LY28-1690-0
File No. S370-36

Program Product

MVS/Extended Architecture System Logic Library: Functional Subsystem Interface

MVS/System Product:

JES3 Version 2	**5665-291**
JES2 Version 2	**5740-XC6**

This publication supports MVS/System Product Version 2 Release 2.0, and contains information that was formerly presented in MVS/Extended Architecture System Logic Library Volume 7, LY28-1227-3, which applies to MVS/System Product Version 2 Release 1.7. See the Summary of Amendments for more information.

First Edition (June, 1987)

This edition applies to Version 2 Release 2.0 of MVS/System Product 5665-291 or 5740-XC6 and to all subsequent releases until otherwise indicated in new editions or technical newsletters. 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 Bibliography, GC20-0001, for the editions that 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 readers' 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 D58, Building 921-2, PO Box 390, Poughkeepsie, N.Y. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

PREFACE

The MVS/Extended Architecture System Logic Library is intended for people who debug or modify the MVS control program. It describes the logic of most MVS control program functions that are performed after master scheduler initialization completes. For detailed information about the MVS control program prior to this point, refer to MVS/Extended Architecture System Initialization Logic. For general information about the MVS control program and the relationships among the components that make up the MVS control program, refer to the MVS/Extended Architecture Overview. To obtain the names of publications that describe some of the components not in the System Logic Library, refer to the section Corequisite Reading in the Master Preface in MVS/Extended Architecture System Logic Library: Master Table of Contents and Index.

HOW THE LIBRARY IS ORGANIZED

SET OF BOOKS

The System Logic Library consists of a set of books. Two of the books provide information that is relevant to the entire set of books:

1. The MVS/Extended Architecture System Logic Library: Master Table of Contents and Index contains the master preface, the master table of contents, and the master index for the other books in the set.

2. The MVS/Extended Architecture System Logic Library: Module Descriptions contains module descriptions for all of the modules in the components documented in the System Logic Library and an index.

Each of the other books (referred to as component books) in the set contains its own table of contents and index, and describes the logic of one of the components in the MVS control program.

ORGANIZATION OF THE COMPONENTS

Most component books contain information about one component in the MVS control program. However, some component books (such as System Logic Library: Initiator/Terminator) contain more than one component if the components are closely related, frequently referenced at the same time, and not so large that they require a book of their own.

A three or four character mnemonic is associated with each component book and is used in all diagram and page numbers in that book. For example, the mnemonic ASM is associated with the book MVS/Extended Architecture System Logic Library: Auxiliary Storage Management. All diagrams in this book are identified as Diagram ASM-n, and all pages as ASM-n, where n represents the specific diagram or page number. Whenever possible, the existing component acronym is used as the mnemonic for the component book. The Table of Book Titles in the Master Preface in MVS/Extended Architecture System Logic Library: Master Table of Contents and Index lists the book titles, the components included in each book (if a book contains more than one component), the mnemonics for the books, and the order number for each book.

HOW TO USE THE LIBRARY

To help you use this library efficiently, the following topics cover

- How to find information using book titles and the master index
- What types of information are provided for each component
- How to obtain further information about other books in the System Logic Library

FINDING INFORMATION USING THE BOOK TITLES

As you become familiar with the book titles, MVS component names and mnemonics, and the book contents, you will be able to use the System Logic Library as you would an encyclopedia and go directly to the book that you need. We recommend that you group the books in alphabetical order for easy reference, or, if you are familiar with MVS, that you to group the books by related functions.

The Table of Book Titles in the Master Preface in MVS/Extended Architecture System Logic Library: Master Table of Contents and Index contains a list of book titles and mnemonics. It provides a quick reference to all the books, and their corresponding components, in the System Logic Library.

FINDING INFORMATION USING THE MASTER INDEX

If you are not sure which book contains the information you are looking for, you can locate the book and the page on which the information appears by using the master index in System Logic Library: Master Table of Contents and Index. For the component books, the page number in an index entry consists of the mnemonic for the component and the page number; for System Logic Library: Module Descriptions, the page number consists of the mnemonic "MOD" and the page number.

For example:

ASM-12 refers to MVS/Extended Architecture System Logic Library: Auxiliary Storage Management, page ASM-12.

MOD-245 refers to MVS/Extended Architecture System Logic Library: Module Descriptions, page MOD-245.

INFORMATION PROVIDED FOR MOST COMPONENTS

The following information is provided for most of the components described in the System Logic Library.

1. An introduction that summarizes the component's function
2. Control block overview figures that show significant fields and the chaining structure of the component's control blocks
3. Process flow figures that show control flow between the component's object modules
4. Module information that describes the functional organization of a program. This information can be in the form of:
   - Method-of-Operation diagrams and extended descriptions.
   - Automatically-generated prose. The automated module information is generated from the module prologue and the code itself. It consists of three parts: module description, module operation summary, and diagnostic aids.

5. Module descriptions that describe the operation of the modules (the module descriptions are contained in System Logic Library: Module Descriptions)

Some component books also include diagnostic techniques information following the Introduction.

FURTHER INFORMATION

For more information about the System Logic Library, including the order numbers of the books in the System Logic Library, see the Master Preface in MVS/Extended Architecture System Logic Library: Master Table of Contents and Index.

CONTENTS

FIGURES

SUMMARY OF AMENDMENTS

Summary of Amendments for LY28-1690-0 for MVS/System Product Version 2 Release 2.0

This publication is new for MVS System Product Version 2 Release 2.0. It contains information that was reorganized from the FSI section in MVS/XA System Logic Library Volume 7, LY28-1230-4, which applies to MVS/XA System Product Version 2 Release 1.7.

Minor technical and editorial changes were made throughout this publication to support MVS/System Product Version 2 Release 2.0.

INTRODUCTION

The functional subsystem interface (FSI) provides an interface between the JESs (JES2 and JES3) and a functional subsystem address space and its functional subsystem application (FSA).

The functional subsystem (FSS) performs such JES functions as device-dependent functions. Thus, FSS processing is an extension of JES processing. FSS processing takes place in its own address space, separate from the JES address space. With the FSS, specific functional processing that JES would have had to perform in its own address space (such as device processing) can instead be isolated from JES and performed by a set of function-specific programs, called the functional subsystem application (FSA). These programs reside in an FSS address space.

The FSA operates under control of a standard task control block (TCB) dispatching structure for an MVS address space. More than one FSS address space can be defined in the system, each of which has its own unique FSS identifier (FSID). At least one FSA exists in each FSS address space (except for the JES3 converter/interpreter FSS address space); an FSA can represent, for example, a single printing subsystem. JES can communicate with each FSS address space and with each FSA.

The functional subsystem interface (FSI) provides two-way communication between JES, and the FSS or the FSA. The FSI is a collection of generalized functions, or services, that JES and the FSS/FSA can request by issuing the FSIREQ macro instruction. The FSI services also allow the FSS/FSA to access JES-controlled spool data sets.

JES uses FSI services to create the FSS address space and the FSA. JES initiates the FSS in response to a specific request, such as an operator command. JES initialization statements and parameters define the FSS. JES:

- Controls its own resources (for example, SYSOUT data sets), even though they are used by the FSS.
- Controls the scheduling of input and output.
- Manages operator communication with the FSS/FSA function.
- Coordinates the termination and restart of the FSS address space.

Figure 1 on page FSI-4 shows the structure of the MVS, JES, and FSS address spaces, the FSA, and the FSI.

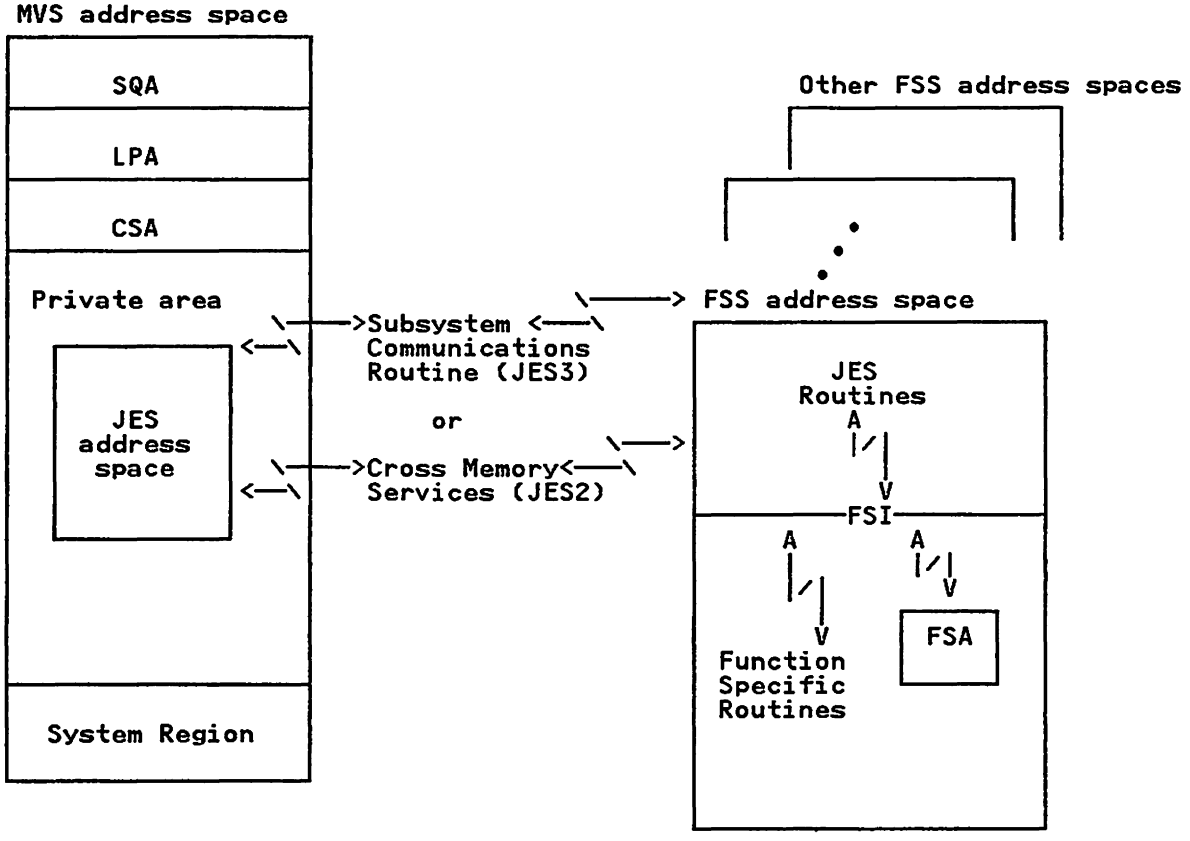

Figure 1. Relationships Between the MVS, JES, and FSS Address Spaces, the FSA, and the FSI

INITIALIZING A FUNCTIONAL SUBSYSTEM (FSS)

An installation defines a functional subsystem (FSS) during JES initialization. It defines the FSS for a specific device or other component and associates this device with the FSS being defined. After JES is initialized and the FSS is started, a functional subsystem application (FSA) becomes directly associated with this specific device or component. Therefore, an FSA is associated with one and only one peripheral device or component.

Once initialization has completed, JES and the FSS/FSA can communicate with each other by using the FSI. For information about how JES2 initializes the FSS and the FSI, see JES2 Logic. For information about how JES3 initializes the FSS and the FSI, see the JES3 Logic Library.

FUNCTIONAL SUBSYSTEM INTERFACE SERVICES

The FSI services allow JES and the FSS and FSA to communicate with each other. The FSI services are requested when JES or the FSA issues the FSIREQ macro instruction. This section describes the FSI services that the FSS, JES, and the FSA can request.

CONNECT/DISCONNECT SERVICES

The CONNECT service informs JES that the FSS or the FSA is initialized and ready to be used. The DISCONNECT service informs JES that the FSS or the FSA is terminated and is no longer ready for work.

COMMUNICATIONS SERVICES

There are three communications services provided by the functional subsystem interface (FSI).

ORDER: JES communicates with the the FSS and the FSA mainly through the ORDER service. For example, an operator device control command generates orders. When JES issues an order, the FSA performs the required processing.

SEND: The SEND service sends responses to JES that describe the outcome of the asynchronous order processing.

POST: The POST service prompts the FSA to issue a new request for a data set.

Note: Because the JES3 converter/interpreter FSS address space does not have an FSA, it does not use the POST service.

DATA SET ACCESS SERVICES

The FSA uses the data set access services to obtain access to an existing JES spool data set and to the spool data set's characteristics. After obtaining access to a spool data set, the FSA can get and free records on the data set.

Note: Because the JES3 converter/interpreter FSS address space does not have an FSA, it does not use these data set access services.

GETDS: The GETDS service gets a particular spool data set (similar to allocating and opening the data set).

RELDS: The RELDS service releases the data set (similar to closing and unallocating the data set).

GETREC: The GETREC service gets records from the spool data set.

FREEREC: The FREEREC service frees the storage occupied by these records.

CHKPT: The CHKPT service saves the environment of the data set currently being processed.

SERVICES USED BY THE FSS

The FSS requests FSI services to inform JES that the FSS or the FSA has been initialized and is ready for work. The FSS issues the FSIREQ macro instruction to request these FSI services. The FSIREQ macro instruction expands to an SSI call (SSI function code 53) for the following services.

CONNECT: JES activates the FSS and the FSA upon receipt of an operator command to start the FSS functions. JES first issues the **MVS START command** to activate the FSS, and the FSS responds with a request for the connect service. The connect service establishes the FSI to JES. JES then activates the FSA by issuing the START FSA order, and the FSA also responds with a request for the connect service.

The connect process occurs at two levels: the FSS connect and the FSA connect.

- The FSS connect service informs JES that FSS initialization has completed and the FSS is ready to be used.
- The FSA connect service informs JES that the FSA is ready to process work on behalf of JES requests.

DISCONNECT: JES deactivates the FSS and the FSA through the disconnect service. JES first issues the STOP FSA order to deactivate the FSA, and the FSA responds with a request for the disconnect service. JES then deactivates the FSS by issuing the STOP FSS order the disconnect service. If the FSS address space abnormally terminates or is cancelled, the FSS cannot respond to JES with the disconnect service. In this case, JES assumes that the FSS address space has failed.

The disconnect process is the reverse of the connect process.

- The FSA disconnect service informs JES that the FSA is no longer available to process work. If the FSS abnormally terminates without disconnecting the FSA, JES disconnects the FSA.
- The FSS disconnect service informs JES that the FSS address space is terminating, and that further use of the FSI for that FSS address space is not permitted.

SERVICES USED BY JES

JES requests FSI services to affect how the FSS and the FSA interact with the peripheral device or component that the FSA supports. JES issues the FSIREQ macro instruction to request these FSI services. The FSIREQ macro instruction expands to a branch and link register (BALR) instruction for these services. The FSS or the FSA satisfies these requests.

JES requests the following FSI services:

ORDER: JES uses the ORDER service to communicate with the FSS and the FSA. The ORDER services are:

- Start FSA: Causes the FSA to request the connect service. This order starts the FSA to interact with the function or device associated with it. This order also causes the FSI to be initialized at the FSA level.

  Note: JES must have already issued the MVS START command to activate the FSS prior to issuing the Start FSA order.
- Stop FSA: Causes the FSA to request the disconnect service, which terminates the FSA.
- Stop FSS: Causes the FSS to request the disconnect service, which terminates the FSS address space.
- Start device: Initializes the device associated with the FSA and allows the FSA to request spool data sets for processing.
- Stop device: Causes the FSA to complete the processing of the current data set and to terminate processing for the device.
- Query: Tells the FSA to obtain information about the current data set and to immediately inform JES.
- Set: Tells the FSA to set or change certain device characteristics.
- Synch: Requests that FSA processing be synchronized to the point of actual processing. JES issues the synch order for repositioning or interruption of this processing. For buffered devices, the synch order results in the FSA issuing the RELDS service against data sets that are in the buffer,

but are not yet visible to the operator. The FSA performs four actions (in the following order) against the data set that is currently visible to the operator:

1. Synchronizes the device to some specified point in the data set that is currently visible to the operator.
2. Repositions the data set from the point of synchronization.
3. Updates certain data set or device characteristics.
4. Interrupts processing of the data set.

- Operator intervention: Informs the FSA that a change to the device setup (such as a change in forms) that involves operator intervention is required. The FSA prepares the device for the change.

POST: When the FSA issues a GETDS request that JES cannot immediately satisfy (for example, because of a change in forms), JES informs the FSA that a spool data set is not available and that the GETDS request was not satisfied. When a spool data set becomes available, JES uses the post service to prompt the FSA to reissue the GETDS request. Until the POST is issued, the FSA effectively waits. However, any spool data sets that were already processing in the FSA continue to process.

SERVICES USED BY THE FSA

The FSA requests FSI services to obtain work for processing. The FSA issues the FSIREQ macro instruction to make requests known to JES. (The FSIREQ macro instruction expands to a branch and link register (BALR) instruction for these services.) JES then satisfies the request.

JES Address Space Processing

The FSA requests the following FSI services that are performed within the JES address space:

GETDS: The FSA uses the GETDS request to access a spool data set. JES satisfies this request by selecting a spool data set according to the selection criteria specified by an installation. This criteria is usually the characteristics matching the current device setup. When responding to a GETDS request, JES releases control of the spool data set to the FSA.

Note: When the FSA issues a GETDS request, JES2 satisfies the request by selecting a group of spool data sets from the JES2 address space and keeps the group in the JES2 portion of the FSS address space. (This group can consist of one or more than one data set.) The next time the FSA issues a GETDS request against a data set in this group, JES2 selects a data set from the group stored within the FSS address space. Therefore, not all JES2 GETDS requests are processed through the JES2 address space. (JES3, however, satisfies a GETDS request by selecting only one data set at a time from the JES3 address space.)

RELDS: The FSA uses the RELDS request to terminate or discontinue FSA processing for an output data set that was originally obtained through a GETDS request. This returns control of the data set to JES. When issuing a RELDS request, the FSA indicates one of the following conditions:

- Complete - The data set has been successfully processed and can be deleted by JES.
- Incomplete - The data set has been only partially printed and JES should requeue the data set to complete its processing.

Note: As with the GETDS request, when the FSA issues a RELDS request, JES2 keeps the data set in the JES2 portion of the FSS address space. When a RELDS request has been issued against every data set in a particular group (obtained through a GETDS request), JES2 returns the group of data sets to the JES2 address space. Therefore, not all JES2 RELDS requests are processed through the JES2 address space.

SEND: The FSA uses the SEND request to respond asynchronously to orders that are issued by JES. (When issuing an order, JES informs the FSA to respond with a SEND request.) For example, when JES issues the START DEVICE order, the FSA uses a SEND request to inform JES that the device has been started. JES determines how the information obtained through the SEND service will be used.

The FSA can also issue an unsolicited SEND request. The JES3 converter/interpreter FSS is the principal user of the unsolicited SEND request.

FSA Address Space Processing

The FSA requests certain FSI services that can be performed within the FSS address space. The FSA requests these services for a data set that has already been obtained through the GETDS service. These requests are:

GETREC: The FSA uses the GETREC service to obtain records from the output data set that it is processing. Unless otherwise indicated by the FSA, JES accesses these records sequentially from the beginning of the data set. The FSA can also use the GETREC service to re-access a data set from the beginning or from a specified CHKPT position within the data set. JES cannot reuse storage buffers associated with these records until the buffers are freed by the FSA through the FREEREC service.

FREEREC: The FSA uses the FREEREC service to release the storage buffers associated with the records that were obtained through the GETREC service.

CHKPT: An FSA periodically uses the CHKPT request to save the status of the output data set that is currently being printed. JES saves this information on the JES spool. The FSA can restart processing at a previously recorded CHKPT.

The following information can be used to diagnose problems in the functional subsystem interface (FSI).

FSI PARAMETER LIST (FSIP)

The FSI parameter list (FSIP) is the common parameter list used by all FSI services. It contains a header section and a request section.

The header section which describes the FSI service that is being requested contains:

- The identifier of the FSS/FSA involved in the request
- The return code from the FSI request processing

The request section describes one or more of the FSI services. These requests are:

CONNECT
DISCONNECT
ORDER
GETDS
RELDS
GETREC
FREEREC
CHKPT
POST
SEND

Note: Each of these sections contains information required or supplied by the processing routines. Refer to the **Control Block Section** of this publication.

For related information on FSI control blocks, see MVS Diagnostic Techniques.

FUNCTIONAL SUBSYSTEM VECTOR TABLE (FSVT)

During the address space creation for an FSS address space, an address space control block (ASCB) is created in the system queue area (SQA), and an address space extension block (ASXB) is created in the private area. FSS connect processing sets the ASXB field, ASXBJSVT, to point to the functional subsystem vector table (FSVT). The FSS connect processing creates the FSVT, and therefore, the FSS must be connected first.

The FSVT contains paired pointers to the functional subsystem control tables (FSCTs). These FSCTs contain the addresses of the FSI service routines. The FSVT and the FSCTs reside in the FSS address space private area.

There is one functional subsystem vector table (FSVT) for every FSS address space. The first entry in the FSVT contains the identifier, 'FSVT'. The remaining entries in the FSVT are organized in pairs. Each pair points to two FSCTs. The first pair of entries point to the two FSCTs that support FSIREQ requests that are made on the FSS level. The second pair of entries in the FSVT point to the two FSCTs that support FSIREQ requests that are made on the FSA level. This FSVT/FSCT structure resides in a single FSS address space.

FUNCTIONAL SUBSYSTEM CONTROL TABLE (FSCT)

There are two functional subsystem control tables (FSCTs) for the FSS level services, and two FSCTs for the FSA level services. The first entry of each FSCT contains the identifier, 'FSCT'; the second entry is reserved for use by JES. These four FSCTs reside in a single FSS address space.

FSS level FSCTs: For the pair of FSCTs on the FSS level:

- The first FSCT contains the address of the JES routine that processes requests made by the FSS through the FSIREQ macro instruction with TARGET=JES specified. This JES routine processes the SEND service.
- The second FSCT contains the address of the FSS routine that processes requests made by JES through the FSIREQ macro instruction with TARGET=FSS specified. This FSS routine processes the ORDER service.

FSA level FSCTs: For the pair of FSCTs on the FSA level:

- The first FSCT contains the addresses of the JES routines that process requests made by the FSA through the FSIREQ macro instruction with TARGET=JES specified. These JES routines process the GETDS, GETREC, FREEREC, RELDS, CHKPT, and SEND services. The addresses of these services reside in the FSCT in this order.
- The second FSCT contains the addresses of the FSA routines that process requests made by JES through the FSIREQ macro instruction with TARGET=FSS specified. These FSA routines process the ORDER and POST services. The addresses of these services reside in the FSCT in this order.

JES issues the FSIREQ macro instruction from the FSS address space. See Figure 2 on page FSI-11 for a detailed control block overview of the FSIREQ macro instruction.

FUNCTIONAL SUBSYSTEM RECOVERY

When an error occurs in the functional subsystem, the FSS/FSA recovery routines generally attempt to stop the FSS, depending on whether the error occurred in the FSS address space, the FSA, or the FSI. If the functional subsystem recovers, the FSS can be restarted by issuing JES commands. If the functional subsystem abnormally terminates, JES ensures that the FSI is terminated so that JES cannot access the functional subsystem address space. If necessary, JES forces a disconnect of the functional subsystem. JES also issues a RELDS request for all outstanding data sets and cleans up any resources related to the terminated functional subsystem.

If the JES address space fails, active FSS address spaces continue processing any work already assigned. If there has not been an intervening IPL, JES reestablishes communication with active FSS address spaces when JES is restarted by a hot start. If there has been an intervening IPL on a particular system, JES3 can still reestablish communication with active FSS address spaces, provided that the FSS address space does not reside in the IPLed system. If the JES support in the FSS address space fails and the failure cannot be resolved, JES issues an error message, takes an SVC dump, and terminates the FSS address space.

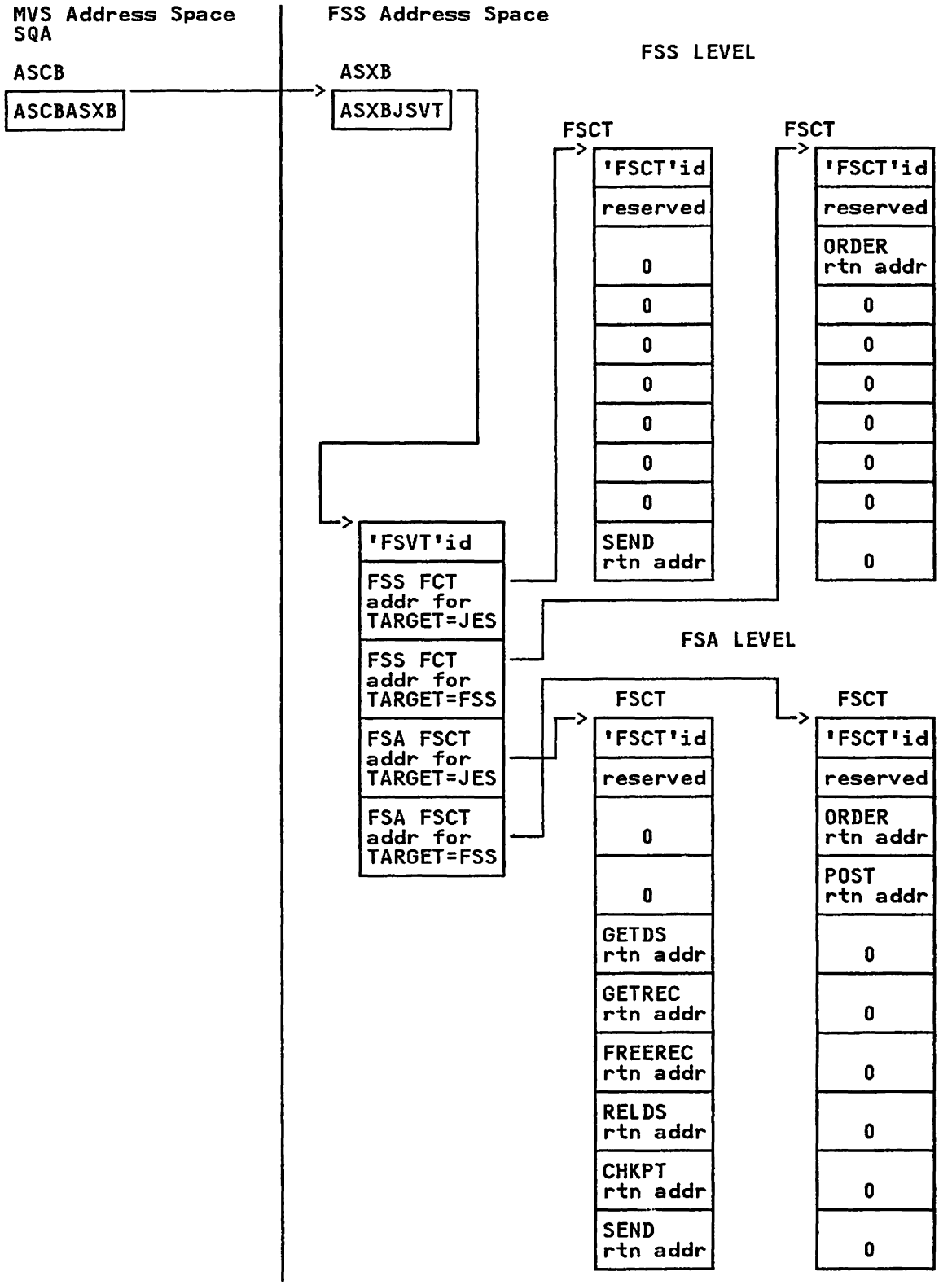

Figure 2. FSIREQ Control Block Structure

METHOD OF OPERATION

There are no Method of Operation diagrams for this component. However, the FSIREQ macro instruction has been included in this section.

FSIREQ - JES FSI REQUEST MACRO INSTRUCTION

The functional subsystem (FSS) requests that the functional subsystem interface (FSI) notifies JES that it, or the functional subsystem application (FSA) is initialized and ready. The FSIREQ macro instruction expands to an SSI call (code 53) for the CONNECT/DISCONNECT services. The FSIREQ macro instruction expands to a branch and link register (BALR) instruction for all other services.

INPUT

The FSIREQ macro instruction is written as follows:

```
FSIREQ
     [REQUEST=requested service]
     [,TARGET=JES or FSS]
     [,PARM=address of the FSI parameter list (FSIP)]
     [,FSID=functional subsystem identifier]
     [,MODE=PC]
```

All of the parameters on the FSIREQ macro instruction are optional. However, if none of the parameters are specified, the address of the FSIP must be provided in register 1, and other information must be provided in the FSIP. The parameters are:

REQUEST The requested service, specified as a register value or as one of the symbolic names listed below. If REQUEST is not specified, one of the equated values for the requested service must previously have been stored in the FSIFUNC field in the FSI parameter list. The REQUEST parameter must be specified, however, for connect and disconnect services. The valid REQUEST values are:

- **FSICON** CONNECT communicates the initiated status of the FSS/FSA to JES and identifies the FSIREQ routines.
- **FSIDCON** DISCONNECT communicates the terminated status of the FSS/FSA to JES.
- **FSIGDS** GETDS gets an existing data set.
- **FSIRDS** RELDS releases a data set.
- **FSIGREC** GETREC gets logical records.
- **FSIFREC** FREEREC frees logical records.
- **FSICKPT** CHKPT writes spool data set checkpoint information.
- **FSISEND** SEND sends a response to JES.
- **FSIORDER** ORDER sends an order to the FSS/FSA.
- **FSIPOST** POST notifies the FSA that JES is ready to accept a GETDS request.

TARGET The subsystem to receive control, either JES or FSS. If TARGET is not specified, JES is the default.

PARM Address of the FSI parameter list (FSIP), which contains the data to be used by the requested service. If PARM is not specified, the address of FSIP must previously have been stored in register 1.

FSID Value that uniquely identifies the FSS/FSA. If FSID is not specified, it must previously have been stored in the FSIFSID field in the FSI parameter list.

MODE PC, if the expanded code runs under a cross memory PC routine. If MODE is not specified, FSIREQ does not run in cross memory mode. This parameter is only valid for JES2.

JES issues only the order and post requests. The FSS/FSA issues all other requests.

OUTPUT

For a connect or a disconnect request, the FSIREQ macro generates a code that invokes an SSI request. For all other requests, FSIREQ invokes the FSI routine whose address is in an FSCT field.

ASSOCIATED CONTROL BLOCKS

Common name	Macro name	Description
ASCB	IHAASCB	Address space control block
ASVT	IHAASVT	Address space vector table
ASXB	IHAASXB	Address space extension block
CHK	IAZCHK	JES checkpoint record area
CVT	CVT	Communications vector table
FSCT	IAZFSCT	Functional subsystem control table
FSIP	IAZFSIP	FSI parameter list
FSVT	IAZFSVT	Functional subsystem vector table
IDX	IAZIDX	Index table
JESCT	IEFJESCT	JES communication table
JSPA	IAZJSPA	JES job separator page data area
PSA	IHAPSA	Prefixed save area
RESPA	IAZRESPA	Response area
SSIB	IEFJSSIB	Subsystem identification block
SSOB	IEFJSSOB	Subsystem options block

For connect and disconnect requests, the caller must have access to CVT, the FSIP, the JESCT, the SSIB, and the SSOB.

Note: The caller must also have obtained storage for the SSIB and the SSOB prior to issuing the FSIREQ macro instruction.

For all other requests, the caller must have access to the ASCB, the ASXB, the FSCT, the FSVT, and the FSIP. If PC mode is specified, the caller must have access to the ASVT; if PC mode is not specified, the caller must have access to the PSA.

INDEX

A

C

D

F

MVS/Extended Architecture
System Logic Library:
Functional Subsystem
Interface

READER'S
COMMENT
FORM

LY28-1690-0

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.

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.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

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.)

MVS/Extended Architecture System Logic Library: Functional Subsystem Interface

LY28-1690-0 S370-36

Reader's Comment Form

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 PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department D58, Building 921-2
PO Box 390
Poughkeepsie, New York 12602

Fold and tape Please Do Not Staple Fold and tape

Printed in U.S.A.

IBM®

LY28-1690-00

LY28-1690-0 S370-36

Printed in U.S.A.