OS/390

Language Environment for OS/390 & VM Vendor Interfaces – IEEE Floating-Point Supplement

SY28-1152-04

OS/390

Language Environment for OS/390 & VM Vendor Interfaces – IEEE Floating-Point Supplement

SY28-1152-04

Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page v.

Fifth Edition, September 1998, Supplement

| This book is a supplement to SY28-1152-04.

This edition applies to Language Environment in OS/390 Version 2 Release 6 (5647-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers' comments may be provided at the back of this publication, or you may address your comments to the following address:

International Business Machines Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie, NY 12601-5400
United States of America

FAX (United States and Canada): 1+914+432-9405
FAX (Other Countries): Your International Access Code +1+914+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)
IBM Mail Exchange: USIB6TC9 at IBMMAIL
Internet e-mail: mhvrcfs@us.ibm.com
World Wide Web: http://www.s390.ibm.com/os390/

If you would like a reply, be sure to include your name, address, telephone number, or FAX number. Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Notices

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 product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Request

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Any pointers in this publication to non-IBM Web sites are provided for convenience only, and do not in any manner serve as an endorsement of these Web sites. IBM accepts no responsibility for the content or use of non-IBM Web sites specifically mentioned in this publication or accessed through an IBM Web site that is mentioned in this publication.

Programming Interface Information

This book documents intended Programming Interfaces that allow the customer to write programs to obtain the services of Language Environment in OS/390.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM	IBMLink	Language Environment
OS/390	VM/ESA	

IEEE is a trademark in the United States and other countries of the Institute of Electrical and Electronics Engineers, Inc.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

About This Book

IBM OS/390 Language Environment for OS/390 & VM (also called Language Environment) provides common services and language-specific routines in a single run-time environment for C, C++, COBOL, Fortran (OS/390 only; no support for VM/ESA, OS/390 UNIX System Services, or CICS), PL/I, and assembler applications. It offers consistent and predictable results for language applications, independent of the language in which they are written.

Language Environment is the prerequisite run-time environment for applications generated with the following IBM compiler products:

- OS/390 C/C++
- C for VM/ESA
- C/C++ Compiler for MVS/ESA
- AD/Cycle C/370 Compiler
- COBOL for OS/390 & VM
- COBOL for MVS & VM (formerly COBOL/370)
- PL/I for MVS & VM
- AD/Cycle PL/I for MVS & VM
- VS FORTRAN and FORTRAN IV (in compatibility mode)

Language Environment supports, but is not required for, an interactive debug tool for debugging applications in your native OS/390 environment. The IBM interactive Debug Tool is available with OS/390 or with the latest releases of the C/C++, COBOL, and PL/I compiler products.

Language Environment supports, but is not required for, VS Fortran Version 2 compiled code (OS/390 only).

Language Environment consists of the common execution library (CEL) and the run-time libraries for C/C++, COBOL, Fortran, and PL/I.

This book documents support for the Institute of Electrical and Electronics Engineers (IEEE) floating-point data type, in conformance with the IEEE 754 standard. This support applies primarily to the OS/390 C/C++ components of Language Environment.

Summary of Changes

Summary of Changes
for SY28-1152-04
for OS/390 Version 2 Release 6
- IEEE Floating-Point Supplement

The following changes apply only to OS/390 Version 2 Release 6.

New Information

Support for IEEE floating-point has been added to the C/C++ components of Language Environment.

Chapter 1. IEEE Floating-Point

Introduction

The IBM S/390 Generation 5 Server includes support for IEEE binary floating-point (IEEE floating-point) as defined by the ANSI/IEEE Standard 754-1985, IEEE Standard for Binary Floating-Point Arithmetic. Starting with Version 2 Release 6, OS/390 (including the Language Environment and C/C++ components) has added support for IEEE floating-point.

Notes:

1. You must have OS/390 Release 6 to use the IEEE floating-point instructions. In Release 6, the base control program (BCP) is enhanced to support the new IEEE floating-point hardware in the IBM S/390 Generation 5 Server. This enables programs running on OS/390 Release 6 to use the IEEE floating-point instructions and 16 floating-point registers. In addition, the BCP provides simulation support for all the new floating-point hardware instructions. This enables applications that make light use of IEEE floating-point, and can tolerate the overhead of software simulation, to execute on OS/390 Release 6 without requiring an IBM S/390 Generation 5 Server.
2. The terms binary floating-point and IEEE floating-point are used interchangeably. The abbreviations BFP and HFP, which are used in some function names, refer to binary floating-point and S/390 hexadecimal floating-point (hexadecimal floating-point), respectively.

The C/C++ compiler provides a FLOAT option to select the format of floating-point numbers produced in a compile unit. The FLOAT option allows you to select either IEEE floating-point or hexadecimal floating-point format. For information on the C/C++ compiler options see the *OS/390 C/C++ User's Guide*.

The C/C++ run-time library interfaces, which formerly supported only hexadecimal floating-point format, have been changed in OS/390 Version 2 Release 6 to support both IEEE floating-point and hexadecimal floating-point formats. These interfaces are documented in the *OS/390 C/C++ Run-Time Library Reference*.

The primary documentation for the IEEE floating-point support is contained in the *Enterprise Systems Architecture/390 Principles of Operation*, and the *OS/390 C/C++ User's Guide*.

IEEE floating-point is provided on S/390 primarily to enhance interoperability and portability between S/390 and other platforms. It is anticipated that IEEE floating-point will be most commonly used for new and ported applications, and in emerging environments, such as Java. Customers should not migrate existing applications that use hexadecimal floating-point to IEEE floating-point, unless there is a specific reason (such as a need to interoperate with a non-S/390 platform).

IBM does not recommend mixing floating-point formats in an application. However, for applications which must handle both formats, the C/C++ run-time library does provide some support which is described below.

Functions

__chkbfp() — Check IEEE Facilities Usage

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
int __chkbfp(void);
```

General Description

The system sets a flag in the secondary task control block (STCB) when IEEE floating-point hardware facilities or simulated facilities (including additional floating-point (AFP) registers in hexadecimal floating-point) are first accessed by a task. The __chkbfp() function returns the state of this flag.

Returned Value

0 IEEE floating-point facilities (including AFP registers in hexadecimal floating-point mode) have *not* been used by the task.

1 IEEE floating-point facilities have been used by the task.

Related Information

- "__fp_level() — Determine Type of IEEE Facilities Available" on page 13

__fp_btoh() — Convert from IEEE Floating-Point to Hexadecimal Floating-Point

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
int __fp_btoh(void *src_ptr, int src_type,
              void *trg_ptr, int trg_type,
              int rmode);
```

General Description

The __fp_btoh() function converts data in IEEE floating-point format, pointed to by *src_ptr*, to hexadecimal floating-point format, and stores the hexadecimal floating-point value at the location pointed to by *trg_ptr*. *src_ptr* and *trg_ptr* point to C floating-point variables of type float, double, or long double as indicated by *src_type* and *trg_type*. Valid values for *src_type* and *trg_type* are _FP_FLOAT, _FP_DOUBLE, and _FP_LONG_DOUBLE. *rmode* specifies rounding mode for inexact mappings. Valid values are:

Value	Description
_FP_BH_NR	No rounding
_FP_BH_RZ	Rounding toward zero
_FP_BH_BRN	Biased round to nearest
_FP_BH_RN	Round to nearest
_FP_BH_RP	Round toward +infinity
_FP_BH_RM	Round toward –infinity

Returned Value

If invalid *src_type*, *trg_type*, or *rmode* is specified, __fp_btoh() returns –1. Otherwise, it returns the following values:

0 Zero (IEEE floating-point +zero or –zero value mapped to hexadecimal floating-point +zero or –zero value, respectively).

1 Underflow (IEEE floating-point value is too small to map to hexadecimal floating-point). In this case **trg_ptr* is set to the hexadecimal floating-point value corresponding to the smallest convertible IEEE floating-point value.

2 Success (with rounding performed as indicated by *rmode*).

3 Overflow (IEEE floating-point value is too large to map to hexadecimal floating-point). In this case **trg_ptr* is set to the hexadecimal floating-point value corresponding to the largest convertible IEEE floating-point value.

Related Information

- "__fp_htob() — Convert from Hexadecimal Floating-Point to IEEE Floating-Point" on page 11

__fp_cast() — Cast between Floating-Point Data Types

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
int __fp_cast(int mode, void *src_ptr, int src_type,
                        void *trg_ptr, int trg_type);
```

General Description

The __fp_cast() function casts between C floating-point data types, when the data format does not match the format specified by the FLOAT compiler option. The *mode* parameter indicates the format of source and target floating-point values pointed to by *src_ptr* and *trg_ptr*. Valid values for the *mode* parameter are _FP_HFP_MODE for hexadecimal floating-point format and _FP_BFP_MODE for IEEE floating-point format.

src_type and *trg_type* indicate the C data type (float, double, or long double) of the source and target floating-point values, respectively. Valid values for *src_type* and *trg_type* are _FP_FLOAT, _FP_DOUBLE or _FP_LONG_DOUBLE.

Returned Value

If invalid values for *mode*, *src_type* or *trg_type* are specified, __fp_cast() returns –1. Otherwise, it performs the requested cast and returns 0.

Related Information

- "__fp_setmode() — Set IEEE or Hexadecimal Mode" on page 15
- "__isBFP() — Determine Application Floating-Point Mode" on page 18

__fpc_rd() — Read Floating-Point Control Register

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
void __fpc_rd(_FP_fpcreg_t *fpc_ptr);
```

General Description

The __fpc_rd() function stores the contents of the floating-point control (FPC) register at the location pointed to by *fpc_ptr*.

Returned Value

None

Related Information

- "__fpc_rs() — Read Floating-Point Control Register and Change Rounding Mode Field" on page 7
- "__fpc_rw() — Read and Write the Floating-Point Control Register" on page 8
- "__fpc_sm() — Set Floating-Point Control Register Rounding Mode Field" on page 9
- "__fpc_wr() — Write the Floating-Point Control Register" on page 10
- "__fp_read_rnd() — Determine Rounding Mode" on page 14

__fpc_rs() — Read Floating-Point Control Register and Change Rounding Mode Field

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
void __fpc_rs(_FP_fpcreg_t *cur_ptr, _FP_rmode_t rmode);
```

General Description

The __fpc_rs() function stores the current contents of the floating-point control (FPC) register at the location pointed to by *cur_ptr* and then sets the rounding mode field of the FPC based on the value specified by *rmode* as follows:

Value	Rounding Mode
_RMODE_RN	Round to nearest
_RMODE_RZ	Round toward zero
_RMODE_RP	Round toward +Infinity
_RMODE_RM	Round toward –Infinity

Note: When processing IEEE floating-point values, the C/C++ run-time library math functions require IEEE rounding mode of round to nearest. The C/C++ run-time library takes care of setting round to nearest rounding mode while executing math functions and restoring application rounding mode before returning to the caller.

Returned Value

None

Related Information

- "__fpc_rd() — Read Floating-Point Control Register" on page 6
- "__fpc_rw() — Read and Write the Floating-Point Control Register" on page 8
- "__fpc_sm() — Set Floating-Point Control Register Rounding Mode Field" on page 9
- "__fpc_wr() — Write the Floating-Point Control Register" on page 10
- "__fp_swap_rnd() — Swap Rounding Mode" on page 16

__fpc_rw() — Read and Write the Floating-Point Control Register

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
void __fpc_rw(_FP_fpcreg_t *cur_ptr, _FP_fpcreg_t *new_ptr);
```

General Description

The __fpc_rw() function stores the current contents of the floating-point control (FPC) register at the location pointed to by *cur_ptr* and then replaces the contents of the floating-point control (FPC) register with the value pointed to by *new_ptr*.

Note: When processing IEEE floating-point values, the C/C++ run-time library math functions require IEEE rounding mode of round to nearest. The C/C++ run-time library takes care of setting round to nearest rounding mode while executing math functions and restoring application rounding mode before returning to the caller.

Returned Value

None

Related Information

- "__fpc_rd() — Read Floating-Point Control Register" on page 6
- "__fpc_rs() — Read Floating-Point Control Register and Change Rounding Mode Field" on page 7
- "__fpc_sm() — Set Floating-Point Control Register Rounding Mode Field" on page 9
- "__fpc_wr() — Write the Floating-Point Control Register" on page 10
- "__fp_swap_rnd() — Swap Rounding Mode" on page 16

__fpc_sm() — Set Floating-Point Control Register Rounding Mode Field

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
void __fpc_sm(_FP_rmode_t rmode);
```

General Description

The __fpc_sm() function changes the rounding mode field of the floating-point control (FPC) register based on the value of *rmode* as follows:

Value	Description
_RMODE_RN	Round to nearest
_RMODE_RZ	Round toward zero
_RMODE_RP	Round toward +infinity
_RMODE_RM	Round toward –infinity

Note: When processing IEEE floating-point values, the C/C++ run-time library math functions require IEEE rounding mode of round to nearest. The C/C++ run-time library takes care of setting round to nearest rounding mode while executing math functions and restoring application rounding mode before returning to the caller.

Returned Value

None

Related Information

- "__fpc_rd() — Read Floating-Point Control Register" on page 6
- "__fpc_rs() — Read Floating-Point Control Register and Change Rounding Mode Field" on page 7
- "__fpc_wr() — Write the Floating-Point Control Register" on page 10
- "__fpc_rw() — Read and Write the Floating-Point Control Register" on page 8
- "__fp_swap_rnd() — Swap Rounding Mode" on page 16

__fpc_wr() — Write the Floating-Point Control Register

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
void __fpc_wr(_FP_fpcreg_t *fpc_ptr);
```

General Description

The __fpc_wr() function replaces the contents of the floating-point control (FPC) register with the value pointed to by *fpc_ptr*.

Note: When processing IEEE floating-point values, the C/C++ run-time library math functions require IEEE rounding mode of round to nearest. The C/C++ run-time library takes care of setting round to nearest rounding mode while executing math functions and restoring application rounding mode before returning to the caller.

Returned Value

None

Related Information

- "__fpc_rd() — Read Floating-Point Control Register" on page 6
- "__fpc_rs() — Read Floating-Point Control Register and Change Rounding Mode Field" on page 7
- "__fpc_rw() — Read and Write the Floating-Point Control Register" on page 8
- "__fpc_sm() — Set Floating-Point Control Register Rounding Mode Field" on page 9
- "__fp_swap_rnd() — Swap Rounding Mode" on page 16

__fp_htob() — Convert from Hexadecimal Floating-Point to IEEE Floating-Point

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
int __fp_htob(void *src_ptr, int src_type,
              void *trg_ptr, int trg_type,
              int rmode);
```

General Description

The __fp_htob() function converts data in hexadecimal floating-point format, pointed to by *src_ptr*, to IEEE floating-point format, and stores the IEEE floating-point value at the location pointed to by *trg_ptr*. *src_ptr* and *trg_ptr* point to C floating-point variables of type float, double, or long double as indicated by *src_type* and *trg_type*. Valid values for *src_type* and *trg_type* are _FP_FLOAT, _FP_DOUBLE, and _FP_LONG_DOUBLE. *rmode* specifies rounding mode for inexact mappings. Valid values are:

Value	Description
_FP_HB_NR	No rounding
_FP_HB_RZ	Rounding toward zero
_FP_HB_BRN	Biased round to nearest
_FP_HB_RN	Round to nearest
_FP_HB_RP	Round toward +infinity
_FP_HB_RM	Round toward –infinity

Returned Value

If invalid *src_type*, *trg_type*, or *rmode* is specified, __fp_htob() returns –1. Otherwise, it returns the following values:

0 Zero (hexadecimal floating-point +zero or –zero value mapped to IEEE floating-point +zero or –zero value, respectively).

1 Underflow (hexadecimal floating-point value is too small to map to IEEE floating-point). In this case **trg_ptr* is set to the IEEE floating-point value corresponding to the smallest convertible hexadecimal floating-point value.

2 Success (with rounding performed as indicated by *rmode*).

3 Overflow (hexadecimal floating-point value is too large to map to IEEE floating-point). In this case **trg_ptr* is set to the IEEE floating-point value corresponding to the largest convertible hexadecimal floating-point value.

Related Information

- "__fp_btoh() — Convert from IEEE Floating-Point to Hexadecimal Floating-Point" on page 3

__fp_level() — Determine Type of IEEE Facilities Available

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
int __fp_level(void);
```

General Description

The system provides simulation of IEEE floating-point hardware (including additional floating-point registers in hexadecimal mode). The __fp_level() function determines the level of IEEE floating-point support available.

Returned Value

0 No IEEE floating-point support available.

1 IEEE floating-point simulation is available.

2 IEEE floating-point hardware is available.

Related Information

- "__chkbfp() — Check IEEE Facilities Usage" on page 2

__fp_read_rnd() — Determine Rounding Mode

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <float.h>
__fprnd_t __fp_read_rnd(void);
```

General Description

For an application running in IEEE floating-point mode, the __fp_read_rnd() function returns the current rounding mode indicated by the rounding mode field of the floating-point control (FPC) register. For an application running in hexadecimal floating-point mode, __fp_read_rnd() returns 0.

Returned Value

For an application running in IEEE floating-point mode, __fp_read_rnd() returns the following:

Value	Rounding Mode
_FP_RND_RZ	Round toward 0
_FP_RND_RN	Round to nearest
_FP_RND_RP	Round toward +infinity
_FP_RND_RM	Round toward –infinity

For an application running in hexadecimal floating-point mode, __fp_read_rnd() returns 0.

Related Information

- "__fp_setmode() — Set IEEE or Hexadecimal Mode" on page 15
- "__fp_swap_rnd() — Swap Rounding Mode" on page 16
- "__isBFP() — Determine Application Floating-Point Mode" on page 18

__fp_setmode() — Set IEEE or Hexadecimal Mode

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
void __fp_setmode(int mode);
```

General Description

The __fp_setmode() function sets a flag to tell C/C++ run-time library functions whether to interpret parameters as IEEE floating-point or hexadecimal floating-point values based on the value of *mode* as follows:

Value	Description
_FP_MODE_RESET	Use the FLOAT compile option to determine the format of floating-point parameters.
_FP_HFP_MODE	Interpret parameters as hexadecimal floating-point values.
_FP_BFP_MODE	Interpret parameters as IEEE floating-point values.

Note: The compiler defines the __BFP__ macro if the FLOAT(IEEE) compile option is chosen. Otherwise, it undefines the __BFP__ macro. Headers related to floating-point, <float.h>, <limits.h>, and <math.h>, use the __BFP__ macro to select floating-point-type-specific bindings for functions and constants at compile-time. Applications which use __fp_setmode() must use the _FP_MODE_VARIABLE macro to prevent type-specific compile-time binding of functions and constants as illustrated by the following example:

```
#define _FP_MODE_VARIABLE
#include <float.h>
#include <limits.h>
#include <math.h>
...
```

Returned Value

None

Related Information

- "__fp_cast() — Cast between Floating-Point Data Types" on page 5
- "__fp_swap_rnd() — Swap Rounding Mode" on page 16
- "__isBFP() — Determine Application Floating-Point Mode" on page 18

__fp_swap_rnd() — Swap Rounding Mode

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <float.h>
__fprnd_t __fp_swap_rnd(__fprnd_t rmode);
```

General Description

For an application running in IEEE floating-point mode, the __fp_swap_rnd() function returns the current rounding mode specified by the rounding mode field of the floating-point control (FPC) register and sets the rounding mode field in the FPC based on the value of *rmode* as follows:

Value	Rounding Mode
_FP_RND_RZ	Round toward 0
_FP_RND_RN	Round to nearest
_FP_RND_RP	Round toward +infinity
_FP_RND_RM	Round toward –infinity

Note: When processing IEEE floating-point values, the C/C++ run-time library math functions require IEEE rounding mode of round to nearest. The C/C++ run-time library takes care of setting round to nearest rounding mode while executing math functions and restoring application rounding mode before returning to the caller.

For an application running in hexadecimal floating-point mode, __fp_swap_rnd() returns 0.

Returned Value

For an application running in IEEE floating-point mode, __fp_swap_rnd() function returns the following:

Value	Description
_FP_RND_RZ	Round toward 0
_FP_RND_RN	Round to nearest
_FP_RND_RP	Round toward +infinity
_FP_RND_RM	Round toward –infinity

For an application running in hexadecimal floating-point mode, __fp_swap_rnd() returns 0.

Related Information

- “__fp_read_rnd() — Determine Rounding Mode” on page 14
- “__fp_setmode() — Set IEEE or Hexadecimal Mode” on page 15
- “__isBFP() — Determine Application Floating-Point Mode” on page 18

__isBFP() — Determine Application Floating-Point Mode

Standards

Standards / Extensions	C or C++	Dependencies
	both	OS/390 V2R6

Format

```
#include <_Ieee754.h>
int __isBFP(void)
```

General Description

The __isBFP() function determines the application floating-point mode.

Returned Value

__isBFP() returns 1 if the floating-point mode of the caller is IEEE, and returns 0 if the floating-point mode of the caller is hexadecimal.

Related Information

- "__fp_read_rnd() — Determine Rounding Mode" on page 14
- "__fp_setmode() — Set IEEE or Hexadecimal Mode" on page 15
- "__fp_swap_rnd() — Swap Rounding Mode" on page 16

Communicating Your Comments to IBM

OS/390
Language Environment for OS/390 & VM
Vendor Interfaces
– IEEE Floating-Point Supplement

Publication No. SY28-1152-04

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a reader's comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
  - FAX: (International Access Code)+1+914+432-9405
- If you prefer to send comments electronically, use this network ID:
  - IBM Mail Exchange: USIB6TC9 at IBMMAIL
  - Internet e-mail: mhvrcfs@us.ibm.com
  - World Wide Web: http://www.s390.ibm.com/os390

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies

Optionally, if you include your telephone number, we will be able to respond to your comments by phone.

Reader's Comments — We'd Like to Hear from You

OS/390
Language Environment for OS/390 & VM
Vendor Interfaces
– IEEE Floating-Point Supplement

Publication No. SY28-1152-04

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.

Today's date: _____

What is your occupation?

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

How did you use this publication?

[ ]	As an introduction	[ ]	As a text (student)
[ ]	As a reference manual	[ ]	As a text (instructor)
[ ]	For another purpose (explain)		

_____

_____

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:

Name	Address
Company or Organization	
Phone No.	

Reader's Comments — We'd Like to Hear from You
SY28-1152-04

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie NY 12601-5400

Fold and Tape | **Please do not staple** | Fold and Tape

Cut or Fold
Along Line

Program Number: 5647-A01

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SY28-1152-04