
FSPLI Programmers Reference

September, 1988



Sirius Software, Inc.
875 Massachusetts Avenue, Suite 21
Cambridge, MA 02139

Telephone: (617) 876-6677
FAX: (617) 234-1200
E-mail: support@sirius-software.com
World Wide Web: <http://sirius-software.com>

August 5, 2010

© 2010 Sirius Software, Inc.

Proprietary Notices

The following products:

- *FSPLI*

is a proprietary product of Sirius Software, Inc.:

Sirius Software, Inc.
875 Massachusetts Avenue, Suite 21
Cambridge, Massachusetts 02139
USA

Model 204® is a proprietary product of Computer Corporation of America, a wholly-owned subsidiary of Rocket Software, Inc., which owns the trademark:

Rocket Software Corporate Office
M204 Division
275 Grove Street
Suite 3-410
Newton, Massachusetts 02466-2272
USA

Contents

Proprietary Notices	ii
Contents	iii
Abstract	v
Chapter 1: The FSPLI Communications Model	1
Model 204 3270 Data Streams	1
Using FSPLI with Structured Data Streams	2
Chapter 2: FSPLI Application Program Design	3
Without Session Management	3
With Session Management	4
Chapter 3: FSPLI Subroutines	5
FSALT — See if Screen is in Alternate Mode	5
FSATTN — Send Out-of-Band Attention Interrupt	6
FSBIND — Reconnect Unbound Full Screen Connection	7
FSBPTR — Return Pointer to Local Connection Buffer	8
FSBSIZE — Return Connection Block Size	9
FSCBLEN — Return Required Size of FSCB	10
FSCLOSE — Close Full Screen Connection	10
FSCOPYI — Copy Data To Local Buffer	11
FSCOPYO — Copy Data From Local Buffer	12
FSEOP — See if Connection is in End of Page Pause Mode	13
FSERW — See if Data Stream Starts with Erase	13
FSFILEO — Open a Model 204 File or Group	14
FSFREEM — Free Non-PL/I Storage	15
FSGETM — Allocate Non-PL/I Storage	16
FSINIT & FSINITI — Initialize Full Screen Connection	16
FSINPUT — Pass Single Line of Input to Model 204	18
FSLOGIN — Login to Model 204	19
FSLOGON — Logon to Model 204 with Account	20
FSOPEN — Open New Full Screen Connection	22
FSREAD — Read Data Stream from Model 204	23
FSRESHO — Send a "Reshow" Input Stream to Model 204	24
FSSEQ — See if Connection is in Sequential Read Mode	25
FSUBIND — Unbind Full Screen Connection	26
FSVIS — See if Connection is Sequential and Visible	27

FSWRITE — Write Data Stream to Model 204	27
Appendix A: Fast/CRAM Session Management	29
Fast/CRAM UNBIND Function	29
Fast/CRAM BIND Function	29
Fast/CRAM BREAK Function	29
Appendix B: IMSIT use of FSBIND and FSUBIND	31

Abstract

FSPLI is a package of subroutines that provide PL/I programs with a high-level Application Programming Interface for accessing *Model 204* via full-screen (IODEV=11) connections. Using the facilities of *FSPLI* a PL/I program may maintain an arbitrary number of connections to *Model 204*. These connections support raw 3270 data streams as well as structured data streams used to exchange information between *Model 204* dollar functions and remote applications. *FSPLI* dramatically simplifies the programming for this type of access by shielding the programmer from details of the 3270 data streams used by *Model 204*.

When used with *Fast/CRAM*, *FSPLI* supports session management that enables a single logical session with *Model 204* to be transferred amongst different address spaces. This makes *FSPLI* ideal for supporting access from IMS applications to *Model 204*, since the load balancing in IMS can result in terminal interactions moving between the various message processing address spaces. The Sirius IMS/DC interface makes use of *FSPLI*.

While *FSPLI* was originally designed to support applications coded in PL/I, it is also callable from BAL (IBM assembly language) and COBOL. The FSOPEN and FSBIND subroutines accept a parameter that specifies the programming environment to be used with the connection. The language setting for the current connection is then used by subsequent *FSPLI* subroutine calls for the connection. Note that the language code can be changed by issuing an FSUBIND/FSBIND sequence.

FSPLI connects to a *Model 204* **ONLINE** as if it were a 3270 terminal. As a result, *FSPLI* deals with the orderly exchange of 3270 data streams. In the normal course of activity, *Model 204* writes a data stream to the "terminal" and gets back a *read modified* response stream.

1.1 Model 204 3270 Data Streams

Model 204 produces as output 3270 data streams which have the following format:

- Command code.
This will be either WRITE, ERASE/WRITE or ERASE/WRITE/ALTERNATE. The WRITE command is used only when a *Model 204* thread is operating in *line at a time* mode (that is, not READ SCREEN and not the editor) and the current line being written is not the first line on the screen. The ERASE/WRITE commands are used for the first line of a screen in line at a time mode and for all READ SCREEN and full screen editor interactions.
- Write Control Character (WCC).
- Data, starting with SBA and not including any link control or framing characters.

Model 204 expects all input 3270 data streams to have the following *READ MODIFIED* format:

- Attention Identifier byte (AID). Indicates what ending key was hit (ENTER, PF_n, CLEAR, etc.).
- Current cursor location, a pair of printable characters which represent a 3270 buffer address.
- Data, if present, generally starting with an SBA-loc-loc sequence which identifies the location of the first modified field.

Three input streams have special meaning to *Model 204*. An AID of PA1 indicates an attention request and results in the cancellation of the current *Model 204* request, if any. An AID of PA2, or CLEAR, indicates a request to re-paint the 3270 screen and causes an ERASE/WRITE or ERASE/WRITE/ALTERNATE (as appropriate) with all of the data required to repaint the entire screen. This is useful when the screen has been altered, perhaps by an asynchronous message.

FSPLI strictly supports a half duplex, flip-flop protocol. *Model 204* waits for an initial data stream from the *FSPLI* application, then writes a reply and waits again for an input stream. Flow control information is provided to resynchronize programs which violate this protocol.

By convention, the first data stream which an *FSPLI* application sends identifies the model number of the 3270 terminal supported for that connection. *Model 204* uses this model number to determine the physical characteristics of the terminal, as shown in the following table:

3270 Model Number Definitions

This table presents the valid 3270 models supported by *Model 204* and the physical characteristics assumed for each model.

2	24 lines of 80 columns in primary mode, no alternate mode defined
3	24 lines of 80 columns in primary mode, 32 lines of 80 columns in alternate mode
4	24 lines of 80 columns in primary mode, 43 lines of 80 columns in alternate mode
5	24 lines of 80 columns in primary mode, 27 lines of 132 columns in alternate mode

1.2 Using FSPLI with Structured Data Streams

Once *FSPLI* has established a connection with *Model 204*, and logged in via *FSLOGIN* or *FSLOGON*, the *FSINPUT* routine can be used to invoke an *APSY* subsystem. The User Language programs in the subsystem can use straightforward dollar functions to exchange images of data with the *FSPLI* application. This approach bypasses the limitations of the *WRITE IMAGE ON TERMINAL* and *READ IMAGE FROM TERMINAL* statements.

FSPLI Application Program Design

FSPLI supports two different application models. The simplest model uses a static connection between the *FSPLI* application and *Model 204* and does not require *Fast/CRAM*. If *Fast/CRAM* is present, then a more sophisticated model with session management is supported.

2.1 Without Session Management

The following logic may be used without *Fast/CRAM* to construct a simple *FSPLI* application:

1. Allocate FSCB (FSGETM with FSBLLEN).
2. Open connection (FSOPEN).
3. Allocate buffer (FSGETM with FSBSIZE).
4. Set model and associate buffer with connection (FSINIT or FSINITI).
5. Get data stream from *Model 204* (FSREAD).
6. Copy data from connection buffer (FSCOPYO). If protocol error (*Model 204* expecting data), repaint the screen with a call to FSRESHO and repeat. Perform one of the following, as appropriate:
 - Process structured data sent from User Language dollar function. Prepare structured reply for User Language dollar function.
 - Output 3270 data stream on terminal, read response.
7. Send response (structured data or 3270 READ MODIFIED stream) back to *Model 204* (FSCOPYI and FSWRITE). Ignore any protocol errors.
8. Go to step 5, above.
9. Terminate connection (FSCLOSE)
10. Free storage for session (FSFREEM with FSBPTR, then FSFREEM for FSCB). This may be the result of *Model 204* termination, or it may be in response to a special data stream entered at the terminal or sent by *Model 204*.

2.2 With Session Management

The following logic may be used when *Fast/CRAM* is installed to support conversational applications with IMS-style MPP load balancing:

1. Allocate FSCB (FSGETM with FSBLLEN).
2. Bind Session (FSBIND). Using a session key formed from the end-user's terminal ID. Remember if session was re-bound, or if new session was started.
3. Allocate buffer (FSGETM with FSBSIZE).
4. Set model and associate buffer with connection (FSINIT or FSINITI).
5. If new session was started (first interaction of session), login to *Model 204* via FSLOGIN or FSLOGON, open any required files and send the line of input to fire up the application using FSINPUT. Otherwise, process reply message from end user terminal, using one of the following as appropriate:
 - Process response from end-user's terminal, update structured data.
 - Read response from end-user.
6. Send response *Model 204* (FSCOPYI and FSWRITE). This will be either structured data or a 3270 READ MODIFIED stream. Ignore any protocol errors. *Model 204* should be waiting for input from the *FSPLI* application.
7. Get reply data stream from *Model 204* (FSREAD and FSCOPYO). If protocol error, re-display last screen with a call to FSRESHO and repeat.
8. If processing structured data, prepare output for the end-user's terminal, otherwise we already have a 3270 data stream.
9. Queue a message to the end-user's terminal.
10. Close or unbind session with *Model 204* If this is the last message from the application, use FSCLOSE to close the session with *Model 204*. Otherwise, use FSUBIND to unbind from the *Model 204* session using the end-user's terminal ID as the session key (optionally saving a fullword of state data).
11. Use FSBPTR and FSFREEM to free the data buffer, and then FSFREEM for the FSCB.
12. Exit to IMS MPP controller. Control will get returned at step 1, above when the next message from the end-user is received.

CHAPTER 3 ***FSPLI Subroutines***

This section describes the various subroutines that comprise the *FSPLI* API. Included with each routine is a description of its function, its arguments, and its return codes. These routines are all described using PL/I syntax, although they may also be called by COBOL or assembly language programs, depending upon the language code provided to **FSOPEN** or **FSBIND**.

3.1 FSALT — See if Screen is in Alternate Mode

FSALT determines whether the last erase-write from *Model 204* was an Erase Write Alternate (EWA) or not.

```
FSALT  PROCEDURE ( FSCB_PTR )
        RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR;      /* FSCB for connection */
```

PL/I FSALT Routine Declaration

FSALT returns a fullword integer value as follows:

- **1 (Connection in Alternate Mode)**

The most recent Erase Write was an Erase Write Alternate.

- **0 (Connection Not in Alternate Mode)**

It is also possible that the connection is not active.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

3.2 FSATTN — Send Out-of-Band Attention Interrupt

FSATTN signals an out-of-band attention over the indicated FSPLI connection. This causes any outstanding *Model 204* request for the connection to be cancelled and *Model 204* will respond with a new output data stream, so that FSREAD should be called next.

Note: An in-band attention is normally signalled by sending a PA1 data stream to *Model 204*, using FSWRITE. FSATTN, however, may be called even when *Model 204* has output for the connection. It is not recommended that FSATTN be used to reestablish synchronization in the case of protocol violations.

```

FSATTN PROCEDURE ( FSCB_PTR )
    RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR;      /* FSCB for connection */
    
```

PL/I FSATTN Routine Declaration

FSATTN returns a fullword integer value as follows:

- **0 (Attention Sent)**

An attention interrupt has been reflected to *Model 204*. *Model 204* will respond with another 3270 data stream, so FSREAD should be called next.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

3.3 FSBIND — Reconnect Unbound Full Screen Connection

The full screen BIND routine, FSBIND, resumes a connection between a FSPLI program and the indicated copy of *Model 204* that was previously unbound. A sixteen character *bind key* is used to identify the previously established session. If the requested session cannot be located, then a new session is created.

```

FSBIND: PROCEDURE ( FSCB_PTR, CHANNEL_NAME, LANG_CODE,
                   KEY ) RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR,      /* FSCB for connection */
    CHANNEL_NAME CHAR(8), /* identifies M204 copy */
    LANG_CODE     FIXED BIN(31), /* language of caller:
                                0 -> PL/I
                                1 -> COBOL
                                2 -> BAL (assembly) */
    KEY CHAR(16);        /* session key */

```

PL/I FSBIND Routine Declaration

All of the information pertaining to this connection is maintained in a FSPLI Connection Block (FSCB), which is passed as a pointer argument. The function FSBLLEN may be used to determine the required length for each FSCB.

Note: The FSCB is updated asynchronously by CRAM routines. Therefore, the FSCB **must** reside in virtual storage for the duration of the FSPLI connection. It must not be freed or moved until FSCLOSE has been called to terminate the FSPLI connection.

This implies that neither PL/I automatic storage nor PL/I heap storage, that is, that controlled by the ALLOCATE statement, may be used for an FSCB. This is because PL/I storage is automatically freed when a PL/I program terminates, and an application may terminate without calling FSCLOSE for all of its FSPLI connections. Further, an FSCB should not reside in a dynamic area which may move in storage, such as an IMS SPA. The routines FSGETM and FSFREEM are provided to solve these memory management problems.

The desired copy of *Model 204* is identified by its eight-character CRAM channel name, that is, that value specified for the *Model 204* parameter CRFSCHNL. The default for this *Model 204* parameter is "M204FULL". If the indicated *Model 204* is not active, the connection is refused with an appropriate error code.

The session key identifies a previously unbound session that is to be reestablished. If an unbound session with this key cannot be found for the indicated copy of *Model 204*, the call is processed as an FSOPEN call.

The return code from FSBIND is set as follows:

- **32 (Bind Successful)**

The unbound session identified by the session key was re-bound.

- **0 (New Connection Established)**

An unbound session with the identified key could not be found. A new session was opened instead. The application must allocate a buffer with length returned from FSBSIZE and associate it with the FSCB using FSINIT or FSINITI.

- **-28 (No Connections)**

Although the indicated *Model 204* IODEV=11 channel is active, an unbound session matching the session key could not be found and there were no connections currently available for establishing a new session. An open request may be successful at a later time.

- **-32 (Invalid Channel ID)**

No *Model 204* with the indicated IODEV=11 channel can be found. The bind request is rejected, but an open request may be successful later once the requested *Model 204* is up.

- **-36 (Insufficient CSA)**

Although the indicated *Model 204* IODEV=11 channel is active, an unbound session matching the session key could not be found and there is insufficient CSA space for establishing a new session. An open request may be successful at a later time. The open request is rejected, but may be successful at a later time.

3.4 FSBPTR — Return Pointer to Local Connection Buffer

FSBPTR returns a pointer to the buffer previously associated with the indicated FSPLI connection by a call to FSINIT or FSINITI. The connection is identified by a pointer to its FSCB. This pointer can then be used with FSFREEM to free the buffer.

Note: FSBPTR can be called after an FSPLI connection has been closed by a call to FSCLOSE, as long as the FSCB for the connection has not yet been freed.

```
FSBPTR: PROCEDURE ( FSCB_PTR )  
          RETURNS ( PTR );  
  
DCL FSCB_PTR PTR;          /* identifies connection */
```

PL/I FSBPTR Routine Declaration

FSBPTR returns a pointer to the local data buffer for the connection, or NULL if the FSCB pointer is invalid or there is no local data buffer associated with the indicated FSPLI connection.

3.5 FSBSIZE — Return Connection Block Size

FSBSIZE returns the block size, in bytes, for an FSPLI connection previously established by FSOPEN. The connection is identified by a pointer to its FSCB. The block size may then be used with FSGETM to allocate a local data buffer for the connection. This buffer must then be associated with the connection by a call to FSINIT or FSINITI.

```
FSBSIZE: PROCEDURE ( FSCB_PTR )  
          RETURNS ( FIXED BIN(31) );  
  
DCL FSCB_PTR PTR;          /* identifies connection */
```

PL/I FSBSIZE Routine Declaration

If the value returned by FSBSIZE is positive, it is the block size for the connection. A negative value is an error return, as follows:

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

- **-8 (Connection Not Open)**

The indicated connection was never successfully opened, the call is ignored.

3.6 FSCBLEN — Return Required Size of FSCB

FSCBLEN returns the required size of an FSCB, in bytes. This function may be called at any time and serves to insulate an FSPLI application from the details of the FSPLI interface routine implementation.

```
FSCBLEN: PROCEDURE ( )  
          RETURNS ( FIXED BIN(31) );
```

PL/I FSCBLEN Routine Declaration

The value returned may be used with FSGETM to allocate an FSCB for a FSPLI connection.

3.7 FSCLOSE — Close Full Screen Connection

The full screen close routine, FSCLOSE, removes an FSPLI connection previously established by FSOPEN or FSBIND. The connection is identified by a pointer to its FSCB. This routine or FSUBIND **must** be called prior to freeing the FSCB or the buffer established by FSINIT, if any.

```
FSCLOSE: PROCEDURE ( FSCB_PTR )  
          RETURNS ( FIXED BIN(31) );  
  
DCL FSCB_PTR      PTR;      /* FSCB for connection */
```

PL/I FSCLOSE Routine Declaration

The return code from FSCLOSE is set as follows:

- **0 (Connection Closed)**

The indicated connection has been closed successfully.

- **-4 (FSCB Pointer Invalid)**

The indicated FSCB is incorrect.

- **-8 (Connection Not Open)**

This call was a noop, since the indicated connection was never successfully opened.

3.8 FSCOPYI — Copy Data To Local Buffer

FSCOPYI copies a 3270 data stream from a data area provided by the caller to the local data buffer for a connection.

```

FSCOPYI: PROCEDURE ( FSCB_PTR, DATA_PTR, DATA_LEN )
           RETURNS ( FIXED BIN(31) );

DCL  FSCB_PTR  PTR,          /* FSCB for connection */
     DATA_PTR PTR,          /* source data area */
     DATA_LEN FIXED BIN(31); /* length of data to
                               to copy */

```

PL/I FSCOPYI Routine Declaration

FSCOPYI returns a fullword integer value as follows:

- **0 (Copy Successful)**

The data from the caller's area has been copied into the local data buffer for the connection.

- **-4 (Invalid FSCB Pointer)**

The first argument does not point to a valid FSCB, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-24 (Data Stream too Long)**

The data stream in the caller's data area is too long for the connection's local data buffer. The call has been ignored. The routine FSBSIZE can be used to ensure that the caller's data stream is not too long for the connection.

3.9 FSCOPYO — Copy Data From Local Buffer

FSCOPYO copies a 3270 data stream from the local data buffer for a connection into a data area provided by the caller. This may be used to copy a data stream into PL/I heap storage, for example.

Note: This may be the last stream sent to *Model 204* or the last stream received from *Model 204*, depending upon the state of the connection.

```

FSCOPYO: PROCEDURE ( FSCB_PTR, DATA_PTR, DATA_LEN )
           RETURNS ( FIXED BIN(31) );

DCL  FSCB_PTR  PTR,           /* FSCB for connection */
     DATA_PTR PTR,           /* target data area */
     DATA_LEN FIXED BIN(31); /* length of data area
                               on input, length of data
                               stream upon return */

```

PL/I FSCOPYO Routine Declaration

The FSCOPYO call returns a fullword integer value as follows:

- **0 (Copy Successful)**

The current data stream in the local data buffer has been copied to the caller's data area and its length returned to the third argument.

- **-4 (Invalid FSCB Pointer)**

The first argument does not point to a valid FSCB, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-24 (Data Stream too Long)**

The local data buffer for the connection holds more data than can be fit in the caller's data area. The call has been ignored. The routine FSBSIZE can be used to ensure that the caller's data area is large enough to hold the maximum data stream for the connection.

3.10 FSEOP — See if Connection is in End of Page Pause Mode

FSEOP examines the local data buffer to see if it contains a data stream to format a 3270 for pausing at end of page.

```
FSEOP  PROCEDURE ( FSCB_PTR )
        RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR;      /* FSCB for connection */
```

PL/I FSEOP Routine Declaration

FSEOP returns a fullword integer value as follows:

- **1 (Connection at EOP Pause Mode)**

The data buffer contains a stream to format a 3270 for pausing at the end of page.

- **0 (Connection not in EOP Pause Mode)**

It is also possible that the connection is not active.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

3.11 FSERW — See if Data Stream Starts with Erase

FSERW examines the local data buffer to see if it contains a data stream that begins with an erase-write (EW) or erase-write-alternate (EWA) stream. If not, FSRESHO can always be used to generate a stream that completely re-paints the screen.

```
FSERW  PROCEDURE ( FSCB_PTR )
        RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR;      /* FSCB for connection */
```

PL/I FSERW Routine Declaration

FSERW returns a fullword integer value as follows:

- **1 (Data Stream Begins with Erase)**

The data buffer contains a stream that re-formats a 3270.

- **0 (Data Stream does not Begin with Erase)**

It is also possible that the connection is not active.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

3.12 FSFILEO — Open a Model 204 File or Group

FSFILEO sends the sequence to open a file or group over a FSPLI connection, provided that the connection is the state of requesting a sequential lines of input (that is, not in full screen mode and not sending output to the FSPLI program). If *Model 204* requests a password, then the provided password is sent. FSFILEO shields the programmer from the details of the 3270 data streams used by *Model 204*. The FSSEQ and FSEOP functions may be used to ensure that the connection is in an appropriate state for this call.

```
FSFILEO PROCEDURE ( FSCB_PTR, FNAME_PTR, PWD_PTR )
                RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR    PTR,      /* FSCB for connection */
     FNAME_PTR  PTR,      /* pointer to blank padded
                          8-character file/group name */
     PWD_PTR    PTR;      /* pointer to blank padded
                          10-character password */
```

PL/I FSFILEO Routine Declaration

FSFILEO returns a fullword integer value as follows:

- **0 (File/Group Opened)**

The indicated file or group is *probably* open and the resulting output data stream is in the local data buffer for the connection. The check for successful open is weak. FSREAD does not need to be called.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-52 (Incorrect Session State)**

The connection to *Model 204* is not in the "waiting for sequential input" state. This could be the result of a READ SCREEN statement in *Model 204* or the write of a structured datastream. The FSSEQ function can be used to avoid this error.

- **-56 (Open Failed)**

The open request was not followed by *Model 204* reading a sequential line of input. The resulting data stream is in the local buffer.

- **-60 (End of Page Pause)**

The connection to *Model 204* is in an "end of page pause" state, this call ignored. Call FSATTN and then try again. The FSEOP function can be used to avoid this error.

3.13 FSFREEM — Free Non-PL/I Storage

FSFREEM frees storage previously allocated by FSGETM.

```
FSFREEM: PROCEDURE ( STRG_PTR )
           RETURNS ( FIXED BIN(31) );

DCL STRG_PTR PTR; /* area to free, returned by a
                  previous call to FSGETM */
```

PL/I FSFREEM Routine Declaration

FSFREEM returns a fullword integer value as follows:

- **0 (Storage Freed)**

The indicated storage has been freed.

- **-44 (Free Request Failed)**

The free request could not be completed, probably because the pointer was invalid or the area had already been freed. This call is ignored.

3.14 **FSGETM — Allocate Non-PL/I Storage**

FSGETM allocates storage which is not associated with the PL/I environment. The storage is allocated in subpool 99 by a conditional GETMAIN. This routine is intended for allocating the FSCB and connection buffer, using lengths returned by FSCBLEN and FSBSIZE, respectively.

```
FSGETM: PROCEDURE ( LENGTH )  
          RETURNS ( PTR );  
  
DCL LENGTH    FIXED BIN(31); /* length of storage  
                               requested */
```

PL/I FSGETM Routine Declaration

FSGETM returns NULL if the requested storage is unavailable, or if the requested amount is invalid, otherwise it returns a pointer to the start of the allocated storage.

Note: Storage allocated by FSGETM should be freed by FSFREEM.

3.15 **FSINIT & FSINITI — Initialize Full Screen Connection**

FSINIT and FSINITI initialize an FSPLI connection previously established by FSOPEN or reestablished by FSBIND. The connection is identified by a pointer to its FSCB. The indicated physical buffer is associated with the connection and the specified 3270 model number is communicated to *Model 204*. FSINITI allows a terminal ID to be passed to *Model 204* so that monitoring routines will correctly identify the end user. This routine must be called prior to the communication routines FSREAD, FSWRITE, and FSATTN.

```

FSINIT: PROCEDURE ( FSCB_PTR, BUFR_PTR, MODEL, EXTENDED )
          RETURNS ( FIXED BIN(31) );

FSINITI: PROCEDURE ( FSCB_PTR, BUFR_PTR, MODEL, EXTENDED,
                    TERMID ) RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR  PTR,          /* identifies connection */
     BUFR_PTR  PTR,          /* buffer for connection */
     MODEL    FIXED BIN(31), /* 3270 model number */
     EXTENDED  FIXED BIN(31), /* non-zero if extended
                               data streams supported */
     TERMID   CHAR(8);      /* terminal ID of end user */

```

PL/I FSINIT & FSINITI Routine Declarations

The return code from FSINIT and FSINITI is as follows:

- **0 (Connection Initialized)**

The indicated connection has been initialized, *Model 204* has prepared a formatting data stream, so FSREAD should be called next.

- **Return Code -4 (FSCB Pointer Invalid)**

The FSCB pointer is invalid. The call is rejected.

- **-8 (Connection Not Open)**

The indicated connection was not successfully opened or re-bound. The call is ignored.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated after the FSOPEN or FSBIND call. This call is rejected.

- **-20 (Protocol Error)**

This should not happen and is an internal error in FSPLI.

- **-40 (FSCB Already Initialized)**

The indicated FSCB has already been initialized by a call to FSINIT or FSINITI. Current call is ignored.

3.16 **FSINPUT — Pass Single Line of Input to Model 204**

FSINPUT passes a single line of input to *Model 204* over a FSPLI connection, provided that the connection is in the state of requesting a sequential line of input (that is, not in full screen mode and not sending output to the FSPLI program). FSINPUT shields the programmer from the details of the 3270 data streams used by *Model 204*.

The FSSEQ function may be used to ensure that the connection is in an appropriate state for this call. If *Model 204* is currently holding output for the connection, indicating a protocol violation, the FSPLI application should ignore its input stream and call FSREAD to receive the 3270 data stream from *Model 204*.

```

FSINPUT PROCEDURE ( FSCB_PTR, INPUT_PTR, INPUT_LEN )
                RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR,      /* FSCB for connection */
     INPUT_PTR    PTR,      /* pointer to line of input
                           to send to Model 204 */
     INPUT_LEN    FIXED BIN(31); /* length of line */

```

PL/I FSINPUT Routine Declaration

FSINPUT returns a fullword integer value as follows:

- **0 (Line Sent)**

The line of input has been sent to *Model 204* as a 3270 data stream over the FSPLI connection. *Model 204* will respond with another 3270 data stream, so FSREAD should be called next.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-24 (Length of Input Line Invalid)**

Either a negative length was passed for the line or the length exceeds 77 characters. The call is ignored.

- **-52 (Incorrect Session State)**

The connection to *Model 204* is not in the "waiting for sequential input" state. This could be the result of a READ SCREEN statement in *Model 204* or the write of a structured datastream. The FSSEQ function can be used to avoid this error.

3.17 FSLOGIN — Login to Model 204

FSLOGIN performs a login sequence over a FSPLI connection, provided that the connection is the state of requesting a sequential lines of input (that is, not in full screen mode and note sending output to the FSPLI program). FSLOGIN shields the programmer from the details of the 3270 data streams used by *Model 204*. The FSSEQ function may be used to ensure that the connection is in an appropriate state for this call.

```

FSLOGIN PROCEDURE ( FSCB_PTR, ID_PTR, PWD_PTR )
                RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR,      /* FSCB for connection */
     ID_PTR        PTR,      /* pointer to blank padded
                             12 character login ID */
     PWD_PTR       PTR;      /* pointer to blank padded
                             10 character password */

```

PL/I FSLOGIN Routine Declaration

FSLOGIN returns a fullword integer value as follows:

- **0 (User Logged In)**

The user is logged in to *Model 204* with the given ID and the resulting output data stream is in the local data buffer for the connection. FSREAD does not need to be called.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-52 (Incorrect Session State)**

The connection to *Model 204* is not in the "waiting for sequential input" state. This could be the result of a READ SCREEN statement in *Model 204* or the write of a structured datastream. The FSSEQ function can be used to avoid this error.

- **-56 (Login Failed or no Password Requested)**

Either the login request failed, or *Model 204* did not request a password. The resulting data stream is in the local buffer.

- **-60 (End of Page Pause)**

The connection to *Model 204* is in an "end of page pause" state, this call ignored. Call FSATTN and then try again. The FSEOP function can be used to avoid this error.

3.18 FSLOGON — Logon to Model 204 with Account

FSLOGON performs a logon sequence with ID and account number over a FSPLI connection, provided that the connection is the state of requesting a sequential lines of input (that is, not in full screen mode and not sending output to the FSPLI program). FSLOGON shields the programmer from the details of the 3270 data streams used by *Model 204*. The FSSEQ function may be used to ensure that the connection is in an appropriate state for this call.

```
FSLOGON PROCEDURE ( FSCB_PTR, ID_PTR, PWD_PTR
                   ACCT_PTR ) RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR,      /* FSCB for connection */
     ID_PTR       PTR,      /* pointer to blank padded
                           12 character login ID */
     PWD_PTR      PTR,      /* pointer to blank padded
                           10 character password */
     ACCT_PTR     PTR;      /* pointer to blank padded
                           8 character account */
```

PL/I FSLOGON Routine Declaration

FSLOGON returns a fullword integer value as follows:

- **0 (User Logged In)**

The user is logged in to *Model 204* with the given ID and the resulting output data stream is in the local data buffer for the connection. FSREAD does not need to be called.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-52 (Incorrect Session State)**

The connection to *Model 204* is not in the "waiting for sequential input" state. This could be the result of a READ SCREEN statement in *Model 204* or the write of a structured datastream. The FSSEQ function can be used to avoid this error.

- **-56 (Login Failed or no Password Requested)**

Either the login request failed, or *Model 204* did not request a password. The resulting data stream is in the local buffer.

- **-60 (End of Page Pause)**

The connection to *Model 204* is in an "end of page pause" state, this call ignored. Call FSATTN and then try again. The FSEOP function can be used to avoid this error.

3.19 FSOPEN — Open New Full Screen Connection

The full screen open routine, FSOPEN, creates a new connection between a FSPLI program and the indicated copy of *Model 204*.

```

FSOPEN: PROCEDURE ( FSCB_PTR, CHANNEL_NAME, LANG_CODE )
           RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR,      /* FSCB for connection */
    CHANNEL_NAME  CHAR(8), /* identifies Model 204 */
    LANG_CODE     FIXED BIN(31); /* language of caller:
                                0 -> PL/I
                                1 -> COBOL
                                2 -> BAL (assembly) */

```

PL/I FSOPEN Routine Declaration

All of the information pertaining to this connection is maintained in a FSPLI Connection Block (FSCB), which is passed as a pointer argument. The function FSBLLEN may be used to determine the required length for each FSCB.

Note: The FSCB is updated asynchronously by CRAM routines. Therefore, the FSCB **must** reside in virtual storage for the duration of the FSPLI connection. It must not be freed or moved until FSCLOSE has been called to terminate the FSPLI connection.

This implies that neither PL/I automatic storage or PL/I heap storage, that is, that controlled by the ALLOCATE statement, may be used for an FSCB. This is because PL/I storage is automatically freed when a PL/I program terminates, and an application may terminate without calling FSCLOSE for all of its FSPLI connections. Further, an FSCB should not reside in a dynamic area which may move in storage, such as an IMS SPA. The routines FSGETM and FSFREEEM are provided to solve these memory management problems.

The desired copy of *Model 204* is identified by its eight character CRAM channel name, that is, that value specified for the *Model 204* parameter CRFSCHNL. The default for this *Model 204* parameter is "M204FULL". If the indicated *Model 204* is not active, the connection is refused with an appropriate error code. If the indicated *Model 204* is active, but no connections are available, a separate error code is returned.

FSOPEN returns an integer value as follows:

- **0 (Connection Established)**

A connection has been established. The application must allocate a buffer with length returned from FSBSIZE and associate it with the FSCB using FSINIT or FSINITI.

- **-28 (No Connections)**

Although the indicated *Model 204* IODEV=11 channel is active, no connections are currently available. The open request is rejected, but may be successful at a later time.

- **-32 (Invalid Channel ID)**

No *Model 204* with the indicated IODEV=11 channel can be found. The open request is rejected, but may be successful later once the requested *Model 204* is up.

- **-36 (Insufficient CSA)**

This code indicates insufficient CSA space was available to complete the CRAM open for this connection. The open request is rejected, but may be successful at a later time.

3.20 FSREAD — Read Data Stream from Model 204

FSREAD receives a 3270 data stream for the indicated FSPLI connection into the local data buffer associated with the connection by FSINIT or FSINITI. If *Model 204* is expecting input from the connection, indicating a protocol violation, the FSPLI application should send a PA2 data stream and then reissue the FSREAD call. This will cause *Model 204* to retransmit the entire contents of the current screen image for the connection.

```

FSREAD: PROCEDURE ( FSCB_PTR )
          RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR;      /* FSCB for connection */
    
```

PL/I FSREAD Routine Declaration

FSREAD returns a fullword integer value as follows:

- **0 (Read Successful)**

The read has completed, the local data buffer has been updated with a 3270 data stream from *Model 204*. *Model 204* will wait for the 3270 read-modified response data stream, so FSWRITE should be called next.

- **-4 (Invalid FSCB Pointer)**

The first argument does not point to a valid FSCB, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-20 (Protocol Error)**

Model 204 is expecting input from the connection, FSWRITE should be called instead of FSREAD. The call has been ignored.

3.21 **FSRESHO — Send a "Reshow" Input Stream to Model 204**

FSRESHO sends the screen clear sequence over a FSPLI connection. The FSSEQ and FSEOP functions may be used to ensure that the connection is in an appropriate state for this call.

```
FSRESHO PROCEDURE ( FSCB_PTR )  
          RETURNS ( FIXED BIN(31) );  
  
DCL FSCB_PTR      PTR;      /* FSCB for connection */
```

PL/I FSRESHO Routine Declaration

FSRESHO returns a fullword integer value as follows:

- **0 (Reshow Transmitted)**

The screen clear sequence has been sent and *Model 204* should respond with an erase-write or erase-write-alternate data stream to re-paint the screen. Call FSREAD next.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-20 (Protocol Error)**

Model 204 is trying to send output to the session. The call has been ignored. Either call FSREAD and then FSRESHO, or use FSATTN and then FSRESHO.

3.22 FSSEQ — See if Connection is in Sequential Read Mode

FSSEQ examines the local data buffer to see if it contains a data stream to format a 3270 for reading a single line of input.

```
FSSEQ  PROCEDURE ( FSCB_PTR )  
        RETURNS ( FIXED BIN(31) );  
  
DCL FSCB_PTR      PTR;      /* FSCB for connection */
```

PL/I FSSEQ Routine Declaration

FSSEQ returns a fullword integer value as follows:

- **1 (Connection in Sequential Mode)**

The data buffer contains a stream to format a 3270 for a single line of input.

- **0 (Connection not in Sequential Mode)**

None of the sequential input routines (FSINPUT, FSLOGIN, FSLOGON or FSLOGO) may be called.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

3.23 FSUBIND — Unbind Full Screen Connection

The full screen unbind routine, FSUBIND, uses a special *Fast/CRAM* function to associate a sixteen character key with the current connection. Then the caller's connection to the copy of *Model 204* is closed, but *Model 204* still thinks the connection is active. The connection can be subsequently reconnected by a call to FSUBIND. The connection is identified by a pointer to its FSCB.

This routine or FSCLOSE **must** be called prior to freeing the FSCB or the buffer established by FSINIT, if any.

```

FSUBIND: PROCEDURE ( FSCB_PTR, KEY )
           RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR,      /* FSCB for connection */
    KEY CHAR(16);          /* session key */
```

PL/I FSUBIND Routine Declaration

The return code from FSUBIND is set as follows:

- **0 (Connection Unbound)**

The indicated connection has been successfully unbound and may be retrieved by a call to FSUBIND specifying the same value for *key*.

- **-4 (FSCB Pointer Invalid)**

The indicated FSCB is incorrect.

- **-8 (Connection Not Open)**

This call was a noop, since the indicated connection was never successfully opened.

- **-64 (Treated as FSCLOSE)**

Either the current version of CRAM does not support the UNBIND service, or an unbound session exists with the same key value. This call has been treated as an FSCLOSE call.

3.24 FSVIS — See if Connection is Sequential and Visible

FSVIS examines the local data buffer to see if it contains a data stream to format a 3270 for reading a single line of input that is not masked (that is, not a password).

```

FSVIS  PROCEDURE ( FSCB_PTR )
        RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR;      /* FSCB for connection */

```

PL/I FSVIS Routine Declaration

FSVIS returns a fullword integer value as follows:

- **1 (Connection in Sequential Visible Mode)**

The data buffer contains a stream to format a 3270 for reading a single visible line.

- **0 (Connection in Full Screen or Invisible Mode)**

It is also possible that the connection is not active.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

3.25 FSWRITE — Write Data Stream to Model 204

FSWRITE sends *Model 204* the 3270 READ-MODIFIED data stream previously placed in the connection's local data buffer by a call to FSCOPY1. If *Model 204* is currently holding output for the connection, indicating a protocol violation, the FSPLI application should ignore its input stream and call FSREAD to receive the 3270 data stream from *Model 204*. It may also be prudent to then transmit a PA2 sequence to *Model 204*, using FSWRITE. This wM204. to retransmit the entire contents of the current screen image for the connection.

```

FSWRITE PROCEDURE ( FSCB_PTR )
        RETURNS ( FIXED BIN(31) );

DCL FSCB_PTR      PTR;      /* FSCB for connection */

```

PL/I FSWRITE Routine Declaration

FSWRITE returns a fullword integer value as follows:

- **0 (Write Successful)**

The write has completed, the provided data stream has been sent to *Model 204*. *Model 204* will respond with another 3270 data stream, so FSREAD should be called next.

- **-4 (Invalid FSCB Pointer)**

A valid FSCB pointer was not supplied, the call is ignored.

- **-8 (Connection Not Open)**

This call was ignored, since the indicated connection was never successfully opened.

- **-12 (Connection Not Initialized)**

This call was ignored, since the indicated connection has not been successfully initialized by FSINIT or FSINITI.

- **-16 (Model 204 Terminated)**

The *Model 204* at the other end of the connection terminated. This call is rejected.

- **-20 (Protocol Error)**

Model 204 is holding output for the connection, FSREAD should be called instead of FSWRITE. The call has been ignored.

APPENDIX A *Fast/CRAM Session Management***A.1 Fast/CRAM UNBIND Function**

The *Fast/CRAM* UNBIND function disconnects the client side of a *Fast/CRAM* session while preserving the *Model 204* side of the session. A twelve character *session key* is used to identify the unbound session. This ID must be unique for a given CRAM channel. UNBIND also allows the client to pass a single fullword of data to be saved with the session. This fullword will be returned when a client re-binds to the session, as explained below.

A.2 Fast/CRAM BIND Function

The *Fast/CRAM* BIND function takes a CRAM channel name and a session key and attempts to reconnect the client side of a previously unbound session. If an unbound session with the identified key is located, BIND reconnects the current client and returns the fullword of data previously saved for the session. This allows the client to carry on without the *Model 204* application being aware that the client has potentially changed address spaces.

If BIND cannot find an unbound session with a matching session key, then BIND acts like a CRAM OPEN — allocating a new *Model 204* thread to the session. Separate return codes indicate whether a previously unbound session was re-bound or if a new session was created.

A.3 Fast/CRAM BREAK Function

The *Fast/CRAM* BREAK function lets *Model 204* break a connection to the indicated client, effectively issuing a remote "close" for the session. If the indicated session is in the unbound state, it is silently cleared and the next *Fast/CRAM* BIND call specifying that session key will cause a new session to get created. If the indicated session was open, then the next *Fast/CRAM* call from that client will be rejected with a "connection closed" return code.

Sirius has extended the processing for IODEV=29 and IODEV=11 threads so if a site is using *Fast/CRAM*, the BUMP command and thread timeouts will use the *Fast/CRAM* BREAK function on the relevant thread. This is especially useful for cleaning up unbound sessions that have been "forgotten."

APPENDIX B *IMSIT use of FSBIND and FSUBIND*

The *IMSIT* interface package allows *Model 204* User Language applications to communicate with ITC terminals and printers connected to IMS DC and controlled by the custom ACP application. Fully conversational User Language programs may be invoked by other IMS transactions and may transfer control back to different IMS transactions.

IMSIT runs under the ACP Message Processing Program (MPP) in an IMS dependant region. At any given time, the ACP is either processing a message, waiting at a read request for the next message, initializing or terminating.

When an *IMSIT* transaction is selected from an initial screen, the IMS MPP scheduler selects the first available MPP to schedule ACP, which retrieves its message, performs some pre-processing, and then dynamically links to the *IMSIT* interface program. *IMSIT* processes its message, eventually filling in some control blocks and returning to the ACP to output the reply message. This process continues until IMS is shut down.

While an *IMSIT* MPP is processing a message, its region is unavailable to process other messages. When a message for ACP arrives, IMS first attempts to use an idling copy of ACP (that is, one resident in a dependant region and waiting on a message read). If none is available, it will either start a new ACP MPP or queue the message to a busy copy. This parallel processing and load balancing is an important contributor to overall system performance, delivering reduced response times. Tuning the number of MPP regions is similar to tuning the number of servers in a *Model 204* **ONLINE** region.

The processing of *IMSIT* is complicated by the fact that the transaction model for IMS MPPs is *single message in — single message out*, the life of a transaction spans just one interaction with a terminal. *Model 204* on the other hand support a *conversational* transaction model. A single transaction from the *Model 204* point of view might require several interactions with a terminal. Between interactions with the terminal, a *Model 204* application may maintain state data for the transaction in its server area. *IMSIT* bridges the gap, allowing conversational transactions with ITC terminals.

As a result of the load balancing employed by IMS, subsequent interactions between the ITC terminal and *Model 204* might occur in different address spaces. Therefore a session between an ITC terminal and *Model 204* (via the ACP and *IMSIT*) be transferable from one IMS dependant region to another, depending upon how the terminal interaction messages are scheduled. This is accomplished by the FSBIND and FSUBIND functions in *FSPLI*, which are built upon the BIND and UNBIND primitives of *Fast/CRAM*.

When *IMSIT* is invoked by the ACP, it uses FSBIND to reconnect to a *Model 204* *Fast/CRAM* session, using the extended terminal ID as the session key. If a session is rebound **and** the current screen is an initial screen, the previous session is closed and a

new connection obtained from FSOPEN. This covers two cases: 1) the end user got impatient and started a second transaction before getting the response to the first transaction and 2) *IMSIT* was abended after calling FSUBIND.

IMSIT processes its message and uses various *FSPLI* calls to interact with *Model 204*. During that time the *Fast/CRAM* session remains fully connected. When *IMSIT* is ready to respond to its terminal, it fills in a common data structure passed by the ACP, since the ACP will format the response for the ITC terminal. If the current output is the end of a *Model 204* session, *IMSIT* uses FSCLOSE to terminate the *Fast/CRAM* session. Otherwise, it uses FSUBIND to disconnect from the *Fast/CRAM* session. Finally, *IMSIT* returns control to the ACP. The ACP processes the data passed back from *IMSIT*, possibly sending a response message to the ITC terminal.

If the ACP detects an error in the data passed from *IMSIT*, it abends. This is a common occurrence while debugging new applications, since the data in question typically comes from *Model 204* User Language programs. When an abend occurs, the ACP invokes IMSITABD in *IMSIT*. If IMSITABD determines that an unbound *Fast/CRAM* session exists for the current ACP transaction, it re-binds the session and sends an error code back to the *IMSIT* user function executing under *Model 204*. Then the IMSITABD returns, allowing abend processing to clean up ITC and IMS resources for the failing transaction.