SC31-4069-00

3890/XP Enhanced Document Processor

Programming Guide

SC31-4069-00

3890/XP Enhanced Document Processor

Programming Guide

Note!

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

First Edition (January 1998)

This edition is a major revision of GC31-2662-2. This edition now applies solely to the IBM 3890/XP Enhanced Document Processor, not the 3890/XP Document Processor. Information in this manual is subject to change from time to time. Before using this publication in connection with the operation of IBM systems, consult the *IBM System/370, 30xx, 4300, and 9370 Processors: Bibliography of Industry Systems and Application Programs*, GC20-0370, for the editions that are applicable and current.

IBM does not stock publications at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department 72L, 8501 IBM Drive, Charlotte, NC, 28262, U.S.A. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

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 or other legally protectable 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.

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, MG39/201, 8501 IBM Drive, Charlotte, NC 28262-8563, U.S.A. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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 the IBM Director of Commercial Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY, 10594, U.S.A.

Trademarks

The following terms, denoted by an asterisk (*) elsewhere in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

IBM	Personal System/2	PS/2
Operating System/2	OS/2	System/370
C/2	VSE	MVS

The following terms, denoted by a double asterisk (**) elsewhere in this publication, are trademarks of other companies as follows:

Intel	Intel Corporation
Recognition UT 600/1000 XP	Recognition International Inc.

About This Manual

This manual describes the IBM* 3890/XP Enhanced document processors, the Control Program, data area references, Sort programs, channel-attachment communication, and Operating System/2* (OS/2*) Local Area Network (LAN) attachment.

Who Should Read This Manual

This manual is for:

- Customer programmers who write Sort programs
- System analysts and application and system programmers who install, diagnose, and use the 3890/XP Enhanced document processors
- Field support personnel who diagnose problems.

This manual assumes you are familiar with the information in the *IBM 3890/XP Document Processor General Information Manual*.

Terms That You Should Know

Host support means either:

- 3890/XP Multiple Virtual Storage (MVS*) support licensed program (Program No. 5685-037)
  or
- 3890/XP Virtual Storage Extended (VSE*) support licensed program (Program No. 5686-008).

3890/XP Series document processor or the term **XP document processor** means all models of the 3890/XP Series document processor.

Routing number means the document code line routing-and-transit field.

Code line data matching replaces the former terms image matching and image processing.

Image refers to the Image Capture System.

Simulated Document Mode was formerly called Image Run Mode (that is, hardware-generated document images on the 3890/XP).

Control Program means the 3890/XP Series document processor microcode and related modules that are released by IBM on the Install diskette.

Reject Pocket means module-pocket 1/1.

This manual also refers to:

* Trademark of IBM

- IBM 3890/XP High-Level Language Services licensed program (Program No. 5601-300) (World Trade Only)
- IBM 3890/XP Toolkit I licensed program (Program No. 5688-043) (World Trade Only)
- IBM Electronic Payment System licensed program (Program No. 5746-XYA)
- IBM Financial Transaction Processing System licensed program (Program No. 5728-019)
- IBM Check Processing Control System licensed program (Program No. 5734-F11).

How This Manual Is Organized

This manual consists of ten chapters and three appendixes:

- Chapter 1, “Introduction to the 3890/XP Enhanced Document Processors”
- Chapter 2, “Initialization Data Record (IREC)”
- Chapter 3, “Run Profile and Related Files”
- Chapter 4, “Record Headers”
- Chapter 5, “Document-Time Processing”
- Chapter 6, “Code Line Data Matching”
- Chapter 7, “Programmable Endorsement and Item Numbering”
- Chapter 8, “Host Communication”
- Chapter 9, “Power Encoder”
- Chapter 10, “Additional Capabilities”
- Appendix A, “Run Initialization Error Codes”
- Appendix B, “Test Modes”
- Appendix C, “Storage.”

This manual also contains a glossary and an index.

Related Publications

For other information related to the 3890/XP Enhanced, see the following publications:

- *IBM 3890/XP Series Document Processor General Information Manual*, GA34-2012
- *IBM 3890/XP Series Stacker Control Instructions Reference*, SC31-2703
- *IBM 3890/XP MVS Support and 3890/XP VSE Support Program Reference*, SC31-2654
- *IBM 3890/XP Enhanced Operator’s Guide*, GA34-2212
- *IBM 3890/XP Series and 3890/XP Enhanced SPXServ Reference*, SC31-4070
- *IBM 3890/XP High-Level Language Services Program Reference Manual*, SC31-2653 (World Trade only)
- *IBM 3890 Document Processor Machine and Programming Description*, GA24-3612
- *ImagePlus High Performance Transaction System General Information Manual*, GC31-2706
- *IBM Operating System/2 Extended Edition Version 1.3 APPC Programming Reference*, S01F-0295
- *IBM Operating System/2 Extended Edition Version 1.3 APPC Programming Services and Advanced Problem Determination for Communications*, S01F-0299
- *IBM System/360 and System/370 I/O Interface Channel to Control Unit, Original Equipment Manufacturers’ Information*, GA22-6974
- *IBM 3890/XP MVS Support Program Reference Manual*, SC31-3886

- *IBM Operating System/2 Extended Edition Version 1.3 User's Guide, Volume 1: Base Operating System*, 64F2904.
- *IBM Operating System/2 2.1 Information and Planning Guide*, S61G-0913.

Chapter 1. Introduction to the 3890/XP Enhanced Document Processors

This chapter describes the following:

- The 3890/XP Enhanced document processors
- Run initialization
- Document code line definition
- Document processing
- Differences among the 3890/XP Enhanced document processors
- Installing the 3890/XP Enhanced Control Program.

The 3890/XP Enhanced Document Processors

The IBM 3890/XP Enhanced document processors are medium- and high-speed, high-volume document processors that read magnetically and optically inscribed documents to perform check-processing applications such as:

- Sorting
- Microfilming
- Endorsing
- Item numbering
- Encoding
- Image capturing.

The XP Enhanced document processors are driven by a Control Program that resides in an attached IBM Personal System/2* (PS/2*). The Control Program runs under the IBM Operating System/2 (OS/2). This provides an extensive environment for function and programming enhancement.

Sort Programming Options

The IBM 3890/XP Enhanced document processors provide new ways to design and write programs for sorting documents.

In the past, with the 3890 document processors, you wrote Sort programs at a host computer using the low-level set of Stacker Control Instructions (SCI). This required special skills and education, especially for programmers who had been working with financially-oriented, general-purpose, high-level languages such as COBOL.

With the 3890/XP Enhanced document processor, you can:

- Continue to write Sort programs for OS/2 1.x (16-bit applications) supported languages, or migrate to OS/2 2.x (32-bit applications) supported languages.
- Continue to write and use Sort programs written with the Stacker Control Instructions (SCI) provided with the 3890 document processor.

---

* Trademark of IBM

- Mix Sort modules written in an OS/2 1.x (16-bit applications) supported language with Sort modules written in SCI and modules written in OS/2 2.x (32-bit applications) supported languages.

OS/2 Language Functions

Sort program modules written in an OS/2-supported language can use the following facilities.

SPXServ Functions: Sort-program-execution services (SPXServ) functions provide access to the internal data and functions of the 3890/XP Enhanced Control Programs. Each SPXServ function is a callable entry point that copies data to or from an internal data area or performs a control function. The SPXServ library is supplied as part of the XP Enhanced Control Programs.

For more information about the SPXServ functions, see the *3890/XP Series and 3890/XP Enhanced SPXServ Reference*.

HLLServ Functions: High-level-language services (HLLServ) supply additional functions that are tailored to the 3890/XP Enhanced document processors.

For more information about the HLLServ functions, see the *3890/XP High-Level Language Services Program Reference Manual*.

This manual is *recommended reading* for the concepts involved in writing Sort programs in OS/2-supported languages such as Dynamic Link Library modules (DLLs).

Stacker Control Instructions (SCI)

The Stacker Control Instruction (SCI) language is a System/370 macro language. You assemble and link-edit the Sort program. The host access method then loads the resulting object code (machine language) into the XP Series and XP Enhanced document processors through the channel or APPC 6.2 attachment.

SCI macros produce 2-, 4-, or 6-byte SCIs for performing required functions. Some macros can use either 2- or 4-byte addresses. SCIs compare and test data from the document and make module-pocket and feature decisions.

For more information about SCI and SCI macros, see the *3890/XP MVS Support and 3890/XP VSE Support Program Reference* and the *3890/XP Series Stacker Control Instructions Reference*.

Mixing OS/2 Languages and SCI

Your Sort program can consist of modules written entirely in one language or of modules written in different languages. Further, you can include modules that are independent of one another, serving specific functions. The Endorse Control program, which is part of the 3890/XP Toolkit I licensed program, is such an example. In addition, your SCI program (using the SCI “CALLNATV” macro) can call Sort modules that are written in OS/2 languages at various entry points.

Run Initialization

Run initialization is the part of the Control Program that prepares a document processor for operation. The Control Program uses initialization data from several sources to prepare for sort operations.

Initialization Data Record (IREC)

You can define the initialization data record (IREC):

- At the host with the assembler IREC macro
- At the PS/2 with the HLLServ IREC Create Facility function.

The initialization data record basically includes:

- Definitions of the fields on documents
- Requested options or features
- Name of the Run Profile.

IBM host check-processing licensed programs include:

- Electronic Payment System (DOSCHECK)
- Financial Transaction Processing System (FTPS)
- Check Processing Control System (CPCS).

Run Profile

The Run Profile, an ASCII file on the PS/2 disk subsystem, contains additional initialization data specifications for controlling each Sort job. The Run Profile is an extension of the IREC. If the Run Profile is not specified in the IREC, then the Run Profile specified in the Machine Profile is used.

The Run Profile provides the following additional initialization information:

- Names of the Sort modules that make up the Sort program
- The Sort Program Sequence (SPS) event for which each Sort module runs
- Names of the related information files used for the Sort run
- Other initialization data.

IBM supplies a set of Run Profiles with default values. You can use the 3890/XP Toolkit I licensed program or an ASCII editor to change these values to customize Run Profiles for your different Sort program runs.

Machine Profile

The Machine Profile is a file used by the initialization function that provides default initialization requests for values not requested in the current IREC or Run Profile. This data includes the:

- Drive and path for all Run Profiles
- Default Run Profile for Sort runs that do not specify one in the IREC.
- Default drives and paths for other Run Profile data files
- Default UserDLL to be loaded for every initialization
- Default entry points to provide the current Sort program.

You can change the Machine Profile by using the Control Program Machine Profile screens. The Machine Profile is fully described in the following:

- *3890/XP Enhanced Document Processor Operator's Guide.*

Document Code Line Definition

The 3890/XP Enhanced document processors let you configure the document processor to match the code line of your documents. You can change your code line definition at each run initialization or by using the SPX command to write the code line definition table.

You can define up to 15 fields on the document code line. Users in the United States typically configure their machines for the ABA code line. In the ABA code line, the field definitions are:

Field 1 - Amount
Field 2 - Process control
Field 3 - Account number
Field 4 - Optional field for special applications
Field 5 - Routing number
Field 6 - Optional field for special applications
Field 7 - Serial number.

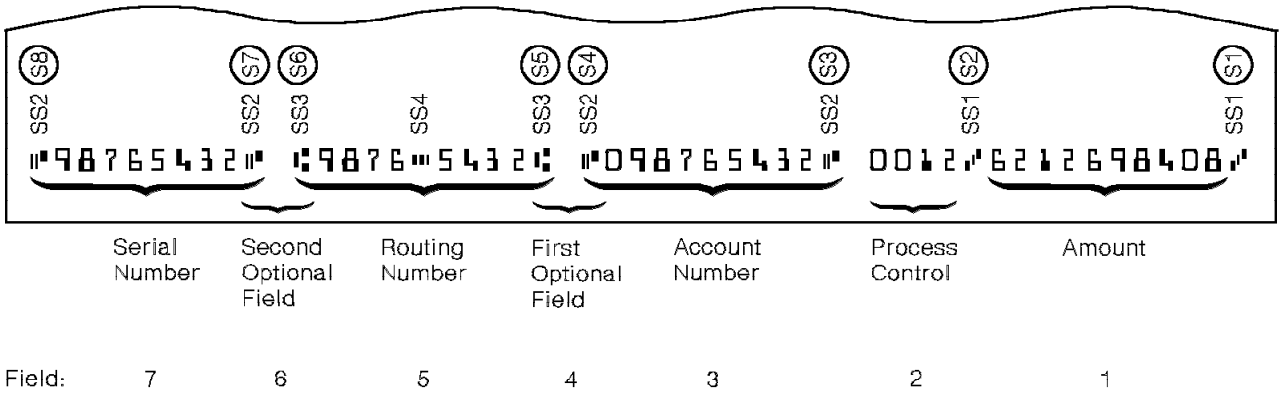

Figure 1-1. Typical Document Layout (ABA Code Line)

Special symbols (SSs) are used to define fields on the document code line. The symbols can be in any sequence that is compatible with your code line. See "Special Symbol Sequence Table (SST) File" on page 3-17. In the example in Figure 1-1, symbols SS1, SS2 and SS3 define the fields and SS4 divides a field into two elements.

Document Processing

Document processing on the 3890/XP Enhanced Document Processor can be performed either online or offline.

Overview

Document processing involves the following basic steps:

1. Initialize the document processor using information from the following:
   - Initialization data record (IREC)
   - Run Profile and related files.
2. Read the document data into the input buffer.
3. Perform the following to prepare the process buffer:
   - Code line verification
   - Code line correction
   - Process buffer format
   - Code line data matching (optional).
4. Run the user-supplied Sort program (an SCI module and/or user OS/2 modules):
   - Module-pocket decision
   - Feature control.
5. Perform the following post-Sort processing:
   - Copy and transmit the module-pocket and the feature control information.
   - Update and transmit the endorsement text and INF number.
   - Space the microfilm.
   - Update pocket counters and feed merge documents.
   - Copy the process buffer to the read buffer for the host to read.
   - Disengage or pause the transport.

Sort Program Sequence (SPS) Events

Sort Program Sequence (SPS) events occur as the document processor runs. Examples of SPS events include:

- The operator initializes the document processor to run a Sort job.
- The operator (optionally) communicates with a Sort program.
- The document processor reads a document.
- The document transport motors stop.

A different Sort module runs for each SPS event. When the operator first initializes the XP document processor for a run, the Run Profile might specify a Sort module to perform the following functions:

- Initialize values in the Sort program.
- Load additional tables.

When the document processor reads a document, the user Sort module makes the module-pocket decision during the SPS document event. Because this part of sorting is time critical, there is no time for database maintenance. The Sort program can only keep data in its program storage while documents are flowing. A message, if any, can be displayed only at the end of the run.

When the document transport stops, however, you can specify another user-written Sort module or entry point to collect the statistics recorded in program storage by the time-critical module and write the information to the disk subsystem, if that is required.

Online or Offline

To switch between host online and host offline processing, you must display the Run screen. For the 3890/XP Enhanced Document Processor, select online or offline with the operator console switch. For the 3891/XP, 3892/XP, and Universal Transport (UT)/XP**, enter the COMM ON or COMM OFF operator commands.

Auxiliary Storage

Auxiliary storage consists of two distinct areas that your Sort modules can use, 3890-emulated auxiliary storage and additional auxiliary storage. (OS/2 language modules can also have one or more privately defined data segments.)

1. The 3890-emulated auxiliary storage contains the following information:
   - Symbol-error-correction information
   - Installed-features information
   - A save area.
2. Additional auxiliary storage contains information such as:
   - Process buffer data and header
   - Endorsement text
   - Sort Program Sequence (SPS) event indicators
   - A “new” save area.

For more information, see “Auxiliary Storage” on page C-7.

Read Record

The 3890/XP Enhanced allows for a larger read record process buffer than the previous 3890 document processor, up to 256 bytes long.

You define the 256 bytes into 15 logical fields and the 12-byte header (field 0). The 3890/XP Enhanced Control Program puts data in the first seven fields from the document code line. Your Sort program puts data in the remaining eight fields. You might use these remaining fields for:

- A depositor’s account number
- An alternate-account-numbering system
- A pass-pocket history of which pocket is used on each pass
- Optical Character Recognition data.

Memory Storage Locking

For the 3890/XP Enhanced, memory storage segments are locked during time-critical operation of the Sorter when documents are being processed. This prevents swap I/O to the PS/2 disk subsystem from causing a machine check on the 3890/XP Enhanced, which stops the machine. The segments locked include user program storage segments as well

** The UT/XP is trademarked by Recognition International Inc as *Recognition UT 600/1000 XP*.

as user segments that are either loaded as part of the user DLL or are allocated during the SPSINIT event.[1]

Because storage locking is restrictive, the segments are unlocked at motor stop time and are locked again when document feeding resumes or when the next initialization occurs.

The amount of additional memory storage available to the user beyond 3 megabytes depends on the amount of free space on the logical drive that contains the swap file. The amount that can be locked depends on the size of memory storage (RAM) and the number of other processes active in the system. You should refer to the SWAPPATH statement in the CONFIG.SYS file. Additional information is given in the *OS/2 User's Guide*. If the MEMMAN statement in CONFIG.SYS defines memory as NOSWAP, less storage will be available.

[1] It is advisable to use static segments defined and loaded as part of the DLL. Dynamically allocated segments can be "lost" when the next initialization or reinitialization occurs. They are recovered when the Control Program itself is ended.

Differences among the Document Processors

There are three basic document processor models:

- 3890/XP or 3890/XP Enhanced document processors
- 3891/XP document processor
- 3892/XP document processor.

Several key differences among these document processors are described in Figure 1-2. Other differences are discussed in the relevant sections of this guide.

Figure 1-2. Differences among the Document Processors

Feature	3891/XP, 3892/XP, or Universal Transport/XP	3890/XP or 3890/XP Enhanced
Use of CONVERGE	The Run Profile specifies the IBM-supplied CONVERGE module (supplied with the Install diskette) to run during the SPSPOST event for endorsement and INF functions. See also the "Full Function Endorse" on page 7-18, which allows you to supply endorsement text without using CONVERGE.	The CONVERGE module is not used.
Endorsement characters	The endorsement characters are larger. The UT/XP prints at 8 characters per inch.	The endorsement characters are smaller.
INF number	The INF number is part of the third endorsement line. The 8-digit INF number prints as 10 digits with two high-order zeros.	The INF number is below the third endorsement line.
Deconfigured features	Deconfigured features can cause an initialization error and stop document processing. This is a serious error.	The 3890/XP and 3890/XP Enhanced cannot deconfigure features.

Installing the XP Enhanced Control Program

There are Install diskettes for each of the following:

- 3890/XP
- 3890/XP Enhanced
- 3891/XP and 3892/XP
- UT/XP

The Install program performs the following:

- Creates and uses the SORTER, CONTROL, and INIT subdirectories
- Changes the system Startup command so that, when you boot or power on the PS/2, the Control Program starts automatically and brings you to the Run screen on the 3890/XP Enhanced.
- Changes the CONFIG.SYS file to use appropriate device drivers.

When installing the OS/2 system, an additional diskette containing the "communication configuration" files is used. There are also separate Install diskettes for the 3890/XP Diagnostics. The 3890/XP Diagnostics Install creates and uses the RAS directory.

Installation Procedure

The 3890/XP Enhanced Control Program must be installed on the PS/2 disk subsystem. To install the control program, insert the Install diskette in the diskette drive of the PS/2 system unit and enter the following command from the **C:>** prompt:

```
A:INSTALL
```

Installation Help

For help with installation, you can review the README notes on the Install diskette.

Insert the diskette in the diskette drive of the PS/2 system unit and enter the following command from the **C:>** prompt:

```
A:README
```

Chapter 2. Initialization Data Record (IREC)

This chapter describes the following:

- Initializing the 3890/XP Enhanced document processor
- IREC and 3890/XP Enhanced Functions
- IREC Contents - General Description
- IREC Contents - Detailed Description.

Initializing the Document Processor

Run Initialization is the part of the Control Program that prepares the document processor for processing documents. The XP Enhanced document processors use the same initialization data record (IREC) that the 3890 document processor uses. The XP Series document processors have additional information in the Run Profile and related files. The XP Enhanced document processor has additional functions as well as additional information in the Run Profile and related files.

The Run Profile is an extension of the initialization data record. The Run Profile can contain initialization commands and specifications that allow the sort programmer to store control files on the PS/2 disk subsystem. For an overview of the files involved, see Figure 2-1 on page 2-2. The Run Profile resides on the PS/2 disk subsystem of the XP Series and XP Enhanced document processors. You can use the 3890/XP Toolkit I or an ASCII editor such as the editor supplied with OS/2 to create your Run Profile and other related initialization files.

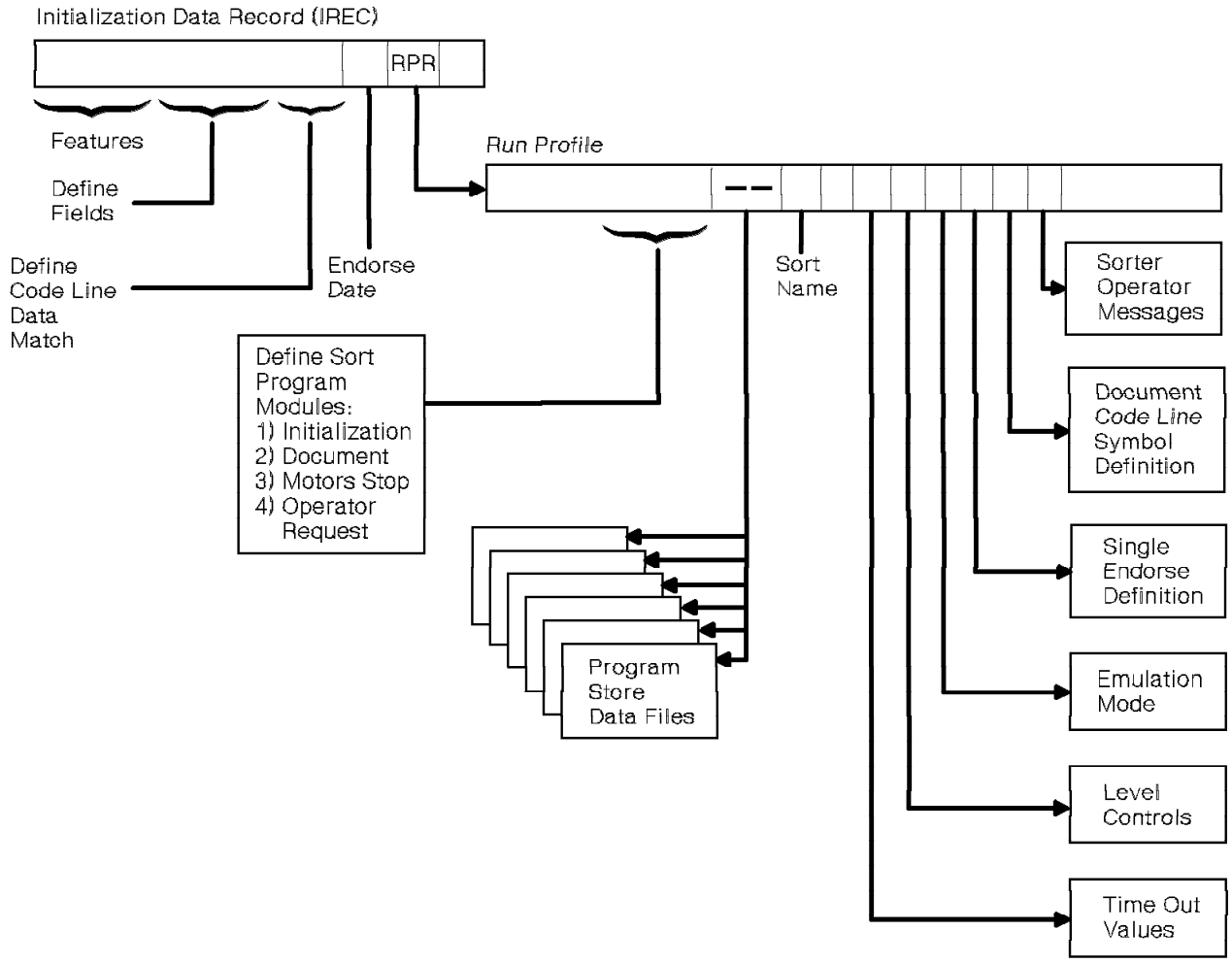

Figure 2-1. 3890/XP Enhanced Initialization Requirements

Initialization online occurs at the end of a program load from the host system (see “Loading the User Control Data” on page 8-4). Initialization offline occurs after the Sort program is loaded, via the operator Load command, from the PS/2 disk subsystem. If you have specified any user-written OS/2 language routines, they must be available as dynamic link library modules (DLLs) on the disk subsystem of the PS/2 and must be named in the Run Profile.

IREC and XP Enhanced Functions

Initialization data, the first 128 bytes of the Sort program (and, therefore, of program storage), specifies the requested functions of the XP Series and XP Enhanced document processors.

Initialization data includes run parameters such as:

1. Feature initialization
    a. Endorser on/off and print position
    b. Item-numbering on/off, print position, and print data
    c. Microfilm on/off, recording mode, and index number data
    d. Optical Character Recognition feature on/off
    e. Power Encoder feature on/off
2. Merge feed on pocket count control
3. Pocket identification
    a. Kill or rehandle
    b. Pocket count
4. Code line data matching control
5. Field record information
    a. Field length
    b. Fixed or variable
    c. Dash transmission
6. High-order zero correction
7. Symbol error correction
8. Test modes
    a. Block Physical Operation (BPO)
    b. Debug mode
9. Run Profile name
10. Endorse date
11. Image Capture System initialization.

Figure 2-2. Feature Initialization Data (Bytes 0 through 127)

Byte(s)	Description
0	Feature initialization byte
1-20	Feature initialization data
21	Control for merge feed, code line data matching, and test modes
22-35	Field record information for fields 1 through 7
36	Field length for field 8
37-54	Pocket kill/rehandle/count
55	Symbol error correction
56-59	Pocket limit values for merge feed
60	High-order zero correction
61	Run Profile name format and endorse date format
62-63	Restart run ID
64	3891/92 XP feature data
65	Saved program indicator
66-67	Reserved
68-69	Field record information for extended code line data match field 8
70-77	Run Profile name
78-79	Feature controls
80	Sort-control of auto-selects (SCA) and routing-number-transparency (TRT) options
81	Initialization for programmable endorsement and item-numbering feature (INF), print position for endorsement, and endorsement enable
82	INF enable
83-87	INF number in unsigned packed decimal
88-95	Endorse date
96-109	Field record information for extended fields 9 through 15
110	Image Capture System initialization
111-113	Reserved
114-115	Image Capture System error list length
116-119	Image Capture System error list pointer
120	Field type definitions for extended code line data matching
121-127	Reserved

IREC Contents — Detailed Description

The following tables provide a detailed description of the contents of the initialization data record (IREC).

Feature Initialization

The feature initialization byte allows the sort programmer to request a change in the status of the 3890 emulated features. (For features specific to the 3891/XP, 3892/XP, and UT/XP, see Figure 2-17 on page 2-14.) The least significant nibble represents user data reset requests that are used to reset the machine data from a previous run initialization.

Figure 2-3. Feature Initialization (Byte 0)

Byte	Bit(s)	Description
0	0	Initializes item-numbering feature (INF). If this bit is on (1), the Control Program uses bytes 1 through 5 or 82 through 87 to initialize the INF. If this bit is off (0), the Control Program ignores bytes 1 through 5 or 82 through 87 *and does not change the existing status of the feature*.
	1	Initializes endorsement. If this bit is on (1), the Control Program uses bytes 6 through 10, 81, or 88 through 95 to initialize endorsement. If this bit is off (0), the Control Program ignores bytes 6 through 10, 81, or 88 through 95 *and does not change the existing status of the feature*.
	2	Initializes Image Capture System. If this bit is on (1), the Control Program uses bytes 11 through 15 and 110 to initialize the Image Capture System. If bits 0 through 3 of byte 110 are off (0), the Image Capture System is *disabled*. If this bit is off (0), the Control Program ignores bytes 11 through 15 and 110 *and does not change the existing status of the Image Capture System*.
	3	Initializes microfilm feature. If this bit is on (1), the Control Program uses bytes 16 through 20 to initialize the microfilm feature. If this bit is off (0), the Control Program ignores bytes 16 through 20 *and does not change the existing status of the feature*.
	4	Resets pocket counters.
	5	Resets user stats.
	6	Initializes code line data match (write) buffers.
	7	Reserved.

Note: In order to perform interim initializations without changing the status of features and other initialization data, you must:

- Specify `*NONE` as the Run Profile name
- Ensure that the IREC byte 0 bits are off.

Item Numbering

You use this data area to specify the Item Numbering feature initialization parameters. You can use this area only when byte 0 bit 0 is on and byte 81 bit 0 is off.

Figure 2-4. Item Numbering (Bytes 1 through 5)

Byte(s)	Bit(s)	Description
1	0	Reserved.
	1-3	Length of the high-order INF segment. This is the length as decoded from unsigned packed binary-coded decimal (BCD) with a maximum of 5 (101). A value of 6 or 7 is not permitted.
	4-5	Reserved.
	6-7	INF print position. This is a binary number that represents one of three possible print positions. Binary Number / Print Position 01 — 1 10 — 2 11 — 3 *Feature disable* — With both bits 6 and 7 off, INF does not operate or indicate errors. **Notes:** 1. When byte 81, bit 0 is on, the INF print position is determined by byte 81, bits 4 through 7. 2. When endorsement is initialized on, the INF print position is determined by the endorsement print position (byte 6, bits 6 and 7, or byte 81, bits 4 through 7).
2-5		Eight-digit unsigned and packed decimal starting number for INF. **Note:** The Sort program controls item number incrementing. See bits 6 and 7 of the Document Data Record Header "Feature Control Byte" on page 4-4.

Note: Bytes 1 through 5 are valid only if byte 81, bit 0 is off. When byte 81, bit 0 is on, these bytes are ignored and bytes 82 through 87 control INF.

Endorsement

You use initialization record byte 6 to specify endorse feature initialization parameters. You can use this field only when byte 0 bit 1 is on and byte 81 bit 0 is off.

Figure 2-5. Endorsement (Bytes 6 through 10)

Byte(s)	Bit(s)	Description
6	0-5	Reserved.
	6-7	Endorsement print position. This is a binary number that represents one of three possible print positions. Binary Number / Print Position 01 — 1 10 — 2 11 — 3 *Feature disable* — With both bits 6 and 7 off, the endorsement feature does not operate or indicate errors. **Note:** When byte 81, bit 0 is on, the print position is indicated by byte 81, bits 4 through 7.
7-10		Reserved.

Note: Bytes 6 through 10 are valid only if byte 81, bit 0 is off. When byte 81, bit 0 is on, these bytes are ignored and byte 81 controls endorsement.

Image Capture System Cycle Number and Date

The image capture system has the next five bytes reserved for future use.

Figure 2-6. Image Capture System Cycle Number (Bytes 11 through 15)

Byte(s)	Bit(s)	Description
11	0-7	Reserved.
12-15		Reserved.

Microfilming

You use initialization record bytes 16 through 20 to define the Microfilm feature initialization data. You can use this field only when byte 0 bit 3 is also turned on.

Figure 2-7 (Page 1 of 2). Microfilming (Bytes 16 through 20)

Byte(s)	Bit(s)	Description
16	0	Reserved.
	1-3	Length of the high-order segment as decoded from unsigned packed BCD with a maximum of 5. A value of 6 or 7 is not permitted.

Figure 2-7 (Page 2 of 2). Microfilming (Bytes 16 through 20)		
Byte(s)	**Bit(s)**	**Description**
	4-5	Reserved.
	6	*On* enables the microfilm. *Off* disables the microfilm; the microfilmer does not operate and does not indicate errors.
	7	*On* enables duo-mode (front). *Off* enables duplex-mode (front and rear) 3890/XP.
17-20		Eight-digit unsigned and packed decimal number for the starting microfilm index number. The actual starting number is this number plus 1.

Note: For machine types 3891/XP and 3892/XP, if INF is activated, the microfilm number automatically becomes the INF number. For 3890 machines, these numbers are separate and must both be specified.

Merge Feed, Code Line Data Matching, BPO, and Debug

Byte 21 is used to control Sort program optional functions.

Function **Definition**

Merge before main Feed from the merge hopper before documents are fed from the main hopper. The sort program uses this function to request a merge document for each stacker pocket.

Merge on pocket count
This function allows the sort programmer to automatically request a merge document to be routed to a stacker pocket when the number of documents in that pocket has reached a specified limit. The limit can be specified in IREC bytes 56 through 58 or by specifying a Pocket Definition Table in the Run Profile.

Code line data match This function reduces the number of rejected documents in the second or subsequent document pass. It uses document code lines from a user-created code line file or actual document data to process a document. This is done by downloading code line data from a host to the code line data match buffers. For more information, see the “Code Line Data Matching Overview” on page 6-1.

Extended code line data match
This function enables subsequent calls to the code line matching procedure and allows code line data matching on fields 8 through 15. For more information, see “Extended Code Line Data Match” on page 6-3.

Block Physical Operation (BPO)
BPO is a sort program test mode that allows the sort programmer to test programs without physically featuring any documents. When BPO is active, module/pocket decisions are generated normally. However, all documents are routed to the reject pocket. For more information, see “Block Physical Operation (BPO) Mode” on page B-1.

Debug mode Debug mode is an online sort program analysis tool. Physical documents can be processed but are not required. Document code lines are received from the code line data match buffers from the host.

Figure 2-8. Merge Feed, Code Line Data Matching, BPO, and Debug

Byte	Bit(s)	Description
21	0	Merge before main option.
	1	Feed merge on pocket count (see bytes 56 through 59).
	2	1 = Perform *standard* code line data matching. 0 = Do not perform code line data matching.
	3	1 = Perform *extended* code line data matching (requires bit 2). 0 = Do not perform extended code line data matching. **Note:** See byte 120 for field type definitions.
	4-5	Reserved.
	6	BPO - Block Physical Operation.
	7	Debug mode.

Field Record Information

Bytes 22 through 36 define code line fields 1 through 8. This includes two bytes for each field 1 through 7 and one byte (byte 36) for field 8, for a total of 15 bytes.

Figure 2-9 (Page 1 of 2). Field Record Information (Bytes 22 through 36)

Byte(s)	Bit(s)	Description
First Byte (22-35)	0	1 = Fixed length. 0 = Variable length.
	1	1 = Send dashes for SS4s in this field.[1] 0 = Do not send dashes for this field.
	2	1 = Perform code line data match for this field. 0 = Do not perform code line data match for this field.
	3-7	This is a binary number that indicates the maximum number of 4-bit hexadecimal characters for this field. If more characters are read than indicated, those in excess are ignored and the field is flagged as *length not valid* in the document data record header.

[1] When dash transmit is specified, on Models C and D Emulation (CMC7) a blank is transmitted as a dash. For multiple consecutive blanks, one dash is transmitted.

[2] When 3890 emulation is in effect, the sum of the second byte (bits 4 through 7 for fields 1 through 7) plus the length of field 8 (byte 36, bits 0 through 7) must not total more than 36. The record, including the 12-byte header, must not exceed 48 bytes.

Figure 2-9 (Page 2 of 2). Field Record Information (Bytes 22 through 36)		
Byte(s)	**Bit(s)**	**Description**
Second Byte[2] (22-35)	0-3	These four bits contain a binary number equal to the maximum number of acceptable digit errors permitted in the field when code line data matching. This number is the *threshold limit*.
	4-7	These four bits should represent the number of bytes assigned for this field. For boundary alignment, you should assign more program storage than is needed to contain the maximum number of characters specified in the first byte. If so, the high-order positions are padded with X'A's. If the number of bytes is too small to contain the number of 4-bit hexadecimal characters specified in the first byte, the initialization data is not correct and will result in an initialization error. If this field is not read from this document, these four bits should be 0.
36	0-7	Field 8 byte length - This field is not read from a document. It is a communication area between the host program and the user-supplied Sort program. If field 8 is not used, byte 36 should contain zeros. See bytes 68 and 69 for *extended* code line data matching with field 8.

Pocket Identification

Each stacker module-pocket is associated with a counter in which the Control Program can count the documents sent to that pocket. Sort programs can identify any pocket as a *kill pocket* or a *rehandle pocket* and can specify whether documents going to that pocket should or should not be counted.

Each pocket is defined by a half-byte:

- Byte 37 defines pockets 1 and 2 in module 1
- Byte 38 defines pockets 3 and 4 in module 1
  and so on up to ...
- Byte 54 defines pockets 5 and 6 in module 6.

Each byte format is the same, so only byte 37 is described. In this byte, bits 0 through 3 define module-pocket 1/1 and bits 4 through 7 define module-pocket 1/2.

Figure 2-10 (Page 1 of 2). Pocket Identification (Bytes 37 through 54)		
Byte(s)	**Bit(s)**	**Description**
37	0-1	Reserved — should be zero.
	2	1 = Count documents. 0 = Do not count documents.
	3	1 = This is a kill pocket. 0 = This is a rehandle pocket.
	4-5	Reserved — should be zero.
	6	1 = Count documents. 0 = Do not count documents.

Figure 2-10 (Page 2 of 2). Pocket Identification (Bytes 37 through 54)		
Byte(s)	**Bit(s)**	**Description**
	7	1 = This is a kill pocket. 0 = This is a rehandle pocket.
38-54		Same format as byte 37, for module-pockets 1/3 to 6/6.

Symbol Error Correction (SEC)

The symbol error correction function attempts to repair field opening and closing special symbols under a specific set of conditions. See "Pre-Sort Processing" on page 5-4 for more information.

Figure 2-11. Symbol Error Correction (Byte 55)		
Byte	**Bit(s)**	**Description**
55	0	1 = Perform symbol error correction. 0 = Do not perform symbol error correction.
	1-7	Each bit represents a field (1 through 7) of input document data. 1 = Field is not a candidate for symbol error correction. 0 = Field is a candidate for symbol error correction.

Pocket Limit Values for Merge Feed on Pocket Count

Pocket counting uses these values to determine when a kill or a rehandle pocket reaches its limit. When the Run Profile includes the PKTDEFtbl keyword to specify the name of the Pocket Definition Table (PDT) file, initialization ignores bytes 56 through 59 and bytes 37 through 54; but it still uses bytes 37 through 54 to indicate kill or rehandle pockets. Byte 21, bit 1 must be specified. The PDT file has pocket limit values for each pocket.

For more information, see "Pocket Definition Table (PDT) File" on page 3-14.

Figure 2-12. Pocket Limit Values for Merge Feed (Bytes 56 through 59)		
Byte(s)	**Bit(s)**	**Description**
56-57	0-3	Reserved — should be zero
	4-15	Binary number, pocket limit for *rehandle pockets*
58-59	0-3	Reserved — should be zero
	4-15	Binary number, pocket limit for *kill pockets*

High-Order Zero (HOZ) Correction

You can request the high-order zero correction function at initialization time to invoke a procedure that repairs character rejects in code line field one (amount field) under a set of specific rules. See “Pre-Sort Processing” on page 5-4 for more information.

Figure 2-13. High-Order Zero Correction (Byte 60)

Byte(s)	Bit(s)	Description
60	0-3	Reserved — should be zero.
	4-7	The binary number that represents the number of high-order positions of field 1 (amount) that can be changed from digit errors to zeros. High-order zero (HOZ) correction occurs if: • Field 1 is specified as a fixed-length field. • Field 1 is read with the correct length. • Positions to the left are either digit errors or zeros.

Run Profile Name, Endorse Date Format, and Restart Run ID

You can use byte 61 to specify the character format of the IREC fields specifying the Run Profile name and the Endorse Date.

Bytes 62 and 63 contain a host-generated number used to verify that the restart file being used is the one that the host is expecting. If a mismatch occurs, machine initialization will fail.

Figure 2-14. Run Profile Name, Endorse Date Format, and Restart Run ID (Bytes 61 through 63)

Byte(s)	Bit(s)	Description
61		Format of the Run Profile name (bytes 70 through 77) and the endorse date (bytes 88 through 95). Valid values are: 00 = EBCDIC 01 = ASCII.
62-63		Restart run ID.

Special Use Bytes

Feature	Definition
Endorsement side	Bit 3 of byte 64 selects the endorsement side on 3891/XP, 3892/XP, and UT/XP machines when you are using the IBM-supplied Converge USERDLL. For more information, see “The Converge DLL” on page 7-16.
Font size	Bits 4 and 5 of byte 64 select the ink jet printer font size on 3891/XP and 3892/XP machines. Large font prints one line of text on a side. Small font allows up to three lines of text per side.

Image count mark size

Bits 6 and 7 of byte 64 control the width of the microfilm count mark on the 3891/XP and 3892/XP machines.

Program stored with the save command

The Control Program sets this flag when the save command sends a Sort program to the disk subsystem. When the file is requested for an offline initialization, program storage files specified in the Run Profile will not be loaded.

Field 8 definition This field (bytes 68 and 69) defines the format of field 8 of the code line.

Figure 2-15. Special Use Bytes (Bytes 64 through 69)

Byte(s)	Bit(s)	Description
64	0	Not used — Field 3 closing symbol for 3890 document processor Models A and B. **Note:** The XP document processor ignores this bit even when in 3890 emulation. For more information, see "Special Symbol Sequence Table (SST) File" on page 3-17.
	1-2	Reserved — should be zero.
	3	Specify side for 3-line endorsement — 3891/3892/UT XP. (via CONVERGE) 0 = Back side. 1 = Front side (also requires byte 78, bit 6).
	4	Specify front ink jet font size — 3891/92 XP. 1 = Large font (allows 1 line) — cannot use CONVERGE. 0 = Small font (allows 3 lines).
	5	Specify back ink jet font size — 3891/92 XP. 1 = Large font (allows 1 line) — cannot use CONVERGE. 0 = Small font (allows 3 lines).
	6-7	Microfilm image count mark size — 3891/92 XP. Valid numbers are 1 and 2.
65	0	1 = This Sort program was placed on disk subsystem using the save command. 0 = This Sort program was not placed on disk subsystem using the save command.
	1-3	Reserved for offline diagnostic routine.
	4-7	Reserved.
66-67		Reserved — should be zero.
68-69		Field 8 record information for *extended* code line data match (if byte 21, bit 3 = 1). Bytes 68 and 69 have the same format as the *first* byte and the *second* byte, respectively, in Figure 2-9 on page 2-9.

Run Profile Name

This field specifies the name of the Run Profile to be used for this initialization. If this field is blank, the Run Profile specified in the Machine Profile is used.

Figure 2-16. Run Profile Name Bytes (Bytes 70 through 77)

Byte(s)	Bit(s)	Description
70-77		Run Profile name (optional). See byte 61 for the format. Specifying the name as *NONE inhibits Run Profile processing. **Note:** Always initialize the XP document processor with the Run Profile name either specified or blank (to default the Run Profile). If you specify *NONE as the Run Profile name for a document processor that is not previously initialized, a run initialization failure occurs.

Additional Features

These bytes specify the initialization requests for the non-emulated features (features not available on the original 3890). Also included in this table is the default for the Sort control of the autoselects function.

Figure 2-17 (Page 1 of 2). Additional Features Bytes (Bytes 78 through 80)

Byte(s)	Bit(s)	Description
78	0	Initializes OCR1 RPQ — 3891/92 XP.
	1	Initializes OCR2 RPQ — 3891/92 XP.
	2	Initializes OCR3 — 3890/XP.
	3	Initializes MICR1 — 3891/3892/UT XP.
	4	Initializes High Read (MICR2) — 3892/UT XP.
	5	Initializes back programmable endorsement — 3891/3892/UT XP.
	6	Initializes front programmable endorsement — 3891/3892/UT XP.
	7	Initializes Roll-On endorser only — 3891/3892/UT XP. **Note:** The UT/XP uses an ink jet graphics definition to emulate a roll-on stamp feature for each document.
79	0	Initializes Power Encoder — 3892/UT XP. **Note:** Encoder verification is always on unless configured off by the operator.
	1-3	Reserved.
	4-7	Roll-On endorser print position, value 1 through 6. If 0, defaults to 1. The values 4, 5, and 6 map to positions 1, 2, and 3.

Figure 2-17 (Page 2 of 2). Additional Features Bytes (Bytes 78 through 80)

Byte(s)	Bit(s)	Description
80	0	Activates Sort-control of auto-selects (SCA) option. You can override this with the Run Profile keyword NOSCA. **Notes:** 1. When using the Image Capture System, this option *must* be enabled. This allows the Sort program to increment the INF number on auto-selected documents. 2. On the UT/XP, documents involved in a hardware autoselect or late module/pocket autoselect are routed to the reject pocket with no featuring applied. After the physical documents are processed, the UT/XP generates dummy records with all digit errors for each of the documents involved in the autoselect. Therefore, if SCA is requested, the user's Sort Control program will only see these dummy records which can be marked for incrementing of the INF number, but no actual INF is printed or endorsement performed.
	1-3	Reserved.
	4	Overrides routing number (field 5) transparency. See "TRT" on page 3-6.
	5-7	Reserved.

Programmable Endorsement and INF

Bytes 81 through 95 control the Item Number feature and the Endorsement feature. These fields specify feature data that was not available in the original 3890 such as:

- 10-digit INF starting number
- INF/Endorse positions 4, 5, and 6
- Endorse data
- Use of an alternate INF number.

Figure 2-18 (Page 1 of 2). Programmable Endorsement and INF (Bytes 81 through 95)

Byte(s)	Bit(s)	Description
81	0	Initializes programmable endorsement and item-numbering feature (INF). If this bit is on, bytes 81 through 87 are used to initialize programmable endorsement and INF (bytes 1 through 10 are ignored). If this bit is off, bytes 1 through 10 are used to initialize programmable endorsement and INF (bytes 81 through 87 are ignored).
	1	Enables or disables programmable endorsement. 1 = Enables endorsement. 0 = Disables endorsement.
	2-3	Reserved.
	4-7	Print position of endorsement and INF. Valid values are 1 through 6.
82	0	Enables or disables INF. 1 = Enables INF. 0 = Disables INF. **Note:** The document processor integrates INF into the endorsement for the 3891/3892/UT XP document processors.
	1	INF number length indicator. 1 = 10-digit INF number (bytes 83 through 87). 0 = 8-digit INF number (bytes 83 through 86). **Note:** For the Image Capture System, a 10-digit INF number is recommended.
	2	Enables or inhibits the printing of an alternate item number. (For more information, see the “Alternate INF Indicator and Data” on page C-8.) 1 = Inhibits the printing of an alternate item number. 0 = Enables the printing of an alternate item number.
	3-4	Reserved.

Figure 2-18 (Page 2 of 2). Programmable Endorsement and INF (Bytes 81 through 95)		
Byte(s)	**Bit(s)**	**Description**
	5-7	Length of the INF high-order segment. Values are 0 through 7.
83-87		INF number in unsigned and packed decimal. If byte 82, bit 1 is off, the INF number is 8 digits long and occupies bytes 83 through 86. If byte 82, bit 1 is on, the INF number is 10 digits long and occupies bytes 83 through 87.
88-95		Endorse date. See byte 61 for the format.

Extended Field Record Information (Fields 9 through 15)

This data area allows you to specify code line fields 9 through 15. This field also specifies which fields can be verified during extended code line data matching.

Figure 2-19. Extended Field Record Information (Bytes 96 through 109)		
Byte(s)	**Bit(s)**	**Description**
96	0	Code line data match indicator for field 9. 1 = Perform code line data match for field 9. 0 = Do not perform code line data match for field 9.
	1-3	Reserved.
	4-7	Code line data match threshold limit for field 9.
97		Number of bytes assigned for field 9.
98-99		Field record information for field 10. (Same format as bytes 96 through 97.)
100-101		Field record information for field 11. (Same format as bytes 96 through 97.)
102-103		Field record information for field 12. (Same format as bytes 96 through 97.)
104-105		Field record information for field 13. (Same format as bytes 96 through 97.)
106-107		Field record information for field 14. (Same format as bytes 96 through 97.)
108-109		Field record information for field 15. (Same format as bytes 96 through 97.)

Image Capture System Initialization

The image capture system initialization bytes specify the detailed initialization information requested for the Image feature.

Figure 2-20. Image Capture System Initialization (Bytes 110 through 119)

Byte(s)	Bit(s)	Description
110		Indicates how the Image Capture System scans document images.
	0 1 2 3	Scan front black-and-white Scan front gray scale Scan back black-and-white Scan back gray scale **Note:** If all these bits are off (0) and byte 0, bit 2 is on (1), the Image Capture System is *disabled*.
	4	Image Capture System online
	5	Image Capture System online fill buffer
	6	Image Capture System ignore compensation errors
	7	Image Capture System ignore analysis errors
111-112		Reserved
113		Offset of DIDM key and document type
114-115		Image Capture suspect list length
116-119		Image Capture suspect list pointer (offset from beginning of program storage).

Extended Field Type Definitions

The extended field defines the data type for fields 8 through 15 of the code line. If you define the field as a digit, the Control Program considers each byte as two binary-coded-decimal (BCD) digits. Otherwise, the Control Program considers the fields as single-byte characters.

Figure 2-21 (Page 1 of 2). Extended Field Type Definitions (Bytes 120 through 127)

Byte(s)	Bit(s)	Description
120	0	For field 8: 1 = Character field; each byte is 1 character. 0 = Digit field; each byte is 2 digits.
	1	For field 9: 1 = Character field 0 = Digit field.
	2	For field 10: 1 = Character field 0 = Digit field.

Figure 2-21 (Page 2 of 2). Extended Field Type Definitions (Bytes 120 through 127)

Byte(s)	Bit(s)	Description
	3	For field 11: 1 = Character field 0 = Digit field.
	4	For field 12: 1 = Character field 0 = Digit field.
	5	For field 13: 1 = Character field 0 = Digit field.
	6	For field 14: 1 = Character field 0 = Digit field.
	7	For field 15: 1 = Character field 0 = Digit field.
121-127		Reserved.

Chapter 3. Run Profile and Related Files

This chapter describes the following:

- Run Profile description
- Run Profile keyword summary
- Run Profile keyword parameters
- Sample Run Profile
- Other initialization data files
- Machine Profile parameters
- Machine Profile SPS defaults.

Run Profile Description

The Run Profile is an ASCII keyword file on the PS/2 disk subsystem. The file extension is RPR and is used to specify additional initialization data. The Run Profile contains functions that can also be requested from IREC data. When these requests conflict, the Run Profile request is used.

When the initialization data record (IREC) does not specify a Run Profile name, the 3890/XP Enhanced document processor defaults to the Run Profile named in the Machine Profile. You specify the name of the default Run Profile in the Machine Profile.

Specifying the Run Profile name as `*NONE` in the IREC inhibits Run Profile processing.

Note: Always initialize the XP Enhanced document processor the first time after program invocation with a specified Run Profile name or a blank Run Profile name. If your initialization data specifies `*NONE` for the Run Profile name for a 3890/XP Enhanced document processor that is not previously initialized, an initialization failure occurs.

You receive several sample Run Profile files with the Install diskette for your document processor. You can edit these sample files to create your own task specific Run Profiles using the Run Profile editor in the 3890/XP Toolkit I or an ASCII editor that supports cursor return line feed (CRLF) characters at the end of each file line.

Run Profile Keyword Summary

The following table summarizes all of the valid Run Profile keywords, and the valid parameters and default settings for each keyword.

Figure 3-1. Summary of Run Profile Keywords and Valid Parameters

Keywords	Valid Parameters	Multiples	Defaults
LCL	Up to 16-character field	No	None
PEDProtect	None	No	Inactive
PGMSTOREdata	File specification, user tag, load address (optional) xxx.PSD	Yes	None
PKTDEFtbl	File specification xxx.PDT	No	Initialization data
PRGENDRSdata	File specification xxx.PED	No	No change
REGCC, NOREGCC	None	No	REGCC
RPQACTive	ZFA, NOZFA DCD, NODCD TRT, NOTRT SCA, NOSCA FILTER1F,FILTER2F, FILTER3F FILTER1B,FILTER2B, FILTER3B	No	NOZFA NODCD IREC 80, 4 IREC 80, 0 No Filters Selected
SORTMSGtbl	File specification xxx.SMT	No	Index displayed
SORTName	Sort program name to be displayed on operator screens	No	None
SYMSEQtbl	File specification xxx.SST	No	ABA.SST
SPXNDPTIME	1 to 1000 seconds	No	1
OCR3Font	OCRAN OCRBN	Yes	None
TRANStbl	File specification xxx.OTT	No	None
3890EMULation	A, B, C, D, RPQ, ORPQ, NONE	No	NONE
USERDLL	Up to 8-character name	Yes	SCIEM
CALLS	Up to 32-character name	Yes	None
SPSxxx	Entry point name *NONE *MP	Yes	*MP

File Specifications

Many of the Run Profile keywords specify a file that must conform to the OS/2 file specification rules. That is, you can:

- Specify a filename (SORT1.XXX)
- Include a drive (C:\SORT1.XXX)
- Include a path (\LEVEL1\LEVEL2\SORT1.XXX)
- Include both a drive and a path (C:\LEVEL1\LEVEL2\SORT1.XXX)

where XXX is a fixed file extension.

If the Run Profile does not specify a drive or a path, OS/2 uses the default drive and the current directory. *For all of the files related to the Run Profile the file extension is fixed, so you must not include it in the file specification*.

File Descriptions

The following rules apply to the files described in this section:

- Columns 1 through 100 of each line or record are used for the keyword and its parameters. (Keywords and parameters are predefined terms used to identify specific initialization requests.) You must separate keywords and parameters by at least one blank; blanks are not permitted within keywords or parameters. Keywords and parameters must be in uppercase, with the exceptions listed in the following sections.

  The format of the keyword record is:

  KEYWORD parm1 parm2 ...

  If you include a keyword without any parameters, the Run Profile assumes the default for that keyword.

- The Run Profile permits no more than one keyword per line or record.
- You must specify USERDLL keywords before the associated CALLS and SPS keywords. You can enter other keyword records in any sequence.
- An asterisk in column 1 indicates that this record is a comment. The asterisk must be in column 1. If an asterisk appears in column 1, the remainder of the line is ignored.
- The Run Profile permits abbreviations of keywords. Required characters are shown in uppercase. Optional characters are shown in lowercase, but, if entered, must be entered as uppercase. Numeric characters must be included in the keyword.
- The following SPSxxx keywords can appear more than once:

  PGMSTOREdata
  OCR3Font
  USERDLL
  CALLS

  Multiple occurrences of other keywords result in a run initialization failure.

- Each record in the file must end with CR/LF (carriage return/line feed). This is supplied by ASCII editors such as the OS/2 “E” editor.

Run Profile Keyword Parameters

This section discusses the Run Profile keyword parameters. Note that all keywords are optional.

LCL

Level Control Label (LCL) specifies a user-defined string that is moved to the LCL table in AUX storage during initialization. LCL support is available for all ASCII data files specified in the Run Profile. Applications can use the LCL fields to verify that the correct level of support files are being used.

For example, assume that a date and time stamp are used as the LCL for all ASCII data files used for an initialization. A Sort program sequence initialization routine could read the contents of the LCL table and fail initialization if it detects an old file; this displays a message to the operator indicating which file is required.

The format of the LCL keyword is:

LCL string

where the string can be 16 characters.

If the string contains blanks, you must precede it and follow it with the single quote character (ASCII 39 or hex 27).

If the string is omitted, a string of 16 blanks is saved for that LCL.

PEDProtect

Programmable Endorsement Data Protect. The PEDProtect keyword blocks both the Endorse Setup/Verify screen and operator use of the ENDorse operator command. This keyword has no parameters and is optional in the Run Profile. When this keyword appears in the Run Profile, it indicates that the endorsements and the data are maintained automatically and that the operator should not have access to these facilities. If you attempt to issue the endorse command, the display screen indicates that the keyword is protecting the endorsement data. When this keyword does not appear in the Run Profile, the 3890/XP operator can alter the endorsements and the data.

PGMSTOREdata (PSD File)

Program Storage Data (PSD) File Specification. The PGMSTOREdata keyword specifies a file that contains program storage data. If the Run Profile does not specify the PGMSTOREdata keyword, program storage is not affected.

The 3890/XP Enhanced Control Program supplies the reserved file extension .PSD; do not include it in your file specification.

The format of the PGMSTOREdata keyword is:

PGMSTOREdata filespec tag [load address]

where tag is a required user tag, and load address is optional.

filespec

This is a file that contains one or more of the following:

- User-supplied Sort program (with or without an IREC)
- SCI subroutine
- Table data.

The Control Program loads program storage data, one file specification per keyword, in the sequence in which the keywords appear in the Run Profile. You can use multiple PGMSTOREdata keyword records.

tag

The user tag can be up to 8 characters long, in uppercase, and is delimited by blanks. If a user tag is not required, you must insert the characters `NULL` as a parameter before you can specify the load address.

load address

The load address is in hexadecimal and specifies where the data is to be loaded into program storage. If the data exceeds available program storage, a run initialization failure occurs. The load address is optional; however, if it is specified, then the user tag is required.

If you do not specify a load address, the data is loaded on the next word boundary following the IREC, SCI program, and data from previous PGMSTOREdata files, if any.

See the SpxGetRPERec function in the *3890/XP Series and 3890/XP Enhanced SPXServ Reference*.

PKTDEFtbl (PDT File)

Pocket Definition Table (PDT) File Specification. The PKTDEFtbl keyword specifies the file that contains the pocket definition table (see page 3-14).

The format of the PKTDEFtbl keyword is:

PKTDEFtbl filespec

The Control Program supplies the reserved file extension .PDT; do not include it in your file specification.

If you specify a PDT file, the Run Profile overrides initialization data bytes 37 through 54 and 56 through 59. If you omit this keyword, the initialization data defines the pocket count processing.

PRGENDRSdata (PED File)

Programmable Endorsement Data (PED) File Specification. The PRGENDRSdata keyword specifies a file that contains your endorsement text. See “Programmable Endorsement Data (PED) File” on page 3-15 for Endorse data file specifications. This data is written to the Active Endorsement Area in AUX storage.

The format of the PRGENDRSdata keyword is:

PRGENDRSdata filespec

The Control Program supplies the reserved file extension .PED; do not include it in your file specification.

If you omit the PRGENDRSdata keyword, the document processor uses the “last PED” instead. If the PEDProtect keyword is not in effect, the document processor displays the Endorse Setup/Verify screen to the operator as soon as initialization completes and you have requested endorse.

The operator can also specify a file as part of the operator ENDorse command. See Figure 7-12 on page 7-14.

REGCC

The REGCC keyword causes the endorsement to print in a position on the back of the document that meets the American National Standards Institute (ANSI) requirement on endorsement position. There are no parameters entered with this keyword.

The XP Enhanced document processors default to REGCC.

NOREGCC

The NOREGCC keyword prints the endorsement in the default position established on the 3891/XP and 3892/XP operator panels. There are no parameters entered with this keyword.

RPQACTive

Request for Price Quotations (RPQs) Active. The RPQACTive keyword activates the 3890 RPQs for the XP Enhanced document processor. The following parameters (which can be in any sequence) are possible:

ZFA **(Zero-for-Able)**
This parameter specifies that the code line data match zero-for-able (ZFA) is active. The document processor (module SCII) forces a match on any digit where the code line data match record contains a zero and the process buffer contains a hexadecimal value of `X'A'`.

NOZFA **(No-Zero-for-Able)**
This parameter specifies that the code line data match zero-for-able (ZFA) is not active.

DCD **(Don’t-Care Digit)**
This parameter specifies that the code line data match don’t-care-digit (DCD) is active. The document processor (module SCII) forces a match on any digit where the host program has placed a hexadecimal value of `X'C'` in the code line data match record.

NODCD **(No-Don’t-Care Digit)**
This parameter specifies that the code line data match don’t-care-digit (DCD) is not active.

Note: ZFA and DCD do not apply for a field if character, rather than digit, comparison is requested. See initialization data byte 120.

TRT **(Routing-Number-Transparency Mode)**
This parameter specifies that the routing-number-transparency mode (formerly transit-routing transparency mode) is active.

The routing number field (field 5) has two formats. The *first format* is four digits, one dash, and four digits. The *second format* is nine digits with the digit to the extreme right being a check digit.

If you request TRT, the document processor (module SCIF) checks the format of the routing number field. If the first format is detected, the field is left unchanged. If the second format is detected, the eight digits to the left are verified against the check digit on the right (modules 10). If the check digit is not valid, the routing-number-not-verified bit is set in the document data record header (byte 9, bit 4). Regardless of check digit validity, the check digit is placed in the document data record header (byte 9, bits 0 through 3) and the field is converted to the first format of four digits, one dash, four digits.

Initialization data byte 80, bit 4 overrides the TRT parameter.

NOTRT (No-Routing-Number-Transparency Mode)
This parameter specifies that the routing number transparency mode is not active. The NOTRT parameter causes initialization data byte 80, bit 4 to be ignored.

SCA (Sort-Control of Auto-Selects)
This parameter specifies that the Sort program controls auto-select documents.

NOSCA (No Sort-Control of Auto-Selects)
This parameter indicates that the Sort program does not control auto-select documents.

Note: If you do not specify SCA or NOSCA in the Run Profile, the initialization data byte 80, bit 0, controls SCA. See also "3890/XP Enhanced Auto-Select Actions" on page 5-12.

FILTER1F (Image Front Side Filter 1)
This parameter requests the front side image color drop out filter number one.

FILTER2F (Image Front Side Filter 2)
This parameter requests the front side image color drop out filter number two.

FILTER3F (Image Front Side Filter 3)
This parameter requests the front side image color drop out filter number three.

FILTER1B (Image Back Side Filter 1)
This parameter requests the back side image color drop out filter number one.

FILTER2B (Image Back Side Filter 2)
This parameter requests the back side image color drop out filter number two.

FILTER3B (Image Back Side Filter 3)
This parameter requests the back side image color drop out filter number three.

Note: To determine which filter has been requested during initialization, you can write native code to execute at the initialization event. You can call the SpxGetFilter function to return the current filter(s) requested. For more information, see the *3890/XP Series and 3890/XP Enhanced SPXServ Reference*.

SORTMSGtbl (SMT File)

Sort Message Table (SMT) File Specification. The SORTMSGtbl keyword specifies the file that contains the customer's Sort message table (see page 3-16).

The format of the SORTMSGtbl keyword is:

SORTMSGtbl filespec

The Control Program supplies the reserved file extension .SMT; do not include it in your file specification. If you do not specify a file, or if you only specify a subset of the total possible messages, the Control Program uses a default message.

SORTName

Sort program name. The SORTName keyword specifies the text that appears in the Sort program name field of the Run and Active operator display screens. This becomes the title of the Sort run.

The format of the SORTName keyword is:

SORTName name

It consists of up to 30 left-aligned characters. If you enter a name that is less than 30 characters, it is padded on the right with blanks. You can request right alignment with leading blanks and enclosing the blanks and the name in single-quotation marks. You can request a single quote to appear in a name by immediately preceding it or following it with another single quote.

SYMSEQtbl (SST File)

Special Symbol Table (SST) File Specification. The SYMSEQtbl keyword specifies the file that contains the special symbol table (see page 3-17).

The format of the SYMSEQtbl keyword is:

SYMSEQtbl filespec

The Control Program supplies the reserved file extension .SST; do not include it in your file specification in the Run Profile. If you do not specify an SST file, the default is the ABA table (see page 3-17).

SPXNDPTIME

Non-document processing time is expressed as time out in seconds. The SPXNDPTIME keyword applies to the following non-document processing events:

- Initialize event
- Motors-stopped event
- Operator event.
- Pause event.

The format of the SPXNDPTIME keyword is:

SPXNDPTIME time

where **time** is a decimal value from 1 to 1000 (default value = 1).

OCR3Font

Optical Character Recognition feature (OCR3) Font Type. See “Optical Character Recognition” on page 10-3 for the font description.

TRANStbl (OTT File)

Translate Output Translation Table (OTT) File. The TRANStbl keyword specifies the file that contains the output translation table for OCR3.

The format of the TRANStbl keyword is:

TRANStbl filespec

The Control Program supplies the reserved file extension OTT; do not include it in your Run Profile file specification.

3890EMULation

The 3890EMULation keyword is an optional keyword that you specify to request 3890 emulation.

If you specify this keyword, the total process buffer read record length must be equal to, or less than, 48 bytes.

The format of the 3890EMULation keyword is:

3890EMULation parm

where **parm** is one of seven possible parameters.

The seven possible parameters are A, B, C, D, RPQ, ORPQ, and NONE which represent the following machines:

- Model A is a 16KB MICR machine
- Model B is a 32KB MICR machine
- Model C is a 16KB OCR machine
- Model D is a 32KB OCR machine
- RPQ is a 48KB MICR machine
- ORPQ is a 48KB OCR machine.

The parameters are mutually exclusive; you can specify only one. If you omit this keyword or specify NONE, the initialization does not request 3890 emulation. (Models E and F are slower versions of A and B. For Model E or F emulation, specify A or B, respectively.)

The 3890EMULation keyword affects the following features:

Process buffer location:

The 3890 process buffer is in program storage, but the process buffer for the XP Enhanced document processor is in additional AUX storage. If you emulate Model A, B, C, D, RPQ, or ORPQ, and the user-supplied Sort program tries to access the process buffer using LPS or SPS, the 3890/XP Enhanced document processor remaps this access to occur at the process buffer address in additional AUX storage. If you specify NONE, process buffer remapping is inactive.

Program storage management:

If you specify 3890 Model A, B, C, D, RPQ, or ORPQ emulation, the program storage above the SCI program that is loaded to address hex FFFF is set to binary zeros at run initialization. If you specify or default to NONE, no part of program storage is set to zeros at run initialization. For more information about program storage, see “Program Storage” on page C-5.

Setting X'0016' in 3890 emulated AUX storage: .

The Control Program uses this keyword to set the hexadecimal value of the highest program storage address plus one (of the emulated model) at address X'0016'. See “3890-Emulated Auxiliary Storage” on page C-7.

Document Data Record Length:

If you specify A, B, C, D, RPQ, or ORPQ with the 3890EMULation keyword, the total length of fields 0 through 8 must not be greater than 48 bytes. For more information about field lengths, see “Process Buffer Fields” on page 5-9.

Input buffer:

If 3890 C, D, or ORPQ emulation mode is specified, the Control Program moves the OCR-3 data to the MICR input buffer by default.

USERDLL

The USERDLL keyword specifies dynamic link library (DLL) files on the XP Enhanced disk subsystem for OS/2 language routines for SPS or CALLS entry points.

The format of this Run Profile command is:

USERDLL filename

where **filename** is the name of a DLL file.

The Control Program supplies the file extension .DLL; do not include it in your Run Profile specification. DLLs must be in a path specified in the LIBPATH keyword of the configuration file, CONFIG.SYS.

You can use up to 10 USERDLL keywords including the SCIEM.DLL by default.

CALLS

The CALLS keyword specifies entry point names that you use in calling OS/2 language modules via the SCI “CALLNATV” macro.

You can specify up to 256 entry points with the CALLS keyword. If you specify an entry point with an SPS keyword, you do not need a CALLS keyword for that same entry point. The entry point must exist in a DLL specified with a USERDLL keyword.

The format of the CALLS keyword is:

CALLS entrypoint

where **entrypoint** is a name of up to 32 characters.

SPSxxx

Sort Program Sequence Keyword Family. There are 10 Sort Program Sequence (SPSxxx) keywords. The SPSxxx keywords specify the calling sequence for Sort modules at the various SPS events. When key events occur, the Control Program runs an ordered list of modules, as specified by SPSxxx keywords. See Figure 3-2.

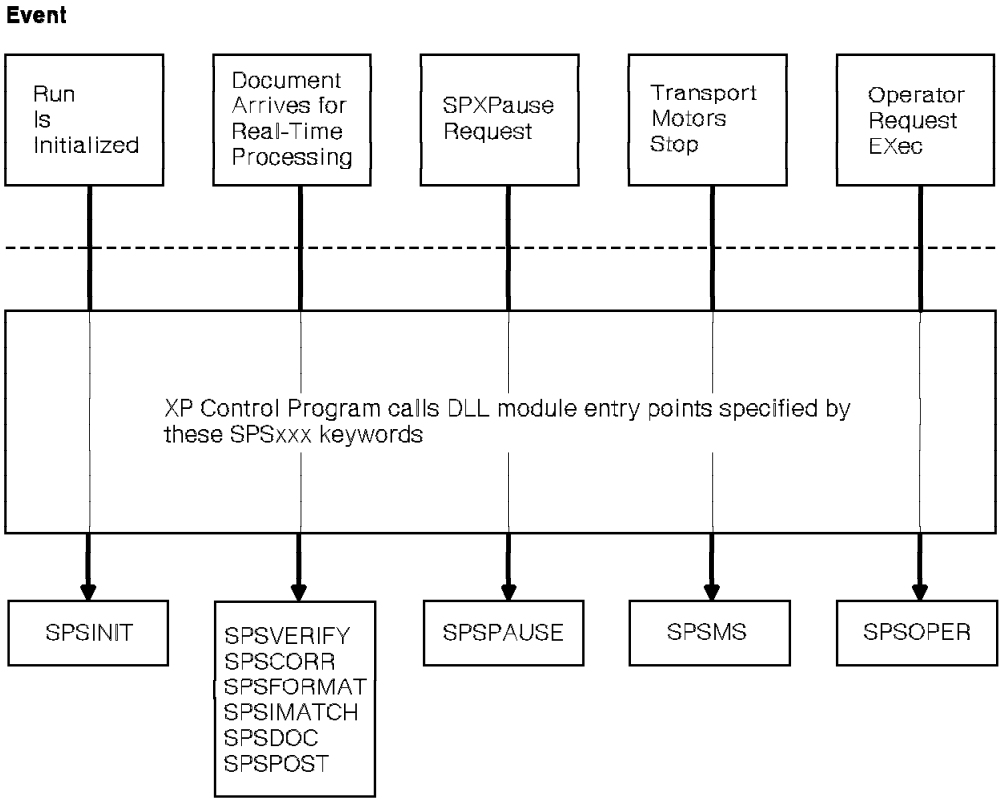

Figure 3-2. SPSxxx Keyword Usage

The format of the SPSxxx keyword is:

SPSxxx entry point-1 entry point-2 ... entry point-n

Each SPSxxx keyword can be specified multiple times. You are allowed a maximum of 32 entry points for each SPS event.

Each entry point must be one of the following:

- Entry point name of a Sort module routine in a DLL file named either in a USERDLL keyword or in the Machine Profile.
- *MP (Machine Profile) - Entry point as specified in the Machine Profile for this SPS event.

- *NONE - No module will be called for this SPS event. This must be the first and only “entry point.”
- (Blank) - Same as *MP.

These are the SPSxxx keywords for the *non-document SPS events*.

SPSPAUSE
SPSINIT
SPSMS
SPSOPER

These are the SPSxxx keywords for the *document-time SPS events*.

SPSVERIFY
SPSCORR
SPSFORMAT
SPSIMATCH
SPSDOC
SPSPOST

The Machine Profile can specify a DLL file that is loaded for every initialization. This is initialized to SCIEM.DLL in the default Machine Profile that comes with the Install diskette.

The installed Machine Profile also specifies the IBM-supplied entry point names for the document-time SPS events required for complete Sort processing. See “Machine Profile SPS Defaults” on page 3-21.

Document-time processing is discussed at greater length in Chapter 5.

The execution sequence of a Sort “program” during document-time is as follows:

1. Every module for SPSVERIFY, in order according to the Run Profile
2. Every module for SPSCORR, in order according to the Run Profile
3. Every module for SPSFORMAT, in order according to the Run Profile
4. Every module for SPSIMATCH, in order according to the Run Profile
5. Every module for SPSDOC, in order according to the Run Profile
6. Every module for SPSPOST, in order according to the Run Profile.

You might supplement the IBM-supplied modules by specifying your own additional module or modules, typically for the SPSDOC event. Entrypoints must be specified in the order in which you want them to run.

Sample Run Profile

```
* SAMPLE RUN PROFILE FOR 3890/XP MODEL A EMULATION
*
* ALL OTHER KEYWORDS ARE COMMENTED OUT, BUT SHOWN AT THEIR DEFAULT STATE.
*
  3890EMULATION A
  SORTNAME 3890 MODEL A EMULATION
*
* RPQACTIVE NOZFA NODCD NOTRT
* SPXNDPTIME 1
* PRGENDRSDATA                * INSERT FILESPEC FOR ENDORSE DATA FILE *
* PKTDEFTBL     * INSERT YOUR FILESPEC FOR POCKET DEFINITION TABLE    *
* LCL    * UP TO 16 CHARACTERS FOR RUN PROFILE LEVEL CONTROL LABEL    *
* SYMSEQTBL C:\INIT\ABA
* SORTMSGTBL C:\INIT\DEFAULT
* REGCC
* PGMSTOREDATA             * ENTER FILESPEC -  TAG - OPTIONAL ADDRESS *
* USERDLL                  * ENTER YOUR OS/2 USER DLL FILE NAME       *
* CALLS                    * ENTER YOUR OS/2 USER ENTRY POINT NAME    *
  USERDLL SCIEM
* SPSINIT *MP
* SPSMS *MP
* SPSOPER *MP
* SPSVERIFY *MP
* SPSCORR *MP
* SPSFORMAT *MP
* SPSIMATCH *MP
* SPSDOC *MP
* SPSPOST *MP
* SPSPAUSE *MP
* USERDLL CONVERGE  *REQUIRED FOR ENDORSE
* SPSPOST CONVERGE  * ON 3891, 3892/XP
```

Other Initialization Data Files

This section includes descriptions and layouts of the various data files used in the 3890/XP Enhanced document processor. These are keyword-oriented and record-oriented files that you can create or edit using the Run Profile editor in the 3890/XP Toolkit I or an ASCII editor.

Pocket Definition Table (PDT) File

The Pocket Definition Table (PDT) file on the disk is an ASCII file and must have a file extension of .PDT. When the Run Profile specifies this file, run initialization ignores IREC bytes 37 through 54 (pocket identification) and bytes 56 through 59 (pocket limit values). The pocket definition table contains a decimal value between 0 and 4095 to indicate the count at which a merge document will automatically be fed to that pocket. The actual number of documents in the pocket can be more than the number specified in the file. The extra documents are the ones that were already in the transport heading for that pocket before the merge document was fed.

The valid keywords for the PDT file are:

MODULE1 Pkt 1, Pkt 2, Pkt 3, Pkt 4, Pkt 5, Pkt 6. Values are decimal counts.

MODULE2 Pkt 1, Pkt 2, Pkt 3, Pkt 4, Pkt 5, Pkt 6. Values are decimal counts.

MODULE3 Pkt 1, Pkt 2, Pkt 3, Pkt 4, Pkt 5, Pkt 6. Values are decimal counts.

MODULE4 Pkt 1, Pkt 2, Pkt 3, Pkt 4, Pkt 5, Pkt 6. Values are decimal counts.

MODULE5 Pkt 1, Pkt 2, Pkt 3, Pkt 4, Pkt 5, Pkt 6. Values are decimal counts.

MODULE6 Pkt 1, Pkt 2, Pkt 3, Pkt 4, Pkt 5, Pkt 6. Values are decimal counts.

LCL Optional keyword that consists of a user-defined character string that is moved to the level control label (LCL) table during initialization.

The following example shows a sample PDT file.

```
MODULE1  50  50  50  50  50  50
MODULE2  50  50  50  50  50  50
MODULE3  50  50  50  50  50  50
MODULE4 100 100 100 100 100 100
MODULE5 100 100 100 100 100 100
MODULE6 100 100 100 100 100 100
```

Each keyword has merge limit counts for the pockets in the specified stacker module. If six counts are not specified, a run initialization error will occur. Each count can range from 0 to 4095. The PDT file sequences the data so that the count for pocket one is the first value, the count for pocket two is the second value, and so on. A minimum of one blank should separate the counts. A value of zero means that the XP Enhanced document processor does not automatically feed a merge document for that pocket. A non-zero value represents the count at which the XP Enhanced document processor automatically feeds a merge document for that pocket. Initialization data byte 21, bit 1 must be set.

If a given stacker module is not defined in the PDT file, the pockets in that module assume counts of zero. Therefore, no merge documents will automatically be fed for any pocket in that stacker module. Multiple records for the same stacker module cause a run initialization failure.

Programmable Endorsement Data (PED) File

The Programmable Endorsement Data (PED) file is an ASCII file and must have a file extension of .PED. The PED file specifies the static endorsement text to be used if endorsing data is not manipulated by the Sort program.

The PED file contains the following keywords:

1. ENDorse
2. LCL

ENDorse

The PED file permits three ENDorse keyword records, each with up to 24 characters of endorsement text. If the file omits the ENDorse keywords, the endorsement is set to blanks. The first ENDorse keyword represents the first line of the endorsement text, the second ENDorse keyword represents the second line of the endorsement, and the third ENDorse keyword represents the third line of the endorsement. Specifying more than three ENDorse keywords causes a run initialization error.

If you enter fewer than 24 characters, the Control Program assumes that the text is left-aligned and pads each line to the right with blanks. You can force right alignment with leading blanks and enclosing the blanks and text in single-quotation marks. You can request a single quote to appear within the text by immediately preceding it or following it with another single quote.

To request a double-wide (DW) character, use a plus sign (+) as the special character. Follow the plus sign with the character that you want to be a double-wide character. The following can be double-wide characters.

- A - Z
- 0 - 9
- <, >

For the 3891/XP and 3892/XP document processors, double-wide characters print as character-underscore.

To print the date, use a percent sign (%) followed by up to seven Ds. This prints as an 8-character date. For further information, see “Control of the Endorse Date” on page 7-13.

The following example shows a sample PDT file.

```
ENDORSE   NATIONAL BANK
ENDORSE   012345  +P+A+I+D
ENDORSE   %DDDDDDD
```

LCL

Optional keyword that consists of a user-defined character string that is moved to the level control label (LCL) table during initialization.

Sort Message Table (SMT) File

The Sort Message Table (SMT) file is an ASCII file and must have a file extension of .SMT. The SMT file is a facility for the sort programmer to specify up to 255 unique messages. These messages are displayed on the Run screen.

The XP Enhanced document processor loads a default message table into storage before loading any user table. You can change the text and the display attributes of any of the default messages.

The valid keywords for the SMT file are:

nnn Where *nnn* is a decimal number from 1 to 255 that indicates the message ID of the operator message. The message ID corresponds to the parameter used in the SCI "SOM" (sort operator message) macro or the SpxPutOpMsgNum function. The SMT file permits leading blanks when the number is fewer than three digits. The SMT file pads the number on the left with zeros for numbers with fewer than three digits. The valid parameters are:

Message severity
Contains one of three codes to designate the severity level and color of the message.

This field consists of a single character as follows:

I Informational (displayed as white)
W Warning (displayed as yellow)
E Error (displayed as red).

Message text
Contains up to 80 characters that define the message. The characters can be entered in uppercase and lowercase. If you enter fewer than 80 characters, the SMT file assumes the message text to be left-aligned and pads the message to the right with blanks. You can request right-aligned with leading blanks and enclosing the blanks and message text in single-quotation marks. You can request a single quote to appear within the message text by immediately preceding it or following it with another single quote.

LCL Optional keyword that consists of a user-defined character string that is moved to the level control label (LCL) table.

The following example shows a sample default SMT file.

```
001 W Hex = 01 Binary = 0000 0001   Default Sort Operator Message
    E  "
nnn E          Customer-Supplied Message Text.
    I  "
255 W Hex = FF Binary = 1111 1111   Default Sort Operator Message
```

If the customer did not supply a message text for *nnn* = 255, the display screen displays the following message (in white).

```
255  Hex = FF Binary = 1111 1111    Default Sort Operator Message
```

Note: The hexadecimal and binary numbers are defaults in the supplied text. The decimal message number is added to the display by the Control Program.

Special Symbol Sequence Table (SST) File

The Special Symbol Sequence Table (SST) file is an ASCII file and must have a file extension of .SST. The purpose of the SST file is to define how the document code line breaks into fields.

You might wish to refer to the ABA table below and the "typical document layout" Figure 1-1 on page 1-4.

The ABA SST table defines the following, for fields 1 through 7.

```
                1   2   3   4   5   6   7

OPEN      SS1 SS1 SS2 SS2 SS3 SS3 SS2
CLOSE     SS1 SS2 SS3 SS3 SS3 SS2 SS2
ALTERNATE 0   0   SS2 0   0   0   0
TRANSMIT  SS4
```

You can define up to 7 fields in the Symbol Sequence Table.

You can request the document processor to close any field with the special symbol (SS) that opens the next field. If a CLOSE special symbol ends a field that could be followed by an optional field, then the optional field is not on the document.

For example, if the account number (field 3) in Figure 1-1 on page 1-4 is closed by the SS3 that opens the routing number (field 5), then optional field 4 is not on that document.

If an ALTERNATE special symbol ends such a field, then the optional field is present (and might or might not have data).

The SST file specifies the CLOSE and ALTERNATE symbols used to set the field-valid flags in byte 1 of the document data record header. In other words, if the first symbol that is read after (to the left of) the third symbol, for example, does not match the third CLOSE symbol or the third ALTERNATE symbol, the XP Enhanced document processor turns on byte 1, bit 3 of the document data record header to indicate an error (length not valid or SS error). See "Document Data Record Header - X'80'" on page 4-2.

Based on the number of fields in your code line, you can have up to seven parameters per keyword. The data is sequenced so that the SS for field 1 is the first value, the SS for field 2 is the second value, and so on. A zero serves as a place holder for the fields that do not need special symbols.

The valid keywords for the special symbol sequence table are:

OPEN The opening symbols.

CLOSE The closing symbols.

ALTERNATE The alternate closing symbols (that open optional fields).

TRANSMIT The symbols transmitted as data in the process buffer.

LCL Optional keyword that consists of a user-defined character string that is moved to the LCL table during initialization.

The following are the valid parameters for the OPEN, CLOSE, and ALTERNATE keywords.

- 0
- SS1
- SS2
- SS3
- SS4
- SS5

The following are the valid parameters for the TRANSMIT keyword.

- SS1
- SS2
- SS3
- SS4
- SS5

Note: These parameters do not need to be in sequence.

The symbols that you specify for TRANSMIT cannot be specified for OPEN, CLOSE, or ALTERNATE. If the symbols do appear, a run initialization error results.

The SST file must contain OPEN and CLOSE keywords; ALTERNATE and TRANSMIT keywords are optional.

- If you do not specify the ALTERNATE keyword, the XP Enhanced document processor assumes there are no alternate symbols.
- If you do not specify the TRANSMIT keyword, the document processor assumes that no symbols are transmitted as data in the process buffer.

Multiple records for the same keyword generate a run initialization error.

Machine Profile Parameters

You update the Machine Profile through the Machine Profile screens on the PS/2. You display the Machine Profile screens by choosing Machine Profile from the Run screen Configure menu. The data maintained in the Machine Profile is data that does not change often.

For additional information, see the *3890/XP Document Processor Operator's Guide*, *3890/XP Enhanced Document Processor Operator's Guide*, and the *3891/XP and 3892/XP Document Processors Operator's Guide*.

Figure 3-3 (Page 1 of 2). Machine Profile Parameters and Format

Parameter	**Format**
Channel address	Two characters (3890/XP Enhanced).
Channel priority	Two characters (3890/XP Enhanced).
Run Profile default name	Up to eight characters.
Run Profile drive/path	OS/2 drive/path. Up to 63 characters.
Default Sort program drive/path	OS/2 drive/path. Up to 63 characters.
Microfilm space count	Decimal 40 960 by default.
Exit to OS/2 password	Up to 8 characters.
Customization password	Up to 8 characters.
OS/2 password limit	Decimal 0 through 9; indicates number of attempts allowed to enter the OS/2 password correctly.
Customization password limit	Decimal 0 through 9; indicates number of attempts allowed to enter the customization password correctly.
Function key names	Up to 8 characters.
Function key commands	Up to 50 characters.
Pocket names	Up to 3 characters.
Sort message table default name	Up to 8 characters.
Sort message table drive/path	OS/2 drive/path. Up to 51 characters.
Symbol sequence table default name	Up to 8 characters.
Symbol sequence table drive/path	OS/2 drive/path. Up to 51 characters.
USERDLL file name	Up to 8 characters.

Figure 3-3 (Page 2 of 2). Machine Profile Parameters and Format	
Parameter	**Format**
Sort program sequence	Ten 32-character entry-point names.
Program store size requested	Number of segments. One segment equals 1 block (64 kb) of memory.

Machine Profile SPS Defaults

The default Machine Profile for the 3890/XP Enhanced document processors specifies the SCI emulation modules that implement the SCI Sort program language. The SCI emulation modules reside in the SCIEM.DLL library and include the following:

for SPSINIT	*NONE
for SPSMS	*NONE
for SPSOPER	*NONE
for SPSVERIFY	SCIV
for SPSCORR	SCIC
for SPSFORMAT	SCIF
for SPSIMATCH	SCII
for SPSDOC	SCID
for SPSPOST	*NONE
for SPSPAUSE	*NONE

For the entries shown, *NONE indicates that NO routine will be called for the event shown on the left (for example, SPSINIT is the initialize event). The entries that show SCIx indicate which IBM-supplied SCI emulation module is called for the event shown on the left (for example, SCIV is the routine that is called for the verify function).

Note: For the 3890/XP Enhanced, the modules in the SCIEM.DLL library are now 32-bit modules and have been renamed. The original modules are also included so that if you are already using them, you do not need to change your program to use these new names. Since the 32-bit versions run faster, you should use them when possible. These 32-bit versions are the defaults in the Machine Profile when the 3890/XP Enhanced Control Program is installed. The modules in the SCIEM.DLL library for a 3890/XP Enhanced are:

- SCIV and SCIV32
- SCIC and SCIC32
- SCIF and SCIF32
- SCII and SCII32
- SCID and SCID32

Note: For all predictor counters to be update, the SCIx modules should be specified.

Run Profiles for specific jobs can specify supplementary or alternate modules by using an SPSxxx keyword with the Machine Profile *MP token or with the actual entry points. The Run Profile overrides the Machine Profile unless it defaults to the Machine Profile as in Figure 3-4. For additional information, see “SPSxxx” on page 3-11.

Figure 3-4. Run Profile and Machine Profile Procedure for Initialize Event. Other SPSxxx keywords are similar.

Run Profile Specification	**Machine Profile Specification**	**Resulting Action**
SPSINIT ABC	Anything	ABC
SPSINIT *NONE	Anything	No module called for this event
SPSINIT *MP or SPSINIT or No SPSINIT keyword	XYZ	XYZ
SPSINIT *MP or SPSINIT or No SPSINIT keyword	*NONE or not specified	No action
SPSINIT ABC *MP GHI	XYZ	ABC, then XYZ, then GHI

Chapter 4. Record Headers

This chapter describes the following:

- Record header overview
- Document data record header
- Exception record header
- SCI error record header.

Record Header Overview

Each time the 3890/XP Enhanced document processors read a physical document, they generate a record consisting of the document data and a record header. The record header gives information about the document as well as a method for controlling document pocketing and features. Because the record header is in the process buffer, it is sometimes referred to as the *process buffer header*. It is also sometimes referred to as the *read record*.

The 3890/XP Enhanced document processors produce three types of record headers:

- Document data record header
- Exception record header
- SCI error record header.

The record header (field 0) is fixed at 12 bytes, 0 through B. Bytes 0 through A contain information about the record. Byte B identifies the record header type. During document-time processing, byte B is 0.

If the XP Enhanced document processor is online to the host, the record header information is part of the read record transmitted to the host.

Some of the record header information is created during pre-Sort processing. The Sort program also creates record header information when it runs. Information from post-Sort processing also becomes part of the record header. See Figure 4-1 on page 4-2.

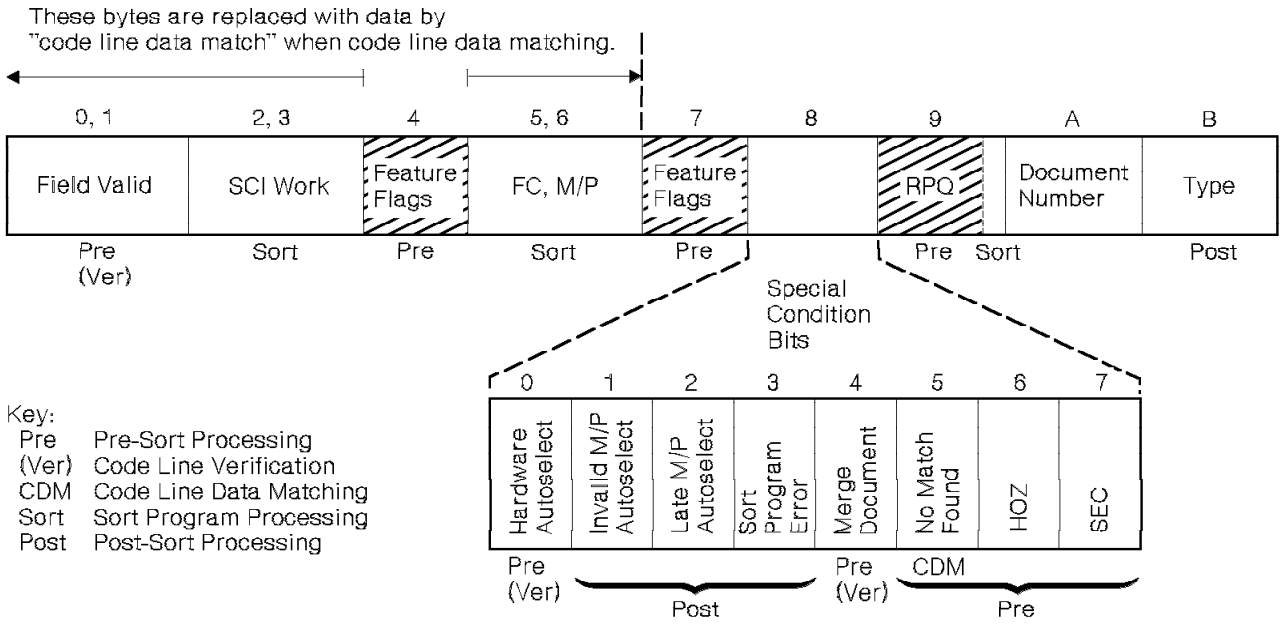

Figure 4-1. Record Header Format

Document Data Record Header - X'80'

The following tables show the bit-specific detail for the document data record header (80).

Record Header Bytes 0 and 1

Figure 4-2 (Page 1 of 2). Record Header Bytes 0 and 1

Byte(s)	Bit(s)	Format
0	0	Code line data match end-of-file (end of host code line data match records).
	1 2 3 4 5 6 7	Field 1 digit error. Field 2 digit error. Field 3 digit error. Field 4 digit error. Field 5 digit error. Field 6 digit error. Field 7 digit error. When any of bits 1 through 7 are on, at least one digit error has occurred in the associated field.
1	0	Forced dummy S1. When this bit is on, it indicates that the opening symbol for field 1 was not on the document or the leading edge of the document was damaged. As a result, the first correct digit forced the field open.

Figure 4-2 (Page 2 of 2). Record Header Bytes 0 and 1

Byte(s)	Bit(s)	Format
	1 2 3 4 5 6 7	Field 1 length-not-valid or SS error. Field 2 length-not-valid or SS error. Field 3 length-not-valid or SS error. Field 4 length-not-valid or SS error. Field 5 length-not-valid or SS error. Field 6 length-not-valid or SS error. Field 7 length-not-valid or SS error. A length-not-valid or SS-sequence error indicates that the opening or closing symbol for a field was missing or not correct or that the number of digits detected in that field was not correct. See "Special Symbol Sequence Table (SST) File" on page 3-17. A length-not-valid error in a specified fixed-length field indicates that the system unit read too many or too few 4-bit hexadecimal characters. A length-not-valid error in a specified variable-length field indicates that the number of 4-bit characters read was more than the maximum number specified in the initialization data record (IREC). **Note:** If the initialization data initializes any field not-to-be-read (IREC bytes 22 through 35, second byte, bits 4 through 7 are zeros), the corresponding length-not-valid bit will be set on to indicate whether the field appears on the document.

Record Header Bytes 2 and 3

Figure 4-3. Record Header Bytes 2 and 3

Byte(s)	Bit(s)	Format
2	0-7	SCI work byte.
3	0-7	SCI work byte. **Note:** Bytes 2 and 3 are the Sort-to-host communication area.

Record Header Byte 4

Figure 4-4 (Page 1 of 2). Record Header Byte 4

Byte(s)	Bit(s)	Format
4	0	Endorsement data not valid. If this bit is on in the header as read by the host, this document had invalid endorsement data. If this bit is on in the header that is presented to the user-supplied SPSDOC Sort program, it is the preceding document that had invalid endorsement data.

Figure 4-4 (Page 2 of 2). Record Header Byte 4		
Byte(s)	**Bit(s)**	**Format**
	1	INF data not valid. If this bit is on in the header as read by the host, this document had invalid user-supplied INF data. If this bit is on in the header that is presented to the user-supplied SPSDOC Sort program, it is the preceding document that had invalid INF data.
	2	Power Encoder verify error.
	3	OS/2 language Sort program time-out error. If this bit is on in the header as read by the host, this document (that is, the document that the user-supplied SPSDOC Sort program was processing when time ran out) was auto selected. If this bit is on in the header that is presented to the SPSDOC Sort program, the preceding document was auto selected.
	4	Image requested for this document. If this bit is on (1), it indicates that a request to capture the document was sent to the Image Capture System. It does not indicate that the image was captured.
	5	OCR3 feature initialized “On.”
	6-7	Reserved.

Feature Control Byte

Figure 4-5 (Page 1 of 2). Feature Control Byte		
Byte(s)	**Bit(s)**	**Format**
5	0	Microfilm space. When this bit is on (1), it indicates that a preset count of 40 960 documents has been microfilmed (approximately 65.53 m [215 ft] of film) and a pause occurs while the film is spaced (152.4 mm [6 in.] or 1244.6 mm [49 in.]). The Sort program can turn this bit on. You can change this count on the 3890/XP Machine Profile Parameters screen or the 3890/XP Enhanced Machine Profile screen.
	1	Microfilm flash. When this bit is on, the microfilm flashlamp will come on when the document is in the correct position.
	2	Increment microfilm high-order segment. When this bit is on, one is added to the value of the high-order segment of the index number, and the low-order segment resets to zero. This occurs before the document is recorded on film.

Figure 4-5 (Page 2 of 2). Feature Control Byte		
Byte(s)	**Bit(s)**	**Format**
	3	Increment microfilm low-order segment. When this bit is on, one is added to the low-order segment of the index number. This occurs before the document is recorded on film. When the low-order segment is all nines and increases by one, it becomes all zeros. This does not affect the high-order segment. **Note:** If bits 2 and 3 are both on at the same time, only the action of bit 2 is taken.
	4	Inhibit printing of endorsement data for programmable endorsement.
	5	Inhibit printing of INF data.
	6	Increment INF high-order segment. When this bit is on, one is added to the high-order segment of the item number, and the low-order segment resets to zero. This occurs before the INF print cycle.
	7	Increment INF low-order segment. When this bit is on, one is added to the low-order segment of the item number. This occurs before the INF print cycle. When the low-order segment is all nines and increases by one, it becomes all zeros. This does not affect the high-order segment. **Note:** If bits 6 and 7 are both on at the same time, only the action of bit 6 is taken.

Note: If you specify both numbers (microfilm and INF), the INF number takes precedence and makes the microfilm number the same as the INF number.

Module-Pocket Byte

The UT/XP module/pocket configuration consists of six physical pockets that are available for sorting in the first module and eight pockets for the remaining four modules. Therefore, if you specify 32, for example, in this byte, the document would physically sort to module two on the UT/XP. However, the logical pocket would be the same as the 3892/XP.

Figure 4-6. Module-Pocket Byte		
Byte(s)	**Bit(s)**	**Format**
6	0	Reserved
	1-3	Module selection, of module-pocket
	4	Reserved
	5-7	Pocket selection, of module-pocket

Record Header Byte 7

Figure 4-7. Record Header Byte 7

Byte(s)	Bit(s)	Format
7	0	Endorsement feature is initialized on.
	1	Sort program set the cancel-sort-processing flag. The Control Program forces the document to module-pocket 1/1.
	2	Power encoding not done for this document due to error, or encode data not valid.
	3	No extended code line data match was attempted.
	4	Process buffer modified by the SCI “SPS” macro or the SpxPutProcBufDat function.
	5	Microfilm initialized on.
	6	INF initialized on.
	7	First document fed after a microfilm space.

Record Header Byte 8

Figure 4-8 (Page 1 of 2). Record Header Byte 8

Byte(s)	Bit(s)	Format
8	0	Hardware-detected auto-select. A hardware-detected auto-select indicates that at least one of the following conditions occurred in relation to this document: • Multiple documents • Short gap between documents • Length greater than approximately 222.3 mm (8.75 in.) • Length shorter than approximately 123.2 mm (4.85 in.) • Distance between leading edge of one document and leading edge of following document too short • SS-sequence error (code line mismatch). If the 3890/XP Enhanced auto-selects a merge document, it automatically feeds another merge document. For additional information, see “3890/XP Enhanced Auto-Select Actions” on page 5-12.
	1	Invalid module-pocket code auto-select. The module-pocket in byte 6 does not exist in this machine.

Figure 4-8 (Page 2 of 2). Record Header Byte 8		
Byte(s)	**Bit(s)**	**Format**
	2	Late module-pocket code auto-select. This indicates that the user-supplied Sort program did not have enough time to make the module-pocket decision. A loop possibly occurred in the Sort program. If this bit is on in the header presented to the host system, the document was being processed by the Sort program when time ran out. If this bit is on in the header presented to the Sort program, the preceding document was auto selected.
	3	SCI error on this document.
	4	Merge document.
	5	No code line data match found in the write buffer. This bit is meaningless for a merge document.
	6	High-order zero correction occurred.
	7	Symbol error correction occurred.

Record Header Byte 9

Figure 4-9. Record Header Byte 9		
Byte(s)	**Bit(s)**	**Format**
9	0-3	Routing number self-check digit.
	4	Routing number not verified (check digit not valid).
	5	Pre-sort digit error
	6	Reserved.
	7	SCI pause. This indicates that, during processing of this document, the SCI program requested a pause in document feeding without forcing a not-ready condition. The XP Enhanced document processor must be online for this to be effective. For additional information, see “SCI Pause and Unit Exception” on page 8-3.

Record Header Bytes A and B

Figure 4-10. Record Header Bytes A and B		
Byte(s)	**Bit(s)**	**Format**
A	0-7	Document sequence number. Each document data record is given a sequence number with a value range of X'00' to X'FF' (X'01' to X'FF' on the UT/XP).
B		X'80'. Identifies this as a document data record header.

Exception Record Header - X'40'

Any incident that interrupts document flow (such as a jam or a selector check) and requires special operator procedures causes an exception record to be created. The exception record helps the host program identify the error and the documents that might need special handling. The exception record is the last record in the block of records transmitted to the host system.

You use the exception record information to do the following:

- Display error documents on the host system display screen to aid in local recovery procedures.
- After you clear the problem documents from the machine, save the information for a later run to continue the job.

Record Header Format

Figure 4-11 (Page 1 of 2). Exception Record Header Format

Byte(s)	Bit(s)	Format
0	0	Quick-stop.
	1	Selector check — 3890/XP Enhanced. MissSort — 3891/XP or 3892/XP.
	2	Reserved.
	3	Microfilm end-of-reel. The film supply has only 1.83 mm to 2.44 mm (6 to 8 ft) remaining. You must install a new supply of film.
	4	Reserved.
	5	Machine check (see Note 2).
	6	Endorser error.
	7	Image suspect list available. A suspect image is one that might be unusable.
1	0 1 2 3 4 5 6 7	Stacker-module 6 quick-stopped. Stacker-module 5 quick-stopped. Stacker-module 4 quick-stopped. Stacker-module 3 quick-stopped. Stacker-module 2 quick-stopped. Stacker-module 1 quick-stopped. Microfilm-module quick-stopped. Right-feed module (RFM) quick-stopped — 3890/XP or 3890/XP Enhanced.
2	0-3	The binary number of the stacker module involved in a quick-stop that is closest to the feed module.
	4-7	Reserved.
3	0-7	The last-read-document sequence number in a quick-stop.
4	0-7	The first-read-document sequence number in a quick-stop.

Figure 4-11 (Page 2 of 2). Exception Record Header Format

Byte(s)	Bit(s)	Format
5	0-7	The number (0 through 255) of the first document not image captured when a quick-stop occurs. This number is only valid when byte 6, bit 0 is on (1).
6	0	Indicates that the Image Capture System caused a quick-stop. If this bit is on (1), byte 5 contains the number of the first document not image captured.
	1	Reserved
	2	Transport Tracking Error (hardware error).
	3	Image Capture System out of sync with the sorter (Image only).
	4-7	Reserved.
7-A	0-7	Item number (last eight digits) of last document captured successfully by the Image Capture System.
B	0-7	X'40'. Identifies this as an exception record header.

Notes:

1. Fields 1 through 15 of the exception record always contain X'A's.
2. A left-feed module (LFM) or a right-feed module (RFM) overrun-quick-stop can cause an out-of-sync condition between the INF and the microfilm index numbers (3890/XP Enhanced only).

   To inform your program of this potential condition, the exception record header byte 0, bit 5 (machine check) is set when the quick-stop caused by register overrun occurs. When this bit is on with INF and microfilm both active, you should synchronize the numbers from the host system.

SCI Error Record Header - X'20'

If a Stacker Control Instructions (SCI) error occurs while the user-supplied Sort program is running and a TSCI time out then occurs, the Control Program produces an SCI error record header. The data part of the record has the document data.

The SCI error record header is a modified document data record header that has additional information about the error:

- The error code to identify the type of error
- The condition code returned as a result of processing the SCI instruction
- The SCI length
- The SCI address at the time of the error
- The SCI address of the last successful branch.

Record Header Format

Figure 4-12 (Page 1 of 2). SCI Error Record Header

Byte(s)	Bit(s)	Description
0-3		Same as document data record header format.
4	0-1	Reserved.
	2-3	SCI length code: 00—Not used. 01—SCI length is 1 halfword. 10—SCI length is 2 halfwords. 11—SCI length is 3 halfwords.
	4-5	Reserved.
	6-7	SCI condition code: 00—Condition code 0 01—Condition code 1 10—Condition code 2 11—Condition code 3.
5	0-3	Reserved.

Figure 4-12 (Page 2 of 2). SCI Error Record Header

Byte(s)	Bit(s)	Description
	4-7	SCI error code: **Error** **Error Condition** **1 (0001)** Field not valid. **2 (0010)** Header byte not valid. **3 (0011)** Length not valid. **4 (0100)** Operation code not valid. **5 (0101)** Link register not valid. **6 (0110)** Table data not valid. **7 (0111)** Parameter not valid. **8 (1000)** Operation not valid. **9 (1001)** Document processor is online. **A (1010)** Address not valid. **B (1011)** Save location not valid. **C (1100)** Displacement not valid. **D (1101)** Execute - subject instruction is an execute. **E (1110)** CALLNATV - Subject routine not located.
6-7	0-15	SCI address of instruction when error occurred.
8-9	0-15	SCI address of last successful branch.
A	0-7	Document sequence number.
B	0-7	X'20'. Identifies this as an SCI error record header.

Native Exception Header - X'02'

The following table describes the layout and definitions for the Native exception header that will be sent to the host following any document event exception condition (pending host support).

Figure 4-13 (Page 1 of 2). Native Error Header

Byte	Bit	Description
0		Document Time Exception Type. See Figure 4-14 on page 4-12 for a description of the value in this byte.
1	0	Exception occurred in the SPS Verify Event.
	1	Exception occurred in the SPS Correction Event.
	2	Exception occurred in the SPS Format Event.
	3	Exception occurred in the SPS Image Match Event.
	4	Exception occurred in the SPS Document Event.
	5	Exception occurred in the SPS Post Sort Event.
	6,7	Reserved.
1		Non Document Time Exception Type. See Figure 4-14 on page 4-12 for a description of the value in this byte.

Figure 4-13 (Page 2 of 2). Native Error Header

Byte	Bit	Description
1	0	Exception occurred in the SPS Pause Event.
	1	Exception occurred in the SPS Run Initialize Event.
	2	Exception occurred in the SPS Motors stopped Event.
	3	Exception occurred in the SPS Operator Event.
	4-7	Reserved.
3-9		Reserved.
A	0-7	Document Sequence Number.
B		X'02'. Identifies this as a Native exception header. Note. Not transmitted, based on host application support, currently maintained in control program memory.

This table will further describe the contents of the values in byte 0 of the native exception header.

Figure 4-14 (Page 1 of 2). Values returned in byte 0 of the Native Exception Header.

Header Byte 0 Value	Description	OS/2 Exception Value	Related Trap(s)
00	No Error (unused)		
01	Access Violation	'C0000005'X	09,0B,0C,0D,0E'X
02	Guard Page Violation	'80000001'X	
03	Guard Page Violation	'80010001'X	
04	Integer Divide by Zero	'C000009B'X	
05	Float Divide by Zero	'C0000095'X	'10'x
06	Float Invalid Operation	'C0000097'X	'10'x
07	Illegal Instruction	'C000001C'X	'06'x
08	Privileged Instruction	'C000009D'X	'0D'x
09	Integer Overflow	'C000009C'X	'04'x
0A	Float Overflow	'C0000098'X	'10'x
0B	Float Underflow	'C000009A'X	'10'x
0C	Float Denormal Operand	'C0000094'X	'10'x
0D	Float Inexact Result	'C0000096'X	'10'x
0E	Float Stack Check	'C0000099'X	'10'x
0F	Datatype Misalignment	'C00000A0'X	'11'x
10	Breakpoint	'C000009F'X	'03'x

Figure 4-14 (Page 2 of 2). Values returned in byte 0 of the Native Exception Header.

Header Byte 0 Value	Description	OS/2 Exception Value	Related Trap(s)
11	Single Step	'C00000A0'X	'01'x
12	Page Error	'C0000006'X	'0E'x
13	Process Terminate	'C0010001'X	
14	Async Terminate	'C0010002'X	
15	Noncontinuable Exception	'C0000024'X	
15	Invalid Disposition	'C0000025'X	
16	Signal	'C0010003'X	

The following is a sample output of an exception condition occurring at an operator event. This data will be written to a file on the fixed disk as C:\CONTROL\EXC.DMP.

This information will be written to file for all non-document events. For Document events the '02' header record will be written.

```
 3890/XP2 Exception Handler Dump Data **** Start ****

Exception Timestamp: Tue Jan 10 17:14:27 1995

exception Access Violation occurred on item   0( 255 Count)
exception occurred during the Operator Non-document Event
Invalid linear address 00000000
Failing code module name : SORTPROB
Failing code module file name : D:\SORTER\SORTPROB.DLL
Failing code Object # 0 at Offset 13f

GS  : 0000     FS  : 150B     ES  : 0053     DS  : 0053
EDI : 0F683B42 ESI : 001A0204 EAX : 00000000 EBX : 0F680000
ECX : 0F680000 EDX : 008EFFEC
EBP : 008EFFA8 EIP : 177E013F EFLG: 00012246 ESP : 008EFFA8
CS  : 005B     SS  : 0053

 Failing instruction at CS:EIP : 005B:177E013F is  mov      al,eax
Information on Registers :
 EAX does not point to valid memory


Thread slot 84 , Id 8 , priority 300

Thread slot 84 , Id 8 , priority 300
Stack Bottom : 008E0000 (008E:0000) ;Stack Top    : 008F0000 (008F:0000)
Process Id : 86 .EXE name : D:\SORTER\XP2.EXE

 This is the restored Register set.
GS  : 0000     FS  : 150B     ES  : 0053     DS  : 0053
EDI : 0F683B42 ESI : 001A0204 EAX : 00000000 EBX : 8001005B
ECX : 00000002 EDX : 008EFFEC
EBP : 008EFFF4 EIP : 0008456E EFLG: 00002246 ESP : 008EFFE8
CS  : 005B     SS  : 0053


 3890/XP2 Exception Handler Dump Data **** End ****


 Stack

 008EFFA8:  008EFFE0  0007B71A  00000053  0F683B42  001A0204  008EFFE0
 008EFFC0:  008EFFD4  0F680000  008EFFEC  00000053  008EFFE0  00000053
 008EFFD8:  0007DDFF  178C0455  008EFFF4  00084544  00036DB7  FFFFFFFF
 008EFFF0:  00036DD4  00000000  1BFBBF68  00000000  206E4120

 ******** Sort 1 ************************
```

Chapter 5. Document-Time Processing

The chapter describes the following:

- Preparing the process buffer
- Pre-Sort processing
- Sort processing
- Post-Sort processing
- Auto-Select actions.

Multithreaded Sort Environment

The 3890/XP Enhanced control program provides isolation for the Application programming interface by maintaining three identical threads for operation. These threads allow the machine to continue operation during a wide range of programming exceptions, occurring in a application.

Each exception will be handled differently based on the event executing when the exception occurs. The following table describes the handling for exceptions on particular events.

Event ***Handling***

Non-document An exception occurring during a non-document event will post a message to the run screen indicating that the exception has occurred. A file named C:\control\exc.dmp is written with specific information on the current exception. See a sample exc.dmp file on page 4-14. The entries registered to execute after the entry causing the exception will not be dispatched. The initialization state of the machine will only be changed during the initialization event (see below). Operator intervention is required to acknowledge the popup message, the use site procedures to proceed with sorter operations.

Initialization Initialization will be handled differently than the other non-document events in that Initialization will fail online or offline. Message "5031 1FX Request not Initialized or Abort Sort set." will be displayed on the run screen. A file "c:\control\exc.dmp" will be written to the disk subsystem for analysis of the exception. The machine must be initialized successfully before processing can continue.

Document An exception at the document time events will act like an abort sort request were issued. The processing on the current item will be suspended at the time of the exception. The item is sent to the reject pocket, and the record data header is updated accordingly. A disengage is issued to the transport, however, the remaining documents in flight will be handled normally, assuming no exceptions. No output file is written for document time exceptions. The '02' application exception record is created for the first document encountering an exception. This record is maintained in control program memory pending host application support.

The 3890/XP Enhanced control program has the ability to preempt sort programs that are running too long in duration. This ability has been extended to also include native language sort programs. Late running programs will cause the item to be routed to the

reject pocket, and the document data header updated to indicate a late module pocket decision.

Preparing the Process Buffer

Sort processing consists of system and application program software that runs while documents flow through the document processor, either online or offline, to the host. This system and application program software consists of the following:

- Code line verification
- Code line correction
- Process buffer format
- Code line data match.

As the documents flow through the 3890/XP Enhanced document processors, the read system automatically reads the characters on each document. These 4-bit characters are right-aligned in individual bytes and stored in the input buffer.

The IBM-supplied SCIEM Sort modules format, verify, and correct the data in the input buffer and move it to the process buffer. They also set flags in the document data record header and step maintenance counts.

"Pre-Sort Processing" on page 5-4 describes the operation of the following SCIEM modules that prepare the process buffer:

- The SCIV module - code line verification
- The SCIC module - code line correction
- The SCIF module - process buffer formatting
- The SCII module - code line data matching.

Pre-Sort processing is followed by Sort processing. The SCID module runs the SCI program by interpreting program storage as stacker control instructions. Other SPSDOC modules can run as well. The module-pocket decision and feature control information is written to the process buffer. If the document processor is online, the process buffer is read by the host as the "read record."

Post-Sort processes the endorsement data. The Run Profile might specify SPSPOST entry points for user-supplied modules as well as the IBM-supplied CONVERGE (for 3891/XP and 3892/XP).

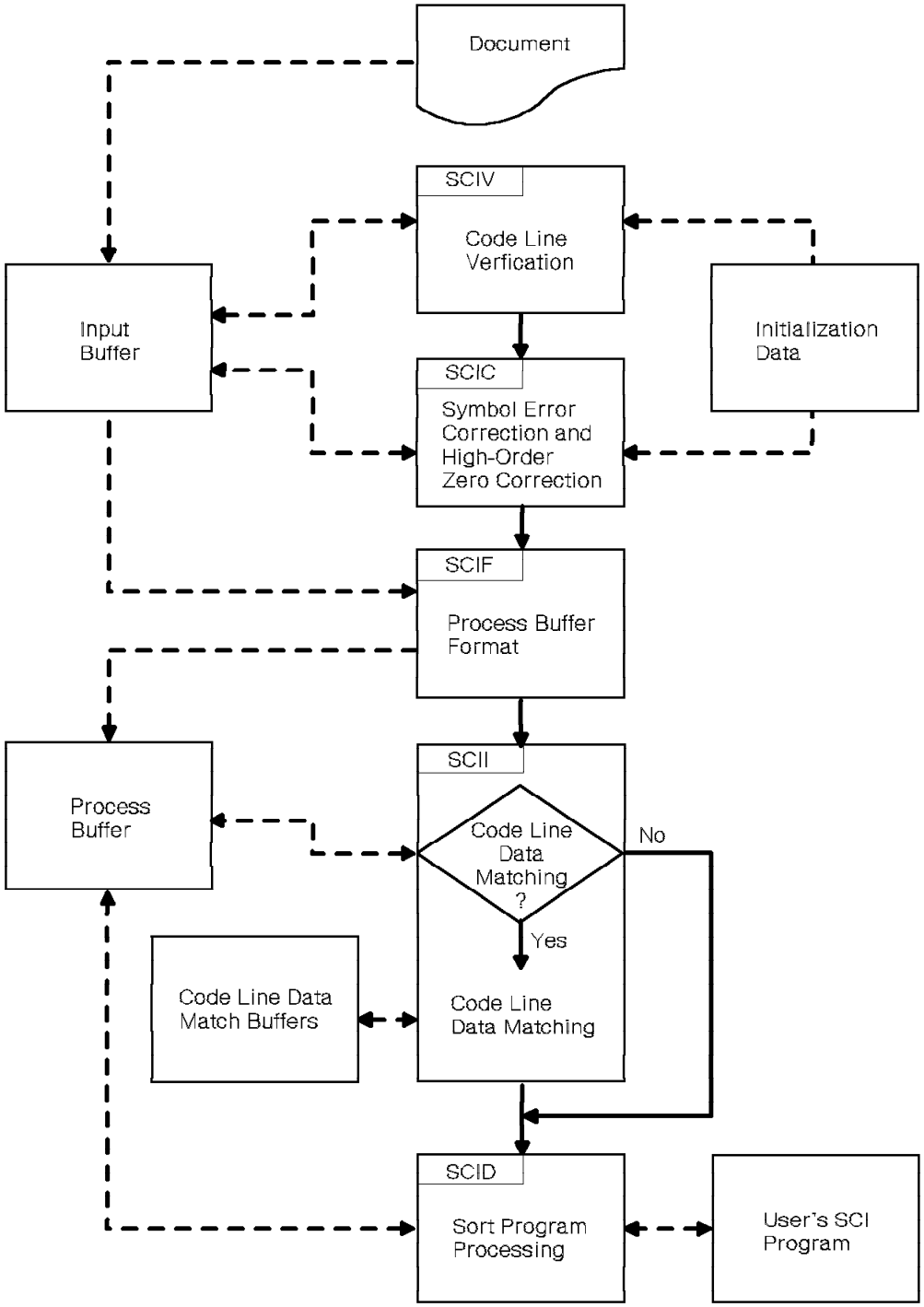

Figure 5-1. Pre-Sort and Sort Processing Functional Overview

Pre-Sort Processing

If you specify the SCIEM Sort modules, specific pre-Sort processing functions occur in the XP Enhanced document processors for the document data before the SPSDOC event occurs.

If you are using an SCI-Sort program, you must specify the SCID entry point defining the SCI emulator at the SPSDOC event time. *It is recommended that you use the Pre-Sort Processing SCIEM modules.*

Code Line Verification (SCIV)

Code line verification (the SCIV module) is the routine that scans the document data in the input buffer. The code line verification routine matches special symbols against a user-defined code line definition table from the SST file to determine the field boundaries.

When the code line verification routine finds a match, it updates the appropriate special-symbol pointer in the input buffer. These pointers are displacements from the start of the input buffer to the special symbol. The pointers effectively break the document data into the desired fields.

All special symbols do not need to appear in the code line. However, those that do appear must be in the proper sequence. If the symbols are not in the proper sequence, the document is auto-selected, and a special symbol sequence error is indicated in the document data record header. If the code line verification routine finds a digit error in the input buffer, it updates the appropriate digit-error counters and indicates a digit error in the document data header record.

The initialization data record (IREC) and the Run Profile specifying an SST file are the source of the information that is used to build the code line definition table. You can define the opening symbol, the closing symbol, and the alternate closing symbol for up to seven fields. You can define a maximum of 15 input fields in the IREC.

See “Special Symbol Sequence Table (SST) File” on page 3-17.

Code Line Correction (SCIC)

The code line correction routine (the SCIC module) provides high-order zero (HOZ) correction for field 1 and symbol error correction (SEC) for all specified code line fields. In your initialization data record bytes 55 and 60, specify how many high-order digits are candidates for HOZ correction in field 1 and which fields are candidates for SEC.

The code line correction routine looks at AUX storage bytes 7 and 8 to determine whether you requested HOZ correction and SEC. See “3890-Emulated Auxiliary Storage” on page C-7.

High-Order Zero (HOZ) Correction

The correction routine attempts high-order zero correction for field 1 if you have requested HOZ correction and the following are true:

- Field 1 is fixed length.
- Field 1 has opening and closing symbols.
- Field 1 is the correct length.

High-order zero correction proceeds from left to right and replaces digit errors with zeros unless the following occurs:

- The field reaches the user-specified number of digits.
- The field encounters a character other than a digit error or zero.

Symbol Error Correction (SEC)

The symbol error correction (SEC) routine consists of opening-symbol correction and closing-symbol correction. For all of the specified fields, as indicated by IREC byte 55, the SEC routine corrects the opening and closing symbols under a specific set of conditions. Any symbols that are defined as data are not field-delimiting symbols and do not go through the SEC routine. See Figure 5-2 on page 5-6 for a flowchart overview of the SEC function. The following list contains the specific conditions for the symbol correction function to successfully correct the symbols:

- The opening symbol of a fixed-length field, if the remainder of that field was read correctly
- The closing symbol of a fixed-length field, if the remainder of that field has been read correctly
- The opening symbol of a variable-length field, if it is the only error in that field and there are no errors in the field to the right.

Opening-symbol correction occurs for a candidate field if the following occurs:

- The opening symbol for that field is not present.
- The opening symbol is present, but the closing symbol is not present and there is no data in the field.

Opening-symbol correction is successful if the following occurs:

- For a fixed-length field of length *n*, a digit error is *n+1* positions from the opening symbol of this or the next higher present field.
- For a variable-length field, only one digit error occurs in the data for the next lower present field.

If opening-symbol correction is successful, the following occurs:

- If the opening symbol is not present, the digit-error position becomes the opening symbol.
- If the opening symbol is present, the existing symbol becomes the field's closing symbol and the digit-error position becomes the opening symbol.

Opening-symbol correction proceeds from SS7 through SS1. If opening-symbol correction is successful, the operation returns to the beginning (SS7).

Closing-symbol correction occurs for fixed-length candidate fields only. If the opening symbol is present for a field of length *n*, the closing-symbol correction is successful if a digit error occurs *n + 1* positions from the opening symbol. If closing-symbol correction is successful, the operation continues with the next available field.

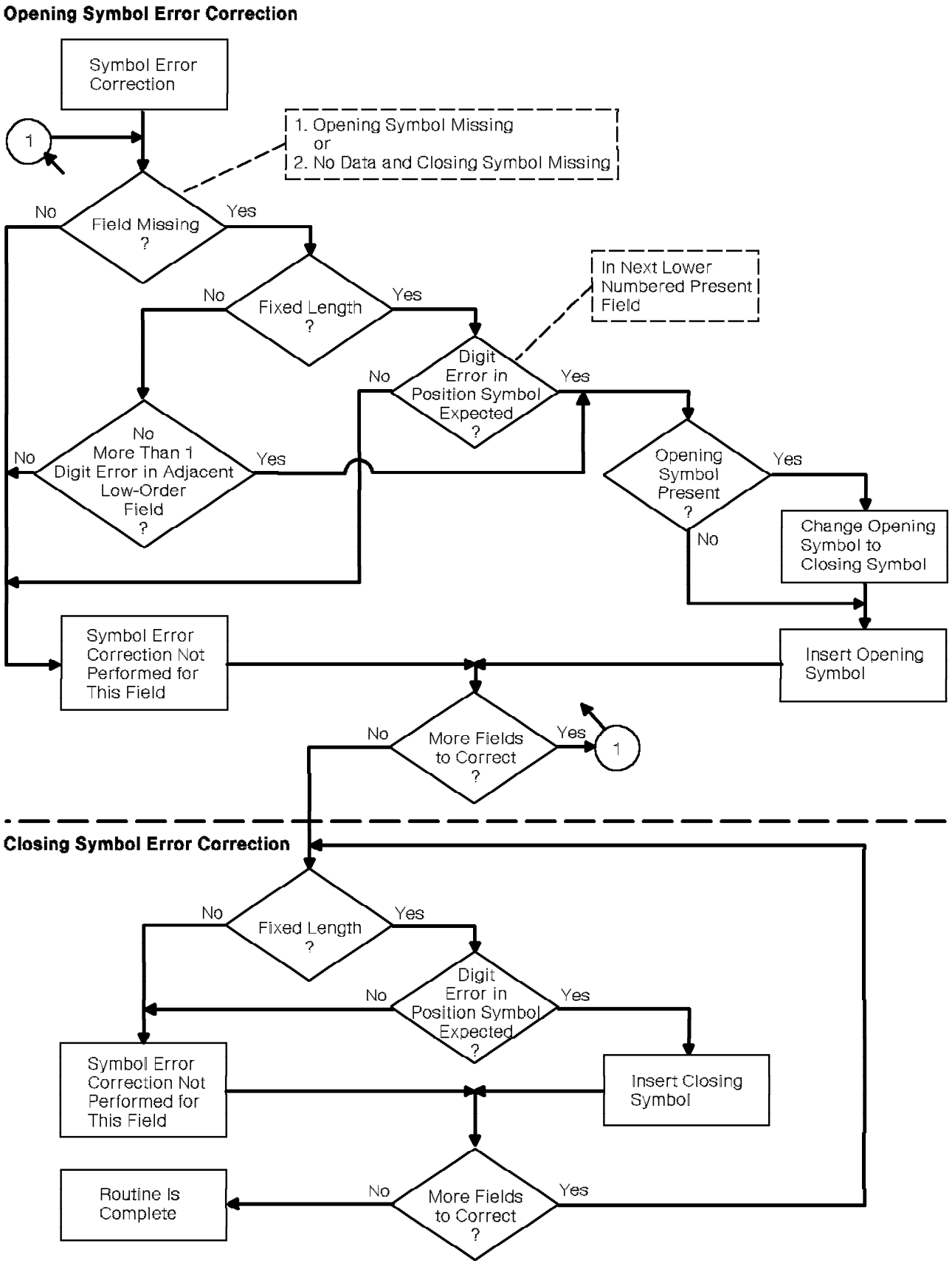

Figure 5-2. Symbol Error Correction (SEC) Routine

Risk and Advisability

If you understand the processing, the document parameters, and the code line, you can identify the possible risks you might take before you request the SEC routine.

Figure 5-3 analyzes a no risk situation, and Figure 5-4 on page 5-8 analyzes a risk situation.

No Risk Situation

		Field 5 Fixed		Field 3 Variable		Field 2 Variable		Field 1 Fixed	
Expected Symbol Sequence	(S6)		(S5)		(S3)		(S2)		(S1)
Actual Symbol Sequence	(S5)				(S3)		(S2)		(S1)
Data		9876-5432	*	543217		321098		7654321098	
Results after Symbol Correction	(S6)	9876-5432	(S5)	543217	(S3)	321098	(S2)	7654321098	(S1)

Figure 5-3. Symbol Error Correction (No Risk)

Example: An unreadable symbol causes a digit error. In Figure 5-3, the opening symbol for field 5 causes a digit error. The symbol sequence takes the closing symbol for field 5 as S5. The digit error appears in the position where the SEC routine expected the opening S5.

The SEC routine changes S5 to S6 and inserts S5 for the digit error. As a result, the SEC routine salvages field 3 and field 5.

Risk Situation

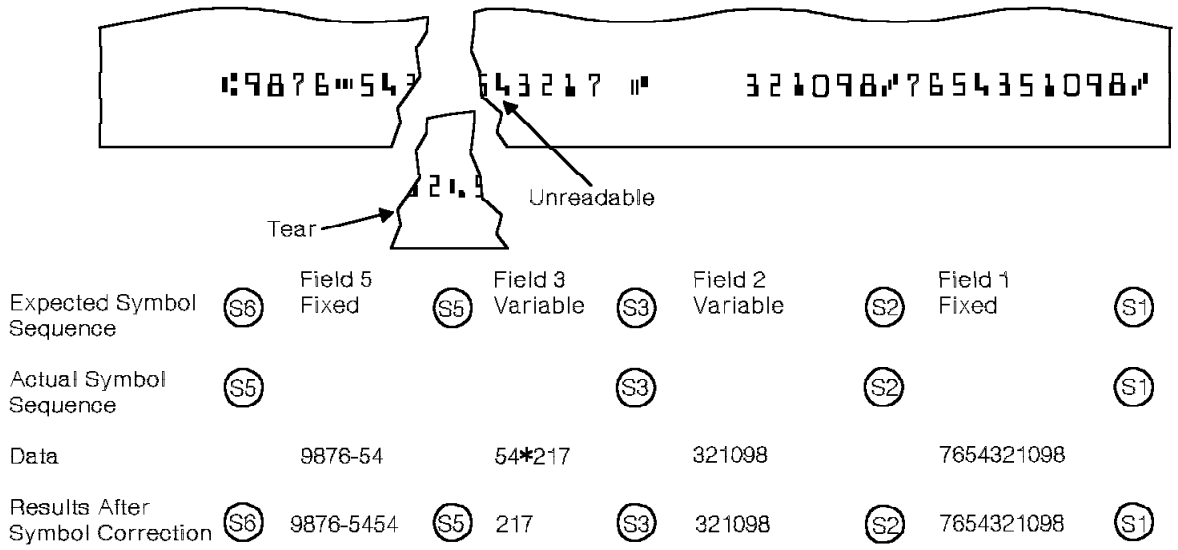

Figure 5-4. Symbol Error Correction (with Risk)

Example: In Figure 5-4, there is a digit error in field 3 (the number **3**) *and* a tear that removes a portion of field 5. The SEC routine operates as it did in the no-risk example, except that the fields and their contents are changed considerably. Although the error conditions and combinations are unique in this example, the risk of obtaining bad data does exist and should be reviewed.

The SEC routine sets byte 8, bit 7 in the document data record header to alert the programmer.

Process Buffer Format (SCIF)

The process buffer format routine (the SCIF module) copies the input buffer to the process buffer and formats it to be compatible with the 3890 Models A, B, C, and D.

The format routine receives control after the code line correction routine. The format routine:

1. Picks up the pointer to the corrected input buffer.
2. Moves the data from the input buffer.
3. Packs the data into the process-buffer fields defined in your initialization data record.
4. Uses X'AA' to pad short or missing fields and check each field for the following:
   - Digit errors
   - Length errors
   - Special-symbol-sequence errors.
5. Uses the error information to build header bytes 0 and 1 in the process buffer and to move header bytes 2 through 11 (unchanged) from the input buffer to the process buffer.

Process Buffer Fields

The process buffer represents data from a single document. The user-supplied Sort program analyzes the process buffer for each document. All fields in the process buffer are accessible by the Sort program. Field 0 is accessible, but only a subset of the field can be written by the customer sort programmer. For more information, see the *IBM 3890/XP Series and 3890/XP Enhanced SPXServ Reference.*

Figure 5-5 shows the format of a typical process buffer.

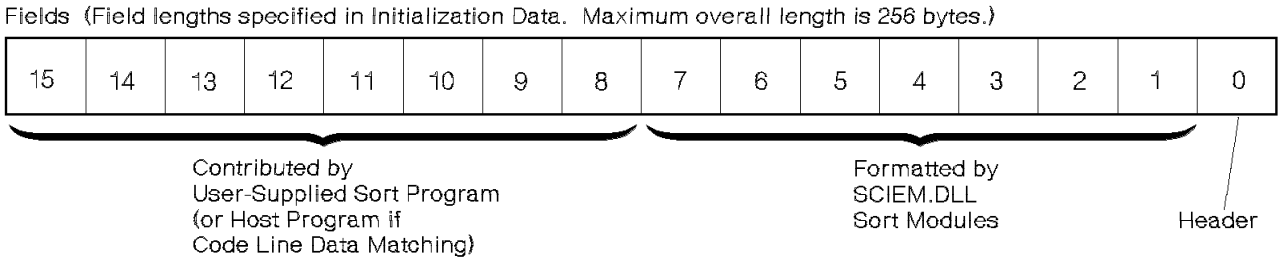

Figure 5-5. Process-Buffer Format

Field 0 is always part of the process buffer, but the initialization data defines the remaining fields (1 through 15).

Note: The total length of all defined fields (1 through 15), plus the 12-byte record header (field 0), must not exceed 256 bytes. When you specify 3890 emulation in the Run Profile, the total length of all fields defined plus the 12-byte record header must not exceed 48 bytes.

Field 0: This field is the document data record header. The record header identifies and describes the process-buffer data and indicates the type of conditions that relate to the data. For more information about the record-header format, see "Document Data Record Header - X'80'" on page 4-2.

Fields 1 through 7: These fields contain information that the XP Enhanced read systems read from the document. The read system formats this document data, one half-byte per character, in the following 4-bit code:

Binary Code	Hexadecimal Character	Binary Code	Hexadecimal Character
0000	0	1010	A (SS1 or fill character)
0001	1	1011	B (SS2)
0010	2	1100	C (SS3)
0011	3	1101	D (SS4 or dash)
0100	4	1110	E (SS5 or reserved)
0101	5	1111	F (digit error)
0110	6		
0111	7		
1000	8		
1001	9		

Fields 8 through 15: These fields contain information that comes from the user-supplied Sort program or, when code line data matching is successful, from the code line data match buffers.

Code Line Data Match (SCII)

The code line data match routine (the SCII module) matches document data in the process buffer with code line data from the host program. If a match occurs, the host-supplied data replaces the document data in the process buffer. If a match does not occur, the code line data match routine sets a flag in the process buffer document data header record.

Note: Code line data matching is compatible with the 3890 document processor. *Extended* code line data matching (when the Sort program calls the matching routine directly) is also compatible with the 3890. For more information about extended code line data matching, see "Extended Code Line Data Match" on page 6-3. (Document fields 1 through 15 are available for matching in *extended* code line data match, whereas only document fields 1 through 7 are used for *standard* code line data match.)

The code line data match routine receives control either:

- After the SCIF module formats the process buffer

  or

- If the Sort program calls the SCII module for an extended code line data match. (This call must be during the SPSDOC event.)

In your initialization data record (IREC), you define the fields the routine matches on and the error threshold for each field.

For more information about the code line data match routine and the SCII module, see "Code Line Data Matching Overview" on page 6-1.

Sort Processing

Sort processing takes place after pre-Sort processing has prepared the process buffer.

SCI Emulation (SCID)

SCI emulation (the SCID module) interprets program storage as Stacker Control Instructions (SCI).

If an SCI error occurs, the SCID module takes the following actions:

1. Interrupts SCI processing
2. Sets document data record header byte 8, bit 3 (SCI error on this document)
3. Continues SCI processing at the SCI error-handling routine.

Note: Your SCI program should have an error-handling routine. The address of the first SCI instruction of the error-handling routine is in program storage (decimal address 146 or 148, see page C-6).

The SCI error-handling routine can set the module-pocket code and the feature control byte for the error document, disengage, if desired, and exit SCI processing for that document. If the routine initiates a disengage, it should display a message telling the operator of the document processor the reason for the disengage.

Alternatively, if your error routine consists of a single-instruction loop, the 3890/XP Enhanced Control Program forces a **Terminate SCI** (TSCI) time out.

If a TSCI time out occurs and the SCI error on this document is set, the Control Program changes the record header to an SCI error record header - X'20'. See “SCI Error Record Header - X'20'” on page 4-10.

Post-Sort Processing

After the user-supplied Sort program processes a document, the Control Program performs specific post-Sort processing functions. These functions include, but are not limited to, the following:

- Call the SPSPOST event module, if any.
- Increment the numbers for INF, if requested by the Sort program.
- Update pocket counters and request merge documents from the transport, as required.
- Copy the module-pocket code and the feature control information from the process buffer record header and forward that data to the transport.
- Forward any Active Endorsement Area (AEA) data for the document to the transport.
- Disengage or pause the transport, as requested.
- Generate the read record by copying the process buffer to a read buffer for transmission to the host program.

If you specify the NOSCA parameter in the Run Profile, the Sort program is bypassed and the features and document data headers are as shown in the auto-select charts on the following pages.

3890/XP Enhanced Auto-Select Actions

The auto-select charts show actions with or without SCI control of auto-selects. See "Run Profile Keyword Summary" on page 3-2 (RPQACTive SCA and No SCA definition).

3890/XP Enhanced
Effects of Auto-Select on Features

Function	***Microcode Auto-Selects***	***Hardware Auto-Selects***	
	Late TSCI or Invalid M/P (1)	*No SCI Control of Auto-Selects*	*SCI Control of Auto-Selects* (2)
Endorse	No	Yes	Sort Control
INF	Yes (no increment)	Yes (no increment)	Sort Control
Microfilm	No flash	No flash	Sort Control
Image	No Image Control Data	No Image Control Data	Sort Control
Actual Pocket Destination	1/1	1/1	Sort Control

3890/XP Enhanced
Effects of Auto-Select on Document Data Headers

Document Data Header	***Microcode Auto-Selects***	***Hardware Auto-Selects***	
Auto-Select Indication *Byte 8*	Bit 1 = Invalid Bit 2 = Late	Bit 0	Bit 0 (3)
M/P Indication *Byte 6*	Sort Control	Ctrl Pgm 1/1	Sort Control
Image Indication *Byte 4*	Bit 4 = 0	Bit 4 = 0	Bit 4 = Sort Control

Notes:

1. A late document autoselect takes precedence over invalid module-pocket (M/P) decision autoselects.
2. On SCI Control of auto-selects, the Control Program attempts to feature the document, as instructed by the Sort program.
3. If the Control Program detects a Special Symbol Sequence error, it sets document data record header byte 1, bits 1 through 7, and it sorts the document to the pocket selected by your Sort program.

3891/XP and 3892/XP Auto-Select Actions

The auto-select charts show actions with or without SCI control of auto-selects. See "Run Profile Keyword Summary" on page 3-2 (RPQACTive SCA and No SCA definition).

3891/XP and 3892/XP
Effects of Auto-Select on Features

Function	***Microcode Auto-Selects***		***Hardware Auto-Selects***	
	Late M/P	*Invalid M/P*	*No SCI Control of Auto-Selects*	*SCI Control of Auto-Selects*
Endorse	No	No	Yes	Sort Control (2)
INF	No	Yes (no increment)	Yes (no increment)	Sort Control (2)
Microfilm	No flash	No flash	No flash	Sort Control (2)
Image	No Image Control Data	No Image Control Data	No Image Control Data	Sort Control (2)
Actual Pocket Destination	1/1	1/1	1/1	1/1 (3)

3891/XP and 3892/XP
Effects of Auto-Select on Document Data Header

Document Data Header	***Microcode Auto-Selects***		***Hardware Auto-Selects***	
Auto-Select Indication *Byte 8*	Bit 2	Bit 1	Bit 0	Bit 0 (3)
M/P Indication *Byte 6*	Sort Control	Sort Control	Ctrl Pgm 1/1	Sort Control (4)
Image Indication *Byte 4*	Bit 4 = 0	Bit 4 = 0	Bit 4 = 0	Bit 4 = Sort Control

Notes:

1. A late document autoselect takes precedence over invalid module pocket decision (M/P) autoselects.
2. On SCI Control of auto-selects, the Control Program attempts to feature the document, as instructed by the Sort program.
3. If the Control Program detects a Special Symbol Sequence error, it sets document data record header byte 1, bits 1 through 7, and it sorts the document to the pocket selected by your Sort program.
4. Auto-selects, with the exception of special symbol sequence errors, will always go to module-pocket 1/1, and the sort designated pocket will be in this byte.
5. The 3891/XP and 3892/XP treats an individual auto-select document and an auto-select group the same. One record is created for an auto-select document or auto-select group.
6. The first document (leading) in an auto-select group is the only document acted on by any of the features.

The feature control byte (5) and the module-pocket code byte (6) in the document data record header reflect decisions made by the Sort program.

1. If *invalid or late module-pocket auto-select* is indicated (byte 8, bit 1 or 2 in the header), the module-pocket and feature decisions are overridden and the following takes place:
   - The document processor sends the document to module-pocket 1/1.
   - The features operate as though byte 5 is all zeros with bit 4 on; that is:
     - Endorsement is inhibited.
     - If INF is enabled, the INF of the previous document is printed.
     - Microfilm does not flash.
     - Image control is not activated.
2. If *hardware-detected auto-select* is indicated (byte 8, bit 0) and sort-control of auto-selects is not active (the RPQACTive parameter NOSCA is specified or IREC byte 80, bit 0 is not set) the same overrides take place. If SCA is active, the Sort program's module-pocket and feature decisions are *not* overridden.

Universal Transport/XP Auto-Select Actions

The auto-select charts show actions with or without SCI control of auto-selects. See “Run Profile Keyword Summary” on page 3-2 (RPQACTive SCA and No SCA definition).

UT/XP
Effects of Auto-Select on Features

Function	***Microcode Auto-Selects***		***Hardware Auto-Selects***	
	Late M/P	*Invalid M/P*	*No SCI Control of Auto-Selects*	*SCI Control of Auto-Selects*
Endorse	No	No	No	No
INF	No	No	No	Sort Control (2)
Microfilm	No flash	No flash	No flash	Sort Control (2)
Image	N/A	N/A	N/A	N/A
Actual Pocket Destination	1/1	1/1	1/1	1/1 (3)

UT/XP Effects of Auto-Select on Document Data Header and Process Buffer Data Available				
Document Data Header	***Microcode Auto-Selects***		***Hardware Auto-Selects***	
Auto-Select Indication *Byte 8*	Bit 2	Bit 1	Bit 0	Bit 0 (3)
M/P Indication *Byte 6*	Sort Control	Sort Control	1/1	Sort Control
Image Indication *Byte 4*	N/A	N/A	N/A	N/A
Process Buffer Data Availability	***Microcode Auto-Selects***		***Hardware Auto-Selects***	
Auto-Select Doc	Yes (4)	Yes (4)	Yes (4)	Yes (4)
Documents Following Auto-Select Doc	Yes (4)	Yes (4)	Yes (4)	Yes (4)

Notes:

1. A late document autoselect takes precedence over invalid module pocket (M/P) decision autoselects.
2. On SCI Control of auto-selects, the UT/XP Control Program attempts to feature the document, as instructed by the Sort program. However, since these documents are received after they have been routed to the reject pocket and featuring is turned off by the UT/XP firmware, the INF (doc serial #) will be incremented if specified by the Sort program although actual item numbering and microfilm flash will NOT be performed.
3. If the Control Program detects a Special Symbol Sequence error, it sets document data record header byte 1, bits 1 through 7, and it sorts the document to the pocket selected by your Sort program.
4. The UT/XP creates a record for each document involved in an auto-select. This includes those items following the actual document causing the autoselect. Each record contains all F's in each field representing digit errors. After receiving these records, the UT/XP pauses. The Control Program gives control to the user's Sort program if one is specified in the SPSPAUSE keyword of the Run Profile. The Control Program then automatically starts the UT/XP feeding documents.

The feature control byte (5) and the module-pocket code byte (6) in the document data record header reflect decisions made by the Sort program.

1. If *invalid or late module-pocket auto-select* is indicated (byte 8, bit 1 or 2 in the header), the module-pocket and feature decisions are overridden and the following takes place:
   - The UT/XP document processor sends the document to module-pocket 1/1.

- The features operate as though byte 5 is all zeros with bit 4 on; that is:
  - Endorsement is inhibited.
  - If INF is enabled, the INF of the previous document is printed.
  - Microfilm does not flash.
  - Image control is not activated.

2. If *hardware-detected auto-select* is indicated (byte 8, bit 0) and sort-control of auto-selects is not active (the RPQACTive parameter NOSCA is specified or IREC byte 80, bit 0 is not set) the same overrides take place. If SCA is active, the Sort program's module-pocket and feature decisions are *not* overridden.

Chapter 6. Code Line Data Matching

This chapter describes the following:

- Code line data matching overview
- Activating code line data matching
- Priming the write buffer
- Creating the host code line data match file
- Extended code line data match
- Using fields 8 through 15
- Search range.

Code Line Data Matching Overview

Code line data matching reduces the number of rejected documents in the second or subsequent document pass. It uses document code lines from a user-created code line file or actual document data to process a document. This is done by downloading code line data from a host to the code line data match buffers.

The IBM-supplied routine for code line data matching (the SCII routine) compares the document code line data that is in the process buffer with the document code line data record from the host. If the comparison is within the user-specified limits for the fields being matched, code line data from the host replaces the data in the process buffer for all fields. Header bytes 0 through 3, 5, and 6 are also replaced. See Figure 4-1 on page 4-2. The user-supplied Sort program then uses the process buffer to make the module-pocket decision for the document.

The only miscompares “allowed” are digit errors or, for character fields, byte errors. Your initialization data record (IREC) specifies the acceptable error threshold for each field in IREC bytes 22 through 35 and 96 through 109. As long as the number of errors does not exceed the threshold, the routine considers the field to be a match.

The code line data matching routine (SCII) also uses the following 3890 document processor RPQs (request for price quotations):

- “Zero-for-able” (ZFA)
- “Don’t-care-digit” (DCD).

These RPQs are set by the RPQACTive keyword in the Run Profile (see page 3-6).

If ZFA is active, the routine considers any digit position a match where the host data contains a zero and the process buffer contains a hex A. If DCD is active, any digit is a match where the host program has placed a hex C in the data.

Notes:

1. The code line data match function is not performed on documents fed from the merge hopper.
2. A candidate field of zero length is automatically *matched*.

Activating Code Line Data Matching

The code line data matching routine (SCII) receives control:

- During the SPSIMATCH event, after SCIF formatting of the process buffer (SPSFORMAT event)

  or

- When the Sort program calls the routine for extended code line data match. This call must be made during the SPSDOC event.

In the initialization data record you specify that you want code line data matching and you define the fields for comparison and the error threshold for each field. See IREC bytes 0, 21, 22 through 36, 96 through 109, and 120.

Priming the Write Buffer

You must prime the code line data match write buffer with code line data records written from the host before matching can begin. IREC byte 0, bit 6 in the initialization data causes the Control Program to prime the code line data match write buffer.

If the matching routine does not find a record in the write buffer that matches the current document, it flags a *no code line data match found* condition in byte 8, bit 5 of the document data record header. If this condition occurs repeatedly, you should determine the relative positions of the document data and the write buffer records and have your host program prime the write buffer with the correct records. You can prime the write buffer by reloading the job. Your Sort program also can request an SCI pause, at which time the host can write new records to the code line data match write buffer.

Creating the Host Code Line Data Match File

The first time the document processor reads documents (first pass) the documents are processed or "captured" as for a normal non-match run, and the document records are read by the host. These records should be received into a disk file (called a code line data match file) at the host. This file must remain in the same "process buffer format" as the document data read by the document processor. This is unsigned, packed, decimal data as in Figure 6-1. The record includes the header.

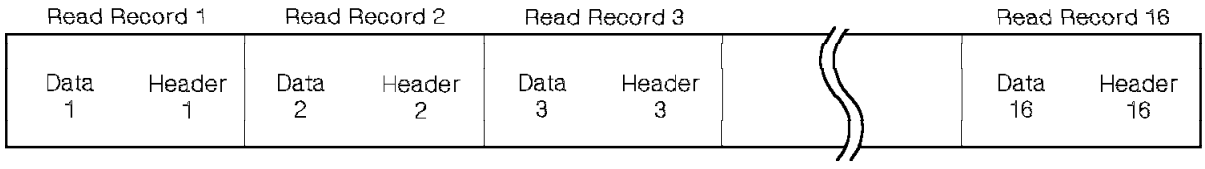

Figure 6-1. Code Line Data Match Write Buffer

On subsequent passes, the records from the host in the write buffer must be in the same sequence as the documents are in as they are read by the document processor. If documents go to more than one pocket on the first pass, you need to create a code line data match file for each pocket. This makes it possible on subsequent passes to keep the code line data from the host in the same sequence that the document processor reads the documents in. If you create only one file per run, you might need to sort the file at the host to match the physical document sequence.

For channel-attached communication, the host writes 64 records to the write buffer, in blocks of 16 records each. End-of-file at the host is indicated by setting byte 0, bit 0 in the headers of the unused records in the last block written to the document processor. You will need an entire block of unused records if the number of records for data matching is a multiple of 16. The channel-attached Control Program stops requesting further writes when it comes to these “end-of-file” records.

For APPC-attached communication, the host indicates end-of-file for the code line data matching pass by deallocating the conversation with “XP.Write.Data”. In this case, the Control Program builds an end-of-file record.

Extended Code Line Data Match

Extended code line data matching provides the code line data match function for all code line fields that can be defined. Only fields 1 through 7 are available for the *standard* code line data matching. The fields for *extended* code line data matching include fields 1 through 15.

When you specify *extended* code line data matching by turning on bits 2 and 3 of IREC byte 21, the matching routine suppresses *standard* matching during the SPSIMATCH event and permits code line data matching under Sort program control during the SPSDOC event. Other initialization data is the same as for the standard code line data match; that is, the same field sizes, formats, and digit error thresholds apply.

The SCI “IMAT” macro specifies the fields for extended matching. Your SCI program can issue “IMAT” any number of times, changing the fields as required. For more information about the SCI “IMAT” macro, see the *3890/XP Series Stacker Control Instructions Reference*

Your OS/2 language Sort routines can call the SpxEIMAT function to specify the fields for matching, followed by a call to SCII. These calls can be repeated as required for each document. See the *3890/XP Series and 3890/XP Enhanced SPXserv Reference*.

Using Fields 8 through 15

Fields 8 through 15 are user fields. If your Sort program puts data in these fields during the capture pass, you can use them in a subsequent pass, perhaps even matching on them. Your host program can also originate information for these fields. Either way, the data in these fields will be copied to the process buffer, assuming a match for the current document.

For example, as you create the code line data match file you assign a sequence number to each record in field 8. The Sort program uses the number assigned to the document on subsequent code line data matching passes.

Search Range

The search range for any document is 15 code line data match records in the write buffer. The range is defined by a series of pointers. The pointers and records pointed to are:

Pointer	Points to the:
Low-limit pointer (LLP)	Beginning record of the search range
High-limit pointer (HLP)	Record that follows the last record in the search range
Search center (SC)	Record in the center of the search range
Starter pointer (STP)	Record which the code line data matching routine currently uses for comparison.

For each document, the code line data matching routine locates the first-to-be-compared code line data at the starter pointer. If there is no match at the starter pointer, the routine then goes to the low-limit pointer and compares consecutive records until it finds a match or until it uses up the search range.

No Match Found

A *no code line data match found* condition is set when the matching routine did not find a match within the search range or when the document is hardware auto selected.

If no match is found, the pointers are updated as follows:

1. For multiple document detect (MDD), short gap, and short leading edge-leading edge (LE-LE) autoselects, the routine:
   - Increases the LLP, HLP, and SC by two records.
   - Sets the STP to the SC.
2. For all other non-compares and autoselects, the routine:
   - Increases the LLP, HLP, and SC by one record.
   - Sets the STP to the SC.

Match Found

If a match is found, the pointers are updated as follows:

1. If the routine finds the match behind the search center, the routine:
   - Does not change the LLP, HLP, and SC.
   - Sets the STP to the record that follows the matched record.
2. If the routine finds the match at the search center, the routine:
   - Increases the LLP, HLP, SC, and STP by one record.
3. If the routine finds the match ahead of the search center, the routine:
   - Increases the LLP, HLP, and SC by the number of records needed to make the SC equal to the STP.
   - Sets the STP to the record that follows the matched record.

Chapter 7. Programmable Endorsement and Item Numbering

This chapter discusses the following topics:

- Initialization
- 3891/XP and 3892/XP endorsements
- 3891/XP and 3892/XP character set
- Regulation CC and the depository bank endorsement area
- 3890/XP Enhanced endorsements
- 3890/XP Enhanced character sets
- INF item-numbering data
- Endorse Setup/Verify screen
- The ENDorse command
- Endorsement control
- Operating principles for endorsement and item numbering
- Fixed endorsement
- Endorse/INF converge DLL
- Full Function endorse
- Roll-On endorser.

Initialization

The programmable endorsement feature and the item-numbering feature (INF) print endorsement text and INF data on the back of documents as they move through the transport. The user Sort module can change the endorsement text and the INF data for each document.

For the 3890/XP Enhanced document processor, you can initialize these features to print the endorsement text and INF data in any of six locations on the back of the document. You can select locations to meet REGCC endorsement requirements or to correspond to 3890 endorsement locations. The three horizontal positions are set at initialization time. The 3890/XP or 3890/XP Enhanced operator mechanically controls the two vertical positions.

For the 3891/XP and 3892/XP document processors, the endorsement text and INF data can be printed on the front side if desired, per the initialization data record (IREC). Back side endorsement can be positioned according to REGCC.

Figure 7-1 on page 7-2, Figure 7-6 on page 7-6, and Figure 7-7 on page 7-7 show the endorsement locations for these machines.

The XP Enhanced document processors do not endorse late and inval id module-pocket auto-select documents; but, if the item-numbering feature is enabled, the document processor prints the INF number of the previous document. See “3890/XP Enhanced Auto-Select Actions” on page 5-12.

For more information about determining print positions, see “Operating Principles for Endorsement and Item Numbering” on page 7-15. For more information about the REGCC keyword, see page 3-6.

3891/XP and 3892/XP Endorsements

The following figure illustrates the endorsement locations for the 3891/XP and 3892/XP.

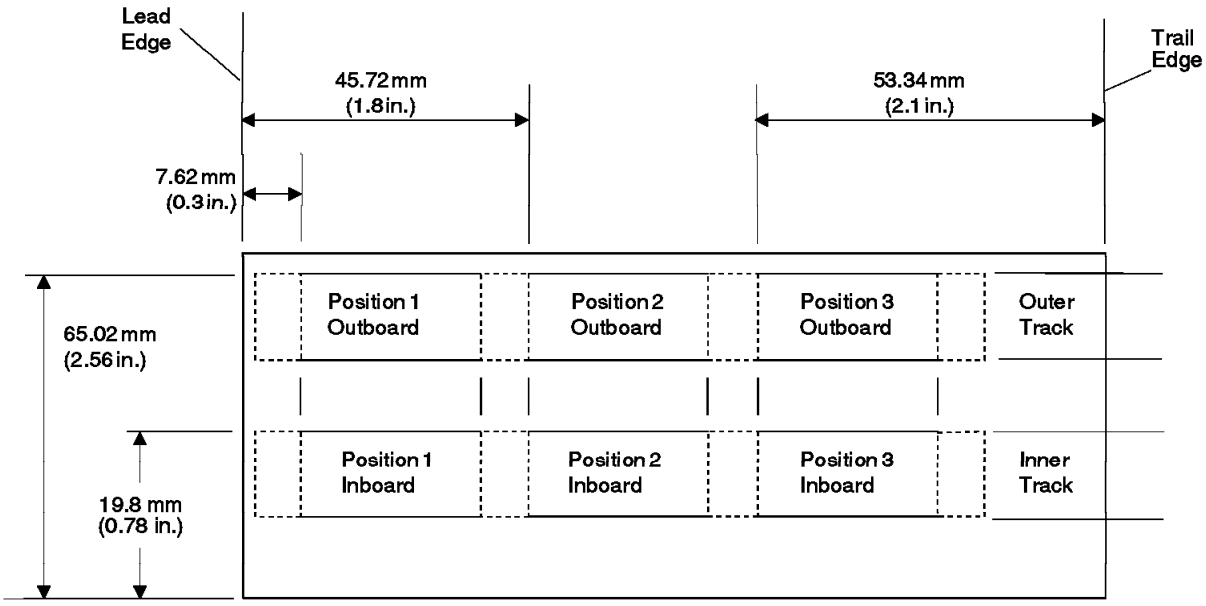

Figure 7-1. Rear View of Document Showing Endorsement Locations (3891/XP and 3892/XP)

3891/XP and 3892/XP Character Set

The following table lists the 3891/XP and 3892/XP document processor character codes that are used for endorsing and item numbering at 6 characters per inch.

Figure 7-2. Character Codes (X'20' - X'7F')

20	Space	30	0	40	@	50	P	60	‘	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	“	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	’	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	spare

Note: The character marked *spare* is a reserved code and prints as a blank space.

Regulation CC and the Depository Bank Endorsement Area

Regulation CC has specified three endorsement areas:

- Payee
- Depository bank (bank of first deposit)
- Transit.

The depository bank endorsement area is specified as the area 3.0 inches from the leading edge to 1.5 inches from the trailing edge. While it is not required that the entire endorsement be wholly contained within this area, it is required that the 9-digit routing number of the depository bank, set off by arrows, reside in the area.

The UT/XP Ink Jet Printer prints 8 characters per inch. This allows for 12 characters per line in the 1.5 inches reserved for the depository bank. This allows for all characters including the arrows to be printed in the payee endorsement area.

The 3891/XP and 3892/XP Ink Jet Printer prints 6 characters per inch. This allows for 9 characters per line in the 1.5 inches reserved for the depository bank. It does not, however, allow for the arrows to be printed in the area. The arrows overflow into the payee endorsement area on one end and into the transit endorsement area on the other end (see Figure 7-3 on page 7-4).

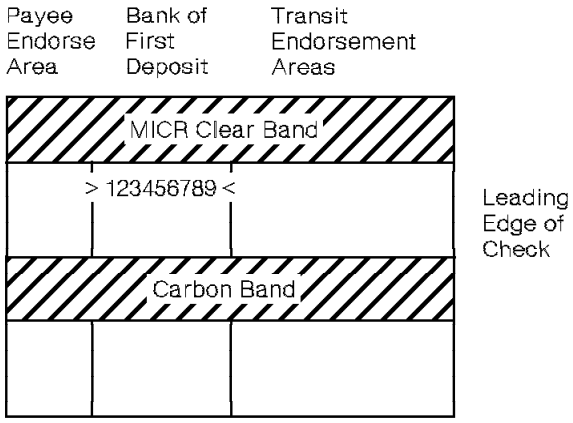

Figure 7-3. Back of Check (Not to Scale) with Routing Number on One Line

If the arrows must be contained in the 1.5-inch area, print the routing number on two lines (see Figure 7-4).

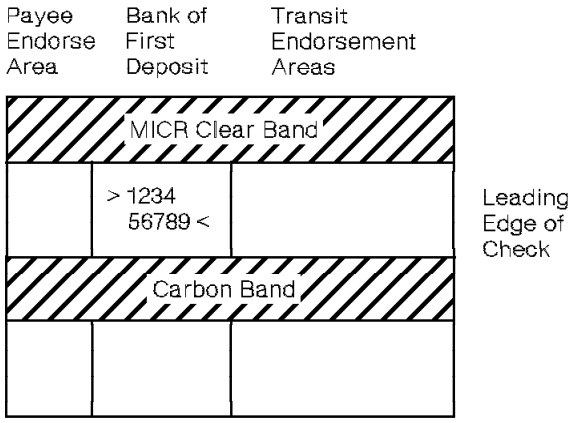

Figure 7-4. Back of Check (Not to Scale) with Routing Number on Two Lines

The 3891/XP, 3892/XP, and UT/XP allow you to specify one of three horizontal print positions (1, 2, or 3) for both the ink jet and Roll-On endorsements. The actual locations of these print positions are initially established manually through the *Customer Engineer (CE) Configuration Menus* on the transport gas panel display. Your CE can show you how to access the Ink Jet menu, one of several menus on the gas panel display.

Note: The print zone values are stored in non-volatile memory. The 3891/XP or 3892/XP must be powered off and back on to update the print zones after you change them on the Configuration Menus.

Proper definition of the print zones ensures that the routing number is contained in the depository bank location if the routing number is left-justified in the Programmable Endorsement Data (PED) file.

An alternative approach is to endorse using the Roll-On endorse option. You can design the endorse plate such that the entire endorsement is contained in the specified depository bank location.

If the Roll-On endorser is being used, proper definition of the print zones ensures that the entire endorsement is contained in the depository bank location. As shown in Figure 7-5, the endorsement can be in one of two horizontal bands and must not overlap into the MICR clear band or carbon band.

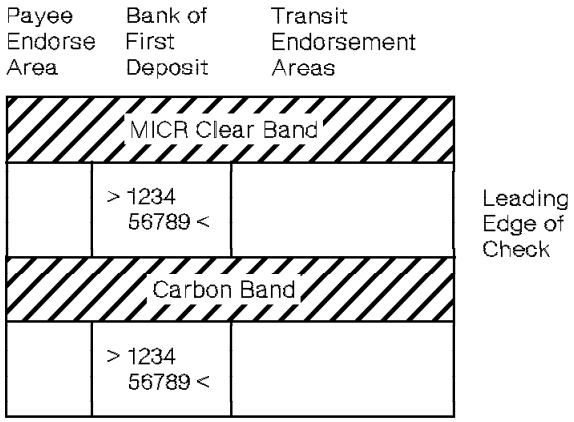

Figure 7-5. Back of Check (Not to Scale) with Roll-On Endorser

3890/XP Enhanced Endorsements

For the 3890/XP Enhanced document processor, the endorsement text consists of 72 alphanumeric characters in three rows of 24 characters each. The endorsement text can contain information such as:

- Bank routing number
- Bank name
- Endorse date.

For the 3890/XP Enhanced, endorsement text can also print in double-wide (DW) characters, twice the width of normal characters.

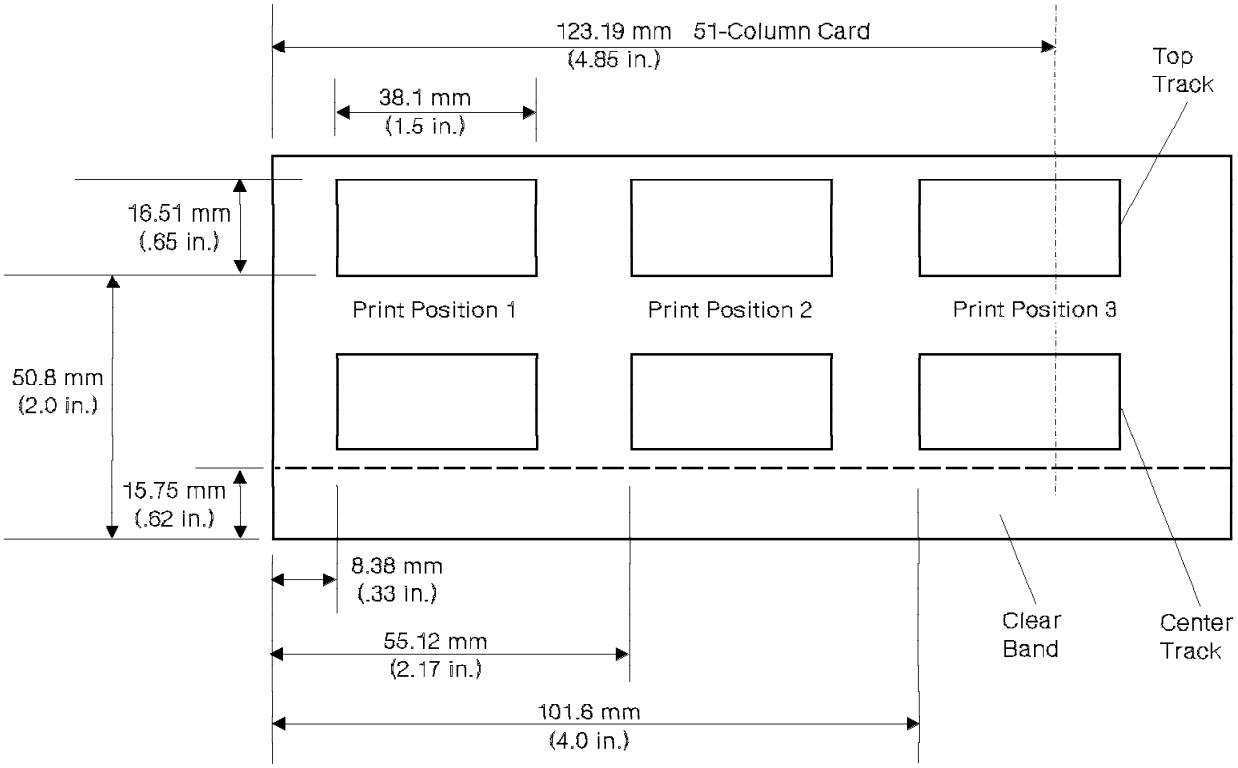

Figure 7-6. Rear View of Document Showing NOREGCC Endorsement Locations (3890/XP or 3890/XP Enhanced)

3890 REGCC Endorsements

The following figure illustrates the endorsement locations for the 3890/XP and 3890/XP Enhanced.

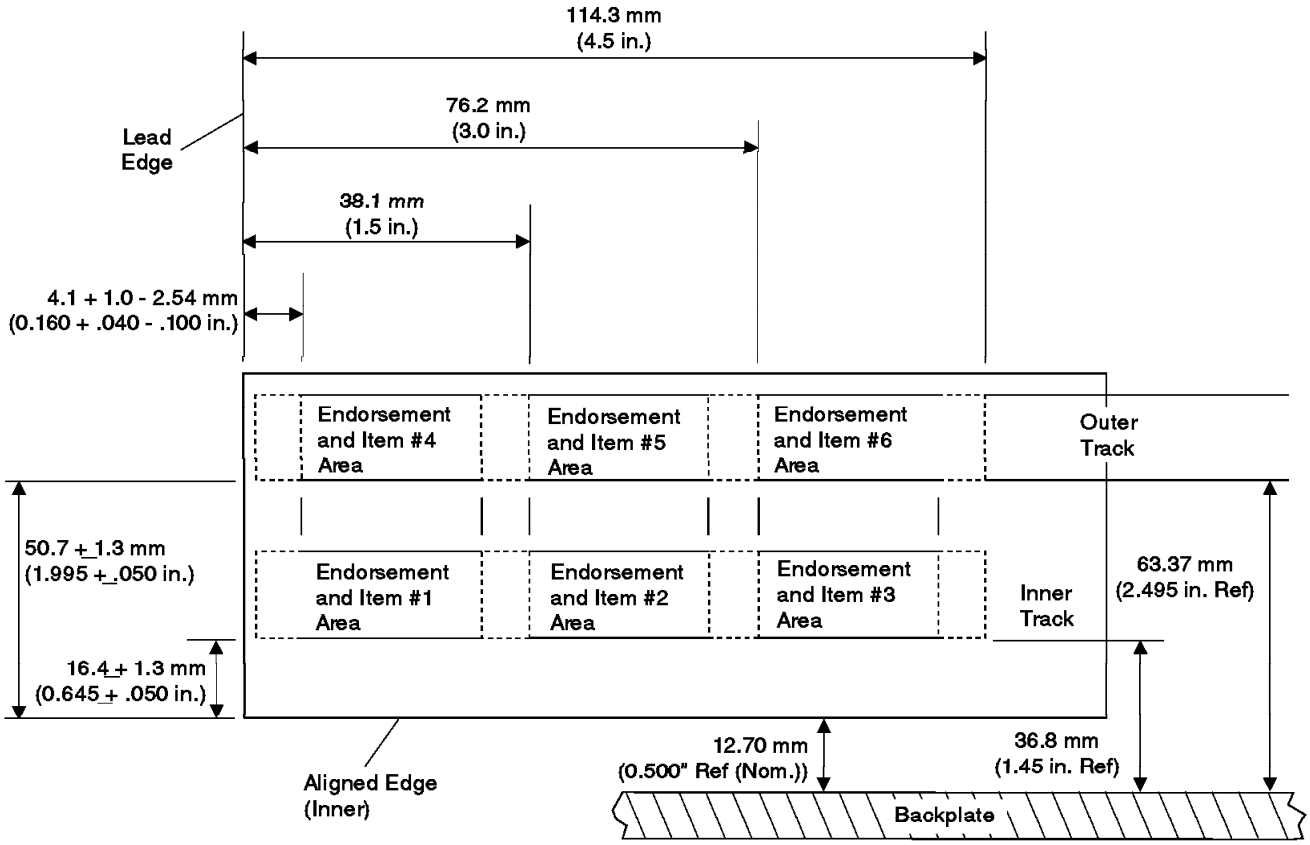

Figure 7-7. Rear View of Document Showing REGCC Endorsement Locations

3890/XP Enhanced Character Sets

For the 3890/XP Enhanced document processor, valid endorsement text characters are:

- Uppercase alphabetic characters A through Z
- Numeric characters 0 through 9
- Special characters
  - Space or blank
  - Ampersand
  - Parentheses
  - Period
  - Dash
  - Equal sign
  - Pound sign
  - Slash
  - Asterisk
  - Comma
  - Dollar sign
  - Colon
  - Left and right arrow
  - Equal sign
  - Hyphen
  - Single quote.

    Note: For a list of characters and graphics, see Figure 7-8 on page 7-9.

- Plus sign character.

  The plus sign special character requests a double-wide (DW) character to print. To print a character as a double-wide character (for the 3890/XP Enhanced only), enter a plus sign in front of that character. For example, enter the following to request a PAID stamp in the endorsement text:

  ```
  +P+A+I+D
  ```

  The plus sign is not a valid character unless you pair it with a valid character (on the same line).

  If you request double-wide characters and you suppress double-wide conversion (with the SpxNoDWConv function), the plus signs print as question marks to indicate invalid characters.

Special codes for large characters can be present in the Active Endorsement Area (AEA). For more information about special large characters, see Figure 7-9 on page 7-10.

24-8 Font for Normal Size Characters:

For the 3890/XP and 3890/XP Enhanced only, the following table lists the valid endorsement characters and their character numbers and hexadecimal values.

Figure 7-8. 24-8 Font Characters

Character	Character No.	Hexadecimal	Character	Character No.	Hexadecimal
Space	032	20	A	065	41
#	035	23	B	066	42
$	036	24	C	067	43
&	038	26	D	068	44
'	039	27	E	069	45
(	040	28	F	070	46
)	041	29	G	071	47
*	042	2A	H	072	48
,	044	2C	I	073	49
-	045	2D	J	074	4A
.	046	2E	K	075	4B
/	047	2F	L	076	4C
0	048	30	M	077	4D
1	049	31	N	078	4E
2	050	32	O	079	4F
3	051	33	P	080	50
4	052	34	Q	081	51
5	053	35	R	082	52
6	054	36	S	083	53
7	055	37	T	084	54
8	056	38	U	085	55
9	057	39	V	086	56
:	058	3A	W	087	57
<	060	3C[1]	X	088	58
=	061	3D	Y	089	59
>	062	3E[1]	Z	090	5A
			'	096	60
				127	7F (Reserved)

[1] A solid triangle prints.

ARW Font for Special Large Characters

A single double-wide (DW) character in transport-printable format is actually a cluster of 2-character codes. For example, the Control Program translates

+A

into the following hexadecimal character code cluster:

80,81

This character code cluster prints a single double-wide character **A**. You can perform the translation in your module and also set a flag to suppress Control Program translation of double-wide endorsement text. For information about suppressing double-wide character translation, see page C-13.

Figure 7-9. ARW Font Characters

Character	Character No.	Hexadecimal	Character	Character No.	Hexadecimal
A	128,129	80,81	V	170,171	AA,AB
B	130,131	82,83	W	172,173	AC,AD
C	132,133	84,85	X	174,175	AE,AF
D	134,135	86,87	Y	176,177	B0,B1
E	136,137	88,89	Z	178,179	B2,B3
F	138,139	8A,8B	0	180,181	B4,B5
G	140,141	8C,8D	1	182,183	B6,B7
H	142,143	8E,8F	2	184,185	B8,B9
I	144,145	90,91	3	186,187	BA,BB
J	146,147	92,93	4	188,189	BC,BD
K	148,149	94,95	5	190,191	BE,BF
L	150,151	96,97	6	192,193	C0,C1
M	152,153	98,99	7	194,195	C2,C3
N	154,155	9A,9B	8	196,197	C4,C5
O	156,157	9C,9D	9	198,199	C6,C7
P	158,159	9E,9F	>	200,201	C8,C9
Q	160,161	A0,A1	<	202,203	CA,CB
R	162,163	A2,A3			
S	164,165	A4,A5			
T	166,167	A6,A7			
U	168,169	A8,A9			

INF Item-Numbering Data

For all of the 3890/XP Enhanced document processors, the Sort program can specify the INF number or the alternate INF data. For more information about printing alternate INF data, see “Alternate INF Indicator and Data” on page C-8.

For the 3890/XP Enhanced only, the INF data consists of 8 or 10 numeric characters printed directly below the endorsement text. For a detailed description of initialization format, see the discussion of bytes 0 through 5 and bytes 81 through 87 on pages 2-5 and 2-16.

10NZ Font for INF Characters

The INF characters are larger than the endorsement characters. The following table lists the characters for the 10NZ font.

Figure 7-10. 10NZ Font Characters

Hexadecimal Digit Specified	Character Printed
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	(blank)[1]
All other characters	?[2]

[1] Only permitted in alternate INF field.

[2] Also flagged as an INF error.

Endorse Setup/Verify Screen

The Endorse Setup/Verify screen automatically appears when the run initialization activates the endorsement feature or when the operator uses the ENDorse command.

When you specify the PEDProtect keyword in the Run Profile, the 3890/XP Enhanced Control Program does not display the Endorse Setup/Verify screen and the ENDorse command is blocked.

The Endorse Setup/Verify screen presents the following information:

- The filename of the Programmable Endorsement Data (PED) file from which the Control Program last loaded the endorsement text.
- The endorsement text. The screen displays the unresolved %D..D and *+DW* variables.
- Endorsement date.

The XP Enhanced document processor operator can view and verify the endorsement text, but cannot change it, except for the date (see below).

The ENDorse Command

The ENDorse command displays the Endorse Setup/Verify screen. Without a parameter, the endorsement text is from the "last read xxx.PED" file. The ENDorse command also accepts a file specification of a PED file as a parameter.

Note: Enter the file specification without the extension PED. The Control Program supplies the file extension. If you do not specify the drive and path, the XP Enhanced document processor searches the drive and path specified in the Machine Profile for Run Profiles.

The ENDorse command copies the file to both the "last-read xxx.PED" and the Active Endorsement Area (AEA).

Endorsement Control

Information for the endorsement feature appears in the initialization data record (IREC) and in the Run Profile. There are two initialization fields that affect the feature:

Endorse date An 8-character field (IREC bytes 88 through 95) that the XP Enhanced document processor can print as part of the endorsement.

Run Profile name An 8-character field (IREC bytes 70 through 77) that contains the name of the Run Profile. The Run Profile can specify a PED file.

The XP Enhanced document processor gives you flexibility in initializing and updating endorsement text, ranging from total operator control of endorsements to total host program control of the endorsements.

Control of the Endorse Date

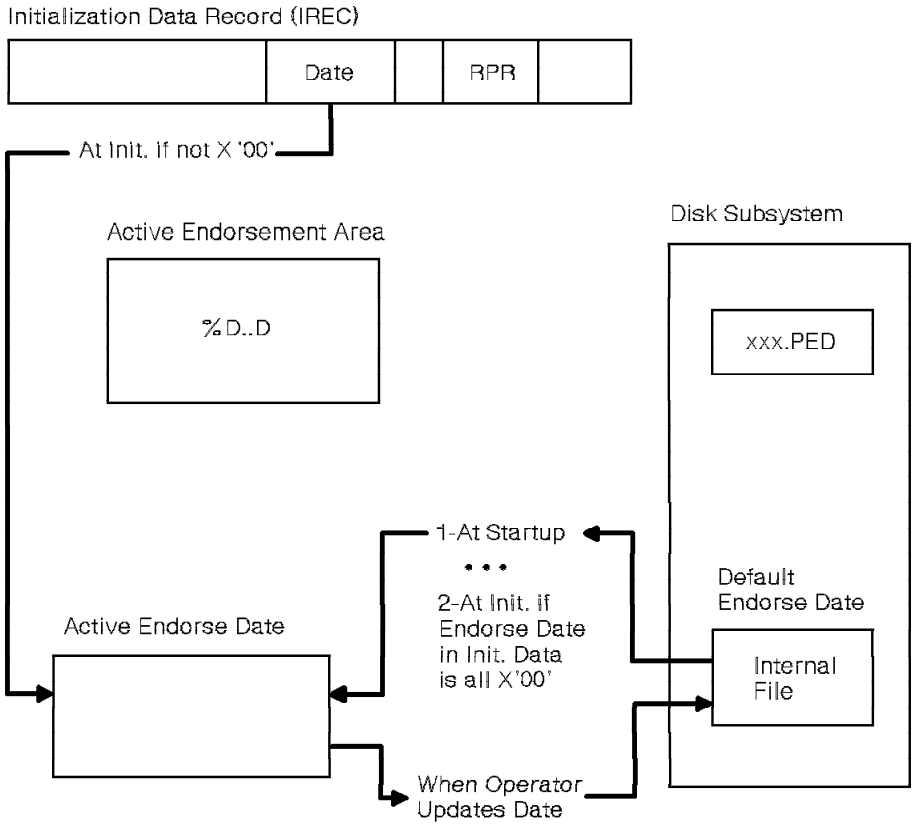

Figure 7-11. Specifying the Endorse Date

The operator sets the Active Endorse Date on the Endorse Setup/Verify screen by changing the “Run Date” input field and pressing the Enter key. The Active Endorse Date is used to resolve the %D..D when documents flow. This allows you to maintain the same date when going past midnight. If the SpxPutNoDateIns function is active, the field prints a question mark (?) for the % and the Ds print as they are.

Each time the operator sets the active endorse date from this display screen, the 3890/XP or 3890/XP Enhanced Control Program puts the new date into the AEA of AUX storage (X'A690-A697'). The SCI “LAS” macro can read the AUX storage area (Load Work Register from Auxiliary Storage).

When the Control Program first starts, the Active Endorse Date is set from an internal file. At initialization time, it is set to the date in the IREC bytes 88 through 95 if given, and the internal file is updated. Therefore, the default endorse date is the last date set.

Note: The Active Endorse Date uses the %D..D variable in the text of the endorsement to position the date in the text. Only the % and the first seven Ds are substituted with the date.

Control of Endorsement Text

You create endorsement text by creating PED files. To create these files, you can use either the 3890/XP Toolkit I or an ASCII editor. For more information, see "Programmable Endorsement Data (PED) File" on page 3-15.

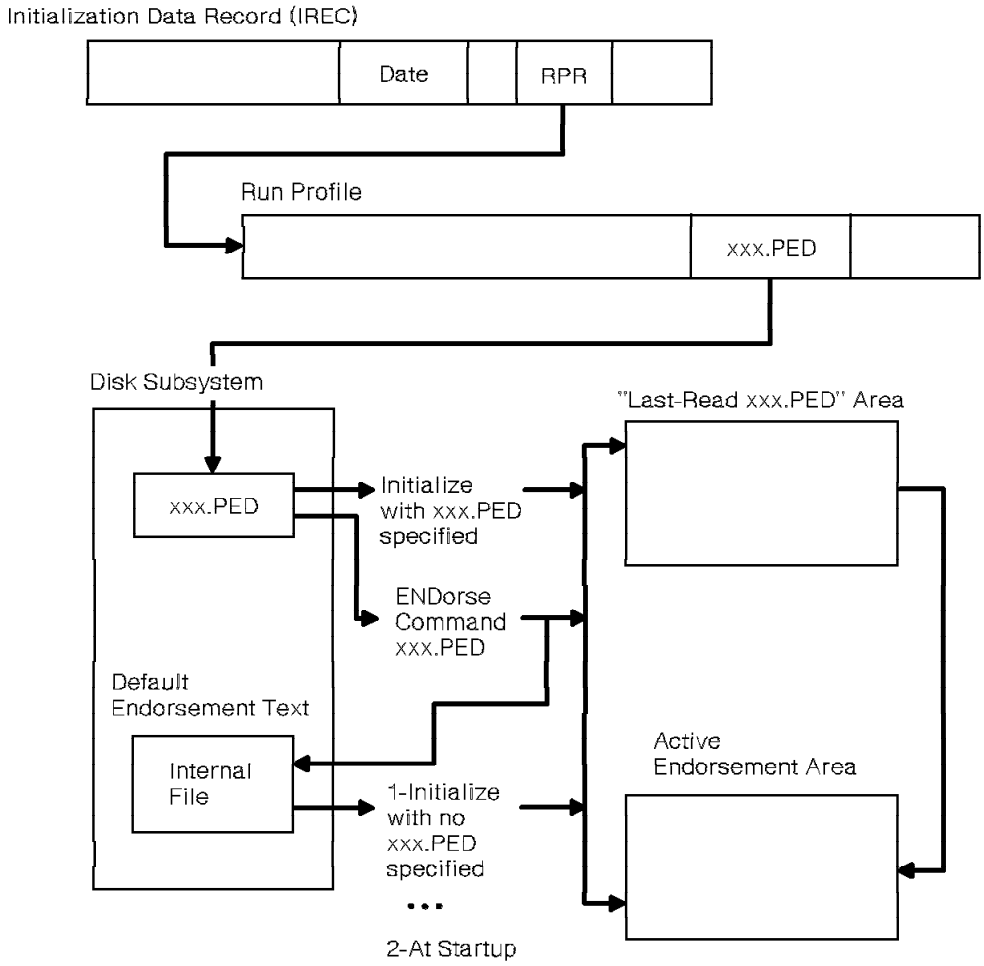

Figure 7-12. Specifying the Endorsement

The Control Program updates the Active Endorsement Area (AEA) from the disk subsystem when any of the following occur:

- The Control Program first starts and updates the AEA from its internal file on the disk subsystem.
- The XP Enhanced document processor initializes a Sort run and updates the AEA from the specified PED file.
- The XP Enhanced operator issues the ENDorse command and specifies a PED file. This action also stores the internal file.

The "Last-Read xxx.PED" Area

The "last-read xxx.PED" area is an unresolved version of the endorsement text. It is used for the Endorse Setup/Verify screen.

The characters %D..D and *+DW* appear in this area. These are updated to the correct coded form in the AEA when the first document processes with the AUX storage flags set correctly.

Single Endorsements

For installations using only one endorsement, you specify the PED file in the Run Profile or the default Run Profile. This sets up the correct endorsement for every job run and makes every initialization the same. You can bypass the automatic display of the Endorse Setup/Verify screen for the endorsement after every initialization by specifying the PEDProtect keyword in the Run Profile.

Note: Some single-endorsement users might decide to omit the PEDProtect keyword to permit the operator of the XP Enhanced document processor to change the date.

If you need a single endorsement for most jobs, you specify a default and have the XP Enhanced operator override the endorsement for the alternate endorsements.

Operator-Controlled Endorsements

Where the XP Enhanced operator is responsible for the date and the endorsement, Run Profiles can omit the PRGENDRSdata keyword (or include the keyword with no parameter).

If the text shown on the Endorse Setup/Verify screen is correct, the operator returns from that display screen. If the Active Endorse Date should change, the operator inputs the correct date on the display screen and confirms the change. If the endorsement should be changed, the operator issues an ENDorse command from the Run screen and specifies a PED file. If the file is not found, the document processor displays the "last read xxx.PED" text and an error message.

Sort-Program-Controlled Endorsements

Although the Sort program changes the endorsement dynamically in the AEA, the Endorse Setup/Verify screen displays the last endorsement text as last loaded.

Operating Principles for Endorsement and Item Numbering

Following are the operating principles for endorsement and item numbering.

- The print position for the endorsement, if initialized on, also determines the INF print position.

  Note: The Control Program checks both the vertical and the horizontal print position to determine the location of the endorsement for the 3890/XP Enhanced document processor. There are six possible positions. If the print head is in the wrong position, an endorse sector error appears on the Run screen after the run is initialized and the Start key is pressed.

- If INF is initialized on, the INF number in AUX storage is updated by using document data record header byte 5, bit 6 and/or bit 7 (the feature control byte).

- Document data record header byte 5, bits 4 and 5 inhibit the printing of the endorsement and INF number.
- IREC byte 82, bit 2 inhibits the printing of an alternate INF number. Changing this bit in the IREC after initialization has no effect.
- Document data record header byte 7, bit 0 and bit 6 indicate that the endorsement and the INF features are on.

Fixed Endorsement

Fixed endorsement is a feature for the 3891/XP, 3892/XP, and UT/XP document processors and can use either the front ink jet, the back ink jet, or both, as specified in the initialization data record.

The 3891/XP and 3892/XP support 15 fixed messages of up to 48 characters each. The UT/XP supports 15 fixed messages of up to 66 characters each. These fixed messages are defined by the user Sort routine at SPSINIT time via the SpxLoadFixM function. The UT/XP allows fixed messages of up to 66 characters per inch.

At SPSDOC time, the user Sort routine calls the SPXFixM function to specify the message numbers for fixed endorsement for the current document, if any. There can be up to three messages for the front side and up to three messages for the back side. For example, a document directed to a particular module-pocket can be given a special fixed endorsement.

The fixed endorsement cannot be printed on the same side with the 3-line “converged” endorsement.

The Converge DLL

The CONVERGE.DLL contains a program that aids you when running a sort program written for a 3890/XP or 3890/XP Enhanced on a 3891/XP, 3892/XP, or UT/XP. The CONVERGE.DLL formats and copies the 3890/XP or 3890/XP Enhanced Active Endorse Area (AEA) to an endorse data area used by the 3891/XP, 3892/XP, and UT/XP.

The CONVERGE.DLL file is supplied with the 3891/XP, 3892/XP and UT/XP control program. To run CONVERGE.DLL, you specify in your Run Profile that CONVERGE is to be run at the SPSPOST event.

CONVERGE.DLL uses these rules when formatting and moving Active Endorse Area data to the data area used by the 3891/XP, 3892/XP, and UT/XP:

1. Lines 1 and 2 each print 24-characters in print lines 1 and 2 of the 3891/XP and 3892/XP endorsement.
2. Print line 3 in the 3891/XP and 3892/XP endorsement has the INF in the rightmost 10 positions.
3. If INF is not being used, then the eight rightmost positions in line 3 are ignored and do not print. The print line is shifted to the right.
4. “Double Wide” printing uses underscore characters, except that the underscore is omitted from the last double-wide character of each word on each line. For example, +P+A+I+D converts to P_A_I_D.

This “converged” endorsement can be printed on the front side or the back side (initialization data byte 64, bit 3).

Full Function Endorse

Full Function endorsing is a 3891/XP, 3892/XP, and UT/XP document processor method of endorsing documents that allows more variations of endorsement data than is available when using the CONVERGE.DLL. You replace CONVERGE with your own sort program module(s) that use the SPXServ functions that allow control of the Endorse feature. With Full Function endorsing, you can perform the following functions beyond those available with the CONVERGE.DLL alone:

- Provide the INF capability on any endorsement line in any line position
- Put any combination of variable, fixed, or serial number data on any endorsement line, front or back
- Provide large font capability for both sides of the document (3891/XP and 3892/XP only)
- Have the placement of the third endorsement line remain constant with or without the INF feature while using the existing Converge DLL
- Invert or rotate any or all of the endorsement lines

SPXServ Functions that support full function Endorse

You can use the following list of SPXServ functions in your sort program to provide full function endorse. For details on the these functions, see the *IBM 3890/XP Series SPXServ Reference.*

SpxLoadFixM
This Spx is the only way to specify the fixed messages to the machine during the initialization process. Any fixed message used during an operation must be loaded during initialization before being requested at document time.

SpxFixM This Spx specifies an all-fixed message endorsement, as well as fixed message support for converged endorsements. If you use SpxFixM and SpxPutInkJetCtrl to specify the same line of endorsement, the machine returns an option state error and nothing is printed.

SpxPutPrtSeq
This Spx is used to specify the sequence of the fixed message, variable text, and serial number on each endorsement line.

SpxInkJetCtrl
This Spx is used to control what is printed on each endorsement line.

SpxPutAltlnf
This Spx puts an alternative item number to the machine. The Spx remains the same.

Note: The SpxGetPrgEndData and SpxPutPrgEndData are used to get and put the 3890/XP and 3890/XP Enhanced Active Endorse Area (AEA). Since full function endorse does not use the AEA, these functions are meaningless when using full function endorse.

Full Function Endorse Usage

To enable full function endorse, your sort program can use these SPXServ functions during the SPSINIT event to perform the following function:

- SpxPutPrtSeq - Set up the requested print sequence.
- SpxLoadFixM - Define up to 15 fixed messages.
- SpxPutInkJetCtrl - Set up any or all lines of endorsement data. This Spx can be used at any event, allowing the endorsement to be changed at any time.

During the SPSPOST event, your sort program can use SPXServ functions to complete or change the endorsement data definitions. Multiple (up to 6) SpxPutInkJetCtrl commands can be issued for each document. The data area initially is all zeros. When an endorsement has been requested, it remains the same until acted on by a subsequent SpxPutInkJetCtrl command. The endorsement data area is cleared between sort program initializations.

Roll-On Endorser

The optional Roll-On (Legend/Date) endorser for the 3891/XP and 3892/XP prints a fixed imprint endorsement on the back of requested documents.

Endorsement Position

The Sort program can specify which one of three horizontal positions is to be used for the placement of the Roll-On endorsement. This specification is done by IREC byte 79, bits 4 through 7 and applies to all documents.

Selection

To use the Roll-On endorser, you must set IREC byte 78, bit 7. The Roll-On endorser can be turned on or off on a per-document basis.

Error Conditions

Various hardware failures can prevent proper Roll-On endorser operation. These will be reflected as Roll-On endorser faults.

The 3891/XP and 3892/XP document processors will not start document feeding with a Roll-On endorser fault condition even if the endorser is not selected. If such a condition exists when the “Feed” key is pressed, the transport motors stop and an error message is displayed.

An attempt to endorse in a horizontal position that would not allow the endorsement to fit entirely on the document causes a “Document Too Short to Endorse in Specified Position” error. Since the endorsement is 1.5 inches long, the horizontal position used must be at least 1.5 inches from the trail edge of the document.

Chapter 8. Host Communication

This chapter discusses the following topics:

- General control
- Restart
- APPC host interface
- System/370* channel interface.

General Control

You operate the 3890/XP Enhanced document processor online to the host system for the following reasons:

- To load the initialization data and the user-supplied Sort programs
- To send document data, sort results, and machine status to the host system
- To write from the host system's code line data match file to the 3890/XP Enhanced document processor when code line data matching.

You can attach the document processor to the host system in one of the following ways:

Channel Interface
The 3890/XP Enhanced offers the System/370 channel-attached method of host attachment.

APPC Several types of host systems offer the APPC method of host attachment through different types of links and general products.

With either communication, the host system controls the document processor by sending commands and receiving responses. This section discusses the following:

- Document-feeding control
  - Offline
  - Online
  - SCI pause and Unit Exception
- Data transfer
- Diagnostic commands.

Document-Feeding Control

From the ready condition, the document processor can control document feeding in the offline condition or the online condition.

Offline

Offline, documents feed continuously or one at a time. The Start key (for the 3890/XP Enhanced) or Feed key (3891/XP or 3892/XP) feeds documents continuously until a pause or a stop occurs. For the 3890/XP Enhanced only, the Single Document key feeds a single document.

* Trademark of IBM

You enable the Start key or the Feed key by performing the following steps (the 3890/XP Enhanced Single Document key is enabled as well):

1. Turn the power on.
2. Display the Run screen.
3. Load a user-supplied Sort program.
4. Set the Sort program switches as required.

The document processor is then ready to start feeding documents.

Online

To feed documents online:

1. Select online processing with the operator console switch (3890/XP Enhanced) or the operator command COMM ON (3891/XP, 3892/XP, and UT/XP).
2. Optionally set the Sort program switches.
3. Enter the Ready condition by pressing the Start or Feed key.

   If features initializations are requested, the transport will be disengaged. Pressing the Start or Feed key a second time starts document feeding.
4. For a *Channel Interface* host attachment, the document processor receives the anticipate CCW from the host system. For an *APPC* host attachment, the document processor receives the start running command from the host system.

Feeding continues under the control of the document processor until a pause or a stop occurs.

Pauses occur when the document data buffer is almost full (24 documents short of the 64-record limit) or the code line data match buffer is almost empty (40 documents short of the 64-record limit). A pause also occurs when any of the following conditions occur:

- A Sort program-requested pause (SCI or SPX)
- A microfilm pause for spacing
- A pocket-full condition
- An Image pause - ICP feed stop.

The document processor automatically resumes feeding when you clear the condition that caused the pause.

The document processor stops when any of the following conditions occur:

- A hardware stop caused by any of the following:
  - Document jam
  - Document runout
  - Stop key
  - Time out (during a pause)
  - Microfilm end-of-reel.
- For a *Channel Interface* host attachment, the host program issues an anticipate-and-stop CCW.
- For an *APPC* host attachment, the host program issues a stop running command.
- The Sort program issues a stop command.

- A late module-pocket auto-select occurs as the result of an SCI error. (The 3891/XP, 3892/XP, and UT/XP do not stop.)
- A code line data match end-of-file condition occurs in debug mode.

When a stop occurs, the document processor might require several more read commands before the not-ready condition occurs. The read operation that transfers the last record from the read buffer must end with a normal end status before the not-ready condition can occur. If the stop occurs because of an exception, the read operation also transfers an exception record. This last block transferred is padded with hex 00s to the required length.

SCI Pause and Unit Exception

The user-supplied Sort program issues the SCI pause to let the host system examine the data and determine what it wants to do with that data. The document processor transfers all read records to the host system before sending unit exception (UX) to indicate the SCI pause. If the code line data matching function is activated, the document processor purges all of the code line data match records from the code line data match buffers.

The SCI pause is activated when the read record header byte 9, bit 7, is set on for the document that started the pause.

After the UX is sent, the document processor pauses until the host system sends a new command. The document processor can stop if an intervention-required condition occurs (such as a jam or a stop key). The host system responds according to what operations are in progress.

When the document processor is not performing code line data matching, the host will respond in one of the following ways, depending on the host program that is operating:

1. For the *Channel Interface* host attachment, the host system replies with the anticipate CCW to start feeding for normal read operations. Usually, the Sort program pauses on batch tickets and lets the host system verify the batch.

   For the *APPC* host attachment, the host system replies with the start running command to start feeding for normal read operations. Usually, the Sort program pauses on batch tickets and lets the host system verify the batch.

2. For the *Channel Interface* host attachment, the host system replies with the anticipate-and-stop CCW and ends the feeding because the host system read the wrong batch ticket.

   For the *APPC* host attachment, the host system replies with the stop running command and ends the feeding because the Sort program read the wrong batch ticket.

3. The host system replies with an initialize sequence to re-initialize the document processor because the host system wants to change the sort pattern.

When the document processor is code line data matching, one of the following occurs:

1. For the *Channel Interface* host attachment, the host system replies with the anticipate-and-stop CCW and ends the feeding.

   For the *APPC* host attachment, the host system replies with the stop running command and ends the feeding.

   Because the host system determines that the documents are not in the expected sequence, you should correct the situation and restart the work.

2. For the *Channel Interface* host attachment, the host system replies with the anticipate CCW to start feeding for normal operations.

   For the *APPC* host attachment, the host system replies with the start running command to start feeding for normal operations.

   Because the document processor purged the code line data match buffers and code line data matching is active, the document processor requests new code line data match data from the host system. After the code line data match data primes the buffers, feeding proceeds normally.

3. The host system replies with an initialize sequence to re-initialize the document processor for another job run.

Data Transfer

At the start of each job, the host system initializes the document processor with new user control data. The user control data consists of the initialization data record (IREC) and optionally the SCI module of the user-supplied SCI Sort program.

Loading the User Control Data

The host system downloads the user control data to the document processor system unit.

For *APPC* information, see "Load Commands" on page 8-14.

For *Channel Interface* information, see "Channel-Command Summary Charts" on page 8-45.

The set-load-mode command assures that the document processor is in the correct state for initialization. See "Loading and the Running Condition" on page 8-5.

The load commands actually transfer the user control data. The 3890/XP Enhanced Control Program loads data sequentially into program storage, starting at byte 0. The first 128 bytes (bytes 0 through 127) are the initialization data record (IREC). The following bytes are SCIs and associated tables.

The end load command initiates the actions and the checks that the IREC specifies by invoking the initialization routine.

For *Channel Interface*, the initialization routine runs between channel end status and device end status of the end load command.

After the initialization routine processes the IREC and the Run Profile, the user-supplied initialization Sort module runs (the SPSINIT event). If any of the routines detect an error, they inform the host system and force the document processor into the not-ready condition. For information about error codes, see Appendix A, "Run Initialization Error Codes."

Running Condition: The running condition prevents the host system from re-initializing the document processor before the host system accepts all of the data that the preceding operations produced. The Active light on the operator panel indicates the running condition.

The running condition implies that the host system has started document feeding and that the document processor can create new data. On the 3890/XP Enhanced document processor, the running condition continues during document jams; if more jams occur after you clear the jam, the runout operation produces new read records and exception records. On the 3891/XP and 3892/XP document processors, a runout operation never occurs; when the sorter is not ready, it enters a not-running state.

The document processor enters the running condition when it accepts an anticipate CCW or the start running command. When the document processor is in the not-ready condition, it resets the running condition when the following occurs:

- There are no documents trapped in the transport.
- The sorter produces a unit exception (UX) condition.

For more information about the UX condition, see "SCI Pause and Unit Exception" on page 8-3.

Loading and the Running Condition: Loading the user-supplied control data causes an initialization attempt that destroys the records that have been created but not yet accepted by the host system. To prevent loss of records, the document processor blocks the load procedure while the possibility of generating more read records or exception records still exists. This protection accepts the first command of the load sequence only when the document processor is in the not-running condition and the ready condition.

The document processor accepts the set-load-mode command, the first command in the load sequence, only when the not-running condition and the ready condition are present. For more information about document feeding from the ready condition, see "Document-Feeding Control" on page 8-1.

Diagnostic Commands

Warning: The diagnostic commands are powerful tools that should be used with caution. If you use the diagnostic commands to change control areas within your Sort program, you might change the operation of your sorter.

The document processor aids the following diagnostic commands:

- Diagnostic Initiate (for *Channel Interface*)
- Diagnostic Key
- Diagnostic Read
- Diagnostic Write.

The following are some specific rules that you should follow when you use the diagnostic commands:

1. The document processor does not allow the diagnostic sequences when in the running-and-ready condition.
2. Each read or write diagnostic sequence must start with a diagnostic key command. For example, diagnostic key, diagnostic read, diagnostic read is not a valid command sequence. Diagnostic key, diagnostic read, diagnostic key, diagnostic read is a valid command sequence.
3. Writing data is only permitted in program storage. Writing data is not a recommended technique.
4. When you use diagnostic commands to read auxiliary storage data, you must never read across a 256-byte boundary in a single command. There is no interlock to prevent this but, when it occurs, data read-back is unreliable. **Use with caution.**
5. When you use diagnostic commands to read program storage data, you must never read across a 64KB boundary in a single command. There is no interlock to prevent this but, when it occurs, data read-back is unreliable. **Use with caution.**

Safeguards

The document processor protects the diagnostic commands against unintentional use. All diagnostic commands, except the diagnostic key command, return the unit-check (UC) status as the initial status unless the last command was a diagnostic key command. Because diagnostic commands can destroy internal document processor data, the document processor rejects the commands if it is in the running condition and the not-IRV condition.

Operation

The diagnostic read/write commands transfer the data to and from the document processor while the control data, transferred by the diagnostic key command, directs the data.

The first byte of the control data in the diagnostic key places the data in one of the two following formats:

1. 3890 emulation format
2. 3890/XP Enhanced document processor format.

   Note: The 3890/XP Enhanced format permits access to larger storage facilities.

See the following overview examples for the format of the diagnostic keys.

Diagnostic Key Data - 3890 Emulation Format:

```
Byte 0     (First byte written)

       bits 0-2   Reserved.
       bit  3     0  - Declares this format.
                  1  - 3890/XP Enhanced Format - see
                       following section.
       bits 4-5   Reserved.
       bits 6-7   Coded storage type

                  00 - Control    (Command Reject on following Read/Write).

                  01 - Auxiliary  (Note: AUX storage format is not
                                         identical to format for Models A and B.
                                         AUX Storage is Read only.
                                         See "Auxiliary Storage" on page C-7.)

                  10 - Main  (Program Storage).

                  11 - Error (Command Reject is on following Read/Write).
Byte 1     Hexadecimal length of data to be transferred—
           ignored if byte 0, bit 3 is 1.
           (Here X'00' means transfer 256 bytes.)

Byte 2-3   3890 hexadecimal address of first byte to be transferred.

Byte 4-7   Reserved and ignored for this format.
```

Diagnostic Key Data - 3890/XP Enhanced Format:

```
Byte 0     (First byte written)

       bits 0-2   Reserved.
       bit  3     1  - Declares this format - must be 1.
                  0  - 3890 Emulation Format - see preceding section.
       bits 4-7   Reserved.

Byte 1     3890/XP Enhanced Component - hexadecimal code as follows:

                  0  - Not valid (Command Reject on Read/Write).
                  1  - Program Storage (Read/Write capability).
                  2  - Reserved for future use (Command Reject on Read/Write).


Byte 2-3   Hexadecimal length of data to be transferred
           (maximum length permitted is 4096).

Byte 4-7   3890/XP Enhanced hexadecimal address of first
           byte to be transferred.
```

Note: Length or address is in the IBM form of high/low, not in the Intel** form of low/high.

Restart

The purpose of restart is to save enough data records within the document processor so that if a system- or host-level crash occurs, the user is able to retrieve data without having to rebatch and rerun the work. Restart is not intended to protect the user against a loss of power to the document processor or a severe system failure within the machine.

Description

Restart data is saved at all times that the document processor is processing documents online to a host. This is referred to as a "paper run." Only one Restart file (C:\CONTROL\RESTART.DAT) is maintained in the document processor. The restart file contains the data for the last online paper run. Data is not saved to this file while running offline. The host program during the Paper run can supply a unique RUN ID by setting bytes 62 and 63 of the initialization data record (IREC). This is used in a subsequent Restart run.

The document processor associates an index number with every record written in the Restart file. The first record after an online initialization is assigned an index number of 1, and the following records are assigned sequentially higher numbers. The Restart Index Number (RIN) is the number associated with the oldest (first-read) record in the Restart file.

During the run, the records are kept in a 64KB buffer which is written to the file after the run or when the document processor stops. Therefore, in a large run, the buffer might wrap several times (the exact number depends on the record size). Note that, whenever the RIN is 1, all the records processed since the last online initialization are in the Restart file.

The restart data is retrieved by the host in the process referred to as Restart. When the host wants to Restart it must send the initialization data record (IREC) that specifies a Run Profile named "Restart" and set bytes 62 and 63 to the same RUN ID that was used in the paper run. The Restart IREC must *NOT* contain feature initialization. The IREC and SCI code must be less than 64KB in size.

During a Restart run initialization, the document processor performs checks to validate the run. The RUN ID sent is compared against the RUN ID saved in the file. They must match or an initialization error occurs. The record length specified by the new IREC must match the record length stored in the file or an initialization error occurs. The initialization failure codes are available to the host as sense byte 1 and are presented to the operator as messages. The RIN is made available to the host by doing a diagnostic read to Auxiliary storage address X'00D4'. The RIN is a 4-byte unsigned binary number in the "IBM format" high to low (for example, RIN 1A3B is stored as 00 00 1A 3B). The RIN is always available, and the host should read it before starting a Restart run.

Once a successful Restart initialization occurs, the transport is stopped but the Control Program remains on the Active screen until all data has been taken by the host. The host

** Trademark of Intel Corporation

must take the data in the same manner that it takes normal document data. All document-type records (for example, type 80 or 20 headers) are sent along with any exception-type records (for example, type 40 headers). No pad records are sent other than to pad the last block transferred to the required length. The records are sent oldest record first. After all of the records are transferred, the sense byte 1 is set to an end-of-transfer completion code.

It is the responsibility of the host to inform the operator if, for any reason during the Restart run, data transfer cannot be completed. Once the host has informed the operator that the transfer has failed, the operator can force the 3890/XP Enhanced document processor to end by switching the online switch to offline. There are no messages or instructions to the operator from the 3890/XP Enhanced during this operation.

Restart Completion Codes (Sense Byte 1)

The return codes set for the restart operation include one message to indicate a successful run, and three initialization errors. These messages/return codes are handled identically to any other run initialization return code, returning the host status in sense byte 1, and displaying a message on the Run screen. The following completion codes are used with Restart:

X'95' The record length supplied in the Restart IREC does not match the record length from the Restart file.

X'96' The Restart file is empty or unusable.

X'97' The RUN ID from the Restart IREC does not match the RUN ID from the Restart file.

X'98' The data from the Restart file has been transferred to the host (end of transfer).

APPC Host Interface

The APPC host interface component of the 3890/XP Enhanced Control Program lets you use a variety of hardware and software products to connect your document processor to different hosts. APPC is the implementation of the Systems Network Architecture (SNA) LU 6.2.

This section gives you details that you need when you write host application programs for your communications network. You should also be familiar with the material in "General Control" on page 8-1.

For more information about the System/370 channel interface, see "System/370 Channel Interface" on page 8-35.

Allocate Parameters

There are several different types of LU 6.2 conversations. The parameters that the host system uses at conversation allocation determine the type of LU 6.2 conversation that will run.

The host system uses two required LU 6.2 conversations and one optional LU 6.2 conversation with the following allocate parameters. The host system is aware of all of these conversations.

The following is a required conversation with the "XP.Inbound" transaction program. The host system sets up this conversation to transfer all the commands except the write command (which is for code line data) from the host system to the document processor:

- **type**(BASIC_CONVERSATION)
- **sync_level**(CONFIRM)
- **security**(NONE)

The following is a required conversation with the "XP.Outbound" transaction program. The host system sets up this conversation to transfer all the responses from the document processor to the host system:

- **type**(BASIC_CONVERSATION)
- **sync_level**(NONE)
- **security**(NONE)

The following is an optional conversation with "XP.Write.Data". The host system sets up this conversation to transfer code line data from the host system to the document processor whenever code line data matching is active:

- **type**(BASIC_CONVERSATION)
- **security**(NONE)

Record Level Buffering

The channel-attached document processor transfers the document records to and from the host system in fixed 16-record blocks (subdivided into four sets of four-record transfers each). With APPC, the communication network that connects the host system to the document processor controller is different for each customer installation. The blocking size for the document records is a system tuning parameter.

In the APPC-attached document processor, APPC handles the record level buffering and sends and receives logical records. Each logical record that APPC transfers from the host

system to the document processor contains a single command. Each logical record that APPC transfers from the document processor to the host system contains a single response. The host system can set up the read record response to contain multiple document records.

The use of BASIC conversations lets the Control Program copy multiple logical records to the APPC buffers on each call to APPC. The host system cannot distinguish what groups of logical records that the Control Program copied together. APPC accumulates these logical records until a complete request unit-sized collection of data is available. The OS/2 EE system configuration parameters determine the request unit (RU) size; the Control Program does not know the RU size. APPC also buffers the records that are transferred from the host system to the document processor.

Sense Data

If the host system requests sense data directly from the document processor the same way that the channel attachments request sense data, the performance of the document processor would degrade. The communication network resources and the associated path lengths between the host system and the document processor would cause poor performance. To prevent this, the document processor sends the sense data to the host system only when the sense data changes. The host system maintains a copy of the sense data for its own operation.

Offline Operation

The document processor can run in the offline condition. When this occurs, the document processor restricts the set of APPC operations to allocation related functions.

The document processor restricts the APPC operations when it is in the offline condition by holding the document processor logically, as seen across the APPC interface, in the not-ready, not-running condition, regardless of the physical condition of the document processor. In addition, the diagnostic operations receive the command reject response when the document processor is in the offline condition.

Conversation Initiation

The host system starts document processor conversations. Transaction programs are pieces of code that communicate with other transaction programs and are contained in an application program. The Control Program contains up to three transaction programs.

Command and Response Logical Records

You can send each command or response by issuing an APPC SEND_DATA verb. (For more information about verbs, see “APPC Verbs for the 3890/XP Enhanced” on page 8-27.) In APPC basic conversations, the first two bytes of each logical record contain a length field. The data for the logical record immediately follows the length field. The next two bytes that the SEND_DATA verb sends represent the command code or the response code. To form the logical record that is sent to APPC, the document processor, or the host system, appends any data that accompanies the command code or the response code. Figure 8-1 on page 8-12 displays the format of a logical record.

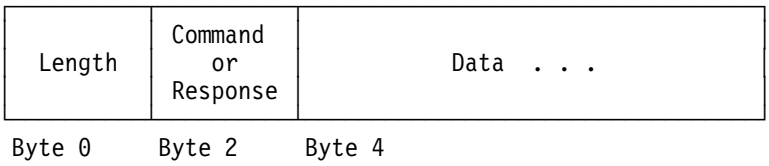

Figure 8-1. Logical Record Format

Host-Initiated Command Sequences

The host system can send the document processor several commands that correspond to the CCW interface. The host system groups the valid command sequences into the following four sets:

1. Commands that you can issue at any time.

   The document processor also allows conversation allocation and deallocation at any time.

   (For more information about these commands, see "General Commands." For more information about conversation allocation and deallocation, see "Conversation Initiation and Termination" on page 8-23.)

2. Commands that initiate the load mode.

   You can enter the load mode only when the document processor is in the not-running condition and in the ready condition.

   (For more information about these commands, see "Load Commands" on page 8-14.)

3. Commands that you can use during the normal document feeding operation.

   To start feeding documents online, the document processor must be in a ready condition and in an initialized condition.

   (For more information about these commands, see "Run Commands" on page 8-16.)

4. Commands that are part of the diagnostic command sequences.

   The document processor does not allow the diagnostic command sequences when it is in a running-and-ready condition or when it is in an offline condition.

   (For more information about these commands, see "Diagnostic Commands" on page 8-18.)

Note: You cannot use more than one set of commands from the last three sets in the preceding list at the same time. You must complete each set of commands before starting another set. You can, however, nest the diagnostic commands on top of a load sequence.

General Commands

You can issue the general commands when the document processor is in any state; the document processor does not need to be online to accept these commands.

Request Sense ID

The request sense ID command requests that the document processor transfer machine identification data to the host system.

You should issue an APPC FLUSH verb after the request sense ID command. The APPC FLUSH verb transfers the command from the host APPC buffers to the document processor. The document processor replies with a sense ID response. For more information about the sense ID response, see page 8-20.

Set Allocation Status

The set allocation status command sends the allocation status string from the host system to the document processor. The set allocation status command lets the host application program send the identifying information (such as the application program ID, the sorter ID, and other identifying information) for display. The 3890/XP Enhanced document processor stores the status string in memory and displays it on the document processor Run screen.

The following field is on the 3890/XP Enhanced document processor Run screen:

`Host Allocation ID`

The document processor uses the allocation status string to fill in the blank space in the allocation status field. The allocation status string replaces any previous data that was in this field. The allocation status string is padded on the right with spaces to cover up any non-blank characters that could have been in a previous allocation status string. When the host system does not allocate the conversations, the allocation status field displays the `«Not Allocated»` message. When the host system first allocates the conversations, the allocation status field shows the `«Allocated By Unknown Host»` message.

The allocation status string can be from 0 to 40 characters long. The host system encodes the allocation status string in the EBCDIC character set and the 3890/XP Enhanced document processor converts the allocation status string for display. The conversion uses the ***G*** translation table file that is specified in the Communication Manager Workstation Profile. The supplied default table converts any character. EBCDIC characters that do not have an ASCII equivalent translate as spaces.

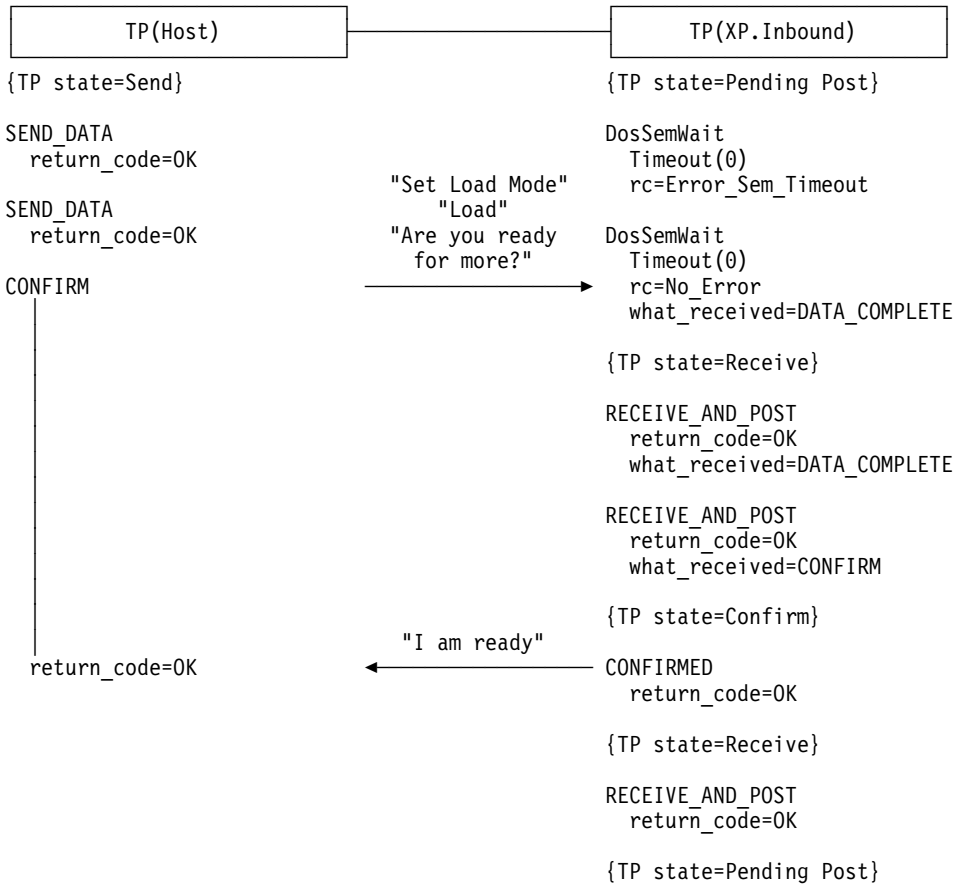

Figure 8-2. Flow of Control Data from the Host to the 3890/XP Enhanced Document Processor

Load Commands

The load commands constitute the load mode. The document processor enters the load mode only when the document processor is in a not-running condition and in a ready condition. Figure 8-2 illustrates the flow of information caused by individual APPC verbs in the load mode. (For more about the information that APPC verbs pass, see “Conversation Flow Diagrams” on page 8-22.)

Set Blocking

The set blocking command sets the blocking size of the read record response. You use the set blocking command to improve the performance on host systems with large APPC CALL overheads. On these types of host systems, you cannot afford to call APPC to receive each document record. The set blocking command gives you the ability to receive many document records on each call to APPC.

As an alternative, you can specify the **fill**(BUFFER) parameter on your APPC RECEIVE verbs. The **fill**(BUFFER) parameter, however, can make it difficult for APPC to transfer the final status change response to your buffers.

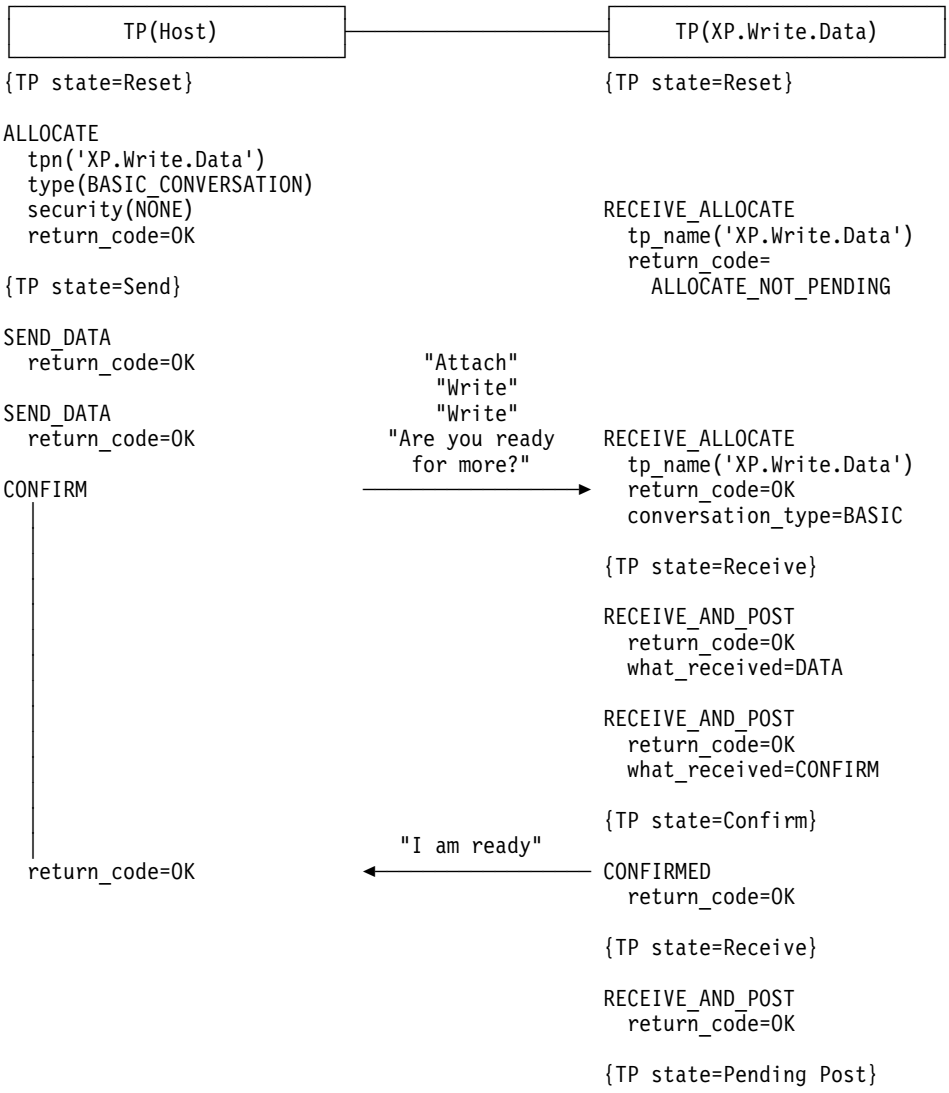

Figure 8-3. Normal Flow of Code Line Data from the Host to the 3890/XP Enhanced Document Processor

The set blocking command specifies the following parameters:

- The number of document records that the 3890/XP Enhanced document processor places in each read record response.

  When a host system first allocates the document processor, the number of document records that the document processor places in each read record defaults to 1. Valid values are 1 through 127. The 3890/XP Enhanced document processor rejects any value that falls outside of this range and makes no change in the read record response.

- The number of read record responses that the document processor should send between issuing APPC FLUSH verbs.

  When the host system first allocates the document processor, the number of read record responses that the document processor sends between the APPC FLUSH verbs defaults to zero. When this read record response number is zero, the document processor never issues an APPC FLUSH verb after the read record response. The host system does not normally need any APPC FLUSH verbs

following the read record response because APPC flushes each RU automatically when it is full and the document processor always flushes the sense data response following the last read record response. The host system should only specify a non-zero value in this read record response number if the host system needs the non-zero value for buffer management.

Set Load Mode

The host system uses the set load mode command to prepare the 3890/XP Enhanced document processor for the reception of the control data. The set load mode command assures that the document processor is in the correct state for initialization. The Control Program sets the not-initialized condition in the sense data and prepares for the load data. The 3890/XP Enhanced document processor replies with a sense data response. For more information, see "Sense Data" under "General Responses" on page 8-19.

Load

The load command transfers the control data from the host user program to the document processor. The Control Program accepts the load command if the set load mode command has successfully placed the document processor in the load mode and if the document processor has not exceeded the load-data-byte count. For more information about the load command, see "Loading the User Control Data" on page 8-4.

Within the limits of APPC, the host system transfers any amount of control data on each load command. Large transfers of control data are the most efficient. You can issue an APPC CONFIRM verb following the load command every time you send 50,000 to 100,000 bytes of control data. Although the document processor does not require the APPC CONFIRM verb, the verb helps to prevent the exhaustion of the LU buffers.

End Load

The end load command causes the Control Program to exit the load mode. The end load command calls the run initialization routine, which performs the actions and checks that initialization requires and processes the Run Profile. The document processor then runs the user-supplied SPSINIT event Sort module, if any.

If any of these routines detect an error, the document processor enters the not-ready condition. If the initialization is successful, the Control Program clears the not-initialized condition in the sense data and checks the initialize-code-line-data-match-buffers bit in the initialization data (bit 6 in byte 0). If you set the initialize-code-line-match-buffers bit and the conversation with "XP.Write.Data" is allocated from a prior run, the document processor purges the conversation and deallocates the conversation to prepare for priming the code line data match buffers.

You should issue an APPC FLUSH verb after the end load command. The 3890/XP Enhanced document processor replies with a sense data response. For more information, see "Sense Data" under "General Responses" on page 8-19.

Run Commands

The document processor uses the run commands during document feed operation. To start document feeding online, the document processor should be in a ready condition and in an initialized condition. Figure 8-3 on page 8-15 shows the flow of information that the individual APPC verbs cause during code line data matching. For more information about the information that APPC verbs pass, see "Conversation Flow Diagrams" on page 8-22.

Start Running

The start running command puts the document processor into the running condition and assures that the document processor is in the correct state for running. The Control Program sets the running condition in the sense data. When you send the start running command, you indicate that you are ready to receive the document records that the user-supplied Sort program generates. The 3890/XP Enhanced document processor replies with a sense data response. For more information, see "Sense Data" under "General Responses" on page 8-19.

If you activate code line data matching, the start running command also prepares the Control Program for receiving code line data match records. After the start running command, the Control Program looks for the LU 6.2 allocate conversation message for the "XP.Write.Data" conversation. The code line data match records should follow on the XP.Write.Data conversation. The document processor rejects all other commands if the XP.Write.Data conversation receives the commands. While the XP.Write.Data conversation remains allocated, you are committed to delivering code line data match records for document processor operation.

You should issue an APPC FLUSH verb after the start running command.

Write

The write command transfers code line data match records to the document processor and must be sent on the XP.Write.Data conversation (each write command transfers one code line data match record). The document processor rejects the write command if the "XP.Inbound" transaction program receives it. You specify the code line data match record length in the initialization data.

You can issue an APPC CONFIRM verb following the write command every time the document processor sends 3,000 to 6,000 bytes of code line data match records. Although the document processor does not require the APPC CONFIRM verb, it helps to prevent exhausting the LU buffers at the document processor and synchronizes the host-user program and the document processor. The synchronization between the host user program and the document processor reduces the overhead that an SCI pause causes. If you do not use an APPC CONFIRM verb, the LU 6.2 pacing controls the flow of data to "XP.Write.Data" The LU 6.2 pacing blocks permission for RU transmission from the host systems LU while a number of RUs wait for "XP.Write.Data" to copy from the local LU. The receive-pacing-limit value in the transmission service mode profile specifies the number of RUs that you can send.

After the write command transfers the last code line data match record for a particular run, the host system should deallocate the conversation with "XP.Write.Data" to show that it has reached the end of the code line data match records. When code line data matching occurs, the Control Program pauses the separator whenever it needs code line data match records. When the host interface sees that it has reached the end of the code line data match records, it stops pausing the separator. It also puts an end-of-file record in the write buffers that follows the last code line data match record that was received from the host system.

Note: An end-of-file record is a record with the code line data match end-of-file bit set (bit 0 in byte 0 in the code line data match record header).

The end-of-file record informs the code line data matching routine that you have reached the last code line data match record.

Stop Running

The stop running command stops the document processor transport. You should issue an APPC FLUSH verb after the stop running command. The 3890/XP

Enhanced document processor replies with a sense data response. For more information, see “Sense Data” under “General Responses” on page 8-19.

Diagnostic Commands

The diagnostic commands make up the diagnostic command sequences.

When the document processor is in a running-and-ready condition or in an offline condition, it does not accept the diagnostic command sequences.

The diagnostic command sequences provide access to the various storage areas within the document processor controller. For more information about diagnostic commands, see “Diagnostic Commands” on page 8-5.

Diagnostic Key

The diagnostic key command sets up the document processor to perform the request diagnostic read command or the diagnostic write command. The eight data bytes that the diagnostic key command transfers specify the following information about the data:

- Storage type
- Starting address
- Byte count.

For information about the format of the diagnostic key data, see “Operation” on page 8-6.

Request Diagnostic Read

The request diagnostic read command requests that the document processor transfer the diagnostic read data from the document processor storage area (specified by the diagnostic key data) to the host system. The document processor rejects the request diagnostic read command if the previous diagnostic key command did not contain valid diagnostic key data. The 3890/XP Enhanced document processor replies with a diagnostic read response. For more information about the diagnostic read response, see “Diagnostic Responses: Diagnostic Read” on page 8-21.

If you expect the diagnostic read data before continuing, you should issue an APPC FLUSH verb after the request diagnostic read command. You would normally issue an APPC FLUSH verb after the last command sequence in a collection of diagnostic command sequences.

Diagnostic Write

The diagnostic write command transfers the diagnostic write data from host system to the document processor storage area (specified by the diagnostic key data). The document processor rejects the diagnostic write command if the previous diagnostic key command did not contain valid diagnostic key data.

Document-Processor-Initiated Responses

Document-processor-initiated responses are the replies that the document processor sends back to the host system. The responses include status information and data. The three types of document-processor-initiated responses are:

- General responses
- Run responses
- Diagnostic responses.

General Responses

The document processor sends the general responses when it is in any state.

Sense Data

The sense data response sends the sense data from the document processor to the host system. The document processor sends the sense data response each time that any part of the sense data changes. If more than one bit changes at the same time, the document processor sends only one sense data response to get new sense data to the host system.

The sense data is two bytes long and the document processor always sends both bytes together.

Sense Byte 0 The document processor's state indicator.

Bit 0	Reserved
Bit 1	**Intervention Required:** This bit is set when the document processor is in the not-ready condition.
Bit 2	Reserved
Bit 3	Reserved
Bit 4	Reserved
Bit 5	Reserved
Bit 6	**Not Initialized:** This bit is set when the document processor is in the not-initialized condition.
Bit 7	**Running:** This bit is set when the document processor is in the running condition.

Sense Byte 1 The run initialization error code. (For a list of error codes and error code descriptions, see Appendix A, "Run Initialization Error Codes.")

The document processor sends the sense data response when the sense data changes for any reason. Therefore, the document processor sends the sense data response for the following commands:

- Set load mode
- End load
- Start running
- Stop running.

The document processor also sends the sense data response after a conversation allocation and after a ready condition. Occasionally, the document processor sends the sense data response when no changes occur in the sense data. For example, a not-initialized condition might have already been set when the document processor processes a set load mode command. The Control Program sets the not-initialized condition again and sends the sense data response to the host system.

The 3890/XP Enhanced document processor issues an APPC FLUSH verb after the sense data response except when a start running command causes a running condition and code line data match is not active.

Command rejected

The command rejected response sends a message to the host system stating that the command that you just sent to the 3890/XP Enhanced document processor is invalid. The command rejected response occurs when the document processor receives a command that is one of the following:

- Not in the document processor command set
- Not in a recognized format

- Not processed because of the present machine state.

To help diagnose an invalid command, the document processor sends the following codes back to the host system with the command rejected response:

- The sense data (in the same format that it is in for the sense data response)
- The invalid command code
- The previous command code
- The reason code for the rejection.

The document processor sends the command rejected response to the host system when the document processor receives an invalid command.

The 3890/XP Enhanced document processor issues an APPC FLUSH verb after the command rejected response.

Sense ID

The sense ID response transfers the machine identification data from the document processor to the host system. The identification data has a variable length. The first four bytes contain the machine number, the model number, and the stacker configuration data. The document processor reserves the rest of the bytes for future enhancements. The sense ID response is the response to the request sense ID command.

The 3890/XP Enhanced document processor issues an APPC FLUSH verb after the sense ID response.

Figure 8-4. Normal Flow of Document Records from 3890/XP Enhanced Document Processor to the Host

Run Responses

The document processor sends the run responses only during the normal document feeding operation. Figure 8-4 on page 8-20 displays the flow of information that the individual APPC verbs cause during normal document feeding operation. (For more about the information that APPC verbs pass, see “Conversation Flow Diagrams” on page 8-22.)

SCI Pause

The SCI pause response sends an indication to the host system that the SCI programmed pause that the user-supplied program requested earlier (by setting bit 7 in byte 9 in a document record header) has completed. The SCI pause response is called UX in the System/370 Channel Interface. If code line data matching occurs and the conversation with “XP.Write.Data” is still allocated, the document processor purges and deallocates the conversation to prepare for priming the code line data match buffers. The document processor requires a start running command from the host system to restart document feeding. If code line data matching occurs, you need to prime the code line data match buffers again.

The SCI pause response is a response to the start running command. The document processor sends the SCI pause response whenever an SCI programmed pause completes. The document processor sends the SCI pause response immediately before clearing the running condition. After the document processor sends the SCI pause response, the host system can do all the things that are allowed in the not-running condition, including sending a start running command or sending a set load mode command.

Read Record

The read record response transfers the document records to the host system. Each read record response transfers a block of document records. The set blocking command sets the block size and the block size defaults to one. You fix the record length in the initialization data.

When the document processor needs to transfer to the host system a response other than the read record response and has already buffered part of a read record response, the document processor pads the remaining slots for document records with X'00' to complete the read record response and transfers the fixed-size read record response to the host system.

Note: The first document record in the read record response always contains actual document record data.

The read record response is a response to the start running command. The document processor sends the read record response each time there are enough document records available from user-supplied Sort program processing.

Diagnostic Responses: Diagnostic Read

The diagnostic read response is a response to the request diagnostic read command. It transfers the diagnostic read data to the host system. The diagnostic key data determines the byte count that the document processor sends.

The document processor only uses diagnostic responses during the diagnostic command sequences.

The 3890/XP Enhanced document processor issues an APPC FLUSH verb after the diagnostic read response.

Conversation Flow Diagrams

The conversation flow diagrams show the design of the host interface conversations. When an APPC verb causes the LU to send information across the network, the conversation flow diagrams show the resulting flow of information as an arrow. Some APPC verbs cause the LU to suspend the program's processing until the LU completes the processing of the verb; the conversation flow diagrams show the suspension of program processing as a vertical line under the verb.

The conversation flow diagrams show some parameters with the verbs and the OS/2 function calls. The conversation flow diagrams show the supplied parameters as `parameter_name(supplied_value)`. The conversation flow diagrams show the returned parameters as `parameter_name=returned_value`. The parameters that the flow diagrams show are significant to the design of the host interface conversations. The parameters that the conversation flow diagrams do not show are not significant to the design.

The conversation flow diagrams display the state of each transaction program in brackets ({ }). The conversation flow diagrams show the state of the transaction program at the beginning of each flow diagram and at each time the state of the transaction program changes.

Conversation Initiation and Termination

Although the host system initiates the conversations, the document processor must perform the following steps on the PS/2 to complete initiation:

1. Properly start and configure the OS/2 Communication Manager.
2. Start the attach manager. Normally, when you start the communication manager, the attach manager starts automatically.
3. Start the Control Program.
4. Issue an APPC RECEIVE_ALLOCATE verb with the local transaction program.

For more information about transaction programs and verbs, see "APPC Verbs for the 3890/XP Enhanced" on page 8-27.

You can activate a conversation from the host system by following the preceding steps. You allocate and deallocate the conversation that transfers the write data during code line data matching as needed. The following sections discuss the details of the sequence for starting and stopping the two required conversations that the Control Program uses.

Note: The document processor does not need to be online for conversation allocation and deallocation.

Allocation

This section describes the activity that occurs during normal conversation initiation. Figure 8-5 on page 8-24 shows the flow of information that the individual APPC verbs cause during conversation initiation. (For more information about the information that APPC verbs pass, see "Conversation Flow Diagrams" on page 8-22.)

Before you allocate the conversations, the Control Program keeps an APPC RECEIVE_ALLOCATE verb active for the "XP.Inbound" transaction program. The attach manager matches the APPC RECEIVE_ALLOCATE verb with the first appropriate incoming allocate. The host system sends the allocate to the document processor.

The queue-allocates-time-out value, which is in the remotely-attachable-transaction program profile for each transaction program, specifies a time-out value for the incoming allocates. Although the Control Program does not place any requirements on the queue-allocates-time-out value, the value zero is suggested for the "XP.Inbound" transaction program so that the document processor immediately rejects incoming allocates if the APPC RECEIVE_ALLOCATE verb is not active in the Control Program.

When the attach manager matches the APPC RECEIVE_ALLOCATE verb, the Control Program issues an APPC RECEIVE_AND_WAIT verb to get the confirm indicator. Then the Control Program issues an APPC CONFIRMED verb to let the host system know that the document processor is ready to allocate the conversation with the "XP.Outbound" transaction program and process the host system commands. The Control Program then issues an APPC RECEIVE_AND_POST verb to change the "XP.Inbound" transaction program to the pending post state (the normal state for this conversation).

After the Control Program issues an APPC RECEIVE_AND_POST verb, it issues the APPC RECEIVE_ALLOCATE verb for the "XP.Outbound" transaction program. The TP-receive-time-out value in the remotely-attachable-transaction program profile for the "XP.Outbound" transaction program is one minute. The one-minute time-out value gives the host system time to issue the other allocate verb and the receive verb. The host

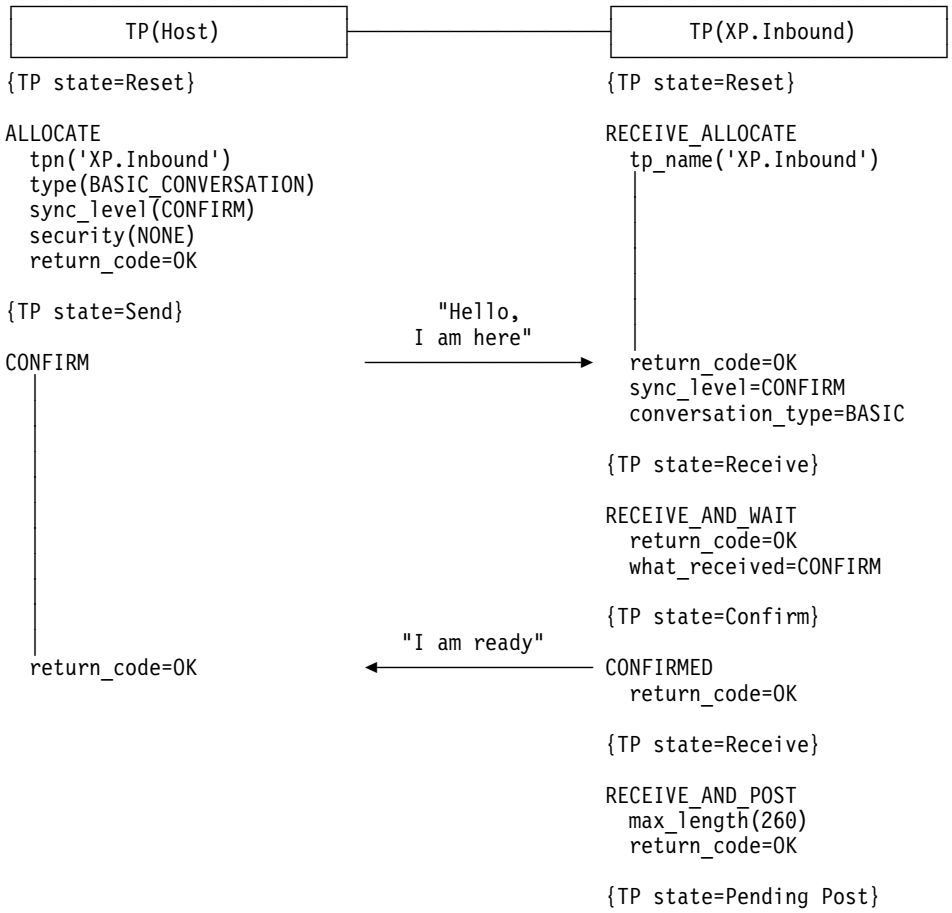

Figure 8-5 (Part 1 of 2). Conversation Initiation

system should issue both of the verbs within the one-minute time-out value to successfully allocate the required 3890/XP Enhanced document processor conversations.

The Control Program then issues an APPC RECEIVE_AND_WAIT verb to get the send indicator and to change the “XP.Outbound” transaction program to the send state (the normal state for this conversation). After the Control Program issues the APPC RECEIVE_AND_WAIT verb, it issues an APPC SEND_DATA verb and an APPC FLUSH verb to transfer a sense data response to the host system.

Deallocation

This section describes the activity that occurs at conversation termination during normal and forced termination.

Normal Termination: This section describes the activity during normal conversation termination. Figure 8-6 on page 8-25 shows the flow of information that the individual APPC verbs cause during normal conversation termination. For more information about the information that APPC verbs pass, see “Conversation Flow Diagrams” on page 8-22.

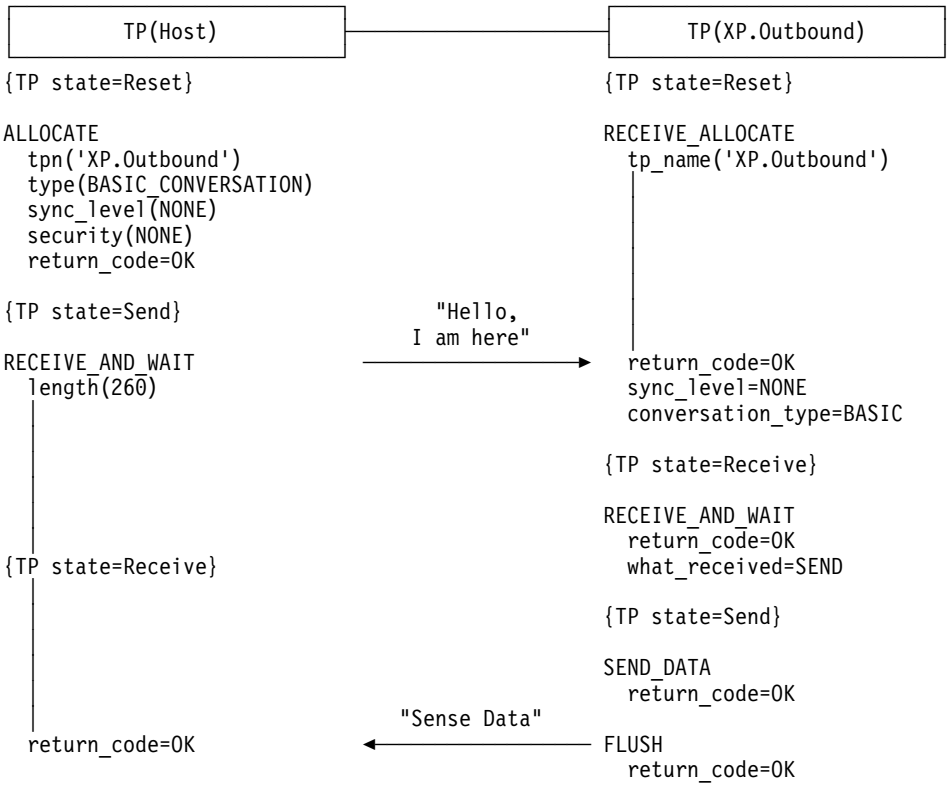

Figure 8-5 (Part 2 of 2). Conversation Initiation

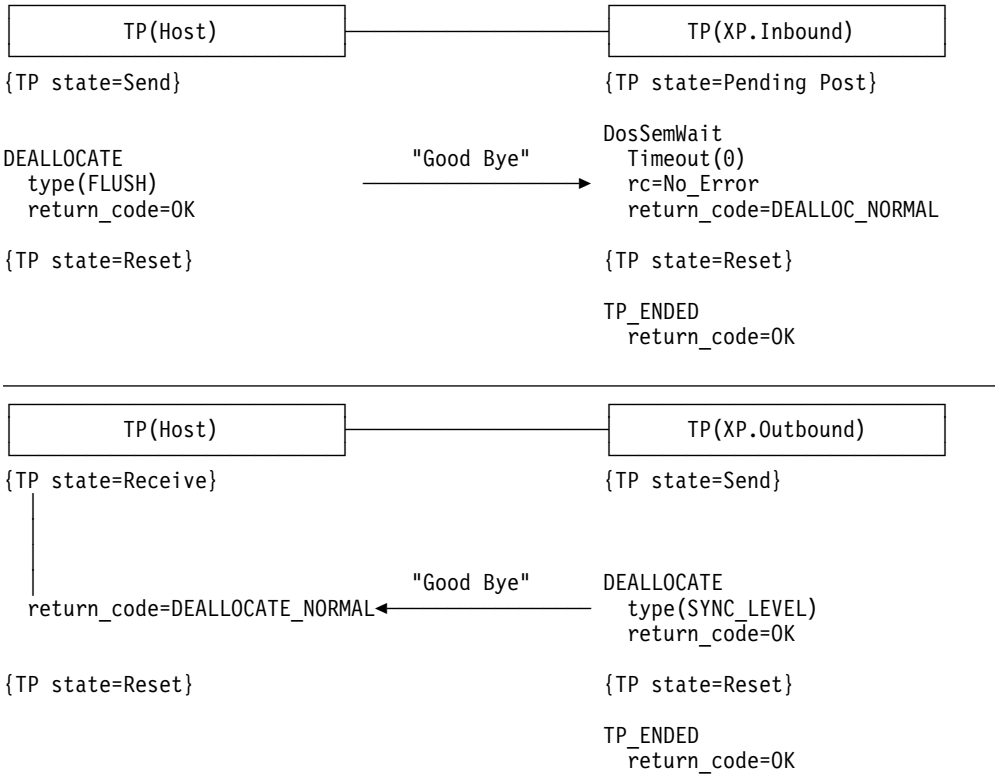

Figure 8-6. Conversation Termination

The host system issues an APPC DEALLOCATE verb to the “XP.Inbound” transaction program. When the XP Control Program sees the APPC DEALLOCATE verb, it finishes

sending all queued data and issues an APPC DEALLOCATE verb from the "XP.Outbound" transaction program. The Control Program then ends all active transaction programs with the APPC TP_ENDED verbs. After the Control Program ends the active transaction programs, it prepares for the next attach. For more information about preparing for an attach, see "Allocation" on page 8-23.

If the 3890/XP Enhanced document processor is online when the APPC DEALLOCATE verb arrives from the host system, the 3890/XP Enhanced document processor goes into a not-initialized condition. When the document processor enters a not-initialized condition, it erases the Control Program host interface buffers and stops the document processor transport.

Forced Termination (XP Enhanced Device Program Exit): The document processor can terminate the conversations at any time. An error recovery sequence or the 3890/XP Enhanced document processor control program termination can cause conversation termination. In either case, the document processor issues an APPC TP_ENDED verb for all active transaction programs. The APPC TP_ENDED verb sends the DEALLOCATE_ABEND_PROG error return code on all active conversations to the host system and does the following:

- Frees the conversations
- Ends the transaction programs
- Ends the allocation of the 3890/XP Enhanced document processor conversations.

APPC Verbs for the 3890/XP Enhanced

APPC for the OS/2 Extended Edition uses the base set and several option sets of the LU 6.2 verbs. The following section is a list of verb subsets that the host interface uses.

Note: The following list of verb subsets is not the entire OS/2 EE implementation of APPC verbs.

APPC Verb Subsets

Verb	Description
RECEIVE_ALLOCATE	Tells APPC that the 3890/XP Enhanced document processor control program is ready to communicate with a partner transaction program that has issued an allocation request. APPC generates IDs for both the new transaction program and the new conversation. If an incoming allocate or a request is not present, the new transaction program waits for one.
TP_ENDED	Ends the identified transaction program and frees up the associated resources.
ALLOCATE	Allocates a conversation connecting the local transaction program to a remote transaction program. The APPC ALLOCATE verb assigns a new resource ID to the conversation. The host system issues the APPC ALLOCATE verb prior to any verbs that refer to the conversation.
CONFIRM	Sends a confirmation request to the partner transaction program and waits for a reply. This verb lets the local program and the partner program synchronize their processing with one another. The APPC CONFIRM verb also causes the LU to flush the send buffer.
CONFIRMED	Sends a confirmation reply to the partner transaction program. This verb lets the local program and the partner program synchronize their processing.
DEALLOCATE	Deallocates the specified conversation. This verb can perform the function of the APPC FLUSH verb or the APPC CONFIRM verb before it deallocates the conversation.
FLUSH	Flushes the send buffer of the local LU by sending all buffered information to the partner LU.

RECEIVE_AND_POST	Immediately returns control to the local program and asynchronously receives information that the partner program sends.
RECEIVE_AND_WAIT	Waits for information to arrive on the specified conversation and then receives the information or receives information already available.
SEND_DATA	Sends data to the partner transaction program.
CONVERT	Provides conversion of character strings. For example, it does ASCII to EBCDIC or EBCDIC to ASCII conversion.

For additional information about APPC verbs, see the *Operating System/2 Extended Edition Version 1.3 APPC Programming Reference* and the *Operating System/2 Extended Edition Version 1.3 APPC Programming Services and Advanced Problem Determination for Communications*.

Operation Codes and Logical Record Formats

This section describes the operation codes and logical record formats of each command and response.

Operation Codes

The following list is a numerical list of the operation code (op code) for each command and response. For an alphabetical list by command or response name, see “Logical Record Formats” on page 8-29.

Op Code (hex)	Command or Response Name
0100	Request Sense ID
0200	Set Load Mode
0300	Load
0400	End Load
0500	Start Running
0600	Write
0700	Stop Running
0800	Diagnostic Key
0900	Request Diagnostic Read
0A00	Diagnostic Write
0B00	Set Allocation Status
0C00	Set Blocking
0101	Sense Data
0201	Command Rejected
0301	Sense ID
0401	SCI Pause
0501	Read Record
0601	Diagnostic Read

Logical Record Formats

The rest of this section discusses the Control Program logical record formats for the commands and the responses, arranged in alphabetical order by command or response name. Each record description contains the following information about the command or response:

Offset Offset of the parameter from the beginning of the logical record
Length Length of the parameter field in bytes
Value Constant value defined for the parameter (if applicable)
Field Name Name of parameter field or description of constant value.

The Control Program stores all the values in memory in left-to-right character format, the high-order byte (which is the most significant) at the low address (which is byte 0, the byte to the extreme left). If you look in the memory at a logical record with operation code X'0301', you see the string in that order (in hexadecimal). For example, a sense ID response logical record can appear as

```
00080301FF389073.
```

Command Rejected:

Offset	Length	Value	Field Name
0	2	X'000C'	Logical record length (LL)
2	2	X'0201'	Response operation code
4	1		Sense byte 0
5	1		Sense byte 1
6	2		The invalid command operation code
8	2		The previous command operation code
10	2		Reason code for rejection
		X'0001'	This command not in command set
		X'0002'	Present machine state not correct for this command
		X'0003'	Unrecognized command record format
		X'0004'	Valid Diagnostic Key Data absent
		X'0005'	Device offline
		X'0006'	Invalid parameter value

Diagnostic Key:

Offset	Length	Value	Field Name
0	2	X'000C'	Logical record length (LL)
2	2	X'0800'	Command operation code
4	8		Diagnostic key data

Diagnostic Read:

Offset	Length	Value	Field Name
0	2		Logical record length (LL).
2	2	X'0601'	Response operation code.
4	N		Diagnostic read data. (The length of data transferred is determined by the diagnostic key data.)

Diagnostic Write:

Offset	Length	Value	Field Name
0	2		Logical record length (LL).
2	2	X'0A00'	Command operation code.
4	N		Diagnostic write data. (The length must match the length of data to be transferred specified in the diagnostic key data.)

End Load:

Offset	Length	Value	Field Name
0	2	X'0004'	Logical record length (LL)
2	2	X'0400'	Command operation code

Load:

Offset	Length	Value	Field Name
0	2		Logical record length (LL)
2	2	X'0300'	Command operation code
4	N		Control data

Read Record:

Offset	Length	Value	Field Name
0	2		Logical record length (LL)
2	2	X'0501'	Response operation code
4	N		Document records

Request Diagnostic Read:

Offset	Length	Value	Field Name
0	2	X'0004'	Logical record length (LL)
2	2	X'0900'	Command operation code

Request Sense ID:

Offset	Length	Value	Field Name
0	2	X'0004'	Logical record length (LL)
2	2	X'0100'	Command operation code

SCI Pause:

Offset	Length	Value	Field Name
0	2	X'0004'	Logical record length (LL)
2	2	X'0401'	Response operation code

Sense Data:

Offset	Length	Value	Field Name
0	2	X'0006'	Logical record length (LL)
2	2	X'0101'	Response operation code
4	1		Sense byte 0
5	1		Sense byte 1

Sense ID:

Offset	Length	Value	Field Name
0	2		Logical record length (LL).
2	2	X'0301'	Response operation code.
4	1	X'FF'	Sense ID byte 0.
5	2		Machine type.
		X'3890'	3890/XP Enhanced machine type.
		X'3891'	3891/XP machine type.
		X'3892'	3892/XP machine type.
7	1		Model number and stacker configuration.
		X'7X'	3890/XP Enhanced model number (7) and the number of stackers installed (X).
		X'1X'	3891/XP model number (1) and the number of stackers installed (X).
		X'1X'	3892/XP model number (1) and the number of stackers installed (X).
8	N		The rest of this variable length response is reserved for future enhancement.

Set Allocation Status:

Offset	Length	Value	Field Name
0	2		Logical record length (LL)
2	2	X'0B00'	Command operation code
4	N		Allocation status string (0 to 40 characters)

Set Blocking:

Offset	Length	Value	Field Name
0	2	X'0008'	Logical record length (LL).
2	2	X'0C00'	Command operation code.
4	2		Number of document records per read record response.
6	2		Number of read record responses between flushes; zero implies never flush read record responses.

Set Load Mode:

Offset	Length	Value	Field Name
0	2	X'0004'	Logical record length (LL)
2	2	X'0200'	Command operation code

Start Running:

Offset	Length	Value	Field Name
0	2	X'0004'	Logical record length (LL)
2	2	X'0500'	Command operation code

Stop Running:

Offset	Length	Value	Field Name
0	2	X'0004'	Logical record length (LL)
2	2	X'0700'	Command operation code

Write:

Offset	Length	Value	Field Name
0	2		Logical record length (LL)
2	2	X'0600'	Command operation code
4	N		Code line data match record

System/370 Channel Interface

This section applies to the 3890/XP Enhanced only. The 3891/XP and 3892/XP document processors do not support the channel interface.

Channel Attachment

The channel is a method of communication between the 3890/XP Enhanced and the System/370 host system. Through the channel, the 3890/XP Enhanced attaches to all host systems implementing the System/370 and System/370-XA architecture. Although the 3890/XP Enhanced attaches to a byte channel or a block channel, the block multiplex channel is the recommended attachment.

The communication link between the channel and the 3890/XP Enhanced document processor is the I/O interface. The I/O interface uses information formats and control signal sequencing as a uniform method for attaching and controlling the system unit. Information in the form of data, status and sense indicators, control signals, and I/O-device addresses transmits over the time-and-function-shared lines of this interface. All transmissions of information interlock with the corresponding response signals.

The document processor communicates through strings of commands. For more information about the strings of commands, see the *System/360 and System/370 I/O Interface Channel to Control Unit, Original Equipment Manufacturers' Information*.

Channel Commands

Channel commands are instructions in the channel program. The following are the types of channel commands that are available:

- Control
- Sense
- Read
- Write
- TIO (Test I/O).

You use channel command words (CCWs) to give the document processor specific commands. There are CCWs for each type of channel command except the TIO command. Although the channel sees the TIO command as a CCW, it is simply a status check instruction specified at the programmer's discretion. The channel program, which directs channel operations, is made up of one or more CCWs.

Figure 8-7 lists the CCWs and hexadecimal codes for each channel-command type.

Figure 8-7. Channel Commands

Command Type	Channel Command Word (CCW)	Hexadecimal Code
Control	Anticipate	23
	Anticipate and Stop	33
	Set Load Mode	43
	Load	53
	Load High	73
	End Load	63
	No-Op	03
	Diagnostic Initiate	D3
	Diagnostic Key	F3
	Diagnostic Read	B4
	Diagnostic Write	B3
Sense	Sense	04
	Sense I/O	E4
Read	Read 0	02
	Read 1	12
	Read 2	22
	Read 3	32
	Read Configuration Data	E2
Write	Write 0	01
	Write 1	11
	Write 2	21
	Write 3	31
TIO	Test I/O	00

Status Indicators: Status Byte

The status byte reports the condition of the 3890/XP Enhanced document processor at the time that the status byte is transmitted. The status byte is transmitted to the channel in the following situations:

- During the initial selection sequence
- To present the channel end status at the end of data transfer
- To present the device end signal and any associated conditions to the channel (the I/O device remains busy during an operation until the channel receives the device end status)
- To present device end status and signal that the system unit is now available
- To present any previously stacked status
- To present any externally initiated status.

After the channel receives the status byte, the byte is reset and not presented again.

Format of the Status Byte:

Bit	Name	Code
0	Attention	Not used
1	Status modifier	SM
2	Control unit end	Not used
3	Busy	Busy
4	Channel end	CE
5	Device end	DE
6	Unit check	UC
7	Unit exception	UX

Figure 8-8. Status Byte

Status Bit Assignments

The specific bits in the status byte are presented and set to a binary value of 1 to indicate status conditions of the system unit. The number of bits presented during a single transmission varies with the number of conditions present. If the condition assigned to a specific bit is not present, then that bit is not presented and remains a binary value of 0. Definitions of the bit assignments and the conditions under which they are presented are the following:

Bit 0 **Attention**: This bit is not used.

Bit 1 **Status Modifier**: This bit is included in the ending status to the anticipate CCW when the 3890/XP Enhanced commits to a write operation.

Bit 2 **Control Unit End**: Not used.

Bit 3 **Busy**: This bit is presented during the initial-selection sequence if the preceding command is being processed from presentation of the initial status to device end acceptance for that command. The TIO (X'00') causes this response only during an operation when no device-end status is available; any other available status accompanies this bit.

Bit 4 **Channel End**: This bit is presented when there are no more data cycles requested for an operation.

Bit 5 **Device End**: This bit is presented when an operation is completed; it is also presented when the system unit changes from the not-ready condition to the ready condition.

Bit 6 **Unit Check**: This bit is presented to request a sense CCW. This status is also returned to the TIO CCW. (X'00') when the 3890/XP Enhanced is in the not-ready condition.

Bit 7 **Unit Exception**: This bit is presented to the anticipate CCW as a result of an SCI pause.

Sense Indicators

The 3890/XP Enhanced document processor reserves two bytes of storage for information about the success or failure of the various CCWs. When the channel program issues a sense CCW, a sense operation follows in which the document processor transmits one or both of these sense bytes to the channel. Information transferred by the sense operation is more detailed than that supplied by the status byte and describes reasons for the unit check indication. It also indicates, for example, that the system unit is in the not-ready condition.

Sense Byte 0

With the exception of the intervention-required condition, the not-initialized condition, and the running condition, the first sense byte, sense byte 0, is reset in the following situations:

- When a command other than a sense CCW, a no-op CCW, or a TIO CCW is received (the initial status does not include the busy status)
- When the system CCW or selective-reset CCW is received
- When a power transition forces the 3890/XP Enhanced into the offline condition.

Format of Sense Byte 0:

Bit	Name	Code
0	Command reject	CMR
1	Intervention required	IRV
2	Bus-Out check	BUS
3	Equipment check	ECK
4	Data check	DAT
5	Overrun	Not used
6	Not initialized	NI
7	Running	RUN

Figure 8-9. Sense Byte Format

Bit Assignments for Sense Byte 0

Bit 0 **Command Reject**: Indicates that an invalid command was issued. This bit is set by all the command codes not listed in Figure 8-7 on page 8-36.

Bit 1 **Intervention Required**: This bit is set when 3890/XP Enhanced is in the not-ready condition.

Bit 2 **Bus-Out Check**: This bit is set whenever the channel receives a data byte or a command byte in even parity.

Bit 3 **Equipment Check**: This bit is set when a failure within the 3890/XP Enhanced is detected. An internal parity error during any data transfer or command decode sets this bit.

Bit 4 **Data Check**: This bit is set when a stop sequence or an internal check is detected during certain CCWs. For additional information about the CCW codes, see "Channel-Command Summary Charts" on page 8-45. The internal check verifies that the number of bytes transferred is a multiple of four. The 3890/XP Enhanced discards any data that it receives.

Bit 5 **Overrun**: Not used.

Bit 6 **Not Initialized**: 3890/XP Enhanced is in the not-initialized condition until the run initialization routine runs successfully. 3890/XP Enhanced enters the not-initialized condition when the offline condition occurs or when the run initialization routine finds invalid data.

Bit 7 **Running**: This bit is set if 3890/XP Enhanced is in the running condition when it receives the sense CCW.

Sense Byte 1

The second sense byte, sense byte 1, is the run initialization error code. The 3890/XP Enhanced performs various validity checks during run initialization. If these checks receive correct responses, the 3890/XP Enhanced returns an error code of X'00'. If these checks detect a wrong response, the 3890/XP Enhanced returns a non-zero error code and enters or remains in the not-initialized condition. The error code transmits to the channel as sense byte 1 and displays an error message for the operator. For a list of the error codes and the error code descriptions, see Appendix A, "Run Initialization Error Codes." For more information about CCW codes, see "Channel-Command Summary Charts" on page 8-45.

Loading User Control Data

You can use one of the following methods to load user control data to the 3890/XP Enhanced:

Method 1

1. One set-load-mode CCW (no data)
2. Any number of load-high CCWs (4096 data bytes per load-high CCW)
3. One load CCW (if remaining data is less than 4096 bytes)
4. One end-load CCW (no data).

Note: Method 1 is the quickest way to load data.

Method 2

1. One set-load-mode CCW (no data)
2. Any number of load CCWs (4096 data bytes maximum per load CCW)
3. One end-load CCW (no data).

If any load CCW or load-high CCW detects insufficient storage space, the CCW returns a code of X'10' in sense byte 1 and resets the load mode. If any load-high CCW detects that a previous load CCW did not transfer 4096 bytes of load data, the CCW puts a code of X'11' in sense byte 1 and resets the load-mode CCW.

Read/Write Data

The records produced as the 3890/XP Enhanced reads items make up most of the transferred data. The 3890/XP Enhanced creates these records, one for one, at the throughput rate. Also, when the 3890/XP Enhanced is using code line data matching, the 3890/XP Enhanced writes corresponding code line data match records at the same rate.

During normal running, the host system transfers all the data to the 3890/XP Enhanced in 16-record blocks by the normal SIO. The first command, the anticipate CCW, withholds device-end status until the 3890/XP Enhanced is prepared to transfer a block of write data. When the 3890/XP Enhanced requires a block of write data, the host system presents the device-end-status modifier; when a block of read data is ready, the host system presents the device-end status alone.

Therefore, the ending status of the anticipate CCW commits the 3890/XP Enhanced to a read-CCW operation or a write-CCW operation. If the 3890/XP Enhanced requires both operations, the write-CCW operation occurs. The CCW address in the channel-status-word (CSW) area of host storage indicates which operation occurred.

Normal SIO for Read/Write Operations

For more information about the transfer-data commands, see “Commands to Transfer Data” on page 8-46.

The following example is a simple channel program. There are variations that are more complex. For this example, assume that the 3890/XP Enhanced is correctly initialized and ready and that code line data matching is active; command chaining is set for all of these CCWs, with the exception of the write-3 CCW and the read-3 CCW.

```
SIO ----> Anticipate (or Anticipate-and-Stop)
     .--- TIC + 32 (Transfer in channel 32 bytes to Read 0)
     :
     :    Write 0
     :    Write 1
     :    Write 2
     :    Write 3  ---> Interrupt from Write operation
     :                  (no Command Chain Flag)
     '--> Read 0
          Read 1
          Read 2
          Read 3  --->  Interrupt from Read operation
```

The Control Program sends the anticipate CCW to the 3890/XP Enhanced. Because code line data matching is active, the document processor needs code line data match records so the 3890/XP Enhanced responds with the status modifier (SM) status as an ending status. The SM status causes the host-channel program to fall through the TIC + 32 and perform the write-X CCWs (write CCWs transfer code line data match records to 3890/XP Enhanced). After the write-3 CCW, the chain ends. After the chain ends, the host system restarts the channel program with the anticipate CCW and follows the same path until the 3890/XP Enhanced no longer requires additional code line data match records.

When the document processor does not need code line data match records, the anticipate CCW ends without the SM status and the TIC + 32 performs the read-X CCWs (read CCWs transfer the 3890/XP Enhanced read records to the host system). After the read-3 CCW, the chain ends. The anticipate CCW restarts the channel program. The channel program is then restarted with the anticipate CCW and the channel program follows the same path until the 3890/XP Enhanced needs additional code line data matches or enters the not-ready condition. Either the UC status or the UX status breaks the chain. The host system must restart the channel program to continue.

Each read CCW and write CCW transfers four records. The user fixes record length in the initialization data. Block length is always 16 records. Because the 3890/XP Enhanced withholds both the not-ready condition and the SCI pause status until all the read data has been sent, either status condition can cause a transmission with less than 16 real records. In this case, the remainder of the block is filled with X'00'. The not-ready condition or the SCI pause status is indicated to the next anticipate CCW.

Sequence Checking and Retry

The suffix numbers (0 through 3) of the read CCWs and write CCWs are used to identify retry and to check sequence. Whenever the suffix number of a command matches that of the preceding command, data transfer occurs to or from the area affected by the preceding command. Whenever the suffix number of a command is not equal to or greater than the preceding command suffix number, the 3890/XP Enhanced returns command reject. The suffix number kept in the 3890/XP Enhanced updates only during initial selection of read, write, and anticipate CCWs. The suffix number lets other commands be inserted between monitored commands without interfering with the sequence checking. Of the affected commands, the 3890/XP Enhanced accepts the read-0 CCW and the write-0 CCW after only an anticipate CCW or another read/write 0-0 CCW; the operation is the same in either event.

The suffix number kept by the 3890/XP Enhanced updates only when the channel-end status or the device-end status produces a read-3 CCW or a write-3 CCW. After this event, the next anticipate CCW that is received resets the internal suffix number and advances the internal read-out pointer to the next block of 16 records. In addition, all four read commands return the command reject status unless the last previous anticipate CCW committed the 3890/XP Enhanced to a read operation. The write CCWs return the command reject status in the same way.

Interface Sequences

The channel adapter monitors the following asynchronous interface sequences. These sequences change the command operation:

Selective reset

The selective reset sequence saves the state of all channel adapter I/O ports in a log. After the selective reset sequence saves the log, the channel adapter performs the functions of the system reset sequence described below.

System reset

The system reset sequence resets all status and sense byte 0 information and the adapter-interface-control logic. The channel can re-try an aborted command. The following are the results of a system reset sequence:

- If the system reset sequence ends the command during the initial selection sequence, the command decode stops and no initial status is produced.
- If the system reset sequence ends the command during a data transfer to the channel, the data transfer stops. The data is then held in the channel adapter data buffer and the no-ending status is produced.
- If the system reset sequence ends the command during a data transfer from the channel, the data transfer stops. All data that is received is discarded and no-ending status is produced.
- If the system reset sequence ends the command after sending the channel-end status and before sending the device-end status, the ending operation of the command ends and the device-end status is sent.

Interface disconnect

The interface disconnect sequence disconnects all the inbound bus and tag lines, except the metering-in line and the request-in line. The channel can re-try an aborted command. The following are the results of an interface disconnect sequence:

- If the interface disconnect sequence ends the command during the initial selection sequence, the command decode stops and no initial status is produced. Depending on when the command decode stops, sense byte 0 can reset.
- If the interface disconnect sequence ends the command during a data transfer, see the following stop sequence.
- If the interface disconnect sequence ends the command after channel-end status is sent and before device end is sent, the ending operation is normal.

Stop

The stop sequence stops all data transfers. The short-CCW-count message in the CCW command charts indicates the ending status results. For more information about CCW command charts, see "Channel-Command Summary Charts" on page 8-45. The following are the results of a stop sequence:

- All data that is sent to the channel remains in the channel adapter data buffer.
- If sense byte 0 indicates a data check, all data that is received for the aborted CCW is discarded.
- If sense byte 0 does not indicate a data check, the data that is received for the aborted CCW is transferred to the 3890/XP Enhanced control unit.

Error detection

The error detection sequence stops command processing. "Channel-Command Summary Charts" on page 8-45 discusses the status and sense conditions for each CCW. The following are the results of an error detection sequence:

- If the error detection sequence detects an error during the initial selection sequence, the command decode stops. A UC status is sent and the appropriate bit is set in sense byte 0.
- If the error detection sequence detects an error during a data transfer, data transfer stops. Ending status of the channel-end status, the device-end status, and the UC status is sent and the appropriate bit is set in sense byte 0. The channel-adapter data buffer holds any data that is transferred to the channel. The channel-adapter data buffer discards any data that it receives from the channel.
- If the error detection sequence detects an error after the channel-end status is sent, the command processing stops. The device-end status and the UC ending status are sent and the appropriate bit is set in sense byte 0.

Channel-Command Summary Charts

This section contains charts that specify the operation details, the status, and the sense information for each CCW. Note that some commands split the channel-end status and the device-end status and that some commands do not split the channel-end status and the device-end status.

The definitions for the status and the sense codes used in these charts are listed alphabetically as follows:

Code	Definition
BUS	Bus-Out check
CE	Channel end
CMR	Command reject
DAT	Data check
DE	Device end
ECK	Equipment check
IRV	Intervention required
NI	Not initialized
RUN	Running
SM	Status modifier
UC	Unit check
UX	Unit exception.

Commands to Transfer Data

CCW - Anticipate (X'23'); Anticipate and Stop (X'33'):

Initial Status:

Status	Sense (Byte 0)	Cause
CE		Normal
UC	IRV	Not ready
	NI	Check sense byte 1
	ECK	Malfunction detected
	BUS	Even parity
UX		SCI pause
UC and UX		Not ready and SCI pause

Resulting Action: Regardless of the status, if the preceding anticipate CCW committed to a read CCW or a write CCW and was followed finally by a read-3 CCW or a write-3 CCW, the command destroys that 16-record block that was just transmitted.

Ending Status:

Status	Sense (Byte 0)	Cause
DE and SM		A block of 16-write records is required. This is a commit to write.
DE		A block of 16-read records is available. This is a commit to read.
DE and UX		SCI pause.
DE and UC	IRV	Not ready and not SCI pause.
	ECK	Malfunction detected.

CCW - Read 0 (X'02'); Read 1 (X'12'); Read 2 (X'22'); Read 3 (X'32'):

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	CMR	Read-command sequence violation or not read operation
	NI	Check sense byte 1
	ECK	Malfunction detected
	BUS	Even parity

Resulting Action:

Status	Action
00	The 3890/XP Enhanced sends requested data; the 3890/XP Enhanced resends data if the suffix matches the suffix of the preceding read CCW.
UC	No action.

Ending Status:

Status	Sense (Byte 0)	Cause
CE and DE		Data transfer complete
CE, DE, and UC	DAT	Short CCW count or internal check
	ECK	Malfunction detected

CCW - Write 0 (X'01'); Write 1 (X'11'); Write 2 (X'21'):

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	CMR	Write-command sequence violation or not commit to write
	NI	Check sense byte 1
	ECK	Malfunction detected
	BUS	Even parity

Resulting Action:

Status	Action
00	3890/XP Enhanced receives data into the correct area.
UC	No action.

Ending Status:

Status	Sense (Byte 0)	Cause
CE and DE		Data transfer complete and 3890/XP Enhanced ready to receive next write command
CE, DE, and UC	DAT	Short CCW count or internal check
	ECK	Malfunction detected
	BUS	Even parity

CCW - Write 3 (X'31')

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	CMR	Write-command sequence violation or not commit to write
	NI	Check sense byte 1
	ECK	Malfunction detected
	BUS	Even parity

Resulting Action:

Status	Action
00	3890/XP Enhanced receives data into the correct area.
UC	No action.

Ending Status:

Status	Sense (Byte 0)	Cause
CE		Data transfer complete
DE		3890/XP Enhanced ready for next command
CE, DE, and UC	DAT	Short CCW count or internal check
	ECK	Malfunction detected
	BUS	Even parity
DE and UC	ECK	Malfunction detected

Commands to Transfer Control Data

CCW - Set Load Mode (X'43')

Initial Status:

Status	Sense (Byte 0)	Cause
CE		Normal
UC	IRV	Not ready
	RUN	3890/XP Enhanced is running
	BUS	Even parity
	ECK	Malfunction detected

Resulting Action:

Status	Action
CE	The 3890/XP Enhanced sets the load mode, the load address, and the NI.
UC	No action.

Ending Status:

Status	Sense (Byte 0)	Cause
DE		Action complete
DE and UC	ECK	Malfunction detected

CCW - Load (X'53')

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	CMR	• Not load mode • Attempted load beyond SCI area (check sense byte 1)
	BUS	Even parity
	ECK	Malfunction detected

Resulting Action:

Status	Action
00	3890/XP Enhanced loads data sequentially, as per load pointer.
UC	3890/XP Enhanced resets the load mode if sense byte 1 is X'10'.

Ending Status:

Status	Sense (Byte 0)	Cause
CE		Data transfer successful or short CCW count. A maximum of 4096 bytes can be transferred.
DE		3890/XP Enhanced ready for next command.
CE, DE, and UC	BUS	Even parity.
	ECK	Malfunction detected.
DE and UC	ECK	Malfunction detected.

CCW - Load High (X'73')

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	CMR	Check sense byte 1 for one of the following conditions: • Not load mode • Attempted load beyond SCI area • Previous load transfer not 4096 bytes.
	BUS	Even parity.
	ECK	Malfunction detected.

Resulting Action:

Status	Action
00	3890/XP Enhanced loads data sequentially, as per load pointer.
UC	3890/XP Enhanced resets the load mode if sense byte 1 is X'10' or X'11'.

Ending Status:

Status	Sense (Byte 0)	Cause
CE		Data transfer successful; 4096 bytes must be transferred.
DE		3890/XP Enhanced ready for next command.
CE, DE, and UC	BUS	Even parity.
	ECK	Malfunction detected.
	DAT	Short CCW count.
DE and UC	ECK	Malfunction detected.

CCW - End Load (X'63')

Initial Status:

Status	Sense (Byte 0)	Cause
CE		Normal
UC	CMR	• Not load mode • Attempted load beyond SCI area (check sense byte 1)
	BUS	Even parity
	ECK	Malfunction detected

Resulting Action:

Status	Action
CE	3890/XP Enhanced resets the load mode and requests run initialization.
UC	No action.

Ending Status:

Status	Sense (Byte 0)	Cause
DE		Run initialization ends without error. (3890/XP Enhanced resets NI.)
DE and UC	NI	Run initialization detects error. (Check sense byte 1.)
DE and UC	ECK	Malfunction detected.

CCW - Diagnostic Key (X'F3')

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	BUS	Even parity
	ECK	Malfunction detected
	RUN	Running

Resulting Action:

Status	Action
00	3890/XP Enhanced loads eight bytes into the diagnostic area and checks the validity of the key data.
UC	No action.

Ending Status:

Status	Sense (Byte 0)	Cause
CE		Data transfer complete.
DE		3890/XP Enhanced ready for next command.
CE and DE		Invalid key detected. Key not given to 3890/XP Enhanced.
CE, DE, and UC	BUS	Even parity.
	ECK	Malfunction detected.
	DAT	Short CCW count.
DE and UC	ECK	Malfunction detected.

CCW - Diagnostic Read (X'B4')

Initial Status:

Status	Sense (Byte 0)	Cause
UC	CMR	Not preceded by the diagnostic key data or the key data is not valid
	BUS	Even parity
	ECK	Malfunction detected
00		Normal

Resulting Action:

Status	Action
00	3890/XP Enhanced reads as per the diagnostic key data
UC	No action

Ending Status:

Status	Sense (Byte 0)	Cause
CE and DE		Action complete or short CCW count
CE, DE, and UC	ECK	Malfunction detected

CCW - Diagnostic Write (X'B3')

Initial Status:

Status	Sense (Byte 0)	Cause
UC	CMR	Not preceded by the diagnostic key data or the key data is not valid
	BUS	Even parity
	ECK	Malfunction detected
00		Normal

Resulting Action:

Status	Action
00	3890/XP Enhanced writes as per the diagnostic key data
UC	No action

Ending Status:

Status	Sense (Byte 0)	Cause
CE		Action complete or short CCW count
DE		3890/XP Enhanced ready for next command
CE, DE, and UC	ECK	Malfunction detected
	BUS	Even parity
DE and UC	ECK	Malfunction detected

CCW - Diagnostic Initiate (X'D3')

Initial Status:

Status	Sense (Byte 0)	Cause
UC	CMR	Not preceded by the diagnostic key data or the key data is not valid
	BUS	Even parity
	ECK	Malfunction detected
CE		Normal

Resulting Action:

Status	Action
CE	3890/XP Enhanced initiates action as per the diagnostic key data
UC	No action

Ending Status:

Status	Sense (Byte 0)	Cause
DE		Action complete
DE and UC	ECK	Malfunction detected

Other Commands

CCW - Sense (X'04')

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	BUS	Even parity
	ECK	Malfunction detected

Resulting Action:

Status	Action
00	3890/XP Enhanced sends sense bytes.
UC	No action.

Ending Status:

Status	Sense (Byte 0)	Cause
CE and DE		Action complete. Sense bytes sent.
CE, DE, and UC	ECK	Malfunction detected.

CCW - No-Op (X'03')

Initial Status:

Status	Sense (Byte 0)	Cause
CE		Normal
UC	IRV	Not ready
	BUS	Even parity
	ECK	Malfunction detected

Resulting Action:

Status	Action
CE	No action
UC	No action

Ending Status:

Status	Sense (Byte 0)	Cause
DE		Action complete
DE and UC	ECK	Malfunction detected

CCW - Read Configuration Data (X'E2')

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	BUS	Even parity
	ECK	Malfunction detected

Resulting Action:

Status	Action
00	3890/XP Enhanced sends 96 bytes of read-configuration data.
UC	No action.

Ending Status:

Status	Sense (Byte 0)	Cause
CE and DE		Normal
CE, DE, and UC	DAT	Short CCW count
	ECK	Malfunction detected

CCW - Sense I/O (X'E4')

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal
UC	BUS	Even parity
	ECK	Malfunction detected

Resulting Action:

Status **Action**

00 3890/XP Enhanced sends the following 12 bytes:

X'FF38907s0000000040E2zzzz'

where:

- Byte 4 (7s) is the model number/number of stacker modules installed. 7s can range in value from 71 to 76 (model number = 7, number of stackers installed = 1-6).
- Bytes A and B (zzzz) are the count of read-configuration data for the 3890/XP Enhanced, which is set at 96 bytes (256 bytes is the reserved maximum).

UC No action.

Ending Status:

Status	Sense (Byte 0)	Cause
CE and DE		Normal.
CE, DE, and UC	DAT	Short CCW count—4 or 12 bytes must be transferred.
	ECK	Malfunction detected.

CCW - Test I/O (X'00')

Initial Status:

Status	Sense (Byte 0)	Cause
00		Normal.
Any stacked status.		Any stacked or available status is sent.
UC	IRV	Not ready.
	BUS	Even parity.
	ECK	Malfunction detected.
Busy		Command in progress, but ending status not yet produced.

Chapter 9. Power Encoder

The Power Encoder is a feature on the 3892/XP and UT/XP document that encodes documents. It can encode the amount field on a check, thus replacing the conventional unit encoding machines, or it can encode the entire MICR line for MICR rejects.

Several SPXServ functions are provided for programming the Power Encoder. For more information, see the *3890/XP Series and 3890/XP Enhanced SPXServ Reference*.

This chapter discusses the following topics:

- Document flow
- Printing
- Programming options
- Print verification
- Transport action.

Document Flow - 3892/XP

The rate at which documents flow through the Power Encoder depends on the transport paths used and the modes of operation in effect.

The Power Encoder has two major transport paths, high-speed and low-speed (shown schematically in Figure 9-1).

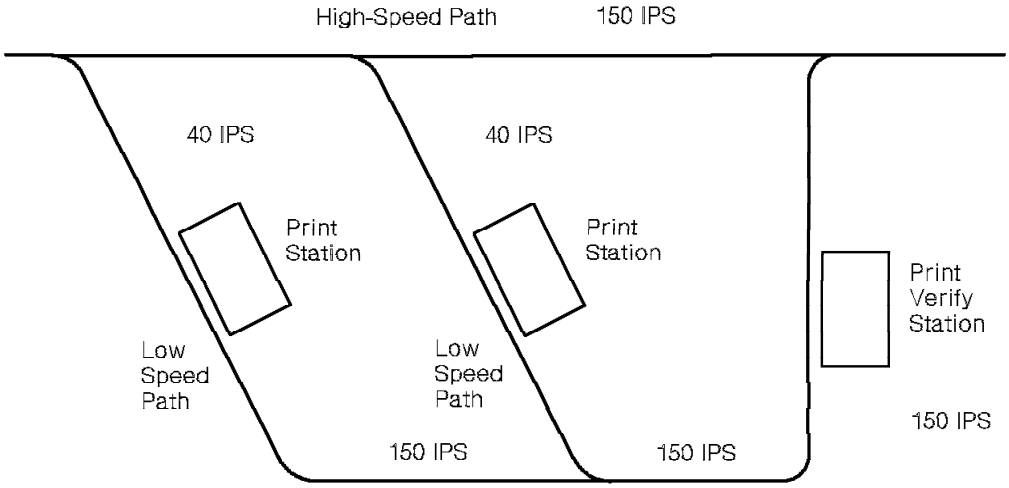

Figure 9-1. Transport Paths, Inches per Second (IPS)

When the encoder is not selected, the transport uses a “high-speed path” that supports a document feed rate of 1000 six-inch documents per minute.

When the encoder is selected, documents flow alternately between the two “low-speed paths” in a *dual path mode* that results in a throughput of 500 six-inch documents per minute (plus or minus 10 percent).

It is also possible to restrict document flow to a single low-speed path, resulting in a throughput of 250 six-inch documents per minute (plus or minus 10 percent).

The operator can set this *single path mode* to circumvent a print station that:

- Is low on ribbon
- Has a broken or missing ribbon
- Has a broken print mechanism
- Exhibits poor print quality.

Once repairs are made, the operator can revoke the single path mode and set the dual path mode. The operator can set or revoke the single path mode only while document feeding is stopped, from the operator panel.

Document Flow - UT/XP

The rate at which documents flow through the Power Encoder (see Figure 9-2) depends on the feed rate that you set. If two encoders are selected, the rate is 1000 6-inch documents per minute. You can change the feed rate with the SPXServ function SpxPutHSPESetup.

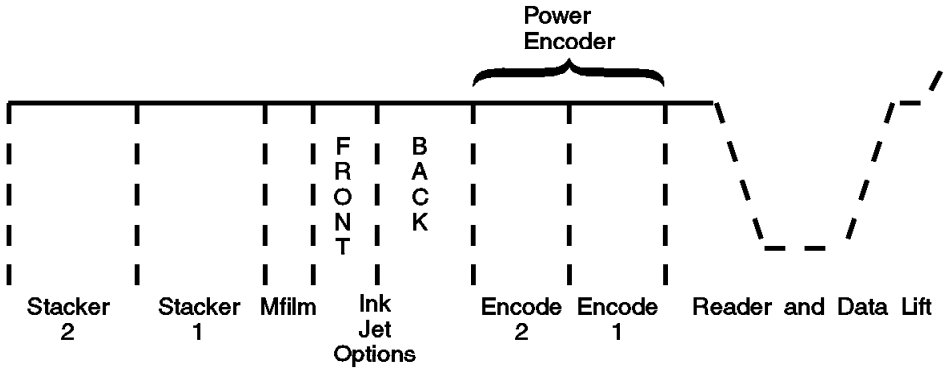

Figure 9-2. Universal Transport Paths

It is possible to restrict the use of a single encoder. The UT/XP Control Program requires reinitialization and sets the feed rate to 600 documents per minute automatically. Each encoder prints a minimum of 10 characters and not more than 17 characters. If more than 17 characters are to be encoded, both encoders must be used and the feed rate must be set to 600 documents per minute. If less than 18 characters are to be encoded (but at least 10 printable characters), the feed rate can remain at 1000 documents per minute. The UT/XP alternates encoders for each document.

Printing

For a given encoder configuration, characters are printed or encoded with a fixed character pitch and character font. The configuration is determined at the time the customer orders the encoder feature.

Character Print Location

All characters are printed on a single line with boundaries, as described in Figure 9-3 on page 9-3. These boundaries are based on the ANSI X9.13 specification.

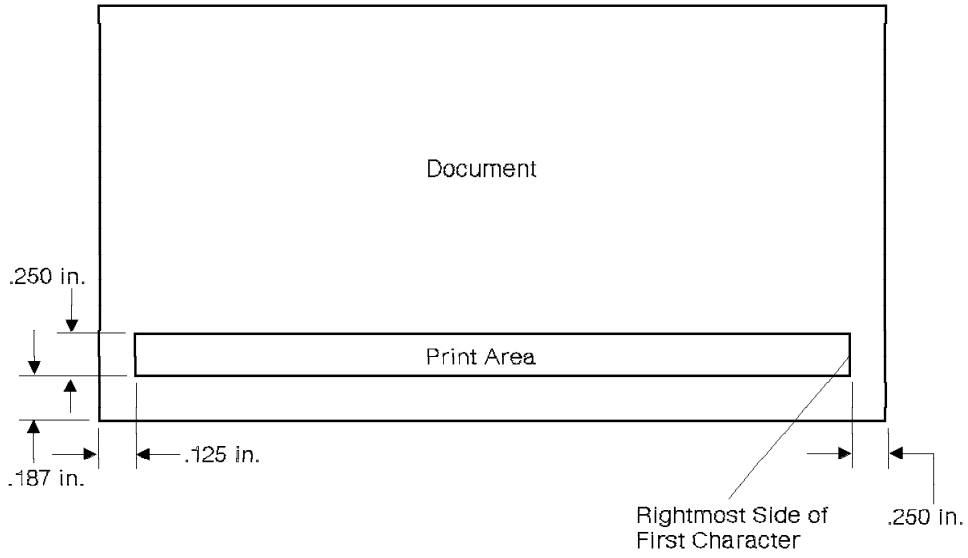

Figure 9-3. Encoder Print Boundaries

Character Font

The encoder supports the following character font:

Figure 9-4. Character Font Supported			
Font	**Pitch**	**As Defined By:**	**Ink**
E13B	8 CPI	ANSI 9.27 (Print Specification) ANSI X9.13 (Placement and Location Specification) ISO R 1004-1969	Magnetic

Character Set

The encoder supports the following character set:

E13B Characters	
Character	Hex Code
0 to 9	30 to 39
Space	20
Transit	3C
Amount	24
On Us	23
Dash	2D

Figure 9-5. Character Set Supported

Ribbon Optimization - 3892/XP

The encoder optimizes ribbon usage by reversing the ribbon motion after each document is printed to take up any unused area on the ribbon. Complete ribbon reversal might not always be achieved during full field encoding because the next document can enter the encode station before ribbon reversal is completed. Ribbon reversal is always completed during amount-only encoding.

Programming Options

The Sort module can express (via the SPXServ functions) two kinds of options for programming the encoder. For detailed information about the SPXServ functions used, see the *3890/XP Series and 3890/XP Enhanced SPXServ Reference*.

Error Thresholds

Prior to starting document feeding, the Sort module establishes error threshold values by calling the SpxPutEncSetup function for the following:

- Consecutive document reject errors
- Consecutive document substitution errors
- Cumulative document reject errors
- Cumulative document substitution errors.

The threshold values for all of these error categories can be different from one another but are common to both encode paths.

Error threshold values range from zero to 255. *A value of zero disables the check* for the corresponding error category and resets any accumulated counts for that error category.

The default value for all thresholds is zero. No checking for consecutive or cumulative document errors is done unless the Sort module establishes non-zero threshold values.

Once established by the Sort module, threshold values remain in effect until the transport is powered off or new threshold values are supplied by the Sort module.

Document Disposition Modes

When a document with reject characters or character substitutions is detected, the document is *dispositioned*. This means the document goes to either:

- The *normal* pocket, as specified by the Sort module for the document (the Sort decision)

  or
- The transport's *reject* pocket.

The Sort module establishes the *disposition modes* to be used when the run is initialized, prior to starting document feeding, by calling the SpxPutEncErrDisp function. Separate dispositions can be specified for:

- Reject documents
- Character substitution documents
- Encoder device error documents.

Dispositions pertain to both encode paths. The default disposition is reject.

Print Verification

Print verification identifies, in real-time, documents that have been mis-encoded or poorly encoded. The data is read from a document immediately after it is encoded (the *verify data*) and is compared with the data that was to be encoded (the *encode data*). The print verify station ignores documents that have not been encoded.

When the station detects verification errors, an action is taken based on error thresholds specified by the Sort module. The Sort module also defines how documents are *dispositioned* after an error occurs.

The print verify station detects the following types of errors:

Reject Characters
: A reject character is indicated by the verification sub-system when a character that is read cannot be identified.

Character Substitutions
: A character substitution is indicated by the verification subsystem when a character encoded on the document does not match the character that was to be encoded.

Single Document Errors

A single document error is defined as any single document having one or more verification errors. The error is further classified by the type of error.

Reject Characters
: Any document with one or more reject characters and no character substitutions

Character Substitutions
: Any document with one or more character substitutions and zero or more reject characters

Consecutive Document Errors

A consecutive document error occurs when ‘n’ consecutive documents from an encode path have verification errors of a single type, where ‘n’ is a threshold.

Reject Characters
: ‘n’ consecutive reject error documents from an encode path

Character Substitutions
: ‘n’ consecutive substitution error documents from an encode path

Cumulative Document Errors

A cumulative document error occurs when ‘n’ documents out of the last 2048 documents from an encode path have errors of a given type, where ‘n’ is a threshold.

Reject Characters
: ‘n’ reject error documents out of the last 2048 documents from an encode path

Character Substitutions
: ‘n’ substitution error documents out of the last 2048 documents from an encode path

Transport Action

The action taken by the transport in response to verification errors depends on the document disposition mode and error thresholds.

Single Document

The document is *dispositioned* or pocketed according to the disposition mode.

Document feeding pauses if the disposition is *reject*. During the pause the Control Program obtains additional information about the document in error and flags the verification error in the document data record header byte 4, bit 2. Feeding then resumes without operator intervention.

If the disposition is *normal*, feeding does not pause and verification errors are not flagged.

Threshold Reached

When a threshold is reached, document feeding stops. Processing of remaining documents (endorsing, filming, encoding, and pocketing) is not affected. The Feed key must be pressed to resume operation.

A message identifying the error type and the encode path is displayed to the operator.

Through the SpxGetEncStatus and SpxGetEncVStat functions, the Sort module can ascertain the reason for the stop. The appropriate threshold counter is reset.

Other Errors

Other errors detected in the encoder hardware, such as ribbon breaks or hammer checks, cause document feeding to stop. The error is flagged in the document data record header byte 7, bit 2, but feeding does not stop.

Invalid data, detected by the SpxPutEncData or SpxPutHSPEData function, is also flagged in header byte 7, bit 2, but feeding does not stop.

Chapter 10. Additional Capabilities

This chapter discusses the following additional capabilities for the 3890/XP Enhanced document processors:

- Image Capture System
- Optical character recognition
- Single pick
- High read
- Logical merge.
- Full function endorsing for the 3891/XP and 3892/XP
- SPSPAUSE
- Re-Initialization with enhanced path support

Image Capture System

The Image Capture System is supported on the 3890/XP Enhanced document processor.

The Image Capture System module is located between the microfilm module and stacker module one. Documents are processed by the 3890/XP Enhanced in the same manner whether or not the Image Capture System is installed. For tracking purposes, the 3890/XP Enhanced sends an image data record to the Image Capture System when the system is active.

Initialization

The Image Capture System must be *enabled* so the hardware will be ready to perform as *requested for each document*. The enabling data is specified in the initialization data record (IREC), bytes 11 through 15, byte 110, and bytes 113 through 119.

At the completion of initialization, the control program displays a status message to the operator and returns the status in sense byte 1 to the host (if online).

Figure 10-1 (Page 1 of 2). IREC Data Used for the Image Capture System

Byte(s)	Bit(s)	Description
0	2	Initializes Image Capture System If this bit is on (1), the Control Program uses bytes 11 through 15 and 110 to initialize the Image Capture System. If byte 110, bits 0 through 3, are off (0), the Image Capture System is *disabled*. If this bit is off (0), the Control Program ignores these bytes *and does not change the existing status of the Image Capture System*.
11	0-7	Two-digit user cycle number (unsigned packed BCD)
12-15		Eight-digit user cycle date (unsigned packed BCD)

Figure 10-1 (Page 2 of 2). IREC Data Used for the Image Capture System

Byte(s)	Bit(s)	Description
80	0	Activates sort-control of auto-selects (SCA) option You can override this with the Run Profile keyword NOSCA. **Note:** When using the Image Capture System, this option *must* be enabled. This allows the Sort program to increment the INF number on auto-selected documents.
81	0	Initializes programmable endorsement and item-numbering feature (INF) If this bit is on, bytes 81 through 87 are used to initialize programmable endorsement and INF (bytes 1 through 10 are ignored). If this bit is off, bytes 1 through 10 are used to initialize programmable endorsement and INF (bytes 81 through 87 are ignored). **Note:** For the Image Capture System, we recommend you employ a 10-digit INF number.
110	0 1 2 3	Image Capture System front black-and-white Image Capture System front gray scale Image Capture System back black-and-white Image Capture System back gray scale **Note:** If these bits are off (0), the Image Capture System is *disabled* when byte 0, bit 2, is on (1).
	4	Image Capture System online
	5	Image Capture System online fill buffer
	6	Image Capture System ignore compensation errors
	7	Image Capture System ignore analysis errors
111-112		Reserved
113		Offset of DIDM key and document type
114-115		Image Capture error list length
116-119		Image Capture error list pointer

Sort Program

Your Sort program *must request* front or back scan (or both) and must request black-and-white or gray scale (in any combination) in 3890-emulated AUX storage location hex 1000, *for each document*. The following bits are used:

Bit 0 = scan front black-and-white.
Bit 1 = scan front gray scale.
Bit 2 = scan back black-and-white.
Bit 3 = scan back gray scale.

You can use the SCI “SAS” macro or SpxPutISFFeatCtl function.

The Sort program must increment the INF number for each document, whether or not the INF feature is enabled.

Document Data Record Header - Sent to Host

Byte 4, bit 4, of the document data header is the *image requested indicator* to the host system. If the bit is on (1), it indicates that a request to capture the document was sent to the Image Capture System. It does not indicate that the image was captured.

Exception Record Header - Sent to Host

Byte 0, bit 7, of the Exception Record Header on (1) indicates that a list of suspect images is available. A suspect image is one that might be unusable.

Byte 5 is the number, 0 through 255, of the first document not image captured when a quickstop occurs.

Byte 6, bit 0, on (1) indicates that the Image Capture System module has caused a quickstop. If this bit is on (1), byte 5 is valid.

Byte 6, bit 1, on (1) indicates that the Image Capture System module has detected a machine check.

Byte 6, bit 3, on (1) indicates that the Image Capture System module tracking numbers do not agree with the sorter tracking numbers.

Bytes 7-A contain the item number of the last document that the Image Capture System module successfully captured. For additional information, see the *ImagePlus High Performance Transaction System General Information Manual*.

Optical Character Recognition

The high performance Optical Character Recognition (OCR3) feature reads an OCR font code line on the front of the document in an area specified by the user. This feature is supported on the 3890/XP Enhanced document processor only.

It is referred to as OCR3 to distinguish it from OCR1 and OCR2, which are RPQs (request for price quotations) for the 3891/XP and 3892/XP document processors.

The high performance OCR feature can read the following fonts:

- OCRA numeric
- OCRB numeric
- Additional fonts available through RPQs.

The code line data read by the hardware is placed in the OCR input buffer for processing by the 3890/XP Enhanced. If 3890 C or D emulation mode is specified, the OCR data is moved to the MICR input buffer by the 3890/XP Enhanced Control Program. Otherwise, if the SCIEM modules (SCIV, SCIC, SCIF, SCII, SCID) are to be used, you must copy the data from the OCR input buffer to the XP Enhanced Control Program's magnetic ink character recognition (MICR) input buffer before these modules run. Therefore, your routine must be defined in the Run Profile as the first module to run during the SPSVERIFY event. For example, your Run Profile should have a line such as

```
SPSVERIFY yourmodule *MP
```

SpxGetOcr3Data and SpxPutInptBufDat are the appropriate functions for copying the data.

Data in fields 1 through 7 of the MICR input buffer should be numeric only. Verify, Correction, and Format errors occur if alpha data is found in the MICR input buffer in fields 1 through 7. Except for SCII, the IBM-supplied SCIEM modules file work on fields 1 through 7 only.

The operator can disable the high performance OCR feature, from the Run screen. An indicator on the 3890/XP Enhanced transport operator panel indicates the state of the feature, on or off.

Initialization

Initialization for the OCR feature is specified in the initialization data record (IREC) and the Run Profile. The Control Program uses this to initialize the feature either “on” or “off” and to specify the font types. When the feature is initialized, it sends a return code to the Control Program. The Control Program, in turn, communicates the feature’s status to the operator (using an operator message) and, if online, to the host as sense byte 1.

To initialize the OCR feature, the following data is required:

- IREC bytes 78 (feature enable) and 120 (extended field type definitions)
- Run Profile (3890 emulation, font types, and translate table)
- Scan area on the document (see below).

OCR Capture Processor (OCP)

The OCR Capture Processor (OCP):

1. Takes image data from an optical scanner located within the transport
2. Performs recognition processing against the image
3. Passes the resulting character string back to the Control Program.

Each character recognized in the OCP has a specific byte value that represents this particular character. These representations are primarily ASCII. However, there are certain special symbols that are not defined in the ASCII character set definition. These characters are assigned code-points, or positions, in the ASCII set that are not otherwise used.

The tables below illustrate the hex code-points for the OCRA and OCRB fonts:

OCRA Characters	
Character	Hex Code
0 to 9	30 to 39
Space	20
⌡ Hook	B2
⅄ Fork	B0
⊓ Chair	B1
Long Vertical	7C
Error	1A

OCRB Characters	
Character	Hex Code
0 to 9	30 to 39
Space	20
/ Slash	2F
+ Plus	2B
< Less Than	3C
> Greater Than	3E
– Minus	2D
Long Vertical	7C
Error	1A

Figure 10-2. OCRA and OCRB Hex Code-Points

The OCP translates the character code-points according to the 256-byte translation table you specify on the TRANStbl keyword in the Run Profile (the OTT file). The default translate table is CXLATE.OTT.

This translation is a simple table lookup in which the code-point for each recognized character is used as an offset into a table of byte values. The value pointed to replaces the code-point in the original character string. The resulting string is then sent to the XP Enhanced document processor and is placed in the OCR input buffer.

Scan Area

The operator can adjust the scan area on the document from the command line on the Run screen. After entering the SCAn area command, the operator enters the OCR scan area in the window that appears on the screen. Valid decimal values are 0 through 928. If no scan area is entered, the scan area as last entered is used.

In addition, the Scan Area window allows you to enable blank “characters” and long vertical marks. The window indicates the last state of these enables (on or off) and retains these states unless changed from this window.

Single Pick

The 3891/XP and 3892/XP document processors do not have a hardware single document start key like the 3890/XP Enhanced. Instead, a single pick function is provided by the Control Program.

A call to the SpxSinglePick function by your Sort program causes the Control Program to request just one document from the transport when the Feed button is pressed. The call should be made at SPSINIT time.

Running Offline

If running Offline from the host, the motors stop and control returns to the Run screen. If a call is also made at SPSDOC or SPSMS time, single pick would again be applied to the next Feed. In this way, you can single step through a set of documents and look at Sort program messages for each one, perhaps looking for a document that causes an error. (Messages are displayed only when the Control Program returns to the Run screen after the transport stops.)

Running Online

If running Online to the host, the Control Program sets the Sort Pause Requested flag in the record header. When the machine pauses (rather than stops), documents are flushed to the host and the Control Program then waits for "Host OK to Start" before feeding the next document. The host can download different initialization data at this time and therefore run a job defined or called for by the single "beginning of job" document that was flushed to the host. In this second initialization, do not call for another single pick.

Later in the run, the Sort module can resume single pick, such as when a special "end of job" control document is seen. There should be a number of these "end of job" documents, to allow for the fact that the machine does not pause immediately. These "end of job" documents can be followed by a "beginning of job" special document. The effect is that many different sort jobs can be run as a single operation at the machine and without operator intervention.

High Line Read/MICR2

The High Line Read feature (for the 3892/XP only) allows the reading of a second code line (MICR2) in a location other than the standard area. A second, vertically adjustable read head is supplied for this function. The MICR2 read system provides all of the functions of the MICR system. The SpxGetMicr2Data function allows access to the 266-byte MICR2 read record.

Logical Merge

Logical Merge is a feature for a 3892/XP document processor that is configured without a merge feeder. This function allows you to call the SCI or SPX “feed merge” functions and to use “merge before main” and “merge on pocket count” on a machine that does not have a merge feeder.

Initialization Time (SPSINIT)

Logical Merge begins when your Sort module calls the SPXLogMrg function. Physical merge is ignored until the next initialization, and all merge document requests are handled as described below.

Document Time (SPSDOC)

Your Sort module reads the process buffer, via Spx, and decides whether the current document data describes a merge document according to your definition of a merge document. If so, your module must set the merge document bit in the header (byte 8, bit 4). *This must be done prior to execution of other Sort processing*.

If you are code line data matching, your module must run before the SPSIMATCH event. Your Run Profile should have an entry such as:

```
SPSFORMAT  *MP yourmodule
```

The Control Program sets a “merge requested” flag if a merge document is requested via any of the following:

- The SCI “FM” macro
- The SpxFM function
- Pocket counter reaching its limit
- Merge before main.

Your Sort module looks at this flag, via the SpxLogMrg function, and begins a “tolerance” count; that is, it determines how many non-merge documents you can tolerate before your Sort module must decide whether to stop the machine and what to tell the operator.

The operator places merge documents in the batch of work at appropriate intervals to ensure that the tolerance is met so the machine does not have to be stopped for a merge document to be inserted.

Your SPSINIT Sort module must examine the initialization data for merge before main, for which a tolerance count of zero is appropriate.

Full Function Endorsing for the 3891/XP and 3892/XP

Full function endorsing is a method of endorsing documents without the use of the CONVERGE DLL. Replacing Converge requires you to use two Spx commands, SpxPutPrtSeq and SpxInkJetCtrl. For details on the specific Spx commands, see the *IBM 3890/XP Series and 3890/XP Enhanced SPXServ Reference*.

The functions that are supported by this method of endorsing include:

- INF Capability on any endorsement line in any line position based on the use of the SpxPutPrtSeq and SpxInkJetCtrl Commands
- Endorse Without Converge
- Any combination of Variable, Fixed, or Serial Number data on any endorsement line, front or back
- Large font capability for both sides of the document using any data described above
- The ability to have the placement of the third endorsement line remain constant with or without the INF Feature while using the existing CONVERGE DLL
- The ability to invert and rotate any or all endorsement lines (the same orientation as the 3890/XP Enhanced).

Existing Spx Commands Compatible with this Endorsement Method

The following list includes the Spx commands that this method uses for input Endorse data fields.

SpxPutAltInf: This Spx puts an alternative item number to the machine. The Spx remains the same.

SpxLoadFixM: This Spx continues to be the only way to specify the fixed messages to the machine during the initialization process. Any fixed message used during an operation must be loaded during initialization before being requested at document time.

SpxFixM: This Spx remains the same, and continues to be used for an all-fixed message endorsement, as well as fixed message support for converged endorsements. If you use SpxFixM and SpxPutInkJetCntrl to specify the same line of endorsement, the machine returns an option state error and nothing is printed.

Existing Spx Commands Not Directly Compatible with this Endorsement Method

SpxGetPrgEndData: This Spx is used to get a copy of the programmable endorse data from the 3890 data area. On a 3891/XP or 3892/XP machine, this Spx gets usable data only when used with the CONVERGE DLL.

SpxPutPrgEndData: The **Put** data Spx, like the **Get** data Spx, handles valid data only when used with the CONVERGE DLL.

Full Function Endorse Usage

This section describes how to add this function to your sort programs.

- Irec: No changes
- Regcc or Noregcc keyword addition
- SPSINIT entry(s) addition
- SPSPOST entry addition
- Remove CONVERGE DLL and entry points
- New USERDLL containing the endorse logic using the Spx commands defined in this section.

SPSINIT Time: The SPSINIT Routine in your USERDLL can use Spx commands to complete the following functions.

- SpxPutPrtSeq - Set up the requested print sequence.

- SpxLoadFixM - Define up to 15 fixed messages.
- SpxPutInkJetCtrl - Set up any or all lines of endorsement data. This Spx can be used at any event, allowing the endorsement to be changed at any time.

SPSPOST Time: The SPSPOST routine in the USERDLL can use Spx commands to complete or change the endorsement data definitions. Multiple (up to 6) SPXPutInkJetCtrl commands can be issued for each document. The data area initially sets to all zeros. Once an endorsement line has been requested, it remains the same until acted on by a subsequent SpxPutInkJetCtrl command. The data area is cleared between initializations.

SPSPAUSE

The SPSPAUSE event gives the native programmer the ability to pause the separator from feeding documents. Once paused, this event is handled the same as the other non-document events.

The customer programmer uses an Spx command (SPXPAUSE) to set a flag that is used by the Control Program to request a pause condition. This pause request is added to the existing pause request signals:

- Sort Pause from Process buffer Header byte 9
- Microfilm Pause
- Host Interface Pause
- Pocket full
- Auto-Selects (UT/XP only).

Once the sorter is paused and the in-flight documents have all been processed, the control routine processes the list of routines specified in the current Run Profile with the SPSPAUSE keyword.

The SPSPAUSE event is timed using the SPXNDPTIME parameter as specified in the Run Profile.

Once the paused routines are completed, the Control Program clears the pause request flag and attempts to restart the separator.

If the motors have timed out during the processing of the PAUSED event routines, use the start key to restart the paper flowing.

Re-Initialization with Enhanced Path Support

The 3890/XP Enhanced document processors support a re-initialize request from a sort routine running during the SPSINIT event.

During normal initialization, the sorter processes the initialization record (IREC), processes the Run Profile and associated files, and calls any SPSINIT routines requested. If an SPSINIT routine requests re-initialization, the sorter processes the Run Profile and associated files, and then calls any SPSINIT routines requested. The sorter does not re-process the IREC during re-initialization, except for extracting the name of the Run

Profile. This allows the SPSINIT routine to change the name of the Run Profile. The SPSINIT routine can specify a new path for the sorter to use when accessing the Run Profile and associated files during re-initialization.

Run Initialization Processing

Once the sorter loads program store from the host or .SCI file, processing by the Run Initialization component of the sorter is as follows:

1. Run Init clears the re-initialization request flag and the re-initialization load path data area.
2. Run Init processes the IREC.
3. Run Init processes the Run Profile named in the IREC (or Machine Profile) and processes the associated files referenced in the Run Profile.
4. Run Init locates and loads files as follows:

   Run Profile — If not null, uses the re-initialization load path (as specified by a sort routine through **SpxPutReinitLoadPath**). If re-initialization load path is null, uses the load path specified in the Machine Profile.

   Associated Files — If not omitted, uses the path specified in the Run Profile. If path is omitted in the Run Profile and the re-initialization load path is not null, uses the re-initialization load path. If path is omitted in the Run Profile and the re-initialization load path is null, uses the current directory.[1]

   Note: Associated Files include OTT, PDT, PED, PSD, SMT, and SST files.

   DLL Files — If not null, uses the re-initialization load path. If re-initialization load path is null, uses the path specified by LIBPATH in CONFIG.SYS.

5. Run Init calls any SPSINIT routines requested.

   A SPSINIT routine can:

   - Change the Run Profile name (and format flag) in the IREC (using **SpxPutProgStore**)
   - Change the re-initialization load path (using **SpxPutReinitLoadPath**)
   - Request re-initialization (using **SpxPutReinitReq**)

6. If no errors, aborts, RNIs, or previous re-initializations have occurred, and re-initialization has been requested, then Run Init repeats steps 3 and 5.

---

[1] For the SMT and SST files, Run Init uses the default drive/path specified in the Machine Profile, rather than the current directory. If no path is specified in the Machine Profile, the current directory is used. Run Init always loads DEFAULT.SMT from the C:\INIT directory, and if no SST file is specified, loads ABA.SST from C:\INIT.

Notes:

1. Run Init performs re-initialization one time only. That is, Run Init does not honor a request to re-initialize during re-initialization.
2. Run Init adds a trailing backslash to the re-initialization load path, if needed.
3. For DLL names, Run Init adds the trailing .DLL extension required for complete DLL specifications.
4. Prior to performing re-initialization, Run Init frees the DLL modules loaded during initialization. This frees storage for any new DLL modules to be loaded during re-initialization.
5. Run Init does not change the sorter's actual current directory so that current sorter operation (writes of PED.NAM and LAST.PED) is not affected.

Appendix A. Run Initialization Error Codes

During run initialization, the Control Program performs validity checks. If these checks all receive valid responses, the Control Program sets a return code of X'00'. Otherwise, the Control Program returns one of the following codes.

Note: The following list shows the defined errors with the hexadecimal return code. The first value is the message number displayed on the operator's panel. The second value represents the value sent to the host in sense byte 1. Error codes 00x through 1Fx are equivalent to the 3890 Model A, B, C, or D return codes.

5000 00x Run initialization completed successfully.

Explanation: Run initialization has processed the IREC data, the Run Profile, and any defaults set up in the customization menus. The document processor is ready to start processing.

5001 01x Control unit not ready.

Explanation: Error set by the Channel.

Action: Notify your service representative of the problem.

5002 02x Set load command was processed.

Explanation: Error set by the Channel.

Action: Notify your service representative of the problem.

5003 03x Code line data match buffer initialized with code line data matching OFF.

Explanation: The code line data match buffer was requested to be initialized from the IREC data (byte 0, bit 6, equal to 1) while code line data matching from the IREC data (byte 21, bit 2, equal to 0) was not selected.

If code line data matching is used, then the code line data match flag (byte 21, bit 2) must be set to 1 in the IREC. If code line data matching is not to be used, then the code line data match buffer flag (byte 0, bit 6) must be set to 0 in the IREC.

Action: Correct and reinitialize.

5004 04x Code line data match buffer required but not requested.

Explanation: Code line data matching was requested (IREC byte 21, bit 2, equal to 1) but the code line data match buffers were not requested to be initialized (IREC byte 0, bit 6, equal to 0). If code line data matching is to be used, then the code line data match flag (byte 21, bit 2) must be set to 1 in the IREC. If code line data matching is not to be used, then the code line data match buffer flag (byte 0, bit 6) must be set to 0 in the IREC.

Action: Correct and reinitialize.

5005 05x Field definition requests code line data matching but code line data match is off.

Explanation: Code line data match was requested in a field definition, but code line data matching was not requested (IREC byte 21, bit 2, equal to 0).

If code line data matching is used, then the code line data match flag (byte 21, bit 2) must be set to 1 in the IREC. If code line data matching is not used, then the code line data match flag in the field definition must be set to 0 in the IREC.

Action: Correct and reinitialize.

5006 06X The byte count for fields 0 - 15 > 256 or fields 0 - 8 > 48.

Explanation: The number of bytes that was requested in the field definition for fields 0 through 15 is greater than 256. In emulation mode, the field definition for fields 0 through 8 is greater than 48.

Action: Check the field lengths in IREC bytes 22 through 36 for emulation mode and IREC bytes 96 through 110 for fields 9 through 15. Correct and reinitialize.

5007 07x Microfilm feature not installed or failed initialization.

Explanation: The microfilm feature was requested, but there is no microfilm feature installed or the feature jumpers are not installed.

Action: Check the feature jumpers and IREC byte 0, bit 3. Correct and reinitialize.

5009 09x Endorser feature not installed or failed initialization.

Explanation: The endorse feature was requested, but there is no endorse feature installed or the feature jumpers are not installed.

Action: Check the feature jumpers and IREC byte 0, bit 1 or IREC byte 81, bit 0. Correct and reinitialize.

5010 0Ax INF feature not installed or failed initialization.

Explanation: The INF was requested, but there is no INF installed or the feature jumpers are not installed.

Action: Check the feature jumpers and IREC byte 0, bit 0, or IREC byte 81, bit 0. Correct and reinitialize.

5011 0Bx SCI start/error address less than 00A0 or SCI length =0.

Explanation: The SCI starting address or error address is less than 160, or the SCI routine that is currently loaded has a length of zero.

Action: Check the SCI starting address for 2-byte or 4-byte addresses. Check the length of the current SCI program; it must be greater than zero bytes long. Correct and reinitialize.

5012 0Cx Debug mode requested with code line data matching off.

Explanation: Code line data matching must be ON when debug mode is requested.

Action: Set code line data matching from the initialization record and reinitialize.

5013 0Dx Field 1-15 requested too many digits.

Explanation: The field definitions in the initialization records specified too many digits for one of the fields 1 through 15. This error is set if the number of digits that is specified for this field is greater than two times the number of bytes specified.

Action: Inspect the initialization record. Correct and reinitialize.

5014 0Ex Pocket requested that is not installed.

Explanation: Either a pocket requested in the initialization record is not installed in the document processor or the feature jumpers for that pocket module have not been installed.

Action: Correct the initialization record or have the feature jumpers installed. Reinitialize.

5015 0Fx Microfilm invalid decimal data found.

Explanation: An invalid decimal value was specified in the initialization record for the microfilm starting number.

Action: Inspect the IREC bytes 17 through 20 for valid data. Correct and reinitialize.

5018 12x Code line data matching requested while offline.

Explanation: You can select code line data matching only while the sorter is online to the host.

Action: Inspect the IREC data for validity. Correct and reinitialize.

5019 13x Insufficient storage for code line data match buffers.

Explanation: Error set by the Channel or LU 6.2.

Action: Notify your service representative of the problem.

5024 18x Unable to Lock Required Memory Storage.

Explanation: The Control Program is unable to lock memory storage for a time-critical Sorter operation. The amount of program storage and the amount of storage used by your DLL exceed available memory storage.

Action: Reduce the amount of program storage requested in your machine profile and the amount requested by your DLL. Stop other programs and processes running on the PS/2.

5031 1Fx Request not initialized OR abort sort has been set.

Explanation: This messages appears if the sort routine detects an error during an SPS initialize OS/2-language routine, which runs at the end of the run initialization processing. If one of these error messages occurs, the sort routine detected an error possibly set by the OS/2-language routine. If the OS/2-language routine sets the error, an additional message on the run screen might show more details of the error.

Action: Correct and reinitialize.

5032 20x Run profile not found.

Explanation: The Run Profile was not found. Search the following areas:

1. The Run Profile that was specified in the IREC.
2. If the IREC field is blank, then the Run Profile that was specified in the machine Customization Screens.

In either case, the drive and path specified in the machine Customization Screens are searched.

Action: Verify the drive and path that were specified in the machine Customization Screens. Inspect the IREC data. Correct and reinitialize.

5033 21X I/O error reading file (Run Profile).

Explanation: The Run Profile was found but could not be read.

Action: Verify that the Run Profile is not already open in another session. Correct and reinitialize.

5035 23x Invalid data contained in Run Profile.

Explanation: An invalid character was found in the Run Profile after a valid keyword.

Action: Verify the syntax of the keyword parameters in the current Run Profile. Correct and reinitialize.

5036 24x Invalid keyword in the Run Profile.

Explanation: An invalid keyword, or character in the keyword field, was detected.

Action: Verify the syntax of the keyword parameters in the current Run Profile. Correct and reinitialize.

5046 2Ex *NONE Run Profile requested while not initialized.

Explanation: *NONE can be used as a Run Profile name only when the sorter has been initialized.

Action: If Run Profile processing is not required for the immediate operation, initialize the sorter once with a default Run Profile; then reinitialize with a *NONE profile name.

5047 2Fx Run profile name conversion error.

Explanation: The Run Profile name can be specified in the initialization record in EBCDIC form, as controlled by IREC byte 61.

Action: If IREC byte 61 = 00, check the IREC field for valid EBCDIC characters. If IREC byte 61 = 01, the name field must be specified in ASCII. Correct and reinitialize.

5048 30x Program Storage Data file not found.

Explanation: The Program Storage Data (PSD) file could not be located on the specified directory.

Action: Verify that the PSD file exists on the directory that is specified in the Run Profile. Correct and reinitialize.

5049 31x I/O error reading file. (PSD)

Explanation: The Program Storage Data (PSD) file was found but could not be read. The file might be open in another session.

Action: Close the file and reinitialize.

5050 32x Invalid Filespec. (PSD)

Explanation: The PSD file specification definition, as specified in the Run Profile, is invalid.

Action: Verify the syntax and the drive or the path statement in the current Run Profile. Correct and reinitialize.

5055 37x Invalid address for PSD file.

Explanation: The address, in the PSD keyword, for the offset from the beginning of the program store is beyond the current size of the program store. The size of the program store is maintained from the customization screen.

Action: Increase the size of the program store. Edit the Run Profile to change the address in the PGMSTOREdata keyword. Correct and reinitialize.

5056 38x Insufficient memory to load PSD file.

Explanation: The address for the PSD file is valid, but the file is larger than the current memory that is available from the address to the end of the program store.

Action: Either specify a lower starting address to contain the entire file or increase the size of the program store in the customization screens. Reinitialize.

5063 3Fx Program store address conversion error.

Explanation: The address for PSD files is specified in the Run Profile in ASCII format. An invalid value was found for this address.

Action: Verify that the address that is specified in the PGMSTOREdata keyword is valid ASCII characters (A-F, 0-9). Correct and reinitialize.

5064 40x Pocket Definition Table (PDT) file not found.

Explanation: PDT could not be found in the specified directory. The PDT information can be received from either the Run Profile or the customization screens.

Action: Verify that the table exists on the specified directory. If there is no entry in the Run Profile, use the customization screen entry. Correct and reinitialize.

5065 41x I/O error reading file. (PDT)

Explanation: The file was found but could not be read. The file might be open in another session.

Action: Close the file and reinitialize.

5068 44x Invalid keyword in pocket definition table.

Explanation: There is an invalid keyword in the PDT. The valid keywords are as follows:

MODULE1
MODULE2
MODULE3
MODULE4
MODULE5
MODULE6

* in column one specifies a comment and is allowed.

Action: Verify the PDT keyword. Correct and reinitialize.

5069 45x More than one pocket definition table requested.

Explanation: The PKTDEFtbl keyword has specified two or more files for the pocket definitions in the Run Profile.

Action: Correct and reinitialize.

5070 46x Duplicate keywords in pocket definition table.

Explanation: You specified a keyword more than once in the PDT. For example, there might be two MODULE1 keywords.

Action: Verify the correct entry in the xxx.PDT file for this operation. Either delete one of the two entries or change one of the two entries into a comment field by inserting an asterisk (*) before the keyword. Reinitialize.

5073 49x Keyword parameters in pocket definition do not equal 6.

Explanation: Each keyword in the PDT must contain six decimal parameters. If you do not want a count for a pocket, enter a zero as a place holder.

Action: Verify that each MODULEx (x = 1 through 6) keyword contains six decimal numbers separated by one or more blanks. Correct and reinitialize.

5074 4Ax Keyword parameter value outside valid range 0-4095.

Explanation: The valid parameters for the MODULEx (x = 1 through 6) keywords must be valid decimal numbers in the range 0 through 4095.

Action: Verify the currently selected PDT for syntax and valid decimal values. Correct and reinitialize.

5079 4Fx Pocket attribute conversion error.

Explanation: The values that were specified in the PDT for the merge pocket count values were specified in ASCII format. This error indicates that an invalid character has been detected.

Action: Verify the PDT for syntax and valid parameters. Correct and reinitialize.

5080 50x Programmable Endorsement Data (PED) file not found.

Explanation: PED file xxx.PED could not be located on the specified directory. The PED file can be specified in the current Run Profile or as an extension to the endorse command from the run screen.

Action: Verify that the file exists on the specified directory. Correct and reinitialize.

5081 51x I/O error reading file (PED file).

Explanation: The file was found but could not be read. The file can be open in another session.

Action: Close the file and reinitialize.

5082 52x Invalid file spec (PED file).

Explanation: The file specification given for the PED file is incorrect.

Action: Verify the Run Profile entry PRGENDRSdata for syntax and parameters. Correct and reinitialize.

5083 53x More than 1 PED file requested.

Explanation: Two or more files have been specified for the Endorsement.

Action: Verify that the current Run Profile contains only one programmable endorse definition file designated by the PRGENDRSdata keyword.

5084 54x Invalid programmable endorse keyword.

Explanation: There is an invalid keyword in the PED file. The only valid keywords are ENDORSE and an asterisk (*), which is used to indicate a comment field.

Action: Verify the PED file that is currently specified in the Run Profile for correct keywords and comments. Correct and reinitialize.

5085 55x LAST.PED endorse file not opened when requested.

Explanation: The internal endorse data file could not be opened. File is not saved.

Action: Call your service representative.

5086 56x LAST.PED endorse file error (UNKNOWN).

Explanation: The internal endorse data file had errors in processing.

Action: Call your service representative.

5087 57x LAST.PED endorse file write failure. Not updated.

Explanation: The internal endorse data file could not be written. The file is not saved.

Action: Call your service representative.

5088 58x LAST.PED endorse file close failure. File not closed.

Explanation: The internal endorse data file could not be closed. The file might have lost data.

Action: Call your service representative.

5089 59x More than three ENDORSE keywords in PED data file.

Explanation: More than three ENDORSE keywords are in the PED file. Three is the maximum for this file.

Action: Verify the current PED file for syntax and the number of keywords present. Correct and reinitialize.

5091 5Bx More than 24 characters on line in PED file.

Explanation: The PED file contains a line with more than 24 characters.

Action: Verify the PED file for the correct number of characters per line (24 or fewer). Correct and reinitialize.

5092 5Cx Invalid attribute specified in endorse definition.

Explanation: An invalid character was detected in the endorsement data contained in the PED file. The valid characters are as follows:

A-Z space # $ & , - . / 0-9 < > = : ` ' *

Action: Verify the characters used in the PED file. Correct and reinitialize.

5095 5Fx EBCDIC to ASCII endorse date conversion error.

Explanation: The endorse date can be specified in the initialization record in EBCDIC form, controlled by IREC byte 61.

Action: If IREC byte 61 = 00, check the IREC field for valid EBCDIC characters. If IREC byte 61 = 01, then the date field must be specified in ASCII. Correct and reinitialize.

5096 60x Sort message table file not found.

Explanation: The Sort Message Table (SMT) file could not be found in the specified directory. The xxx.SMT can be specified in the current Run Profile or in the customization screens.

Note: A default sort message table is loaded during every initialization.

Action: Verify that the file exists on the specified directory. Correct and reinitialize. If the default file has been erased, call your service representative.

5097 61x I/O error reading file (sort-message table).

Explanation: The SMT file was found but could not be read. The SMT file might be open in another session.

Action: Close the SMT file and reinitialize.

5098 62x Invalid File Spec (sort-message table).

Explanation: The file specification given for the SMT file is incorrect.

Action: Verify the Run Profile entry SORTMSGtbl for syntax and parameters. Correct and reinitialize.

5099 63x Invalid Parameter in (sort message table) SMT.

Explanation: The valid keywords in the sort message table are decimal numbers 001-255.

Action: Correct and reinitialize.

5102 66x More than 1 Sort message table had been requested.

Explanation: Two or more files are specified by the Run Profile SORTMSGtbl keyword.

Action: Correct and reinitialize.

5105 69x Sort message table keyword outside range (001-255).

Explanation: The valid keywords in the sort-message-table (SMT) file are decimal numbers in the range 001 through 255. These error messages indicate that a value outside the acceptable range has been entered.

Action: Verify the SMT file currently specified for use in this operation. Correct and reinitialize.

5106 6Ax No severity level specified for sort operator message.

Explanation: The valid SMT file entry is shown below.

For text message XXX Y:

XXX is a decimal number in the range 001 through 255.

Y is one of the following:

I for information messages (white)
W for warning messages (yellow)
E for error messages (red).

Action: Correct and reinitialize.

5107 6Bx Invalid severity level specified for sort operator message.

Explanation: A value other than I, E, or W is specified for the message severity level.

For text message XXX Y:

XXX is a decimal number in the range 001 through 255.

Y is one of the following:

I for information messages (white)
W for warning messages (yellow)
E for error messages (red).

Action: Correct and reinitialize.

5111 6Fx Sort Message number conversion error.

Explanation: A value specified for the XXX value could not be converted from ASCII to binary.

For text message XXX Y:

XXX is a decimal number in the range 001 through 255.

Y is one of the following:

I for information messages (white)
W for warning messages (yellow)
E for error messages (red).

Action: Correct and reinitialize.

5112 70x Special symbol sequence table not found.

Explanation: The SST file was not found on the specified directory. The SST table can be specified in the current Run Profile or in the customization screens. A default SST table is loaded for any initialization that does not specify a table for this operation.

Action: Verify that the file exists on the specified directory. Correct and reinitialize.

5113 71x I/O error Reading file (symbol-sequence table).

Explanation: The file was found but could not be read. The file might be open in another session.

Action: Close the file and reinitialize.

5114 72x Invalid file spec (symbol-sequence table).

Explanation: The file specification for the SST file is incorrect.

Action: Verify the Run Profile entry SYMSEQtbl for syntax and parameters. Correct and reinitialize.

5115 73x Invalid data in SST file.

Explanation: The valid parameters for SST entries are as follows:

For the OPEN, CLOSE, and the ALTERNATE parameter, the valid entries are SS1, SS2, SS3, SS4, and SS5 (and 0 as a place holder).

For the TRANSMIT parameter, the valid entries are SS1, SS2, SS3, SS4, and SS5.

Action: Verify the specified SST entry. Correct and reinitialize.

5116 74x Invalid keyword in symbol-sequence table.

Explanation: The valid keywords for SST files are the OPEN, CLOSE, ALTERNATE, and the TRANSMIT keywords. An asterisk (*) indicates a comment line.

Action: Verify the SST for syntax and validity of keywords. Correct and reinitialize.

5117 75x Keyword used more than once in symbol table.

Explanation: A keyword has been specified more than once in the symbol sequence table.

Action: Correct and reinitialize.

5118 76x More than 1 symbol-sequence table requested.

Explanation: A keyword has been specified twice in the SST.

Action: Edit the specified symbol-sequence table. Correct and reinitialize.

5121 79x No OPEN keyword found in symbol-sequence table.

Explanation: An OPEN keyword is required in the SST.

Action: Edit the specified SST. Correct and reinitialize.

5122 7Ax No CLOSE keyword found in symbol-sequence table.

Explanation: A CLOSE keyword is required in the SST.

Action: Edit the specified SST. Correct and reinitialize.

5123 7Bx Duplicate symbol used in field and transmit keywords.

Explanation: If a symbol is used as a parameter of the TRANSMIT keyword, it cannot be used as a parameter of any other keyword in the table.

Action: Edit the specified SST. Correct and reinitialize.

5124 7Cx Too many attributes specified in symbol-sequence table.

Explanation: The OPEN keyword, the CLOSE keyword, and the ALTERNATE keyword can specify up to seven symbols or place holders. The TRANSMIT keyword can specify up to five symbols.

Action: Verify the specified SST for the correct number of parameters. Correct and reinitialize.

5128 80X Native-program file not found or insufficient memory.

Explanation: Either the specified OS/2-language program was not found on the path specified on the LIB statement in the CONFIG.SYS file or there is not enough storage to contain the file. The return codes are as follows:

2 File not found
8 Not enough memory
15 Invalid drive/path
123 Invalid name.

Action: Verify the LIB statement in the CONFIG.SYS file for the correct path statement for the DLL file to be loaded. Also verify the number of segments requested in the customization screens. If you decrease this number, there is more space in which to load PC-language modules. Correct and reinitialize.

5134 86x Too many USERDLL files requested (greater than 10).

Explanation: Up to 10 OS/2-language modules can be loaded for any initialization.

Action: Consider combining one or more OS/2-language modules into one module. Correct and reinitialize.

5140 8Cx ASCII to EBCDIC translation error on a CALLS keyword.

Explanation: The ASCII to EBCIDIC translation on a CALLS keyword failed.

Action: Call your service representative.

5141 8Dx Too many calls specified.

Explanation: A limit of 255 CALLS can be accepted for any single initialization.

Action: Correct and reinitialize.

5142 8Ex Unable to find calls entry point.

Explanation: The specified entry point could not be found in the OS/2-language routine.

Action: Verify that the OS/2-language routine containing the entry point has been loaded before the CALLS keyword in the Run Profile. Correct and reinitialize.

5143 8Fx Too many level control labels specified.

Explanation: A limit of 10 level control labels for any one initialization has been set up.

Action: Correct and reinitialize.

5144 90x Invalid parameter for RPQ keywords.

Explanation: The valid entries for the RPQACTive keyword are:

ZFA	NOZFA	Zero for able RPQ
DCD	NODCD	Don’t care digit RPQ
TRT	NOTRT	Transit field transparency RPQ.
SCA	NOSCA	Sort Control of Auto Selects.

Action: Verify the RPQACTive line of the Run Profile currently in use. Correct and reinitialize.

5145 91x Conflicting parameters for RPQ keyword.

Explanation: Both options were specified for the RPQACTive keyword in the Run Profile.

Action: Verify the RPQACTive line of the Run Profile currently in use. Correct and reinitialize.

5146 92x Multiple parameters for RPQ keyword.

Explanation: The current Run Profile contains more than one RPQACTive keyword.

Action: Verify the number of RPQACTive keywords in the current Run Profile. Correct and reinitialize.

5149 95x Restart Operation Field Length mismatch.

Explanation: This error indicates that the field length read from the Restart file header does not equal the current field definition in the initialization record (IREC).

Action: Correct and reinitialize.

5150 96x The current restart file is empty.

Explanation: This error indicates that the Restart File is either empty or could not be accessed.

Action: Correct and reinitialize.

5151 97x Restart operation identification mismatch.

Explanation: This error indicates that the Restart ID read from the Restart file header does not equal the current Restart Identification in the initialization data record (IREC).

Action: Correct and reinitialize.

5152 98x Restart operation has completed successfully.

Explanation: This message indicates that the Restart operation has completed successfully. The machine is now ready to be initialized for the next operation.

Action: Prepare to initialize.

5153 99x Restart operation was attempted offline.

Explanation: This error indicates that the Restart operation was attempted offline. The machine is now ready to be initialized for the next operation.

Action: Put the machine online and reinitialize.

5162 A2x Duplicate SORTName keywords specified.

Explanation: More than one SORTName keyword was found in the current Run Profile.

Action: Edit the Run Profile currently being used and verify that only one SORTName keyword exists. Correct and reinitialize.

5164 A4x Too many OCR3 fonts requested.

Explanation: More than two fonts have been requested.

Action: Edit the Run Profile currently being used. Correct and reinitialize.

5165 A5x OCR3 was requested but failed initialization.

Explanation: The OCR3 feature was requested but was not installed or failed to initialize.

Action: Check for OCR3 power on. Correct and reinitialize.

5166 A6x An invalid font type was requested for OCR3.

Explanation: An OCR3 font type that is not supported was requested.

Action: Edit the Run Profile currently being used and correct the font type. Reinitialize.

5167 A7x An invalid scan area was specified for OCR3.

Explanation: The specified scan area was not in the allowed range.

Action: Correct and reinitialize.

5168 A8x OCR3 translate table not found or name is not ASCII

Explanation: The OCR3 translate table xxx.OTT could not be located on the specified directory. The return codes are as follows:

4 Too many files open
5 Access denied
32 Sharing violation
87 Invalid parameter
110 Open failed
112 Disk full.

Action: Verify that the file exists on the specified directory. Correct and reinitialize.

5169 A9x I/O error reading the OCR3 translate table.

Explanation: The translate table was found but could not be read. The file might be open in another session.

Action: Close the file and reinitialize.

5170 AAx Invalid data contained in OCR3 translate table.

Explanation: The translate table must be 256 bytes in length.

Action: Verify that the translate table format is correct. Correct and reinitialize.

5171 ABx OCR3 translate table invalid or not specified.

Explanation: If the OCR3 feature is to be used, a translate table must be specified.

Action: Correct and reinitialize.

5172 ACx Model C or D emulation requested without requesting OCR3.

Explanation: Model C or D emulation was requested in the Run Profile but the OCR3 feature is not installed.

Action: Correct Run Profile and reinitialize.

5173 ADx OCR3 failed initialization command.

Explanation: The OCR3 responded to an attempt to initialize with a return code other than 0. The return codes are as follows:

1 Hardware detected failure.
2 Invalid or missing data.

Action: For data errors, check with the systems operator. For hardware failures, notify your service representative of the problem.

5185 B9x SPS entry point not found.

Explanation: An entry point in an OS/2 language routine could not be found. The return codes are as follows:

6 Invalid handle
127 Entry point not found.

Action: Verify that the entry point is in your DLL. Correct and reinitialize.

5186 BAx *NONE is not the first parameter in the SPS keyword.

Explanation: When *NONE is specified in the SPS keyword, it must be the only parameter.

Action: Check the current Run Profile. Correct and reinitialize.

5187 BBx All SPS keywords specify *NONE. No sort routine loaded.

Explanation: The sort routine must contain loaded entry points.

Action: Correct and reinitialize.

5188 BCx *NONE is followed by an additional parameter.

Explanation: When *NONE is specified in the SPS keyword, it must be the only parameter.

Action: Check the current Run Profile. Correct and reinitialize.

5190 BEx Too many sort program sequences specified.

Explanation: The SPS contains too many entries. The allowable limit for any sort program sequence is 32 entries.

Action: Verify the number of entries for each SPS event. Include the entries listed in the machine profile. Correct and reinitialize.

5191 BFx Invalid SPS keyword.

Explanation: An SPS event was requested but the characters following the SPS prefix do not constitute a valid SPS keyword.

Action: Edit the current Run Profile for validity of the SPS entries. Correct and reinitialize.

5208 D0x Invalid parameter for SPXNDPTIME.

Explanation: The value specified for the SPXNDPTIME keyword exceeds the allowable value of 1000 decimals (.0001).

Action: Verify the SPXNDPTIME keyword for the parameter listed in the current Run Profile. Correct and reinitialize.

5210 D2x Multiple parameters for SPXNDPTIME.

Explanation: More than one SPXNDPTIME keyword is specified in the current Run Profile. One SPXNDPTIME keyword is the acceptable limit.

Action: Verify the number of SPXNDPTIME keywords listed in the current Run Profile. Correct and reinitialize.

5213 D5x The scanner feature failed to initialize.

Explanation: The Scanner Feature failed to respond to an initialization request.

Action: Verify that the Scanner Feature ICP Unit is powered ON. Call your service representative if problem persists.

5214 D6x The scanner feature failed to disable.

Explanation: The scanner feature failed to respond to an initialization or disable request.

Action: Verify that the scanner feature of the image capture processor (ICP) unit is powered on. Notify your service representative if you continue to receive these error messages.

5215 D7x The scanner feature initialization failed.

Explanation: The scanner feature initialization data in IREC bytes 114-119 are invalid.

Action: Call your service representative.

5216 D8x The scanner feature initialization failed.

Explanation: The scanner feature initialization data in IREC bytes 114-119 is invalid or the scanner feature of the image capture processor (ICP) unit was powered off.

Action: Notify your service representative of the problem.

5217 D9x The scanner feature is not installed.

5223 DFx User exit time conversion error.

Explanation: The value specified for the SPXNDPTIME keyword is not valid and cannot be translated from ASCII to binary.

Action: Verify the keyword parameters listed in the Run Profile for the SPXNDPTIME keyword. Correct and reinitialize.

5224 E0x Invalid parameter for the 3890EM.

Explanation: The valid parameters for the 3890EMULation keyword are A, B, C, D, ORPQ, RPQ, and NONE.

Action: Review the Run Profile for validity of the 3890-emulation keyword (3890EMULation). Correct and reinitialize.

5226 E2x Multiple parameters for 3890EM keyword.

Explanation: The 3890EMULation keyword is not valid. The valid parameters for the 3890EMULation keyword are A, B, RPQ, and NONE.

Action: Review the Run Profile for validity of the 3890-emulation keyword (3890EMULation). Correct and reinitialize.

5227 E3x MICR-1 reader requested but is not configured.

Explanation: The MICR-1 reader was specified, but is not installed.

Action: Correct and reinitialize.

5229 E5x OCR-1 reader requested but is not configured.

Explanation: The OCR-1 reader was specified, but is not installed.

Action: Correct and reinitialize.

5230 E6x OCR-2 reader requested but is not configured.

Explanation: The OCR-2 reader was specified, but is not installed.

Action: Correct and reinitialize.

5232 E8x Front endorser requested but is not configured.

Explanation: The Front endorser was specified, but is not installed.

Action: Correct and reinitialize.

5233 E9x Roll-on endorser requested but is not configured.

Explanation: The Roll-on endorser was specified, but it is not installed.

Action: Correct and reinitialize.

5234 EAx The request did not complete successfully.

Explanation: A transport request failed because the transport is in an invalid state for this operation. For example, a feature disable is requested but the machine is not initialized.

Action: Correct and reinitialize.

5235 EBx Unable to read the machine status.

Explanation: The command to read the machine status returns an error.

Action: Notify your service representative of the problem.

5236 ECx Encoder is requested but is not configured.

Explanation: The Encoder was requested, but it is not installed.

Action: Correct and reinitialize.

5237 EDx Unable to read the machine configuration.

Explanation: The command to read the machine configuration returns an error.

Action: Notify your service representative of the problem.

5238 EEx Unable to set feed rate to 600 DPM.

Explanation: The encoder feature was selected with only one encoder present. The Control Program attempts to set the feed rate to 600 documents per minute automatically.

Action: Notify your service representative of the problem.

5240 F0x Errors closing a file (DOSCLOSE).

Explanation: A program has finished processing or reading a file, and the OS/2 function call was requested to close the file. The file could not be closed.

Action: Correct and reinitialize.

5241 F1x Microfilm not disabled though disable was requested.

Explanation: A microfilm disable request was sent; however, the microfilm is still active.

Action: Call your service representative.

5242 F2x INF not disabled though disable was requested.

Explanation: An item number feature disable request was sent; however, the feature did not respond.

Action: Call your service representative.

5243 F3x Endorse not disabled though disable was requested.

Explanation: An endorser disable request was sent; however, the feature did not respond.

Action: Call your service representative.

5244 F4x The feature request is not honored; transport is active.

Explanation: Feature requests are only honored when the transport is NOT in the active state.

Action: Retry the requested operation when not active.

5245 F5x OCR3 not disabled though disable was requested.

Explanation: An OCR3 disable request was sent; however, the feature did not respond.

Action: Call your service representative.

5246 F6x Run initialize error code conversion error.

Explanation: The run initialization error could not be recognized as a valid format. The actual error setting was sent to the host.

Action: Call your service representative.

6001 Feed or Merge Hopper Empty.

Explanation:

Action: Place more documents into the main or merge hopper.

6002 MISSORT condition detected.

Explanation: A document was possibly placed in a wrong pocket.

Action: Consult the Transport panel for information on the missort or view the list of documents in the Item Display which gives the proper pocket where the documents should have been placed.

6003 A Cover or Door is Open on the Transport.

Explanation:

Action: Close all covers and doors and press the Feed button again.

6004 OCR reader needs Attention.

Explanation: The Optical Character Recognition (OCR) reader has a problem.

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6005 OCR-2 Reader Lamp Failure.

Explanation: The Optical Character Recognition (OCR) reader has a lamp failure.

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6006 OCR-1 Reader Lamp Failure.

Explanation: The Optical Character Recognition (OCR) reader has a lamp failure.

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6007 OCR Power/Fan Failure.

Explanation: The Optical Character Recognition (OCR) reader has indicated the Power supply has failed or the Fan has stopped.

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6008 Microfilm Camera needs Attention.

Explanation:

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6009 Inkjet Printer needs Attention.

Explanation: The Inkjet printer is reporting a problem with the back or front ink jet .

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6010 Roll-On Endorser needs Attention.

Explanation:

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6011 There is a Document In the Auxiliary Pocket.

Explanation:

Action: Remove documents from the auxiliary pocket.

6012 Jam in the Feeder Area.

Explanation:

Action: Remove jam from location indicated on the Transport Panel.

6013 Jam in the Reader Area.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6014 Jam in the Option Area.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6015 Jam at Module/Stacker #1.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6016 Jam at Module/Stacker #2.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6017 Jam at Module/Stacker #3.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6018 Jam at Module/Stacker #4.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6019 Jam at Module/Stacker #5.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6020 Jam at Module/Stacker #6.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6021 Jam at the Microfilm Area.

Explanation:

Action: Remove jam from location indicated on the Transport Panel. See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

6022 Back Ink Jet Fault.

Explanation:

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6023 Front Ink Jet Fault.

Explanation:

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6024 Microfilm is Low, at End of Reel.

Explanation:

Action: The cassette must be turned over to record the second half of the reel or a new cassette must be loaded into the camera.

6025 Microfilm Camera NOT Ready.

Explanation: See the DISPLAY LAST screen to identify documents that were not Microfilmed. On the DISPLAY LAST screen items identified with an 'F' were not microfilmed.

Action: The cassette may not be installed properly or the microfilm camera may not be selected.

6026 Roll-On Endorser Fault.

Explanation:

Action: Follow directions on the Transport panel or see the documentation to determine the cause of the problem.

6027 Transport Command Format Error.

Explanation: Error detected communicating with the Transport Firmware.

Action: Notify your service representative of the problem.

6028 **Transport Transmission Error.**

Explanation: Error detected communicating with the Transport Firmware.

Action: Notify your service representative of the problem.

6029 **Transport Sorter State Error.**

Explanation: Error detected communicating with the Transport Firmware.

Action: Notify your service representative of the problem.

6030 **Transport Option State Error.**

Explanation: Error detected communicating with the Transport Firmware.

Action: Notify your service representative of the problem.

6031 **Roll-On Endorser Late Action.**

Explanation: The response to the Transport to stamp the endorsement on the document arrived too late. No endorsement occurred on the document.

Action: Reduce the length of processing in your Sort program.

6032 **_____ document(s) in the transport were not read.**

6033 **Program Requested Merge but Merge Feeder is not Present.**

Explanation: The Sort program requested a Merge Document to be fed, but the merge feeder is not configured on the Transport.

Action: Initialize a Sort that uses a DLL for Logical Merge or reconfigure the merge feeder to "ON" status. The current Sort cannot run.

6034 **OS/2 System Error, Code = &RetCode**

Explanation: A program error occurred in the OS/2 interface that was not expected.

Action: Notify your service representative of the problem.

6035 **RS422 Device Driver Error, Code = &RetCode**

Explanation: The RS422 card sends and receives information to the Transport. A program error occurred in the Device Driver interface for the card.

Action: Notify your service representative of the problem.

6036 **Transport Firmware Error, Code = &RetCode**

Explanation: A program error occurred in the Transport Firmware interface that was not expected.

Action: Notify your service representative of the problem.

6037 **Transport Alert Error, Code = &RetCode**

Explanation: A program error occurred in the Transport Firmware interface. An Alert was sent that was not expected.

Action: Notify your service representative of the problem.

6038 Transport Command Reject Error, Code = &RetCode

Explanation: A program error occurred in the Transport Firmware interface. A Command Reject was sent that was not expected.

Action: Notify your service representative of the problem.

6039 Transport Firmware Error, Code = &RetCode

Explanation: A program error occurred in the Transport Firmware interface that was not expected.

Action: Notify your service representative of the problem.

6040 Transport is Offline.

Explanation: The Control Program cannot communicate with the Transport.

Action:

6041 Transport Communication Error, Code = &RetCode

Explanation: The Control Program cannot communicate with the Transport.

Action: Notify your service representative of the problem.

6042 Transport Component Not Properly Installed.

Explanation: A required component is not properly installed.

Action: Notify your service representative of the problem.

6043 Adapter Communication Error, Code = &RetCode Return Code = &RetCode

Explanation: The Control Program cannot communicate with the RS422 Adapter. The return codes are as follows:

1 Read Error.
2 Write Error.
3 Readback Error.
4 Readback Invalid Data Error.

Action: Notify your service representative of the problem.

6044 Adapter Component Not Properly Installed.

Explanation: A required component is not properly installed.

Action: Notify your service representative of the problem.

6045 Jam at the Image Transport Area.

Explanation:

Action: Remove jam from location indicated on the Transport Panel.

6051 Transport is Offline.

Explanation: The Control Program cannot communicate with the Transport.

Action:

6052 Transport is Offline.

Explanation: The Control Program cannot communicate with the Transport.

Action:

6053 Cannot Feed, No Reader Selected. Initialization Required.

Explanation:

Action: Load or Initialize a Sort program.

6054 Cannot Feed, Not Initialized.

Explanation:

Action: Load or Initialize a Sort program.

6055 Transport Communication Error, Code = &RetCode

Explanation: The Control Program cannot communicate with the Transport. The return codes are as follows:

0202 Transmit Timeout, Transport not Available
0282 Transmit Timeout, Transport Available
0284 Receive Timeout, Transport Available
0286 Receive and Transmit Timeout, Transp. Available

Action: Check for power-on. Notify your service representative if the Transport cannot be placed in service.

6056 Transport Communication Recovered, Code = 0

Explanation: Communication re-established. COMM off, COMM on may be required.

Action:

6057 Transport is back Online, Initialization Required.

Explanation: Communication re-established.

Action:

6058 Feature Requested was not Initialized, &FEATNAME

Explanation: The Sort Program feature decision requested a feature that was not initialized.

Action: Correct and reinitialize.

6070 Jam at Encoder.

Explanation:

Action: Remove jam from location indicated on the Transport Panel.

6098 **AS=&EZASER12 MS=&EZSER12**

6099 **AS=&EZASER24 MS=&EZSER24**

6800 **Unable to resolve Encoder Verify Error(s)**

Explanation: The number of doc's or the doc ID's didn't match.

Action: Inspect the last 6 to 8 documents in the reject pocket for Encoding problems.

6801 **Encoder problem. Refer to Transport Display.**

Explanation: Encoder attention was active. Refer to the Transport Display for more information.

7010 **A record transfer to the Scanner Feature failed.**

Severity:

Explanation: This error indicates that the Control Program detected an error condition while attempting to transmit document data to the Scanner ICP.

Action: Warning! An 'Out of sync' condition may exist. Use the recovery procedures for your site to verify that the current job has been processed properly. Retry. Contact Service Representative

7058 **The Image System was found to be disabled.**

Severity:

Explanation: The Image System was found to be disabled when the XP Enhanced Control Program expected it to be initialized.

Action: An 'out of sync' condition may exist. Use recovery procedures for your site to verify the job processed properly. Re-initialize the Image System. Notify your service representative of the problem.

7059 **The Image System was found to be enabled.**

Severity:

Explanation: The Image System was found to be active after being disabled.

Action: Manually disable the Image System. Notify your service representative of the problem.

7060 **ICP Feed Stop won't go away**

Explanation: The ICP Feed Stop won't go away.

Action: Manually disable the Image System (DIS IS). Notify your service representative of the problem.

7062 **The Image Scanner Feature requested a disengage.**

Explanation: The Image Scanner Feature requested a disengage in order to transfer error data to disk.

Action: Start Machine. Notify Service Personnel

7063 The Image Scanner Feature requested a disengage.

Explanation: The Image Scanner Feature requested a disengage in order to transfer the Suspect Item List.

Action: Start Machine.

7064 The Image Scanner Feature requested a disengage.

Explanation: The Image Scanner Feature requested a disengage because the Scanner needs to be compensated.

Action: Compensate the Scanner. Start the Machine.

7065 The Image Scanner Feature requested a disengage.

Explanation: The Image Scanner Feature requested a disengage because the 'Fill Buffer Mode' completed.

Action: View document Image data at the ICP Re-initialize the SCI program

7066 The Image Scanner Feature requested a disengage.

Explanation: The Image Scanner Feature requested a disengage because an unrecoverable error occurred.

Action: Disable the Image Scanner Feature. Notify Service Personnel

7067 The Image Scanner Feature requested a disengage.

Explanation: The Image Scanner Feature requested a disengage because allowable number of parity was exceeded.
WARNING: Document Images for processed documents may require re-handling.

Action: Retry Notify Service Personnel

7068 The Image Scanner Feature requested a disengage.

Explanation: The Image Scanner Feature requested a disengage because the Image Quality Threshold was exceeded.
WARNING: Document Images for processed documents may require re-handling.

Action: Retry Notify Service Personnel

7070 An error occurred while retrieving 3897 exception data.

Severity:

Explanation: An error was detected while attempting to retrieve the Image Feature Exception data.

Action: Use the recovery procedures for your site to determine the action required. Contact your service representative.

7080 An error occurred while sending the Auto-select Modifier data.

Explanation: An error was detected while attempting to transfer the Auto-select Modifier data because a "0" return code was not received via the HSA

Action: Use the recovery procedures for your site to determine the action required. Contact your service representative

Appendix B. Test Modes

The 3890/XP Enhanced Control Program makes it possible for you to test your programs and subroutines by using the Block Physical Operation (BPO) mode and the Debug Mode.

Block Physical Operation (BPO) Mode

BPO is a test mode that lets you test the Sort program, SCI or OS/2 language, by using physical test documents. It is set by byte 21, bit 6 in the initialization data record (IREC).

Test documents that are compatible with the XP or XP Enhanced code line process in the same way that documents process for a normal run except that the documents are not actually sorted into different module-pockets. The BPO mode blocks sorting and routes all the documents to module-pocket 1/1, the reject pocket. You can use these test documents for further testing without sorting them back into their original sequence.

During the run, the INF, the endorsement feature, and the microfilming feature are held inactive, so the document processor does not print on or microfilm documents. If the image feature is installed, IREC byte 110, bit 4 (Image Capture System Online) should be **off**.

The feature control byte (byte 5) in the document data record header reflects any updates to the features of the document processor. Byte 6 of the document data record header contains the module-pocket code that would have been used for the pocket select decision if the BPO mode had not been active.

The following example describes the independence of byte 0 in the IREC from the active state of the BPO mode:

```
If three consecutive initializations are done:

    Run 1 Initialization

         Feature ON, BPO OFF

                     Feature operates on documents.
                     INF and microfilm numbers increment.

    Run 2 Initialization

         BPO ON, Feature NOT affected (byte 0, bit OFF)

                     Feature does NOT operate physically on
                     documents, but does increment number values
                     updated by Run 1.


    Run 3 Initialization

         BPO OFF, Feature NOT affected (byte 0, bit OFF) --

                     Feature operates on documents as in Run 1,
                     with number values updated by Run 2.
```

Debug Mode

Debug mode is a test mode that helps diagnose SCI errors.

In *debug mode,* you can test your program while processing either blank or inscribed documents. In *debug mode* and *simulated document mode* (3890/XP Enhanced only) you can test your program without processing any physical documents. Remember to take the 3890/XP Enhanced out of simulated document mode when you have finished testing. (The 3891/XP and 3892/XP document processors do not have simulated document mode.)

Procedure

Use the following *online* procedures for debug mode:

- You must generate document data in code line data form on tape or disk.
- A host program must write the code line data matches to the XP Enhanced and to read the resulting read records. Carefully check all the output of the Sort program including the following bytes from the document data record header:
  - Byte 2
  - Byte 3
  - Byte 4
  - Byte 5
  - Byte 6
  - Byte 8
  - Byte 9
  - Byte B.
- You must specify in the initialization data record (IREC) the initialization data for the following:
  - Debug mode (byte 21, bit 7)

  - Code Line Data Matching mode (byte 21, bit 2)
  - Prime the write buffers (byte 0, bit 6).

  If you are testing your program on the 3890/XP Enhanced document processor without running documents, select *simulated document mode* from the Run screen.

- For channel, the process ends when the host program writes code line data with header byte 0, bit 0 (code line data end-of-file) set to 1. The 3890/XP Enhanced document processor disengages and stops processing and then enters the *not-ready* condition.
- For APPC, the process ends when the host deallocates the "XP.Write.Data" conversation.

Appendix C. Storage

This appendix describes the following:

- Accessible storage Map for SCI load and store
- Program storage
- 3890-emulated auxiliary storage
- Additional auxiliary storage.

Accessible Storage Map for SCI Load and Store

Figure C-1 is a summary of accessible storage areas for the 3890/XP Enhanced document processor. For a more information, see “Program Storage” on page C-5 and “3890-Emulated Auxiliary Storage” on page C-7.

Figure C-1 (Page 1 of 3). Accessible Storage Map for SCI Load and Store

Storage Area	Hexadecimal Address	Description	Accessible by Load	Accessible by Store
Program Storage	0000-007F	Initialization data (see Note 4 on page C-5)	yes	yes
	0080-009F	SCI pointers	yes	yes
	00A0-	SCI program	yes	yes
3890-Emul. AUX storage	0000 (Note 1)	Indicates hardware type: Zero = 3890/XP Enhanced Not zero = 3890 model A-F	yes	no
	0007, 0008	Symbol error correction and high-order zero parameters (see Note 2 on page C-4)	yes	yes
	0010	User stats	yes	no
	0016	Hexadecimal value of highest program storage address, plus one, of the emulated model Model A or C: 4000 Model B or D: 8000 Model RPQ: C000 Other: 0000	yes	no
	001A	Features installed Bit 0 = 1 (Item number installed) Bit 1 = 1 (Endorser installed) Bit 3 = 1 (Microfilm installed)	yes	no
	001C	Machine type: 1 = 3890/XP Enhanced 2 = 3892/XP 3 = 3891/XP	yes	no
	0060-007F	SCI counters	yes	no
	008C	Model stats Bits 0-3 = Number of Stacker Modules Bit 4 = 1 (Low priority TSCI) Bit 5 = 1 (Debug mode) Bit 6 = 1 (MP is valid) Bit 7 = 1 (End SCI Exec)	yes	no

Figure C-1 (Page 2 of 3). Accessible Storage Map for SCI Load and Store

Storage Area	Hexadecimal Address	Description	Accessible by Load	Accessible by Store
	00A8-00A9	Address of the data byte to the extreme right of process buffer in program storage (3890-emulation only)	yes	no
	00AA	Process buffer data length	yes	no
	00AB	Length of process buffer data and header	yes	no
	00AC	Autoselect reasons	yes	no
	00BC, bit 0	Block Physical Operation	yes	no
	00BC, bit 2	End load completed	yes	no
	00BD, bit 2	Microfilm initialized and loaded	yes	no
	00BD, bit 3	INF initialized and on	yes	no
	00BD, bit 5	Online bit	yes	no
	1000	Image Capture System control Bit 0 = scan front black-and-white Bit 1 = scan front gray scale Bit 2 = scan back black-and-white Bit 3 = scan back gray scale	yes	yes
	20A0-20FB	Pocket counters	yes	yes
	3000-30FF	Save area	yes	yes
	(3)ED0-(3)EF3 (7) (7) (B) (B)	Process buffer (3890-emulation only)	yes	yes
	(3)EF4-(3)EFF (7) (7) (B) (B)	Header for process buffer (3890-emulation only)	yes	no
Additional AUX storage	8000-8FFF	New save area	yes	yes
	A000-A0F3	Process buffer data	yes	yes
	A0F4-A0FF	Process buffer header	yes	no
	A200-A247	Active endorsement data	yes	yes
	A300-A347	Pocket limit table	yes	yes
	A400-A405	Alternative INF indicator and data	yes	yes
	A500-A527	Additional sort message	yes	yes
	A5F0-A5F1	OS/2-language Sort return code	yes	yes
	A600, bit 5	Request-not-initialized	yes	yes
	A601	Verify residual	yes	no

Figure C-1 (Page 3 of 3). Accessible Storage Map for SCI Load and Store

Storage Area	Hexadecimal Address	Description	Accessible by Load	Accessible by Store
	A602	Jam indicator	yes	no
	A610-A61F	SPS event indicator	yes	no
	A680-A683	SBT hit address	yes	no
	A690-A697	Active endorse date	yes	no
	A6A0-A6D1	SPS-operator-entered data	yes	no
	A6E0-A6EB	Jam exception header	yes	no
	A700-A794	Code line definition table	yes	no
	A800-AB21	LCL table (includes count field)	yes	no
	B000-BFE1	PSD table (includes count field)	yes	no
	C000-C00B	Extended SCI error data	yes	no
	C010-C014	INF number (3890/XP Enhanced-maintained)	yes	no
	D000-D127	Input buffer area	yes	yes
	D200	Suppress date substitution in endorsement data	yes	yes
	D201	Suppress DW translation of endorsement text	yes	yes
	D300-D340	Command line input	yes	yes
	D400-D46B	Pocket name table	yes	no
	D500	Cancel sort processing	yes	yes
	D600-D62F	Code line data match pointers	yes	yes

Notes:

1. Any 3890 AUX storage address X'0000-70FF' can be loaded with the LAS instruction. However, only the 3890 addresses listed under 3890-emulated AUX storage contain meaningful emulation data.
2. Same as initialization data bytes 55 and 60 (X'0037' and X'003C').
3. Each pocket counter is in a 2-byte (four hexadecimal digits) 3890-emulated AUX storage location and consists of counter status and document count per pocket.

   The first digit (to the extreme left) is counter status. This digit must be preserved to ensure counter integrity, if the pocket counter data is changed by the load/store SCI operation. There is no internal checking; therefore, the retention of this digit is the responsibility of the programmer.

   ```
   Bit 1     Pocket counter filled
   Bit 2     Pocket counter active
   Bit 3     Kill pocket type
   ```

   The last three digits are the pocket count. This is a 12-bit binary value in

IBM format. (The high-order byte of an integer is stored in memory in the low address of the field.)

Pocket counter addressing is:

M/P	3890-emulated AUX Storage
1/1 - 1/6	20A0 - 20AB
2/1 - 2/6	20B0 - 20BB
3/1 - 3/6	20C0 - 20CB
4/1 - 4/6	20D0 - 20DB
5/1 - 5/6	20E0 - 20EB
6/1 - 6/6	20F0 - 20FB.

4. Merge feed control (byte 21, bit 1), merge feed pocket limits (bytes 56 through 59), and code line data match control are the only initialization data changes that are effective on succeeding documents with no reinitialization. Symbol error correction and high-order zero correction parameters can be effectively changed through 3890-emulated AUX storage locations X'0007' and X'0008'.

   The IFD SCI only affects the feature data.

Program Storage

The size of program storage is a hardware limitation, not an architectural limitation. The 3890/XP Enhanced document processor does not allocate buffers in program storage, so the complete program storage area is available to the user's sort program. If an SCI program tries to read data outside of program storage, an SCI error results.

When the Run Profile specifies 3890 emulation and the SCI program attempts to read process buffer data in program storage (using LPS or SPS), the 3890/XP Enhanced Control Program automatically remaps this to occur from the location of the process buffer in additional AUX storage. It does this by comparing the address specified in the LPS or SPS instruction to the address of the process buffer in the document processor. The 3890 model is specified in the 3890-emulation parameter of the Run Profile. This results in the following addresses being mapped to the process buffer:

Model	Address
3890 Model A, C	3ED0-3EFF
3890 Model B, D	7ED0-7EFF
3890 Model RPQ	BED0-BEFF.

Note: Although 3890 emulation implies a program storage size of 16K, 32K, or 48K, the 3890/XP Enhanced document processor does not check for Sort programs that exceed this limit or that attempt to read areas in program storage above this limit. Up to 64K is available to the SCI program in 3890 emulation. If the user-supplied SCI program has 2-byte addresses (byte 136, bit 0 = 0) and attempts to address a location above 64K, the addressing wraps back to low storage with unpredictable results.

Program Storage Assignments

Figure C-2. Program Storage

Decimal Address	Hexadecimal Address	Contents
0--127	0--79	Initialization data record (IREC)
128--	80--	SCI program header (when using an SCI program)
128-135	80-87	SCI program name
136 (bit 0)	88	Format of bytes 138-159 (0=3890 format, 1= XP or XP Enhanced format)
137	89	Reserved
138--159	8A--9F	3890 format - byte 136 (bit 0) = 0
138-143	8A-8F	SCI error information - byte 136 (bit 0) = 0
144-145	90-91	Address of first SCI - byte 136 (bit 0) = 0
146-147	92-93	Address of first SCI after error - byte 136 (bit 0) = 0
148-159	94-9F	Reserved - 136 (bit 0) = 0
138--159	8A--9F	3890/XP Enhanced format byte 136 (bit 0) = 1
138-143	8A-8F	Reserved - byte 136 (bit 0) = 1
144-147	90-93	Address of first SCI byte 136 (bit 0) = 1
148-151	94-97	Address of first SCI after error - 136 (bit 0) = 1
152-159	98-9F	Name of VSE IREC macro; document processor treats it as part of SCI program - 136 (bit 0) = 1
160--	A0--	SCI program

Note: Byte 136, bit 0, is used only to determine the format of bytes 138-159; this implies no other restrictions.

Auxiliary Storage

There are two areas of storage that the LAS (load-work-register from AUX storage) and the SAS (store-work-register-into-AUX storage) instructions can use. Neither of these areas is auxiliary storage as it existed in the 3890 document processor models, but the addresses are mapped to equivalent information by the 3890/XP Enhanced Control Program.

The first area, 3890-emulated AUX storage, is in the range of X'0000-70FF'. It is described in Figure C-1 on page C-2. Any 3890 document processor area that is not listed in this figure is not maintained by the 3890/XP Enhanced. If you attempt to load an address that is not listed, the 3890/XP Enhanced document processor does not give you an address-not-valid error, but the information it gives you will be meaningless.

The second area, Additional AUX storage, is unique to the 3890/XP Enhanced document processors and contains new information. The Additional AUX storage addresses are mapped to the actual information by the LAS and SAS instructions when they are processed.

Note: This is not contiguous storage space. If you attempt to load or store Additional AUX storage data across field boundaries (defined by the hexadecimal addresses) or if you attempt to load an address that is not listed, the 3890/XP Enhanced gives you an SCI address-not-valid error.

The Additional AUX storage addresses are described in Figure C-1 on page C-2.

3890-Emulated Auxiliary Storage

The XP Enhanced document processors support 3890-emulated AUX storage defined as locations X'n000-n0FF', where *n* is a value 0 through 7. However, the XP or XP Enhanced document processor maintains only the 3890 locations that are listed under "3890-Emulated Auxiliary Storage" in Figure C-1 on page C-2. It sets all other locations to 0.

The Sort program can test byte 0 of 3890-emulated AUX storage to determine whether the document processor is a 3890/XP Enhanced document processor or a 3890 Model A, B, C, D, E, or F.

Byte 0	**Document Processor**
Zero	XP Series or XP Enhanced
Not zero	3890 Models A through F

Additional AUX Storage

The following data area descriptions (sorted alphabetically by name) specify the format and the meaning of the Additional AUX storage data areas.

You can access these through SPXServ functions. You can also access some data areas as AUX storage locations through the SCI SAS, the SCI LAS, and diagnostic operations through the XP Enhanced Control Program host interface.

Active-endorse date (X'A690-A697')
Eight ASCII characters which specify the active date.

Active-endorsement data (X'A200-A247')
Three rows of 24 bytes containing ASCII characters that print in the endorsement area of a document. See "3890/XP Enhanced Character Sets" on page 7-8 for a complete list of valid characters.

The plus sign (+) character is a valid character but must be adjacent to a character (A-Z, 0-9, >, <). When this occurs, the letter is printed in double wide characters (for the 3890/XP or 3890/XP Enhanced document processor) or with underscores (for the 3891/XP and 3892/XP document processors).

Additional sort message (X'A500-A527')
Up to 40 ASCII characters that the user can display at the next transport stop.

Alternate INF Indicator and Data (X'A400-A405')
A 6-byte field where the user can request the system to print an Alternate INF number instead of the XP/XP Enhanced-maintained INF number.

```
Byte  0    X'00' = Use the system INF Number.
           X'01' = Use the alternate INF Number.
Bytes 1-5  Alternate INF Number, which has a value range of X'0-A'.
           X'A' prints as a blank.
           You must specify 10 digits, two per byte.
```

3890/XP or 3890/XP Enhanced data can be alphanumeric, 3891/92 XP data is numeric only.

For more information, see IREC byte 82, bit 2, under Figure 2-18 on page 2-16.

Cancel sort processing (X'D500')
You can change this 1-byte flag from X'00' to X'01' at any point in the document-processing cycle to prevent further processing for the present document. For example, a user's verify routine might decide to prevent any further processing on a document. The content of the process buffer reflects the processing up to the point of the cancel. Bit 1 of byte 7 in the header is set "on" to indicate that the XP Enhanced Control Program sent the document to module-pocket 1/1. Feature control is handled as if the document were an autoselect.

Code line data match pointers (X'D600-D62F')

Twelve 32-bit address pointers in Intel format are used during code line data matching. These are addresses to code line data match data in the host write buffer.

```
PTR 1     Address of the code line data match at the low-search range.
PTR 2     Address of the code line data match at the high-search range.
PTR 3     Address of the code line data match at the center of the
          search range.
PTR 4     Address of the first code line data match to be compared.
PTR 5     Address of the high-address code line data match in the
          Write Buffer.  When this address is reached,
          a pointer must wrap to the reset address.
PTR 6     The reset address for pointers 1 through 4.
PTR 7-12
          The values of pointers 1 through 6 are saved
          prior to pre-sort code line data matching if extended
          code line data matching is permitted.  These values
          are restored if the sort program calls the
          code line data match routine.
```

Code line definition table (X'A700-A794')

```
Byte  0   (X'A700')
          bit 0 - bit 5   SS1 - SS6 is not
                  data if the corresponding bit is on.
Bytes 1-2 (X'A701-A702')
          bit 0   Symbol error correction is
                  allowed if the bit is on.
          bit 1-bit 15
                  Field 1 - Field 15 is not a
                  candidate for symbol error
                  correction if the corresponding
                  bit is on.
Byte  3   (X'A703')
          8-bit binary integer that specifies the
          number of high-order digit errors that can be
          corrected to zeros in the amount field.
Byte  4   (X'A704')
          bit 0   Routing number transparency mode is active.
          bit 1   Code line data match zero-for-able is active.
          bit 2   Code line data match match don't-care-digit is active.
          bit 3   Sort program gets control of
                  auto-selected documents.
Bytes 5-148 (X'A705-794')
          Sixteen 9-byte field-definition records that
          correspond to Fields 0-15.  Each record contains:

          Byte 0  Field-opening symbol (right aligned)
                  (X'00', X'0A')
          Byte 1  Field-closing symbol
                  (right aligned)
          Byte 2  Field alternate closing symbol
                  (right aligned)
          Byte 3  Eight-bit integer value that gives
                  the offset of the low-order digit of
                  the field from the low-order end
                  (right end) of the process buffer
          Byte 4  Eight-bit integer value that gives
                  the maximum number of digits for
                  this field in the process buffer
          Byte 5  Eight-bit integer value that gives
                  the number of bytes for this field
                  in the process buffer
          Byte 6  Eight-bit integer value that gives
                  the code line data match threshold limit
          Byte 7  bit 0  Fixed-length field
                  bit 1  Dash transmission allowed
                  bit 2  Code line data matching allowed
          Byte 8  Reserved.
```

Command line input (X'D300-D340')

The sort program can prompt the operator by displaying a command on the input line of the Run screen. The operator need only press the Enter key. The input line is 65 bytes (ASCII).

INF Number (3890/XP Enhanced-maintained) (X'C010-C014')
The last INF number produced by the XP or XP Enhanced document processor. This number is the first INF number, which the feature control byte increases, as required.

If byte 81, bit 0, of the initialization data is set to off, the INF number is only 8 digits long. In this case, this 10-digit field is left aligned and the 8 digits to the extreme left (high-order) are the INF number; the two digits to the extreme right (low-order) are blanks (X'AA'). For 3890/XP Enhanced mode, all 10 digits are the INF number, unless otherwise specified. MICR or OCR code line characters.

Input buffer area (Recognition Subsystem data, header information, and pointers) (X'D000-D127')
Contains document data as it was received from the transport and corrected by the code line correction processing; information determined during the verify and the correction processing precedes the document data. Each document character is in a hexadecimal format and occupies the nibble to the extreme right of the byte, with the nibble to the extreme left set to zero.

```
Bytes  0-1 Reserved
Bytes  2-3 Sixteen-bit integer in Intel format that gives
           the length of document data.
Bytes  4-19
           Eight 16-bit integer values in Intel
           format that give displacements from the
           low address of the input buffer data
           area (Byte 34) to field-defining
           symbols S1-S8.
Bytes 20-31
           Document header bytes 0-11
           (not necessarily completed as in the
           process buffer).
Byte  32   Autoselect reasons:
           bit 0  Multiple documents detected.
           bit 1  Short gap detected.
           bit 2  Long document detected.
           bit 3  Short document detected.
           bit 4  Short leading edge (LE) to LE space.

Byte  33   Unused
Bytes 34-293
           Document data as received from the
           transport is right-aligned in this area.
           Field-defining symbols have
           been marked by turning on bit 2.
Bytes 294-295
           Used to force a dummy S1 if the opening
           symbol is not sent from the transport.
           Set to X'2020'.
```

Jam exception header (X'A6E0-A6EB')
The last exception header created in the 3890/XP Enhanced document processor.

Jam indicator (X'A602')
Indicates that the system processed the jam after the last document processing cycle.

LCL Table (X'A800-AB21')

You can use the level control label (LCL) table to indicate the version of any file loaded during initialization.

```
Byte  0-1 Sixteen-bit binary integer in Intel format that
          gives the number of entries in the LCL table.
Bytes 2-801
          Ten LCL table entries in ASCII format:

          Bytes  0-15  Sixteen-character LCL
          Bytes 16-23  Eight-character file name
          Bytes 24-27  Four-character file extension (.xxx)
          Bytes 28-80  Fifty-two character file path.
```

OS/2 Language Sort return code (X'A5F0-A5F1')

Two user-defined hexadecimal characters that are set by an OS/2 language Sort routine.

New Save Area (X'8000-8FFF')

Usage defined by the user.

Pocket limit table (X'A300-A347')

36 pocket limit values ordered as MP 1/1 through MP 6/6. These values are 16-bit binary integers in Intel format.[1]

```
Pocket limit = 1234  (X'04D2')
Memory value = X'D204'
```

Pocket name table (X'D400-D46B')

A table of names to be assigned to each pocket on the machine. These 3-character ASCII names are ordered so that the first name is for module-pocket 1/1 and the last name is for module-pocket 6/6.

Process Buffer Data (X'A000-A0F3')

Formatted document data that is right-aligned in the area (for example, the field-1, low-order byte is in location X'A0F3').

Process Buffer Header (X'A0F4-A0FF')

Header bytes 0 through 11.

---

[1] The low-order byte of an integer value is stored in memory in the low address of the field.

PSD Table (X'B000-BFE1')

The Run Profile element table contains the parameters for the program storage data (PSD) files that are loaded into program storage during run initialization. It consists of the following:

```
Byte  0-1  Sixteen-bit binary integer in Intel format that
           gives the number of entries in the PSD table.
Bytes 2-4097
           128 PSD table entries that have the following format:

           Bytes  0-7   Eight-character ASCII filename.
           Bytes  8-15  Eight-character user tag.
           Bytes 16-19  IBM format offset into
                        program storage where data
                        begins.
           Bytes 20-23  IBM format data length.
           Bytes 24-27  Intel format offset into
                        program storage where data begins.
           Bytes 28-31  Intel format data length.
```

Request-not-initialized — RNI (X'A600', bit 5)

When bit 5 of this byte is set to on, a request is made to set the RNI flag to indicate the state of the document processor. If the document processor is online, bit 6 of Sense Byte 0 is set to Not Initialized.

SBT hit address (X'A680-A683')

The 4-byte address in the IBM format of the located table entry.

Suppress date substitution in endorsement date (X'D200')

If this 1-byte field is set to X'00', the system substitutes the active date for any %DDDDDDD string in the endorsement text. Substitution stops with the first character that is not a D or when it fills eight positions.

When this field is set to X'01', no date substitution occurs. This lets the sort program perform its own date substitution. If the sort program does not change the endorsement data, this flag can be set after the first document to indicate that date substitution has already occurred.

Suppress double-wide (DW) translation of endorsement data (X'D201')

When this 1-byte field is set to X'00', any valid double-wide character specifications are translated to the transport codes that cause the large characters to be printed (for the 3890/XP Enhanced) or underscores (for the 3891/XP and 3892/XP).

When this field is set to X'01', no translation occurs. Use this value to perform these translations in advance, thus saving processing time for the sort program. Also, you can save processing time when there are no DW specifications in the programmable-endorsement data. If the sort program does not change the endorsement data, this flag can be set after the first document is processed to indicate that DW translation has already occurred.

SPX operator-entered data (X'A6A0-A6D1')

One to 50 ASCII characters entered by the operator for an operator-initiated processing (SPSOPER event). These characters begin with the low address of the field and are user-defined.

SPS event indicator (X'A610-A61F')

Byte flags that indicate which SPS event is active. A value of X'00' for a specific SPS event byte indicates that the SPS event is not active; a value of X'01' indicates that it is active.

```
Byte  0    Verify input data
Byte  1    Correct input data
Byte  2    Format process buffer
Byte  3    Code line data match
Byte  4    Document sort processing
Byte  5    POST-sort processing
Byte 12    Pause Event
Byte 13    Run-initialization processing
Byte 14    Motors stopping processing
Byte 15    Operator-initiated processing
```

Verify residual (X'A601')

A 1-byte residual from the Verify SCI.

For more information about the Verify SCI, see the *3890/XP Series Stacker Control Instructions Reference*.

Glossary

This glossary defines terms that are used in this manual and in other books in the 3890/XP Series and 3890/XP Enhanced library. This glossary also includes tems and definitions from the *IBM Dictionary of Computing*, SC20-1699. If you do not find the term that you want, see the index.

A

alternate INF data. A 5-byte field in AUX storage that is printed on the back of an item or document instead of the INF number when the alternate INF indicator in AUX storage is set to X'01' (on). You must specify 10 digits, two for each byte. Each half-byte has a value range of X'0-A'. X'A' prints as a blank.

autoselect. The document is sent to module-pocket 1/1 because of hardware-detected errors.

C

CCW. See *channel command word*.

channel command word (CCW). A System/370 word interpreted by the channel as an instruction in the channel program. One or more CCWs make up the channel program. The channel program directs channel operations.

code line data matching. The process of matching data in a file to the code line data on a document.

D

default. An alternative value, attribute, or option that is assumed when none has been specified.

disk subsystem. System facilities that are identified by OS/2 drive and path conventions.

display panel. A predefined display image that can contain information such as instructions, prompts, options, and data.

display screen. An illuminated display surface on which display images are presented.

DLL. See *dynamic-link library*.

dynamic-link library (DLL). A specially-linked collection of Sort modules, loaded by OS/2 facilities, that can be called during the running of Sort programs. The OS/2 loader automatically loads additional DLL files that are needed to resolve labels in the DLL file that is being loaded. The file named in the Run Profile (USERDLL keyword) is of this type.

E

ECP. See *endorse control product*.

EIM. See *extended image match*. Also see *extended code line data match*.

endorse control product (ECP). This product aids the Sort programmer in controlling the endorse feature for each document.

extended code line data match. An operation that permits code line data matching to be done under the control of the SCI program.

extended image match (EIM). See *extended code line data match*.

I

IBM format. The high-order byte of an integer value is stored in the low address of the field; therefore, the integer value in memory appears the same as its hexadecimal format. See *Intel format*.

```
Decimal format:      1234
Hexadecimal format:  04D2
IBM format:          04D2
```

image matching. A term used in the past to describe the process of matching data in a file to the code line data on a document. See *code line data matching*.

Image Scanner feature. The Image System has two major components, the IBM 3897 Image Capture System, which acquires images, and the IBM 3898 Image Processor, which processes images and performs character recognition. The 3897 consists of a Image Scanner module or feature and an Image Capture Processor. The camera, for example, is housed in the Image Scanner module. The Image Capture Processor performs compression on the image data.

IML. See *initial microcode load*.

INF number. An 8- or 10-digit number that is printed on the back of each item or document that the 3890/XP or 3890/XP Enhanced processes if the alternate INF indicator in AUX storage is set to X'00' (off). The Sort program can add one to either the low-order segment or the high-order segment of this number after each document is processed.

initial microcode load (IML). This is the event that loads the 3890/XP or 3890/XP Enhanced Control Program. This

event must occur after every power transition of the 3890/XP or 3890/XP Enhanced.

initialization data record (IREC). The first 128 bytes of program storage. The initialization data record defines the functions of the 3890/XP or 3890/XP Enhanced. It is extended by the Run Profile, which can be named in the initialization data record.

initialization data record (IREC) macro. A host-system macro that can be used to format the initialization data record.

Intel format. The low-order byte of an integer value is stored in the low address of the field. Therefore, the integer value in memory appears to be the reverse of its hexadecimal format. See *IBM format*.

```
Decimal format:      1234
Hexadecimal format:  04D2
Intel format:        D204
```

IREC macro. See *initialization data record (IREC) macro*.

item number. See *INF number*.

K

K byte. See *kilobyte*.

kilobyte (K byte). 1 K byte = 1024 bytes.

L

LAPI. See *LU 6.2 application-program interface*.

LCL. See *level-control label*.

level-control label (LCL). A 16-character string carried in every Run Profile element that can be used to check the change level of that element.

LU 6.2 application-program interface (LAPI). Macros and modules that supply support for the host application to communicate with other application programs using LU 6.2 protocol.

M

M byte. See *megabyte*.

Machine Profile. This file contains parameters defined by the user describing the specific XP or XP Enhanced machine, such as the System/370 channel address. It is loaded with the XP or XP Enhanced Control Program from the disk subsystem. You can use the customization display panels in the XP or XP Enhanced user interface to maintain this file.

megabyte (M byte). 1 M byte = 1 048 576 bytes.

magnetic ink character recognition (MICR). The ability to recognize the characters that run along the 5/8" band at the bottom of a document.

message line. The area of the display screen where messages appear.

MICR. See *Magnetic Ink Character Recognition*.

N

native language. Any language supported by the OS/2 operating system.

O

OCR. Set *Optical Character Recognition*.

Operating System/2 (OS/2). The operating system for the IBM PS/2 computers.

Optical Character Recognition (OCR). The ability to use optical means to identify graphic characters.

OS/2. See *Operating System/2*.

P

panel. See *display panel*.

Program Storage Data (PSD) file. A file on the disk subsystem that is identified in the Run Profile that is loaded without change into program storage during initialization. You use a PSD to load a table into the sorter; a Run Profile might include a PSD entry.

PSD. See *Program Storage Data file.*

PSD table. Run Profile element table. A table that the 3890/XP or 3890/XP Enhanced Control Program builds for the SCI program and OS/2 language Sort programs. The table points to locations in storage of data that is loaded from PSD files during initialization. Such files, identified in the Run Profile, are Run Profile elements.

PSD tag. A tag that is defined by the Run Profile PGMSTOREdata keyword and is available to the SCI and OS/2 language Sort programs through the PSD table. Sort programs use this tag to identify and find PSD files by type.

Q

quick-stop. The hardware-initiated event that stops the transport immediately when a jam or other problem is detected. The operator clears the affected module, then uses a runout to permit trapped documents that are still in the transport to be processed. See *runout*.

R

Run Profile. An ASCII keyword file that controls operation during initialization. This file is available to the XP or XP Enhanced Control Program through the disk subsystem. If no Run Profile is named in the initialization data, the default Run Profile is used. The Machine Profile specifies the default Run Profile and the drive and path for all Run Profiles.

Run Profile Element Table. See *PSD table*.

runout. The hardware-controlled operation that the operator uses during recovery from a quick stop. It permits processing of items that were trapped by the quick stop but not removed by the operator. During runout, the SCI program runs and creates read records, even though all documents that are read during runout appear blank. A jam can also occur during runout.

S

scanner feature. See *Image Scanner feature*.

SCI. See *Stacker Control Instruction*.

SIO. See *start I/O*.

Sort module. See *Sort-program module*.

Sort program. All the user-supplied code for SPS events. In the Run Profile, the Sort program for each SPS event is specified by an ordered series of SPS keywords. Once it is called, a sort-program module can call other sort-program modules. One special sort-program module is the IBM-supplied SCID, which interprets the SCI program that is stored in the program-store area. Like any sort-program module, SCID can be specified in the Run Profile for any of the SPS events, most usually for SPSDOC.

Sort-program module. A routine which is part of a Sort program. Sort modules are entry points within DLL files. You can specify Sort modules to be run as part of Sort programs in the following ways:

- Name the Sort modules in an SPSxxx keyword in the Run Profile.
- Name the Sort module in the Machine Profile.
- Call the Sort module directly from another Sort module.

Sort Program Sequence (SPS). Sort Program Sequence keyword in the Run Profile. This keyword specifies the components of the Sort program for a specific SPS event.

Special Symbol Sequence Table (SST). This table defines the code line for documents.

SPS. See *Sort Program Sequence*.

SPS event. An event that causes a Sort program module to run. SPS events include the following:

- Non-document SPS events:
  - Initialize
  - Motors stopped
  - Operator.
- Document SPS events:
  - Verify
  - Correction
  - Format
  - Image match
  - Document
  - Post sort.

SPXSERV.DLL. See *SPX services dynamic-link library*.

SPX services dynamic-link library (SPXSERV.DLL). This library comes with the 3890/XP Series or 3890/XP Enhanced Control Program. These subroutines are the lowest-level interface for the OS/2 language programmer to 3890/XP or 3890/XP Enhanced services.

SST. See *Special Symbol Sequence Table*.

Stacker Control Instruction (SCI). SCI is a language that is interpreted by the 3890/XP or 3890/XP Enhanced as the Sort program. 3890/XP or 3890/XP Enhanced offers major extensions, including 4-byte addressing and native-language subroutines.

start I/O (SIO). The System/370 instruction that starts all I/O operations by requesting a channel program.

T

Terminate Stacker Control Instructions (TSCI). An interrupting condition that is forced by the system when the Sort program has taken too much time. The TSCI causes the late module-pocket code to occur.

TSCI. See *Terminate Stacker Control Instructions*.

Index

Numerics

A

B

C

D

E

F

H

I

J

K

L

M

N

O

P

R

S

T

U

V

Z

Communicating Your Comments to IBM

3890/XP Enhanced
Document Processor
Programming Guide

Publication No. SC31-4069-00

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 readers' 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:

  United States & Canada: 1-800-955-5259
- If you prefer to send comments electronically, use this network ID:

  IBM Mail Exchange: USIB1362 at IBMMAIL

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.

Readers' Comments — We'd Like to Hear from You

3890/XP Enhanced
Document Processor
Programming Guide

Publication No. SC31-4069-00

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

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.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC31-4069-00

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

International Business Machines Corporation
Payment Solutions
Department 58G MG34/204
8501 IBM Drive
Charlotte NC 28262-8563

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

SC31-4069-00

Cut or Fold
Along Line

File Number
S/370-4300-40

SC31-4069-00

Printed in U.S.A.