call the routine again.

DWF009 Error *nn* from FSERASE macro erasing $VECTOR STAT file.

Explanation: An error was encountered in attempting to erase the temporary vector statistics file. *nn* is the error code returned by the FSERASE macro.

System Action: The DWFIN routine was unable to erase the $VECTOR STAT file, therefore any statistics will be appended onto the existing $VECTOR STAT file.

User Action: Determine why the FSERASE failed (see *VM/SP CMS Command and Macro Reference*). Correct the problem. Call the routine again.

DWF010 Error *nn* from FSERASE macro erasing VECTOR STAT file.

Explanation: An error was encountered in attempting to erase the vector statistics file. *nn* is the error code returned by the FSERASE macro.

System Action: The DWFTM routine was unable to erase the VECTOR STAT file. The current statistics will not be in the VECTOR STAT file, but they will be in the $VECTOR STAT file.

User Action: Determine why the FSERASE failed (see *VM/SP CMS Command and Macro Reference*). Correct the problem. Call the routine again.

DWF011 When the *xxxx* option is used, it must appear as the only VSIM parameter. Here it is ignored.

Explanation: *xxxx* can be DROP, HELP, or ?. One of these options appears in a multiple options list on the VSIM command. When any of these options are used, they must appear as the only VSIM option.

System Action: The specified option is ignored. Other options in the list will be processed as usual.

User Action: Reissue the VSIM command with the option alone.

DWF020 *type-exception* at address *address* while processing vector element *element-no*.

Explanation: An interrupt occurred during a vector operation. *type-exception* specifies what kind of interrupt was encountered. The address of the instruction that was interrupted is shown in *address*. If applicable, the vector element on which the interrupt occurred is specified in *element-no*.

System Action: System interrupt handling will proceed as usual.

User Action: Determine the cause of the interrupt and correct the program.

DWF021 Overflow in statistics data area.

Explanation: The value in one of the data areas in which statistic information is being saved became too large for that data area.

System Action: None.

User Action: Issue VSIM CLRSTAT before attempting to generate statistics.

Chapter 5. Installing the Simulator

To install the 3090 Vector Facility Simulator on VM, ensure that there is at least one free cylinder of 3350 space on the target disk, mount the distribution tape on virtual device 181 and issue these commands:

```
1. FILEDEF IN TAP1 SL 1 VOLID VECSIM

2. FILEDEF OUT DISK VLOAD EXEC A

3. MOVEFILE IN OUT

4. VLOAD
```

Figure 15 shows the files installed on your A-disk after executing VLOAD:

```
VSIM      EXEC
VLOAD     EXEC
VSIM      HELPCMS
DWFLKED   LKEDIT
DWFLOD    LOADLIB
DWFOPT    TEXT
DWFCHK    TEXT
DWFSIM    TEXT
DWFSTAT   TEXT
DWFMSG    TEXT
DWFLKED   TEXT
SAMPLE    TEXT
DWFTXT    TXTLIB
DWFSTAT   TXTLIB
```

Figure 15. Simulator Files

Verify Installation

To verify that the 3090 Vector Facility Simulator is installed correctly on your VM system, activate the simulator and run the sample program provided on the distribution tape. The name of the program is SAMPLE TEXT.

To run the sample program under the simulator, issue these commands:

- `VSIM`

  This activates the simulator environment.

- `LOAD SAMPLE (START`

  This loads and runs the sample program.

Message Indicating Successful Installation

If the following message is displayed when you run the sample program, the simulator is installed correctly:

```
SAMPLE PROGRAM COMPLETED SUCCESSFULLY
```

The simulator is not installed correctly if you don't see the above message.

If the simulator is not installed correctly, repeat the installation procedure and check to be sure all the required files are on your A-disk. If you are still unable to find the cause of the problem, contact the Central Service location at the address shown on the inside front cover of this manual.

Files Needed after Installation

After you are sure the simulator is installed correctly, you can erase files that are not permanently needed. The following is a list of simulator files that are needed permanently. **You should not erase these files.**

1. VSIM EXEC
2. VSIM HELPCMS
3. DWFLOD LOADLIB
4. DWFSTAT TXTLIB

Index

N

O

P

Q

R

S

T

U

V

IBM 3090
Vector Facility Simulator
Program Description/
Operations Manual

READER'S COMMENT FORM

Order No. SC23-0336-0

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

How did you use this publication?

[ ] As an introduction
[ ] As a reference manual
[ ] For another purpose (explain) _____

[ ] As a text (student)
[ ] As a text (instructor)

_____

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

What is your occupation? _____

Newsletter number of latest Technical Newsletter (if any) concerning this publication: _____

If you wish a reply, give your name and address: _____

_____

_____

IBM branch office serving you _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automatic mail-sorting equipment. Please use pressure-sensitive or other gummed tape to seal this form.

SC23-0336-0

Reader's Comment Form

Fold and Tape | Please Do Not Staple | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department 52Q MS 458
Neighborhood Road
Kingston, New York 12401

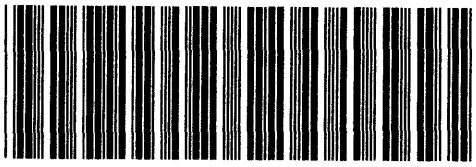
Fold and Tape | Please Do Not Staple | Fold and Tape

IBM®

PRINTED IN U.S.A. SC23-0336-0

SC23-0336-0

SC23-0336-00



PRINTED IN U.S.A. SC23-0336-0