SPERRY UNIVAC

PUBLICATIONS UPDATE

Operating System/3 (OS/3)

Information Management System (IMS) Action Programming in RPG II

User Guide

UP-9206–A

This Library Memo announces the release and availability of Updating Package A to "SPERRY UNIVAC Operating System/3 (OS/3) Information Management System (IMS) Action Programming in RPG II User Guide", UP-9206.

This update for release 8.0 documents guidelines for defining the program information block and buffer size for local workstations.

Copies of Updating Package A are now available for requisitioning. Either the updating package only or the complete manual with the updating package may be requisitioned by your local Sperry Univac representative. To receive only the updating package, order UP-9206–A. To receive the complete manual, order UP-9206.

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ and MZ	Mailing Lists A00, A07, A08, B00, B07, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 28U, 29U, 75, 75U, 76 and 76U (Package A to UP-9206, 7 pages plus Memo)	Library Memo for UP-9206–A
		RELEASE DATE: January, 1983

SPERRY UNIVAC
COMPUTER SYSTEMS

PUBLICATIONS RELEASE

Operating System/3 (OS/3)

Information Management System (IMS) Action Programming IN RPG II

User Guide

UP-9206

This Library Memo announces the release and availability of "SPERRY UNIVAC® Operating System/3 (OS/3) Information Management System (IMS) Action Programming in RPG II User Guide", UP-9206.

The Information Management System (IMS) Action Programming in RPG II User Guide is one of five books replacing the IMS 90 Applications User Guide/Programmer Reference, UP-8614, Rev. 1. Other manuals replacing UP-8614 are:

- IMS Concepts and Facilities, UP-9205
- IMS Action Programming in COBOL and Basic Assembly Language (BAL) User Guide, UP-9207
- IMS Terminal Users Guide, UP-9208
- IMS Data Definition and UNIQUE User Guide, UP-9209

This manual describes and illustrates how to write RPG II action programs. It is presented in nine sections and four appendixes as follows:

Section 1.	Setting the Stage
Section 2.	General Rules for Coding Action Programs
Section 3.	Writing an Action Program
Section 4.	Writing a More Complex Action Program
Section 5.	Special Types of Output Messages
Section 6.	Using Screen Format Services for Format Messages
Section 7.	Action Programming in a Distributed Data Processing Environment
Section 8.	Compiling, Linking, and Storing Action Programs
Section 9.	Debugging An Action Program

LIBRARY MEMO ONLY	LIBRARY MEMO AND ATTACHMENTS	THIS SHEET IS
Mailing Lists BZ, CZ and MZ	Mailing Lists A00, A07, A08, B00, B07, 18, 18U, 19, 19U, 20, 20U, 21, 21U, 28U, 29U, 75, 75U, 76, and 76U (Cover and 287 pages)	Library Memo for UP-9206 RELEASE DATE: September, 1982

Appendix A. Using Device Independent Control Expressions and Field Control Characters

Appendix B. Generating Edit Tables

Appendix C. Summary of IMS Error Codes

Appendix D. Action Program Coding Restrictions

The complete titles and ordering numbers of the books that form the IMS library are:

- Information Management System (IMS) System Support Functions User Guide, UP-8364, Rev. 7
- Information Management System (IMS) Concepts and Facilities, UP-9205
- Information Management System (IMS) Action Programming in RPG II User Guide, UP-9206
- Information Management System (IMS) Action Programming in COBOL and Basic Assembly Language (BAL) User Guide, UP-9207
- Information Management System (IMS) Terminal Users Guide, UP-9208
- Information Management System (IMS) Data Definition and UNIQUE User Guide, UP-9209
- IMS/DMS Interface User Guide, UP-8748, Rev. 1

Additional Copies may be ordered by your local Sperry Univac representative.

Information Management System (IMS)

Action Programming in RPG II

OS/3

User Guide

H

SPERRY UNIVAC

UP-9206

This document was prepared by Systems Publications using the SPERRY UNIVAC UTS 400 Text Editor. It was printed and distributed by the Customer Information Distribution Center (CIDC), 555 Henderson Rd., King of Prussia, Pa., 19406.

PRINTED IN U.S.A.

PAGE STATUS SUMMARY

ISSUE: Update A – UP-9206
RELEASE LEVEL: 8.0 Forward

Part/Section	Page Number	Update Level
Cover/Disclaimer		Orig.
PSS	1	A
Acknowledgment	1	Orig.
Preface	1 thru 3	Orig.
Contents	1 thru 10	Orig.
1	1 thru 10	Orig.
2	1 thru 23 24 25 thru 43	Orig. A Orig.
3	1 thru 23	Orig.
4	1 thru 21	Orig.
5	1 thru 55	Orig.
6	1 2 thru 13	A Orig.
7	1 thru 10	Orig.
8	1 thru 9	Orig.
9	1 thru 28	Orig.
Appendix A	1 thru 15	Orig.
Appendix B	1 thru 19	Orig.
Appendix C	1 thru 7	Orig.
Appendix D	1 thru 4	Orig.
Index	1 thru 13	Orig.
User Comment Sheet		

All the technical changes are denoted by an arrow (→) in the margin. A downward pointing arrow (↓) next to a line indicates that technical changes begin at this line and continue until an upward pointing arrow (↑) is found. A horizontal arrow (→) pointing to a line indicates a technical change in only that line. A horizontal arrow located between two consecutive lines indicates technical changes in both lines or deletions.

Acknowledgment

We are indebted to the many systems analysts and staff members of Sperry Univac branch offices and customer organizations who helped us develop the OS/3 IMS library. They gave us suggestions, answered numerous questions, reviewed the manuals, and provided us with "real-life" programming examples. The customer organizations assisting us include:

- Gay and Taylor Insurance Adjustors, Winston-Salem, NC
- Penn Ventilator Company, Philadelphia, PA
- Victor Valley Community College District, Victorville, CA

The Sperry Univac organizations assisting us include:

- Los Angeles Access Center, Customer Support Services, Los Angeles, CA
- Charlotte Commercial Branch, Raleigh Office, Raleigh, NC
- Charlotte Commercial Branch, Greensboro Office, Greensboro, NC
- Minneapolis Marketing Branch, Minneapolis, MN
- Wellesley General Branch, Wellesley, MA
- Philadelphia Manufacturing Branch, Wayne, PA
- Des Moines Marketing Branch, West Des Moines, IA
- System 80 Benchmark and Demonstration Services, Blue Bell, PA

Preface

This manual is one of a series designed to instruct and guide you in using the SPERRY UNIVAC Information Management System (IMS) for Operating System/3 (OS/3). It describes and illustrates how to write RPG II action programs.

This manual is divided into seven sections and four appendixes. The topics discussed are:

- Section 1. Setting the Stage

  Introduces and defines IMS terminology related to action programming and discusses how IMS and action programs interface.

- Section 2. General Rules for Coding Action Programs

  Discusses special coding considerations in writing action programs with particular emphasis on the RPG II/IMS interface areas.

- Section 3. Writing an Action Program

  Presents simple programming examples illustrating the fundamental properties of action programming – processing input messages and generating output messages.

- Section 4. Writing a More Complex Action Program

  Presents more complex programming examples illustrating the use of internal subroutines and screen format services in action programs.

- Section 5. Special Types of Output Messages

  Describes and provides programming examples of the many types of output action programs can generate – namely, multiple output, continuous output, output-for-input queueing, and output with message switching.

- Section 6. Using Screen Format Services to Format Messages

  Describes how action programs use screen format services to format output messages and receive formatted input.

- Section 7. Action Programming in a Distributed Data Processing Environment

  Describes the IMS transaction facility for handling distributed data processing with IMS.

- Section 8. Compiling, Linking, and Storing Action Programs

  Explains how to compile, link, and store action programs for use during online IMS sessions.

- Section 9. Debugging an Action Program

  Describes how to interpret data provided in a snap dump for debugging purposes.

- Appendix A. Using Device Independent Control Expressions and Field Control Characters

  Describes the use of device independent control expressions and field control characters for formatting messages.

- Appendix B. Generating Edit Tables

  Explains the use of the edit table generator for converting unformatted input into fixed formats.

- Appendix C. Summary of IMS Error Codes

  Presents all error codes returned by IMS as a result of function requests made by action programs.

- Appendix D. Action Program Coding Restrictions

  Presents IMS restrictions for RPG II coding forms.

As one of a series, this manual is designed to guide you in programming and using the OS/3 information management system. Depending on your need, you should also refer to the current versions of other manuals in the series. Complete manual names, their order numbers, and a general description of their contents and use are as follows:

- Information management system (IMS) concepts and facilities, UP-9205

  Describes the basic concepts of IMS and the facilities that IMS offers.

- Information management system (IMS) system support functions user guide, UP-8364

  Describes the procedures to generate, initiate, and recover an online IMS system.

- Information management system (IMS) action programming in COBOL and basic assembly language (BAL) user guide, UP-9207

  Describes how to write action programs in COBOL and BAL, with extensive examples.

- Information management system (IMS) terminal users guide, UP-9208

  Describes terminal operating procedures, standard and master terminal commands, and special purpose IMS transaction codes. Also includes UNIQUE command formats with brief descriptions. The manual is in easel format for ease of use at the terminal.

- Information management system (IMS) data definition and UNIQUE user guide, UP-9209

  Describes how to create defined files for use with UNIQUE or your action programs and explains how to use UNIQUE. Includes extensive examples of data definitions and UNIQUE dialogs.

- IMS/DMS interface user guide, UP-8748

  Describes how to access a data base management system (DMS) data base from IMS.

Contents

FIGURES

TABLES

1. Setting the Stage

1.1. INTRODUCING IMS

The SPERRY UNIVAC Information Management System (IMS) is an interactive, transaction-oriented file processing system. It is interactive because it carries on a conversation with the terminal operator; it is transaction-oriented because for each input message, the terminal operator receives a response or output message. In this way, operators are constantly informed of the results of their inquiries.

1.2. INTERACTING WITH IMS

Action programs process messages

Application programs, called action programs, interact with IMS to process input messages from terminals, perform file retrieval or updating functions, and create output messages.

Languages used – BAL, COBOL, RPG II

You can write action programs in RPG II, COBOL, or basic assembly language (BAL). IMS also provides a set of action programs called the uniform inquiry update element (UNIQUE) that performs file retrieval and updating functions through the use of commands from the terimnal.

Purpose of this manual

This manual tells you how to write action programs in RPG II. Action programs are similar to standard RPG II programs, but must follow specific rules because they operate under the control of IMS.

Read IMS concepts and facilities first

Before you start writing action programs, you must understand how IMS works, and what you (or the IMS administrator) must do to make it work. This information is in the IMS concepts and facilities manual, UP-9205 (current version). We also assume that you know RPG II. For more information about RPG II coding, consult the RPG II user guide, UP-8067 (current version).

Prerequisites for using this manual

Throughout this manual, we assume you've read and understood both UP-9205 and UP-8067. However, as required, we briefly define terms and describe concepts that are directly related to RPG II action programming.

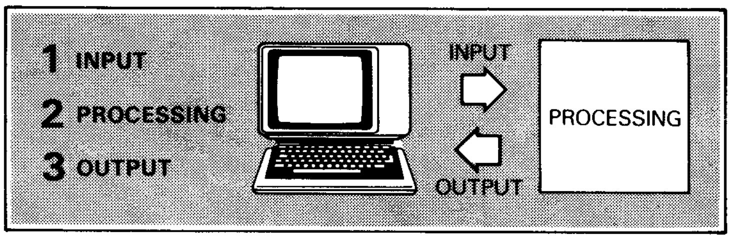
1.3. LET'S DEFINE SOME BASIC IMS TERMS

Action defined

What action programs do

The term **action programming** comes from the fact that the unit of work in IMS is the **action**. An action begins when an operator enters a message at a terminal and ends when a response to that message is returned. This is an important point to remember since the action programs you write are involved primarily with this activity – processing input messages, performing file retrieval or updating, and creating output messages.

An action always consists of three activities:



Transaction defined

A **transaction** is one action or a series of actions.

A **simple** transaction (Figure 1-1) consists of a single action.

Example – Simple transaction

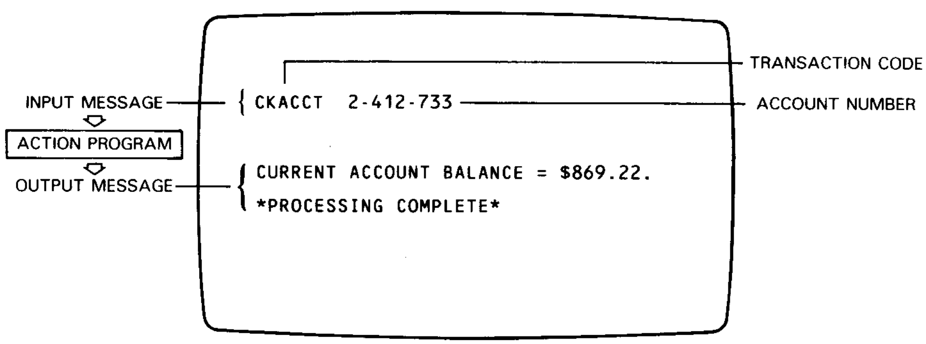


Figure 1-1. **A Simple Transaction.** *In this example, one action program processes the input messsage and produces an output message – the checking account balance for the account specified and a* **processing complete** *notice.*

A **dialog** transaction (Figure 1-2) consists of two or more related actions.

Example – Dialog transaction

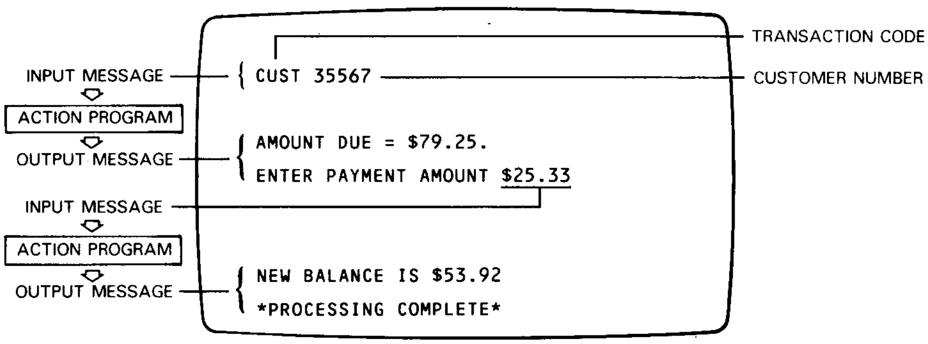


Figure 1-2. A Dialog Transaction. *In this example, two action programs are sequenced to produce amount due information, allow data entry, and compute a new balance for a specific customer account.*

Transaction codes initiate transactions

To begin a transaction, the operator enters a 1- to 8-character transaction code. (In single-thread IMS, the transaction code is 1 to 5 characters.) This code tells IMS the name of the action program that will process the input message.

Transaction code defined

Transaction codes are either the entire input message or a part of it. Transaction codes are defined to IMS at configuration time.

1.4. HOW YOU STRUCTURE TRANSACTIONS

Series of action programs processes transaction

Sometimes a single action program can process the function required. But more often than not, a series of action programs is needed. In either case, we create what we call a transaction structure.

Types of transaction termination

Transaction structure depends on how you terminate action programs. There are four major types of termination:

TYPES OF TERMINATION

- Normal
- External succession
- Delayed internal succession
- Immediate internal succession

From here on, we'll call the termination types normal termination, external, delayed, and immediate succession.

Distinction between termination and succession

Using the words **termination** and **succession** in the same context can be somewhat confusing. In IMS, termination means that an action program is finished processing. Whether you specify normal termination, external, delayed, or immediate succession, you are telling IMS that the current action program is finished processing and is now terminating.

Succession means that although the action program is terminating, the transaction is not complete. A successor action program will continue processing the transaction.

Transaction complete

Normal termination says that the transaction itself is complete. No more processing occurs.

However, external, delayed, or immediate succession tells IMS that another action program follows and will resume processing.

Figures 1-3 through 1-6 illustrate these concepts.

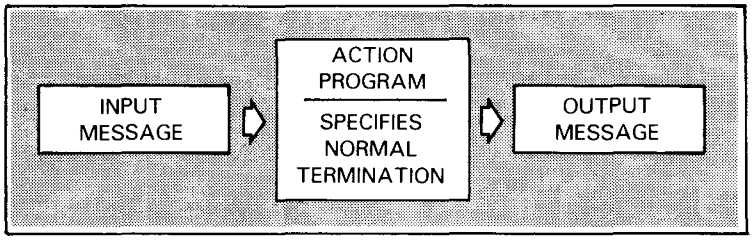


Figure 1-3. Normal Termination

Normal termination

Use normal termination to tell IMS that once your program creates an output message, the transaction is complete. When you don't specify the type of termination, IMS terminates normally. The last action program in a transaction always ends with normal termination.

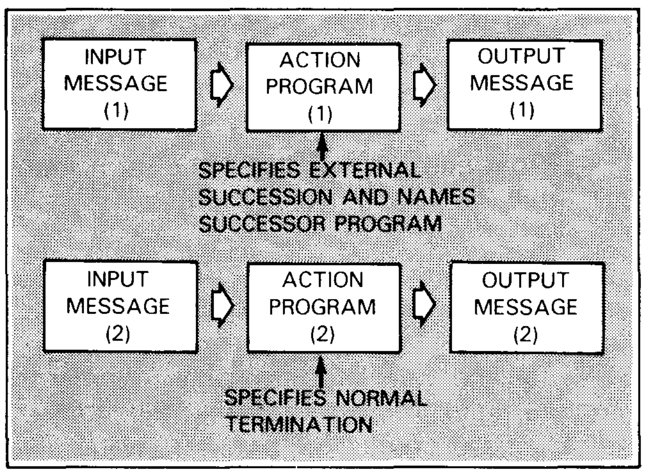


Figure 1-4. External Succession

External succession

Use external succession to tell IMS that the current action program is sending an output message and terminating; however, the transaction is not complete. When the terminal operator enters a second input message, the action program you named as external successor processes the second action, produces an output message, and terminates.

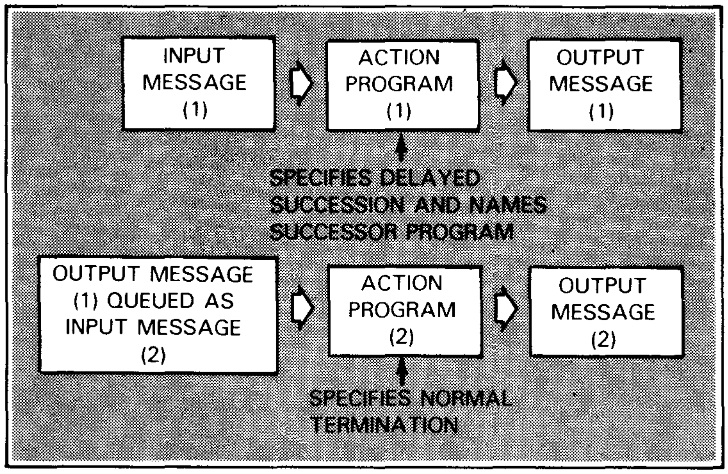


Figure 1-5. Delayed Internal Succession

Delayed succession

Use delayed succession to tell IMS that the current action program has processed an input message and produced an output message; however, that message isn't going to the terminal. Instead, it becomes the input message to the action program you named as successor. The successor program produces an output message that does go to the terminal and terminates. With delayed succession, the second action program uses the output message of the predecessor as its input message. Even though only one input message and one output message are seen at the terminal, internally there are two separate actions, each with an input and output message.

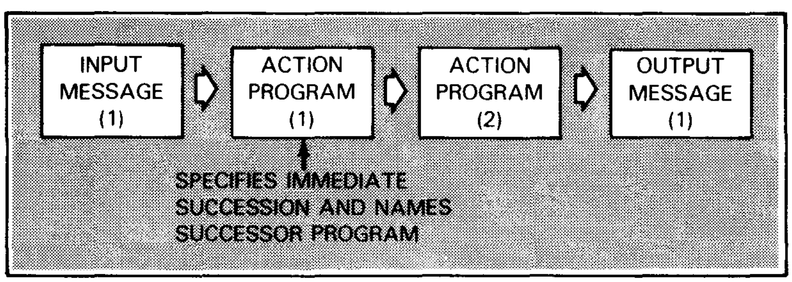


Figure 1-6. Immediate Internal Succession

Immediate succession

Use immediate succession to tell IMS that the current action program processed an input message but is not producing an output message. When it terminates, its successor action program immediately takes up where processing left off, produces an output message and terminates. In immediate succession, there is only one input and one output message. Thus, two action programs are processing a single action.

Combining transaction structures

With these four types of termination or transaction structures there is a good deal of flexibility in structuring transactions. There are basically no limitations as to how you can combine them. For example, you can specify immediate succession, delayed succession, external succession, and finally normal termination, all in turn (Figure 1-7).

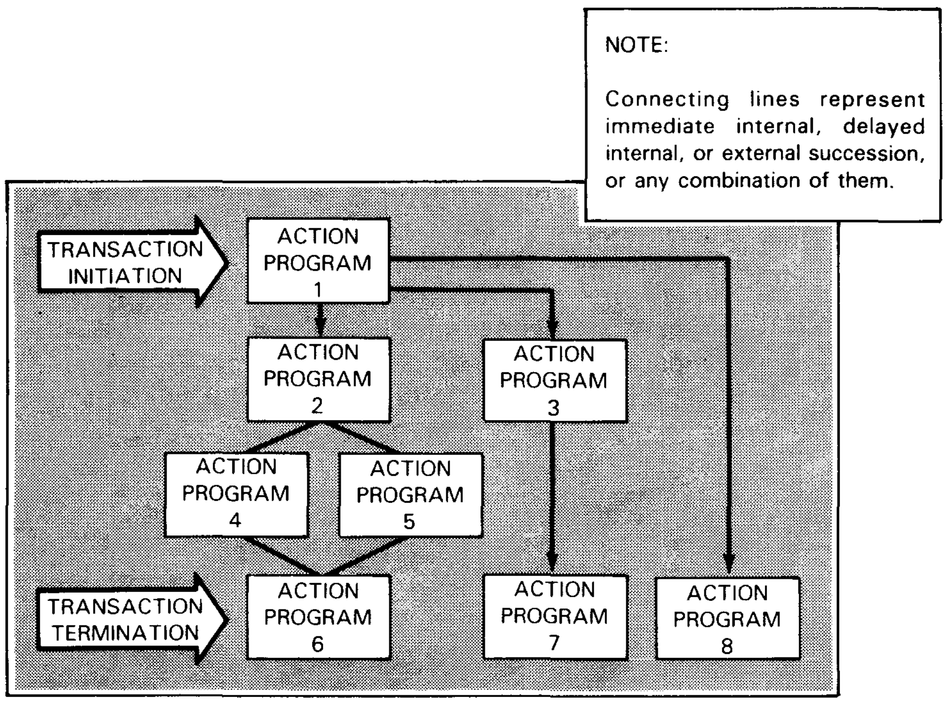


Figure 1-7. Dynamic Transaction Structure

1.5. WRITING REUSABLE ACTION PROGRAMS

Action programs must be serially reusable

You must write action programs so that they are serially reusable. This allows different terminals specifying the same transaction code to take turns using the same action programs. As long as IMS doesn't require the main storage space, action programs remain there after use and aren't reloaded each time they are called.

RPG II turns off indicators and switches

RPG II turns off all indicators and internal switches after each action program execution. When the same program is again initialized for use, RPG II sets on only the 1P indicator.

Action program must reset fields

Since action programs are serially reusable, you must reset all fields to their original value before reexecuting the program. For example, you must blank or zero out any fields you expect to be blank or zero since they may contain values from a previous execution.

1.6. HOW YOUR PROGRAM TALKS TO IMS

Activation record links action program to IMS

To communicate with IMS, an action program must link itself to IMS. This link is the activation record. The activation record handles the control and communication of data between IMS and your action program. The activation record can contain up to six interface areas as shown in Figure 1-8.

Interface area usage

Whether or not you use all six interface areas depends on the needs of your action program. All the interface areas are optional. In the case of the program information block, whether or not you define it in your action program, RPG II automatically returns values to the status code fields after each I/O request. We'll discuss these fields in Section 2.

More information on interface areas

Also, in Section 2, we'll discuss when, why, and how you use the interface areas.

Layout of the activation record in main storage

Figure 1-8 shows how main storage looks when the action program PROG01 is loaded in a multithread IMS system. The layout of the activation record is slightly different in single-thread IMS.

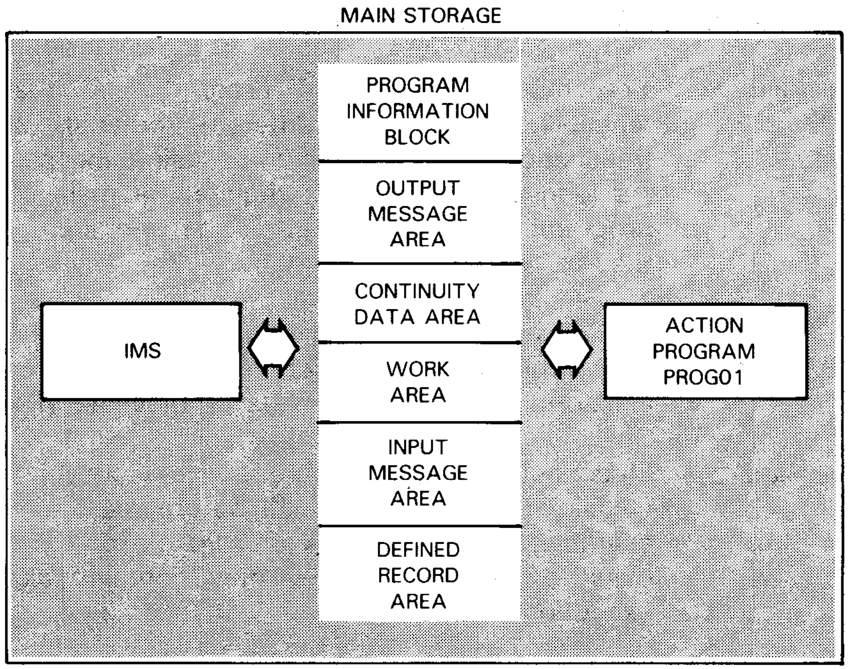


Figure 1-8. The Activation Record in Main Storage

Figure 1-9 shows the relationship between an action program and its interface areas.

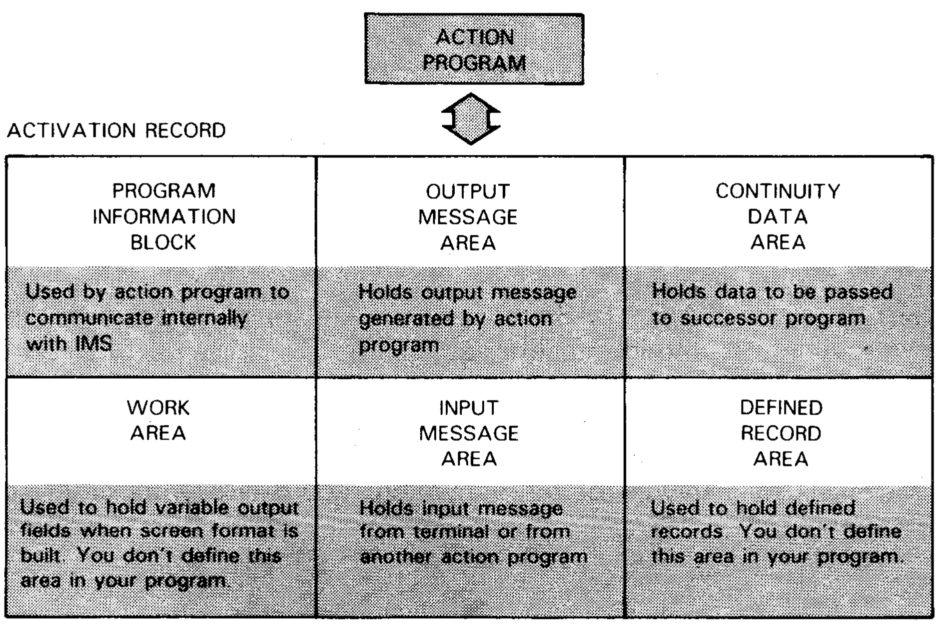


Figure 1-9. The Action Program and Its Interface Areas

2. General Rules for Coding Action Programs

2.1. CODING ACTION PROGRAMS

Action programs similar to normal RPG II programs

Coding action programs is very similar to standard RPG II coding. However, there are some differences since action programs operate under the control of IMS.

Scope of section

In this section, the discussion centers around those coding specifications that distinguish an action program from standard RPG II programs. We won't discuss the standard RPG II coding practices with which you are already familiar. For more information about RPG II coding, consult the report program generator II (RPG II) user guide, UP-8067 (current version).

Most differences on file description form

A sizeable part of this discussion concerns the file description form since the major coding differences for action programs concern this form. In addition, differences in coding for other RPG forms are covered in this section. Where we don't point out differences in coding, assume that action programs conform to the same coding rules as standard RPG II programs. IMS coding restrictions for all coding forms are listed in Appendix D.

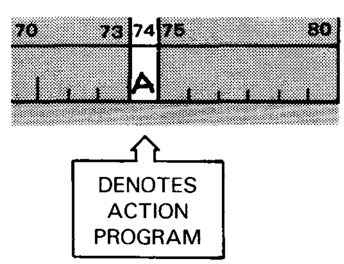
RPG II form names

In our discussion of the various coding forms, we refer to them as the control, file description, file extension, calculations, input, and output forms.

2.2. IDENTIFYING AN ACTION PROGRAM

'A' on control form denotes action program

You denote an action program by placing the letter A in column 74 of the control form. It tells the compiler to generate a program that interfaces with IMS.



Naming the program

Enter the program name in columns 75 through 80. This name is assigned to your program during compilation. When you don't specify a name, RPG II automatically assigns RPGOBJ as the program name. However, since you will undoubtedly have numerous action programs, you will want to give each a unique name.

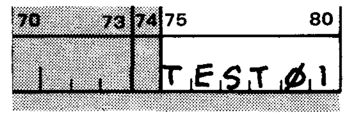


Figure 2-1 shows the control form coding.

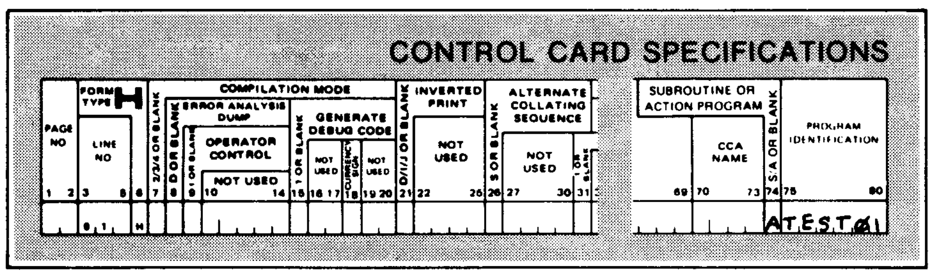


Figure 2-1. Coding the Control Form

Naming restrictions

The program name must conform to the following ....

Naming Restrictions

1–6 be one to six characters;

A start with an alphabetic character (the remainder may be any alphanumeric characters); and

[be left-justified.

2.3. DESCRIBING FILES AND INTERFACE AREAS

Define files as in normal RPG II programs

Use the file description form to describe the files and the interface areas your action program is going to use. Describe all the files the action program accesses just as you would in a standard RPG II program.

File types you can access

Action programs access conventional MIRAM, ISAM, DAM, and SAM files as well as IMS defined files. (You can access IRAM files but you must define them to the IMS configurator as MIRAM files.) Conventional files are data files you create via OS/3 data management. Defined files are files created by IMS from conventional files according to user-supplied definitions. For more information on creating and using defined files, consult the IMS data definition and UNIQUE user guide, UP-9209 (current version).

Where data files are defined to IMS

You identify data files used by an action program in the ACTION section of the IMS configuration and define each of your conventional files in a FILE section. Table 2-1 summarizes the file organization, access modes, and file types used in action programs. See Appendix D for allowable file description form entries.

File organizations, access modes, and file types used by action programs

Table 2-1. Summary of File Types Used by Action Programs

File Type	File Organization	Access Mode	RPG II File Types
IMS Defined File	Indexed	Random	Input/Update/Output*
		Sequential	Input
MIRAM	Indexed	Random	Input/Update/Output*
		Sequential	Input
	Nonindexed (Relative)	Random	Input/Update/Output
		Sequential	Input
	Dedicated Sequential (Disk or Tape)	Sequential	Output
ISAM	Indexed	Random	Input/Update/Output*
		Sequential	Input
DAM	Nonindexed (Relative)	Random	Input/Update/Output
SAM	Sequential (Disk or Tape)	Sequential	Output

*For output files, only ADD is allowed.

Where data files are defined to RPG II

You define all files used by action programs on the file description form and input/output form.

Accessing files in random mode

An action program can access ISAM, DAM, MIRAM, and defined files in random mode by defining them as chained files on the file description form (column 16).

Restrictions on file updating

Operating under IMS, the action program retrieves one record at a time. Updating or deleting of the retrieved record must be done before the next record is retrieved. Records being added to, or deleted from, a file on which updating is being performed cannot be added or deleted between the reading and writing of a record that is being updated. The ADD or DEL specification in columns 16–18 of the output form performs add or delete functions.

Accessing files in sequential mode

An action program can also read ISAM, MIRAM, and defined files in sequential mode. Define them as primary or secondary files and use normal cycle input, or as demand files and use the READ operation on the calculations form.

Writing to sequential files

An action program can write output to a SAM file or dedicated sequential MIRAM file. Sequential input files (disk or tape) are not supported. However, you can read a disk MIRAM file sequentially by defining it as a random file (MODE=RAN) in the FILE section of the IMS configuration.

Where the differences are

The major difference in coding the file description form is the use of the interface areas or activation record. The interface areas and how you code them are described in 2.4 through 2.19.

2.4. DEFINING THE INTERFACE AREAS

Activation record

The activation record handles the control and communication of data between IMS and your action program. The activation record can contain as many as six interface areas:

Interface area names

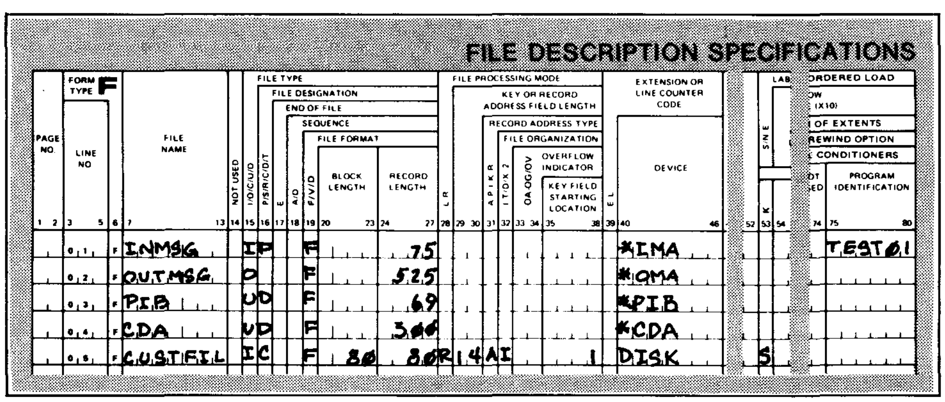
- Input message area (IMA)
- Continuity data area (CDA)
- Output message area (OMA)
- Work area (WA)
- Program information block (PIB)
- Defined record area (DRA)

On the file description form, define the interface areas your action program intends to reference. You never define a work area or a defined record area, although these areas may be part of your program's activation record.

Sample coding of interface areas

Notice in Figure 2-2 that the action program PROG01 has defined one data file, CUSTFIL, and four interface areas. This means that PROG01 intends to reference the input message area, output message area, program information block, and continuity data area during processing.

Example



FILE DESCRIPTION SPECIFICATIONS

Line No (3–5)	Form Type (6)	File Name (7–13)	I/O/C/U/D (15)	P/S/R/C/D/T (16)	F/V/D (19)	Block Length (20–23)	Record Length (24–27)	L/R (28)	Key or Record Address Field Length (29–30)	Record Address Type (31)	File Organization (32)	Key Field Starting Location (35–38)	Device (40–46)	53	Program Identification (75–80)
01	F	INMSG	I	P	F		75						*IMA		TEST01
02	F	OUTMSG	O		F		525						*OMA		
03	F	PIB	U	D	F		69						*PIB		
04	F	CDA	U	D	F		300						*CDA		
05	F	CUSTFIL	I	C	F	80	80	R	14	A	I	1	DISK	S	

Figure 2-2. Defining Files and Interface Areas

Assigning interface area file names

The interface areas are defined just like any other file. You assign a unique file name in columns 7–13 for each interface area. This file name follows the standard rules for file names. The file name you assign can be the same as the interface area name.

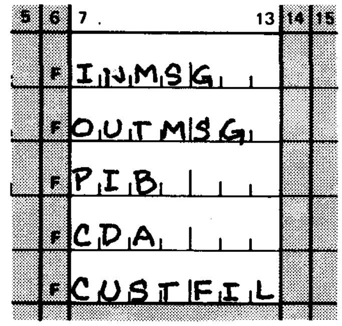


Table 2–2 summarizes the entries you must make.

Acceptable entries

Table 2–2. Coding Interface Areas on the File Description Form. *When you define an interface area, you must make these entries on the file description form.*

Interface Area	File Name (7–13)	File Type (15)	File Designation (16)	Format (19)	Record Length (24–27)	Device Name (40–46)
Input Message Area	Any	I, U	P, S, D	F	16 + message size	*IMA
Output Message Area	Any	U, O	D, blank	F	16 + message size	*OMA
Program Information Block	Any	I, U	D	F	Varies (70 maximum)	*PIB
Continuity Data Area	Any	I, U, O	P, S, D, blank	F	Saved data size	*CDA

2.5. DEFINING THE PROGRAM INFORMATION BLOCK (PIB)

Purpose

Size

The program information block passes control data between IMS and the action program after I/O and at termination. It is a predefined 145-character area. Your action program can access only the first 70 characters. The remaining 75 characters are for IMS internal use only.

RPG II checks status codes

The program information block is always present in the activation record, but you don't need to define it unless you reference it. After each I/O request, RPG II automatically checks the status codes and makes them available whether or not you define the program information block.

Define PIB as input demand or update demand file

You define the program information block in one of two ways:

- as an input demand file; or
- as an update demand file

Type depends on use

Choose input demand if you intend only to read it for data. If you intend to update it, you must define it as an update demand file.

Structure of the Program Information Block

Before discussing the program information block, let's take a look at the data it contains. Table 2-3 summarizes the contents of the program information block; subsection 2.6 is a detailed description of the fields action programs can reference.

Summary of program information block fields

Table 2-3. Contents of the Program Information Block

Characters	Specification	Characters	Specification
1-2	Status-code	47-48	Continuity-data-area-inc
3-4	Detailed-status-code	49-63	Success-unit-id
5-10	Successor-id	49-54	Transaction-date
11	Termination-indicator	49-50	Year
12	Lock-rollback-indicator	51-52	Month
13-20	Transaction-id	53-54	Day
13-14	Year	55-63	Time of day
15-16	Day	55-56	Hour
17-20	Time	57-58	Minute
21-27	Data-def-rec-name	59-60	Second
28-34	Defined-file-name	61-63	Filler
35-36	Standard-msg-line-length	64-69	Source-terminal-chars
37-38	Standard-msg-number-lines	64	Source-terminal-type
39-40	Work-area-length	65-66	Source-term-msg-line-length
41-42	Continuity-data-input-length	67-68	Source-term-msg-number-lines
43-44	Continuity-data-output-length	69	Source-term-attributes
45-46	Work-area-inc	70	DDP-mode

2.6. HOW PROGRAM INFORMATION BLOCK FIELDS ARE USED

Determining Error Status

Status-code

Status-code (positions 1–2) is a half-word binary integer value returned by IMS indicating the completion status of a request. Remember that RPG II still sets *ERROR to indicate the error condition; however, the status code provides more detailed information. The status-code values are:

Status-code values

Status	Value
Successful	0
Invalid key or record number	1
End of file or unallocated optional file	2
Invalid request	3
I/O error	4
Violation of data definition	5
Internal message control error	6
Screen format error	7

When status-code=3 (invalid request)

An invalid request status code is returned when IMS detects an error in a request before passing the request to data management, the control system, or the integrated communications access method (ICAM).

When status-code=4 (I/O error)

IMS returns an I/O error status code when an unrecoverable error is detected by data management, the control system, or ICAM.

When you configure ERET=YES

You specify an error return option for each action program at configuration time. If you choose to accept errors (ERET=YES specified to the configurator), then, regardless of the status-code value, the action program regains control when the request is completed. When an error occurs, *ERROR is set. If you want more information about the error, you must test for the various status codes.

When you don't configure ERET=YES

If the option to reject errors is chosen or defaulted at configuration time, IMS returns control to the action program only when the status code equals 0,1,or 2. When any other status code is returned, the action program doesn't regain control.

Recommendation

We strongly advise that you specify ERET=YES so that your program can regain control and terminate orderly.

Detailed-status-code

Detailed-status-code (positions 3–4) is a half-word binary value returned by IMS following a request when the status code is invalid request (3), I/O error (4), internal message control (6) error, or screen format (7) error. The detailed status code provides more detailed information concerning the error. IMS also returns detailed status codes for invalid key (status code 1) when you use defined files.

Detailed-status-code for I/O error

When the status code is I/O error (4), the detailed status code contains either filenameC + 2 or the error code and subcode returned by the file access method. All file types except MIRAM return a detailed status code of filenameC + 2. MIRAM files return an error code (DM) and subcode. You can find these messages in the system messages programmer/operator reference, UP-8076 (current version).

Detailed status codes for other errors

The detailed status codes for status codes 1, 3, 6 and 7 are listed in Appendix C.

Naming A Successor Program

Successor-id (positions 5–10)

Successor-id identifies the action program that takes control when the current program terminates. You must move the name of the successor action program into successor-id whenever you terminate with external, delayed, or immediate succession.

Size and name

Successor-id is a 6-character field. The name you assign must be left-justified and zero-filled.

When you specify normal termination

When the action program uses normal termination, don't specify a value for successor-id.

Use to find cause of errors

The successor-id field is also used to find and display the cause of errors. To find the cause of an error, check the status-code field, associate a successor-id with each possible error condition, and assign an error code to each condition. When an error occurs, move the error code to the successor-id field and terminate your action program abnormally by moving **A** or **S** to the termination-indicator field. IMS sends the error code from the successor-id field to the terminal after abnormal termination.

Specifying Types Of Termination

Termination-indicator (position 11)

Termination-indicator is a 1-character value that shows the type of termination for the current program. (See 1.4 for a description of the types of termination.) You select the type of termination by moving a specific character to the termination-indicator field.

Default value

When you don't move a value to termination-indicator, IMS assumes normal termination.

Table 2-4 lists the character, type of termination it selects, and IMS operations that take place.

Table 2-4. Termination Indicators

Termination types and IMS operations

To Terminate Current Action Program With:	Move To Termination-Indicator	IMS Operations
Normal Termination	N	Output message is sent to terminal. All resources, including current action program, are released. When you don't move a value to this field, normal termination is assumed.
External Succession	E	Output message is sent to terminal. Any data saved by this program is stored in the continuity data file. All resources, including current action program, are released. Successor action program is scheduled when another input message is received from originating terminal.
Delayed Succession	D	No output message goes to the terminal. Output message is queued as input message to successor action program. Any data saved by the program is stored in the continuity data file. All resources, including current action program are released. Successor action program is initiated by normal scheduling process.
Immediate Succession	I	No output message goes to the terminal. Current action program only is released. Successor action program is immediately initiated and IMS passes to it (intact) the interface areas of the predecessor program.
Abnormally without Snap Dump	A	Sends error message to originating terminal (includes value moved to successor-id). All resources are released. All files are rolled back.
Abnormally with Snap Dump	S	Same as A except a snap dump of current action program and its activation record is also provided. To get a snap dump, specify // OPTION DUMP, JOBDUMP, or SYSDUMP in your IMS job control stream.

Table 2-5 summarizes the types of termination an action program can specify and the associated successor-id entries.

Table 2-5. Summary of Action Program Termination Types and Successor-ids

	Type of Termination					
Program Information Block Data Items	Normal Transaction Termination	Abnormal Transaction Termination	Abnormal Transaction Termination with Snap Dump	Action Termination with External Successor	Action Termination with Immediate Internal Successor	Action Termination with Delayed Internal Successor
Successor-id	Ignored	Termination code	Termination code	Successor program name	Successor program name	Successor program name
Termination-Indicator	N	A	S	E	I	D

Involuntary termination

The termination-indicator field controls voluntary termination of action programs. Action programs can also terminate involuntarily. Involuntary termination occurs when IMS encounters an abnormal condition in the processing of a request issued by an action program. Involuntary termination occurs when action program execution causes a program check or when an execution loop within an action program continues beyond a specified time limit. When any of these conditions occurs, IMS sends a 3-line message to the originating terminal and to the system console, giving the cause of the abnormal termination. Abnormal termination messages are listed in the system messages programmer/operator reference, UP-8076 (current version).

Causes

Result

Obtaining a dump

A snap dump of the action program and its activation record is performed only when // OPTION DUMP, JOBDUMP, or SYSDUMP is specified in the job control stream for executing IMS.

Record Locking and Rollback

Lock-rollback-indicator (position 12)

Lock-rollback-indicator is a 1-character value, set by the action program, that indicates the record lock and rollback functions you want performed at action program termination. Table 2-6 summarizes the possible entries for this field.

Table 2-6. Summary of Record Locks and Rollback

Lock-Rollback-Indicator	Termination-Indicator	Description
H	E, D	Holds all locks imposed by the current action program into the successor program.
R	E, D	Releases all pending locks set by the current action program. Update locks are held into the successor program.
N	E, D, N	Releases all locks for the transaction. Establishes a new rollback point in the audit file. This is the default value.
O	E, D, N	Releases all locks for the action or transaction. Rolls back all updates for this action or transaction. Establishes new rollback point in the audit file.

Default value

IMS checks the lock-rollback-indicator field at action termination for external and delayed succession or normal termination. When you don't specify a value in lock-rollback-indicator, IMS assumes the value N. Don't confuse this with the N signifying normal termination.

Holding of locks

IMS doesn't check the lock rollback indicator when you terminate with immediate succession. All records remain locked since there is only one action taking place in immediate succession and IMS always holds locks for at least the length of the action.

Caution in using R and H options

Use the R and H options only when the termination indicator is set to E for external succession or D for delayed succession. In long transactions, use R and H with caution. Holding of locks across action programs in a multithread environment can cause deadlock. In a single-thread environment, holding locks across actions can decrease response time. In single-thread IMS, you can use the R and H indicators only when you specify RECLOCK=YES in the OPTIONS section of the configuration. See the IMS system support functions user guide, UP-8364 (current version).

Single-thread restriction

Advantages of the N option

Use the N option for long-running update transactions. The N option releases all locks when the termination indicator is set to E for external succession or D for delayed succession. With normal termination, locks are always released and a new rollback point is established. This option also establishes additional rollback points, limits the range of rollback, and reduces the size of the audit file. The audit file contains the before-image of records to be updated. By limiting the number of updates in an action program or by establishing additional rollback points in a long-running transaction, you reduce the size of the audit file and save disk space.

Getting online file recovery

The O option activates online file recovery to roll back files to the previous rollback point. Use the O option for external and delayed succession or normal termination.

Lock for update

If you specify lock for update (LOCK=UP) for a particular file in the FILE section at configuration time, IMS releases record locks when updates are completed rather than at the end of an action. When you use this option, IMS doesn't save before-images in the audit file and doesn't roll back updates at abnormal termination. You can use the R indicator to release locks on uncompleted updates at the end of an action, or the H indicator to hold locks on uncompleted updates into the next action.

Transaction Identification

Transaction-id (positions 13–20)

Transaction-id is a unique identification for a transaction. IMS sets this value for all action programs that are part of the same transaction. The first part is the date in Julian form; the second part is a unique number assigned by IMS. If you require the accurate date and time in your action program, use the transaction-date and time-of-day under success-unit-id.

Defined File Identification

Data-def-rec-name and defined-file-name (positions 21–34)

If your action programs access a defined file, the data-def-rec-name (positions 21–27) and defined-file-name (positions 28–34) fields name the defined file or subfile. Both are 7-character items, left-justified and blank filled. The description of the defined file is contained in the data definition record in the NAMEREC file.

IMS places configured values in these fields

When IMS schedules the first action in a transaction, it places:

- the data definition record specified by the DDRECORD configurator parameter into the data-def-rec-name field; and
- the defined file name specified by the DFILE configurator parameter into the defined-file-name field.

Passing new names to successor program

When your action program terminates in external or delayed succession and the successor program accesses a different defined file, you can pass the new data definition record name and defined file name to the succeeding program either by:

1. placing the new names in data-def-rec-name and defined-file-name; or
2. placing zeros in both fields and allowing IMS to insert the values configured for the successor action.

Using conventional files in successor program

If the successor program accesses only conventional files, your action program should place zeros in data-def-rec-name and defined-file-name. This allows the successor program to access a conventional file that may have contributed to the defined file used in the previous action.

Standard Message Size

Standard-msg-line-length (positions 35–36)

Standard-msg-line-length is a half-word binary integer that shows the maximum line length for a message. IMS obtains this value from the CHRS/LIN configurator parameter.

Standard-msg-number-lines (positions 37–38)

Standard-msg-number-lines is a half-word binary integer that shows the maximum number of lines for a message. IMS obtains this value from the LNS/MSG configurator parameter.

Work and Continuity Area Sizes

Work-area-length (positions 39–40)

Work-area-length is a half-word binary integer. It contains the size of the work area specified at configuration time. You must configure a work area when your action program uses screen format services. RPG II uses this work area to store the variable output fields while the screen is built. This all happens internally. The action program itself doesn't use the work area.

Continuity-data-input-length (positions 41–42)

Continuity-data-input-length is a half-word binary integer. It contains the size of the continuity data record passed by the predecessor program.

Continuity-data-output-length (positions 43–44)

Continuity-data-output-length is a half-word binary integer that defines to the current action program the configured size of the continuity data area. When the current program terminates, this field contains the size of the continuity data area passed to the successor program.

Work-area-inc (positions 45–46)

Work-area-inc is a half-word binary integer. Move a value to this field when you need to increase the size of the configured work area in the successor action program. You do this because you know the configured size will not be large enough to hold the screen that the successor program wants screen format services to build.

Continuity-data-area-inc (positions 47–48)

Continuity-data-area-inc is a half-word binary integer. Move a value to this field when you want to increase the configured size of the continuity data area for the successor action program. IMS adds this increment value to the length of the continuity data record that the current action program is saving. It then compares this value to the configured continuity data area size. The larger value becomes the size of the continuity data area for the successor action program.

Success Unit Identification

Success-unit-id (positions 49–63)

Success-unit-id provides a calendar date and clock time for your action program at the beginning of each success unit. Reference this field when your action program requires an accurate date/time value.

Source Terminal Characteristics

Source-terminal-type (position 64)

Source-terminal-type is a 1-character field containing a type code for the source terminal. The values set by IMS are:

Value	Description
C	System console
F	UTS 400 terminal in native mode (with or without character-protect feature)
N	UTS 10, DCT 500, DCT 1000, or teletypewriter
P	UTS 400 terminal in UNISCOPE mode with FCC-protect feature
T	UTS 400 text editor
U	UTS 400 terminal in UNISCOPE mode with character-protect feature
V	UNISCOPE 100 or UNISCOPE 200 terminal
W	Workstation or UTS 20 terminal
3	IBM 3270 terminal
4	UTS 40 terminal

Source-term-msg-line-length (positions 65–66)

Source-term-msg-line-length is a half-word binary integer that specifies the number of characters per line for the source terminal. For hard copy terminals, this is the configured line length (CHRS/LIN specification in the GENERAL section of the IMS configuration).

Source-term-msg-number-lines (positions 67–68)

Source-term-msg-number-lines is a half-word binary integer that specifies the number of lines for the source terminal. For hard copy terminals, this is the configured number of lines (LNS/MSG specification in the GENERAL section of the IMS configuration).

Source-term-attributes (position 69)

Source-term-attributes is a 1-character field defining specific attributes of the source terminal. The values it can contain are:

Value	Description
A	Screen bypass and Katakana
K	Katakana character set
N	Nonvideo device
S	Screen bypass feature
Z	None of these attributes

Remote Transaction Type

DDP-mode (position 70)

DDP-mode is a 1-character field that identifies the type of remote transaction in distributed data processing. The values set by IMS are:

Value	Description
A	Transaction was initiated because of an ACTIVATE request from an action program (program-initiated transaction).
R	Transaction was initiated by directory or operator routing (operator-initiated transaction).

2.7. HOW TO READ THE PROGRAM INFORMATION BLOCK

Defining PIB as an input demand file

To read the PIB (but not update it), define it as an input demand file on the file description form.

Using status codes to determine processing

Let's assume that in your action program you want to be able to read the status-code and detailed-status-code fields, and based on the values they contain, determine what processing is done. Figure 2-3 shows the file specifications.

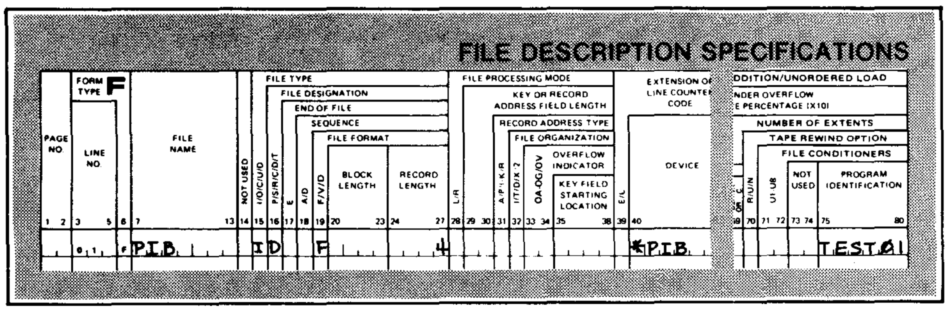
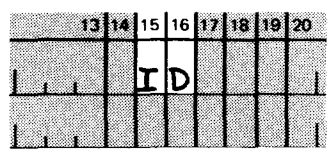


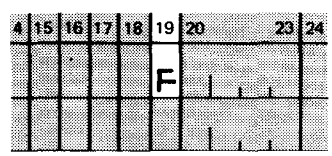
Figure 2-3. Defining the Program Information Block as an Input Demand File

Sample file description form coding



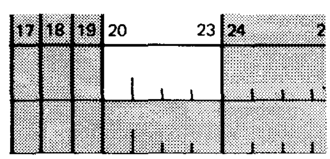
First, name the file. In Figure 2-3, the file name is PIB. Then, enter an I in column 15 for file type and a D in column 16 for file designation.

Column 19 (file format)



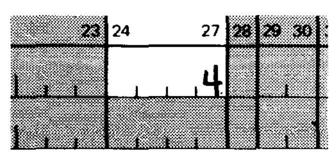
Enter an F in column 19 for file format. For RPG II action programs, the file format entry is always F.

Omit block length



Omit block length (columns 20–23). If you enter a value, it must equal record length.

Columns 24–27 (record length)



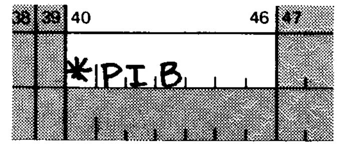
Enter 4 since status-code and detailed-status-code are the first four characters of the program information block. These are the fields you want to read. If you choose, you can reference all 70 characters of the program information block by entering 70 for record length. By doing that, you can read any of its fields during your action program.

Considerations in determining record length

In defining record length, specify at least the number of characters up to and including the field or fields in the program information block that you want to read.

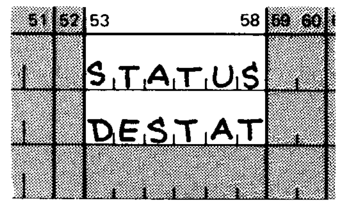
Columns 40–46 (device name)

Specify *PIB. You may not enter any other name.



Input form entries

To get the values for status-code and detailed- status-code into your action program, you have to name these fields on the input form (Figure 2-4). You can assign any name you choose, provided the position you assign to them corresponds exactly to their position in the program information block. Program information block fields defined on the input form that are not read by your action program are flagged at compilation as unreferenced.



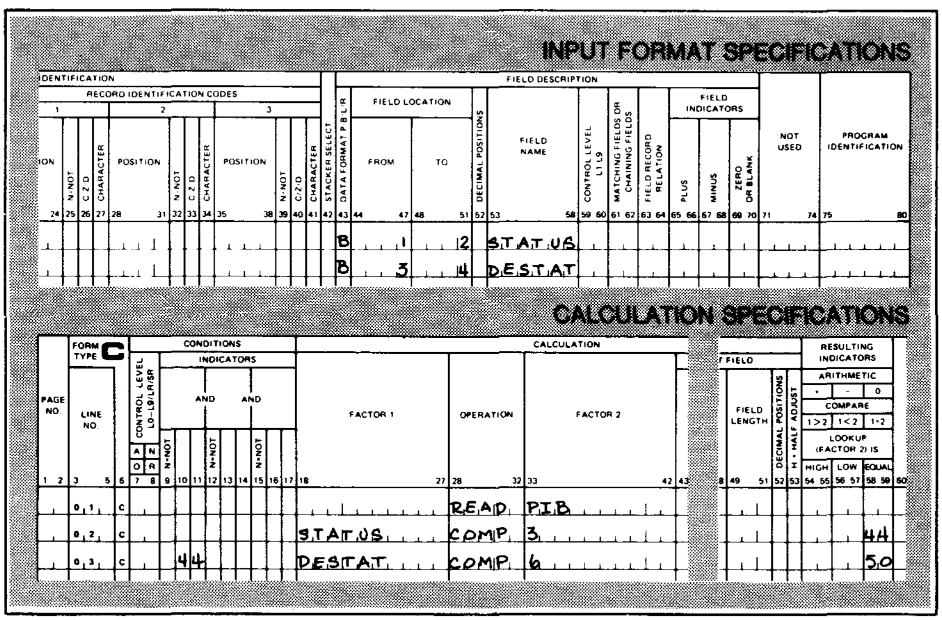
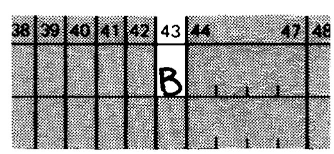


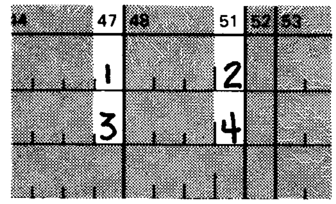
Figure 2–4. Testing Status and Detailed Status Codes

Specify binary fields

In column 43 of the input form, specify B, because status-code and detailed-status-code are binary fields.



In columns 47 and 51, specify the starting and ending positions.



Specifying the READ operation

On the calculation form, specify the READ operation for the file name you assigned to the program information block.

Testing for status codes

To test the status codes and detailed status codes, specify the COMP operation for the field names you specify on the input form. Figure 2-4 shows the coding to test for a status code of 3 and detailed status code of 6.

No end-of-file indicator set on

You may read the program information block as many times as you want. RPG II doesn't set on the end-of-file indicator.

2.8. HOW TO UPDATE THE PROGRAM INFORMATION BLOCK

Defining PIB as an update demand file

To update the program information block, define it as an update demand file. There are many instances when you will need to do this. The most common reason for updating the program information block is to specify types of termination - normal termination, external, delayed, or immediate succession.

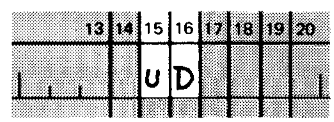
Updating successor-id and termination-indicator

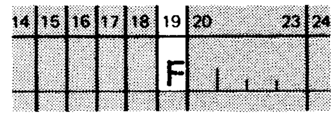
Let's assume your transaction contains two action programs, PROG01 and PROG02. For processing to continue when PROG01 terminates, PROG01 must name its successor and the type of termination. PROG01 does this by updating the program information block. On the output form, it moves the name of the successor program, PROG02, into the successor-id field and moves the termination code, E, D, or I, depending on the type of termination desired, to the termination-indicator field. Now let's take a look at how you code the file description form to allow for this updating.

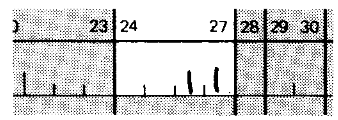
Sample file description form coding

In Figure 2-5, you see how we defined the program information block as an update/demand file in columns 15 and 16, and entered an F for file format in column 19. For record length, we specified 11 since termination-indicator occupies position 11 in the program information block. You must specify at least 11 character positions when updating the termination-indicator field.

Defining record length

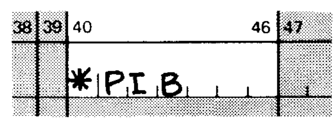






Device name

Enter *PIB in columns 40–46. You can't substitute any other name in these columns.



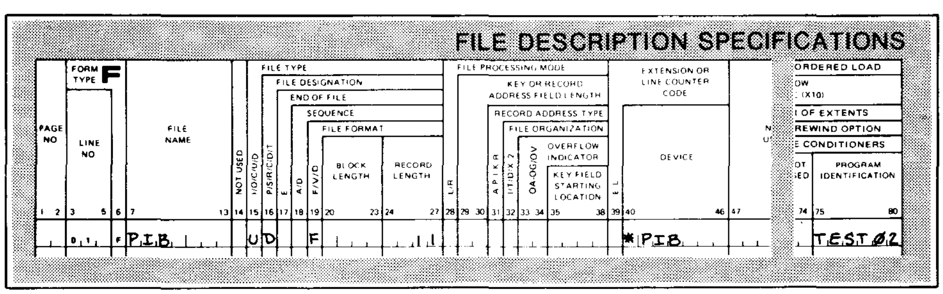
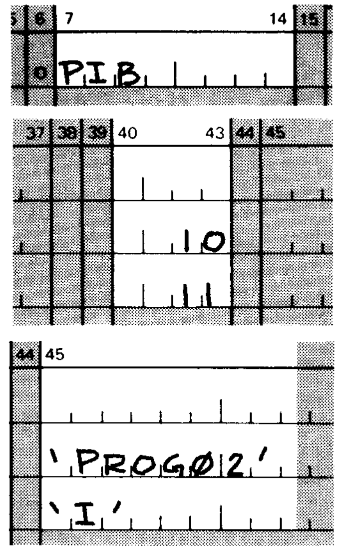


Figure 2–5. Defining the Program Information Block as an Update Demand File

UPDATING THE PROGRAM INFORMATION BLOCK

Sample output form coding

The actual updating of these fields occurs at output. Figure 2-6 shows the output form for PROG01. The file name is PIB. It matches the name assigned to the program information block on the file description form. We defined the end positions for output as 10 and 11, respectively. Position 10 is the end position for successor-id; and 11, for termination-indicator. In columns 45-70, we indicated 'PROG02' as the name of the successor program and 'I' for immediate succession as the type of termination. When PROG01 terminates, 'PROG02' is moved to the successor-id field and 'I' to termination-indicator. IMS then checks the fields to determine what processing takes place next.



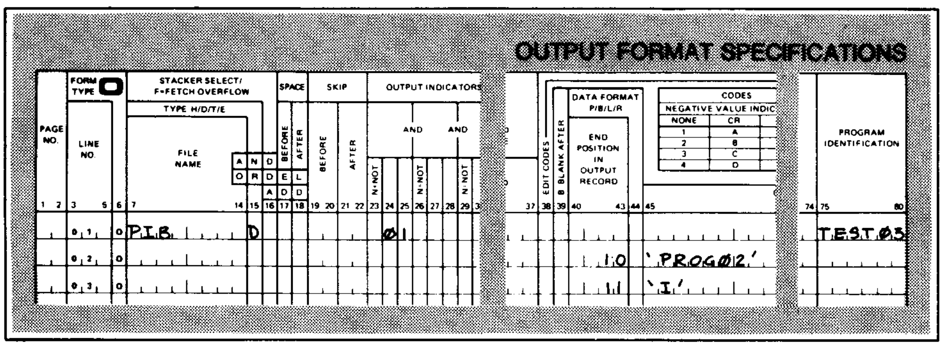


Figure 2-6. Designating a Successor Program and Type of Termination

No READ operation

You don't need to read the program information block before updating it. RPG II does this for you. However, you must define it as an update demand file.

You define it as an update demand file so you can change individual fields. If you define it as an output file, you must supply all fields or the information contained in the fields you don't supply will be overlaid by blanks. Therefore, it is much easier to define it as an update demand file.

When you specify the PIB as update demand and do not supply input specifications, you receive a warning message that there are no input specifications. This is only a warning message and you need not take any action.

When reading the program information block, be aware that the end-of-file indicator is not set on by RPG II.

2.9. DEFINING THE INPUT MESSAGE AREA (IMA)

The input message sent from the terminal goes to the input message area where it awaits processing by the action program. You define an input message area if your action program references it.

Defining the input message area

Generally, the IMA is defined as a primary input file since the input message coming in from the terminal often contains data to be processed by the action program.

Size

The input message area's size is usually specified at configuration time. When the size isn't specified or the size specified is inadequate, IMS allocates an area large enough to handle the entire input message.

Control header

In addition to the input message coming in from the terminal, the input message area also contains a control header. The control header is 16 characters long and contains data generated by IMS related to the input message.

Format of the Input Message Area Header

Table 2-7 lists the fields that comprise the input message area control header.

Summary of header fields

Table 2-7. Input Message Area Control Header Contents

Characters	Specification
1-4	Source-terminal-id
5-12	Date-time-stamp
13-14	Text-length
15	Reserved for system use
16	Auxiliary-device-id
C'1'	Device = Aux1
C'2'	Device = Aux2
C'3'	Device = Aux3
C'4'	Device = Aux4
C'5'	Device = Aux5
C'6'	Device = Aux6
C'7'	Device = Aux7
C'8'	Device = Aux8
C'9'	Device = Aux9

Input Message Header Fields

The input message area control header contains the following items:

Source-terminal-id

Source-terminal-id (positions 1–4)

Source-terminal-id identifies the terminal that sent the input message.

Message-identifier

Message-identifier (positions 5–12)

Message-identifier is a unique identifier for each input message. The first part is the date; the second part is a unique number assigned by IMS. It is given in binary integers.

Text-length

Text-length (positions 13–14)

Text-length is a binary half-word integer that specifies the length of the input message text.

Auxiliary-device-id

Auxiliary-device-id (position 16)

Auxiliary-device-id is the configured number of the auxiliary device transmitting data to the action program. This number is specified in the communications network definition.

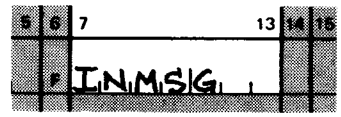
2.10. READING THE INPUT MESSAGE AREA

Defining IMA as an input file

In most circumstances, the input message area is defined as a primary input file since the input message sent from the terminal is the first data you want the action program to process. Consequently, as soon as your action program begins processing, RPG II reads the input message area. Study Figure 2-7 for a moment.

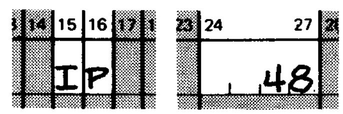
Sample file description form coding

In Figure 2-7, we define the input message area as INMSG in columns 7-13, file name. You must give the input message area a unique name; you can name it IMA.



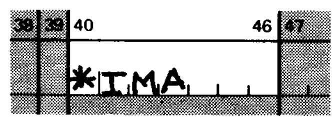
Columns 15,16,24-27

We entered IP for primary input in columns 15 and 16, respectively. The record length entry is 48. This designates the size of the input message (32 characters plus an additional 16 characters for the IMA control header) that this action program is expecting.



Device name

The entry *IMA in columns 40-46 is required. You may not substitute any other name.



Read once only

RPG II reads the input message area only once. After this, any attempt to read this area sets the end-of-file indicator on.

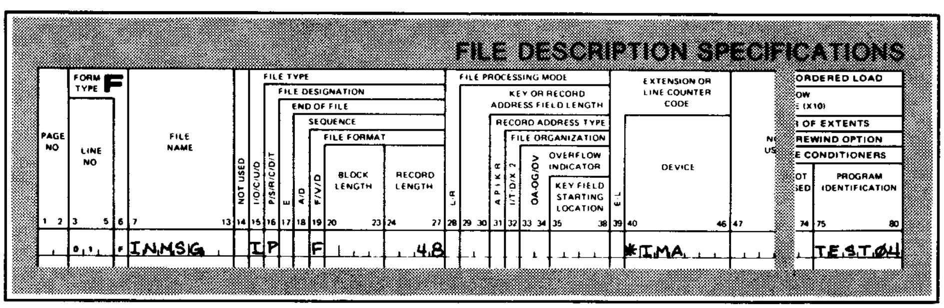


Figure 2-7. Defining the Input Message Area as a Primary Input File

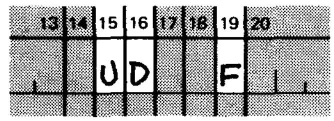
2.11. USING THE INPUT MESSAGE AREA TO PASS DATA

Defining IMA as an update file

Define the input message area as an update file (Figure 2-8) when you want to use it to pass data from the current action program to its successor program.

Saving data in the input message area

Normally, you pass data by means of the continuity data area. However, when you use immediate succession, you can pass data to the successor program in the input message area.



How to pass data

To use the input message area to pass data, define it as an update file. Then, at termination, output to the input message area any data you want to save and pass to the successor program. You would code this operation on the output form as you would to do output to any file.

Successor program using saved data

The successor program defines the input message area as an input or update file depending on how it intends to use the data. To read the data, define it as an input file. To read and update the input message area, define it as an update file. In either case, the data saved in the input message area of the predecessor program is immediately available to the successor program.

Restrictions on reading input message area

Remember, you can only perform a READ operation on the input message area once. If you try it a second time, the end-of-file indicator is set on.

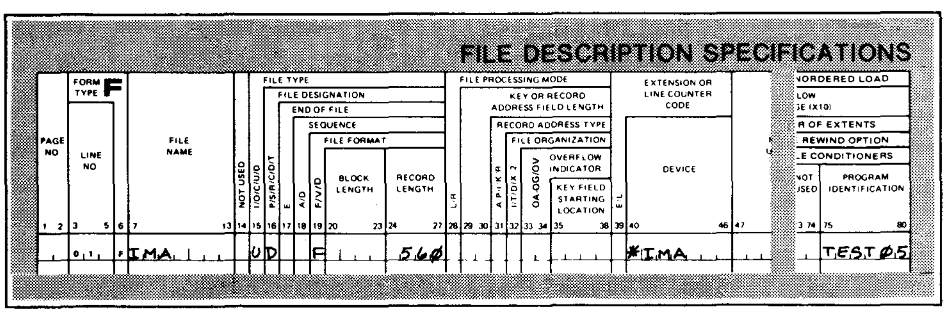


Figure 2-8. Defining the Input Message Area as an Update Demand File

Immediate succession saves interface area contents

When using the input message area to pass data between programs, you must specify immediate succession in the termination-indicator field of the current action program. Only in immediate succession does the input message area remain intact between the time the first action program terminates and the successor program begins processing. Recall that in normal termination, external and delayed succession, the interface areas, including the input message area, are released at the termination of the current program. And in the case of external and delayed succession, the successor program gets its own set of interface areas. In immediate succession, however, all interface areas remain intact. Consequently, the data saved in the input message area of the first program is accessible to the successor program.

All other terminations release interface areas

To save input message area

Remember if you want to use the input message area to pass data:

- on the file description form, define it as an update file;
- on the output form, move the data to be saved to the input message area; and
- specify 'I' for immediate succession in the termination-indicator field.

2.12. DEFINING THE OUTPUT MESSAGE AREA (OMA)

Purpose

The output message area holds the output message that your action program generates. It remains there until it's sent to the terminal.

Size

You must define an output message area when your program produces an output message. The maximum size of the output message area is specified at configuration.

Control header

In addition to the output message sent to the terminal, the output message area contains a control header. This header is 16 characters long and contains data generated by IMS concerning the output message.

Format of the Output Message Area Header

Table 2-8 lists the fields that comprise the output message area control header.

Summary of header fields

Table 2-8. Output Message Area Control Header Contents

Characters			Specification
1-4			Destination-terminal-id
5-6			SFS-options
	5		SFS-type
	6		SFS-location
7-8			Reserved for system use
9-12			Continuous-output-code
13-14			Text-length
15-16			Auxiliary-device-id
	15		Aux-function
	16		Aux-device-no
		C'1'	Device=Aux1
		C'2'	Device=Aux2
		C'3'	Device=Aux3
		C'4'	Device=Aux4
		C'5'	Device=Aux5
		C'6'	Device=Aux6
		C'7'	Device=Aux7
		C'8'	Device=Aux8
		C'9'	Device=Aux9

Output Message Header Fields

The output message area control header contains the following items:

Destination-terminal-id

Destination-terminal-id (positions 1-4)

Destination-terminal-id identifies the terminal to receive the output message. If you don't move a value to this field, the terminal that sent the input message receives the output message.

SFS-options

SFS-type (position 5)

When you transmit an input or input/output screen using screen format services, IMS places a value of I in SFS-type. This means that the screen format can be used for input in the following action. You can change the screen to an output-only screen by placing hexadecimal zero in this field.

SFS-location (position 6)

To build a screen format in dynamic main storage instead of in your output message area, move C'D' to SFS-location. Once you build a screen format in dynamic main storage and you want to send a message from the output message area, you must move hexadecimal zero to this field. Screen format services is discussed in Section 6.

Continuous-output-code

Continuous-output-code (positions 9–12)

Continuous-output-code is a 4-character field that the action program uses when generating continuous output. The contents of this field are returned to the successor program in the input message area. Continuous output is discussed in Section 5.

Text-length

Text-length (positions 13–14)

Text-length is a binary half-word integer that specifies the length of the output message. At the start of program execution, this field contains the configured size of the output message area. Before the output message actually goes to the terminal, RPG II enters a new value into the text-length field. It computes this value by taking the end position for the last field described on the output form, and subtracting 12 characters (16 characters for the output message area header minus 4 bytes for the text-length field). IMS then uses this value to determine the size of the output message going to the terminal. This procedure is further described in 2.14.

Auxiliary-device-id

Auxiliary-device-id (positions 15–16)

Auxiliary-device-id contains two fields: aux-function (15) and aux-device-no (16). The action program moves a value to aux-function when it generates continuous output and when it transmits regular output messages to an auxiliary device. Aux-device-no identifies the configured number for the auxiliary device receiving the output message. This number is specified in the communications network definition.

2.13. FILE SPECIFICATIONS FOR THE OUTPUT MESSAGE AREA

You can define the output messsage area as an output file or as an update demand file.

Defining OMA as an output file

Generally, the output message area is defined as an output file since most action programs generate output messages. Figure 2-9 shows you how to do this.

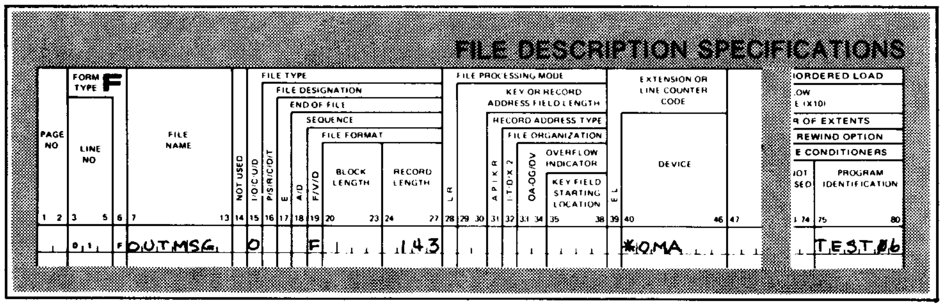
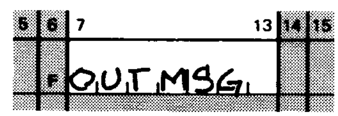


Figure 2-9. Defining the Output Message Area as an Output File

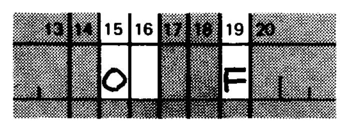
Sample file description form coding

The output message area is defined as OUTMSG in columns 7-13. You must give it a unique name; you can use the name OMA.



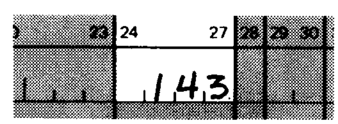
Columns 15,16,19

The file type (column 15) is O for output. Whenever column 15 contains an O, leave column 16 blank. The required entry in column 19 is F for file format.



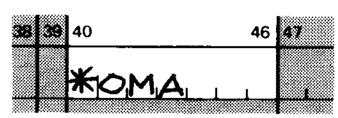
Columns 24-27

In columns 24-27, we entered 143. This is the configured size of the output message, including 16 characters for the control header.



Device name

In columns 40-46 (Device), *OMA is the only acceptable entry.



Defining OMA as an update demand file

Define the output message area as an update demand file when you want to do a READ operation on the output message area. Generally, you read the output message area for one of two reasons:

Reading text-length

1. To determine the value in the text-length field. This field contains the output message area size specified at configuration. Knowing this value is important in determining the size of the output message your action program can create.

Reading data saved by predecessor program

2. To get data saved there by a predecessor program using immediate succession.

Saving data in OMA

You can save data in the output message area with either immediate or delayed succession.

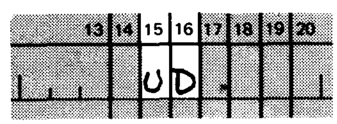
Output message area in immediate succession

With immediate succession, all interface areas of the current action program, including the output message area, remain intact for the successor program. The successor program needs only to read the output message area to get this data.

Output message area in delayed succession

With delayed succession, the output message area of the current action program automatically becomes the input message area of the successor program. Thus, the sucessor program has immediate access to the saved data. If the successor program defines the input message area as the primary file, RPG II reads it as soon as processing begins.

Determining maximum output message area size



In Figure 2-10, all entries are the same as in Figure 2-9, except for columns 15 and 16 where we defined the output message area as an update demand file. We did this in order to read the text-length field to see if the configured output message area size can handle the 143-character output message this program generates. If the configured size is smaller than this, a portion of the message is lost when transmitted to the terminal.

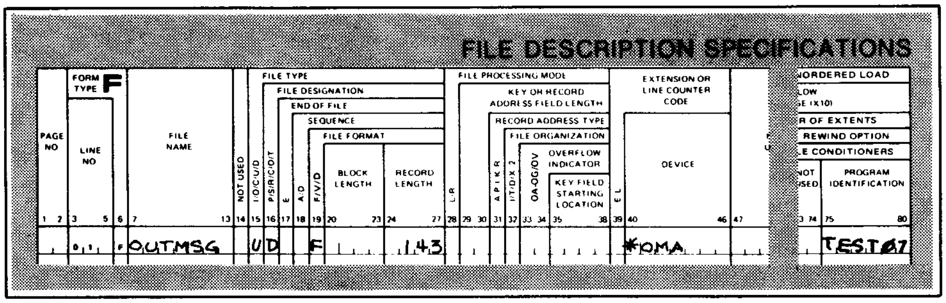


Figure 2-10. Defining the Output Message Area as an Update Demand File

2.14. HOW TO CODE YOUR OUTPUT MESSAGE

RPG II moves value to text-length

When an action program generates an output message smaller than the configured output message area size, RPG II moves a new value into the text-length field before the message is sent to the terminal. Also, when an action program generates more than one message (see Section 5), RPG II moves a value to text-length before each message is sent.

How output message length is determined

RPG II uses the end position of the last field you code on the output form to determine the length of the output message. For this reason, be sure to list last the field with the highest end position. You must also remember to allow 16 characters for the output message header when calculating the end position of the first field.

Allowing for output message header

Suppose your output message has three fields, CUSTNO, NAME, and ACCT. The first field, CUSTNO, is 14 characters long, but you must allow 16 characters for the output message header, so you give the value 30 for the ending position of the first field. NAME and ACCT are each 30 characters.

Example

In Figure 2-11 the field ACCT has the highest value end position, 90, and is listed last on the output form. RPG II computes the value of text-length by taking the value 90 and subtracting 12 characters (16 for the output message area header minus 4 for the text-length field). Consequently, when the output message goes to the terminal, the three fields CUSTNO, NAME,and ACCT all appear on the screen since the value in text-length was large enough to accommodate the three fields.

OUTPUT MESSAGE AREA CODING

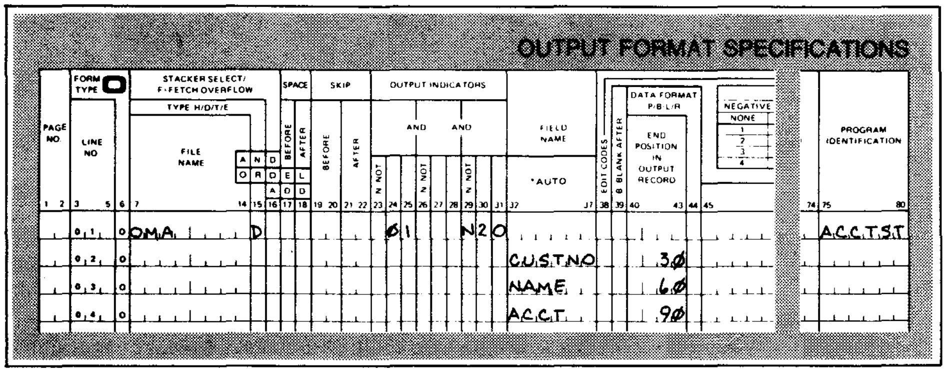


Figure 2–11. Coding the Output Form Determines the Value in Message Length

Incorrect text length

Now look at Figure 2–12. In this case, RPG II looks at the end position on the output form and determines the output text-length field value based on position 60. RPG II computes the value for the text-length field using the end position 60.

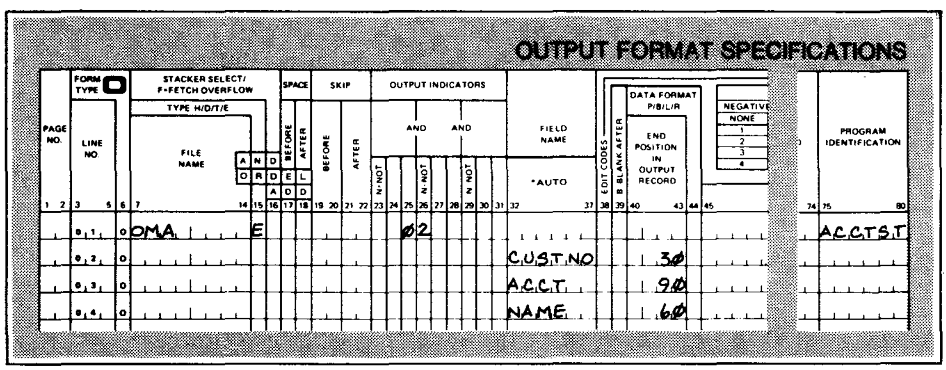


Figure 2–12. How Placement of Output Fields Can Cause Incorrect Message-Length Field

Effect of incorrect text length

When the output message goes to the terminal, only CUSTNO and NAME appear on the screen. IMS overlooks ACCT since the text-length size wasn't big enough. This happens even though the configured size of the output message area is large enough to hold the entire message. You control what goes to the terminal by the way that you list fields on the output form.

When program moves value to text-length

If you wish, you can move a value to the text-length field. This value should equal the actual size of your output message plus four characters for the text-length field itself. RPG II doesn't override this value no matter what you specify as the last entry on the output form.

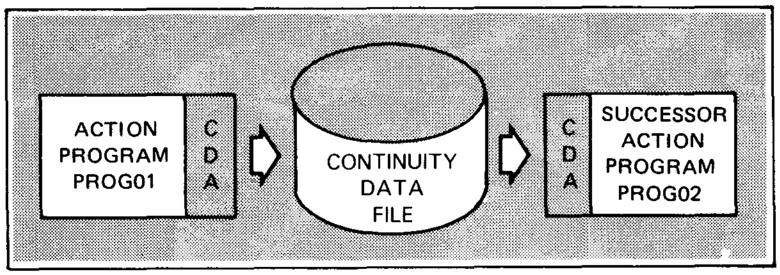
When text-length=0

When message-length is set to zeros, IMS puts out the message *TRANSACTION COMPLETE*.

2.15. DEFINING THE CONTINUITY DATA AREA (CDA)

Purpose

The continuity data area is used to pass data from one action program to its successor. IMS saves this area on disk at the termination of the predecessor action program and restores it at the start of the successor action program. You generally define a continuity data area when you want to pass data between action programs.



Size

Continuity data area size is specified at configuration. How you define it on the file description form depends on how your action program uses it (Table 2-9).

File description form entries

Table 2-9. Defining the Continuity Data Area According to How the Action Program Uses It

How Action Program Uses Continuity Data Area	File Type	File Designation
Saves Data Only	O	Blank
Reads and Updates Saved Data	U	P.S,D
Reads Saved Data Only	I	P,S,D

In 2.16 we'll consider an example where the continuity data area is used in the three ways described in Table 2-9.

2.16. HOW TO USE THE CONTINUITY DATA AREA TO PASS DATA

Example

Consider a case where there is a transaction that contains three action programs – PROG01, PROG02, and PROG03. When it terminates, PROG01 wants to pass data to PROG02. Figure 2-13 shows how you do it.

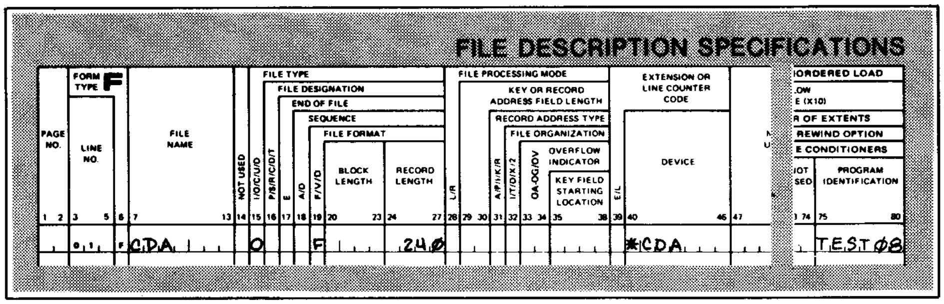
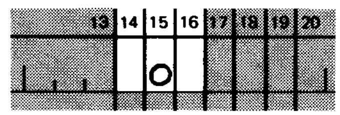


Figure 2-13. Defining the Continuity Data Area when It Saves Data Only

Assigning file name and type

The file name is CDA. In column 15, the entry is O for output file because at output we want to move data to the continuity data area.

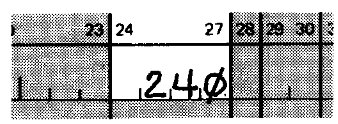


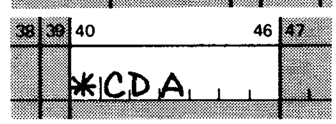
Continuity data file

When an action program terminates, in this case PROG01, any data in the continuity data area is moved to the continuity data file. It is saved there until the successor program is scheduled. In single-thread IMS, the continuity data file is AUDCONF; in multithread IMS, it is CONDATA.

Record length and device name

For record length we enter 240; this is the size of the saved data. In columns 40-46, the required entry is *CDA.





Now, consider Figure 2-14.

CONTINUITY DATA AREA CODING

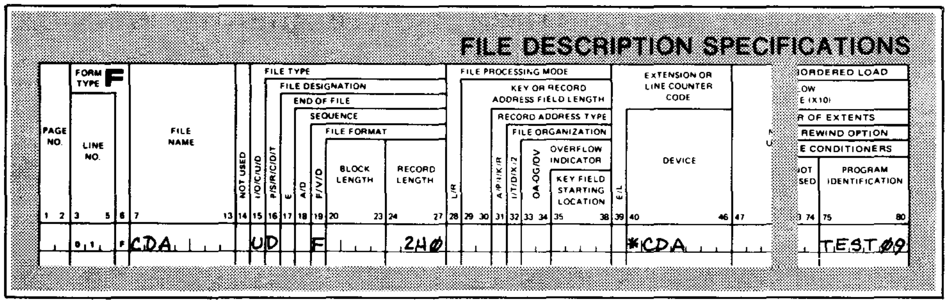


Figure 2-14. Defining the Continuity Data Area when It Reads and Updates Saved Data

Continuity data area as an update file

Figure 2-14 is the file description form coding for PROG02. PROG02 is the middle program in this series. It is designed to read the data saved by PROG01 and update it. Notice the continuity data area is defined as an update/demand file.

Passing data to PROG02

When IMS schedules PROG02, it moves the data saved by PROG01 from the continuity data file to the continuity data area of PROG02. Using the READ operation, this data becomes available to PROG02 for updating.

Passing data to final successor program

When PROG02 terminates, it passes data to its successor, PROG03. Figure 2-15 is the coding for the file description form for PROG03, the last program in the transaction.

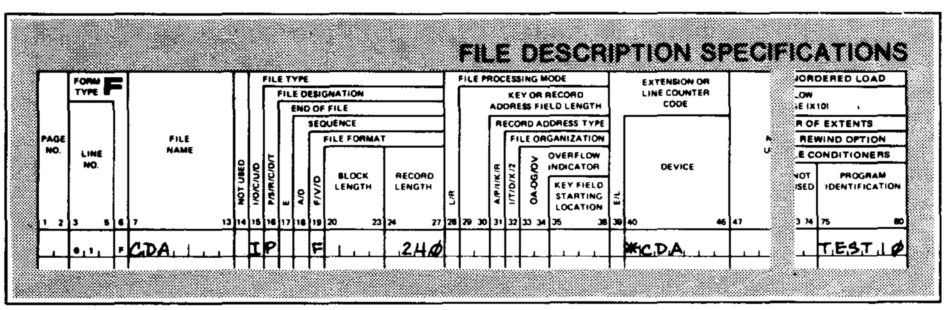


Figure 2-15. Defining the Continuity Data Area when It Reads Data Only

Continuity data area as an input file

Figure 2-15 defines the continuity data area as a primary input file. When IMS schedules PROG03, it moves the saved data of PROG02 from the continuity data file to the continuity data area of PROG03. As soon as PROG03 begins processing, RPG II begins reading this area. The transaction is complete when PROG03 terminates normally.

Normal flow of saved data

Continuity data file

In describing this transaction, we said the saved data went to the continuity data file. This point needs explanation. When an action program defines a continuity data area, any data saved by that program goes to that specific area. When the program terminates, the saved data is written to the continuity data file – AUDCONF in single-thread IMS; CONDATA in multithread IMS. When the successor program begins processing, IMS moves the saved data from the continuity data file to the continuity data area of the successor program.

Saved data flow in immediate succession

Only in immediate succession is this process different. Since all interface areas, including the continuity data area, remain intact between programs, the data stored there is not written to a continuity data file. It remains in main storage and is immediately available to the successor program when processing begins.

Other ways to save data

We might mention again at this point that you can also use the input and output message areas to pass data when specifying immediate succession (see 2.11). In addition, you can use the output message area to pass data when using delayed succcession since the output message area becomes the input message area of the successor program (see 2.13).

2.17. HOW TO VARY CONTINUITY DATA AREA SIZE TO SUIT AMOUNT OF DATA PASSED

Changing continuity-data -area-inc value

You may need to vary continuity data area size from one action program to another depending on the size of the data saved. You do this by changing the value of continuity-data-area-inc in the program information block. You can only increase the continuity data area size for the successor action program, not for the current program.

How IMS determines continuity data area size

IMS determines the continuity data area's size at the termination of each action program based on which length is larger:

- the CDA length specified at configuration; or
- the length specified in the continuity-data-area-inc field in the program information block plus the actual length of the data saved at the termination of the action program.

Increasing continuity -data-area size

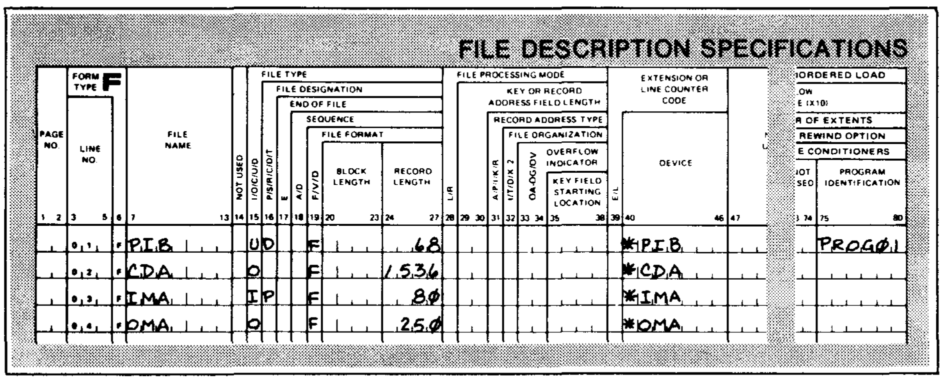
Let's consider once again a series of three action programs, PROGO1, PROGO2, and PROGO3. Assume that the configured continuity data area size is 1536 characters. The data you want to pass in PROGO1 is 1500 characters. You know that PROGO2 will be passing the same data plus additional data to PROGO3. Consequently, PROGO1 needs to increase the continuity data area size for PROGO2, the successor program. To do this, PROGO1 must have already defined the program information block as an update demand file on the file description form. On the output form, you specify an increment value for this field.

Consider Figures 2-16 and 2-17.

Updating the program information block

In Figure 2-16, we defined PIB as an update demand file since PROGO1 updates successor-id, termination-indicator, and for the purpose of this example, continuity-data-area-inc. Recall that you do not need to do a READ operation of the program information block to update it. Also, notice that the CDA is an output file with a configured size of 1536 characters.

Example



FILE DESCRIPTION SPECIFICATIONS

Line No. (3-5)	Form Type (6)	File Name (7-13)	File Type (15)	File Designation (16)	File Format (19)	Record Length (24-27)	Device (40-46)	Program Identification (75-80)
01	F	PIB	U	D	F	68	*PIB	PROGØ1
02	F	CDA	O		F	1536	*CDA	
03	F	IMA	I	P	F	8Ø	*IMA	
04	F	OMA	O		F	25Ø	*OMA	

Figure 2-16. Coding the File Description Form for Program PROGO1

Moving a value to continuity-data-area-inc

In Figure 2-17, we show the values output to the PIB file when PROGO1 terminates. 'PROGO2' is moved to successor-id (end position 10). 'I' is moved to termination-indicator (end position 11). The hexadecimal value '1F4' (500) is moved to continuity-data-area-inc.

Example

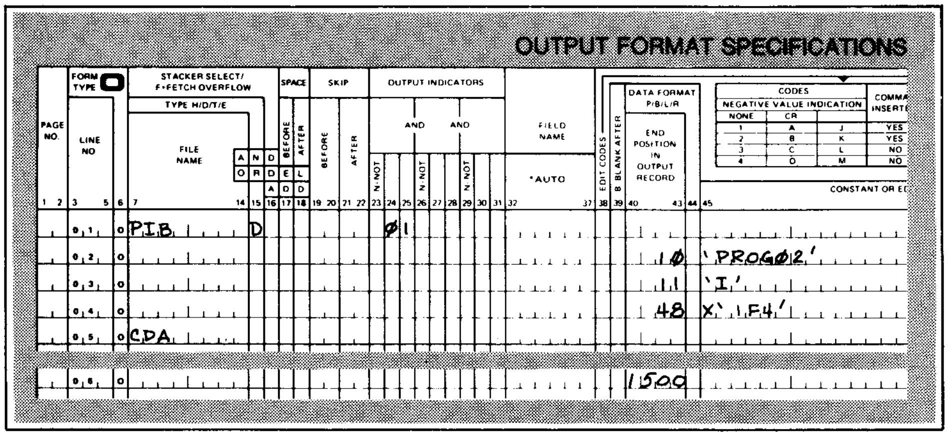


Figure 2-17. Coding the Output Form for Program PROG01

Computing continuity-data-area size for successor

At termination IMS examines all these fields. It compares the value of the continuity data length specified at configuration (1536) to the sum of the continuity-data-area-inc (500) plus the length of the data saved by PROG01 at termination. Since the saved data (1500 characters) plus 500 is larger than 1536, IMS increases the continuity data area size for PROG02 to 2000 characters.

Continuity-data-output-length and continuity-data-input-length

The actual length of the saved data is specified in the continuity-data-output-length field in the program information block of the current action program. When IMS schedules the successor program, this value is moved to continuity-data-input-length in the program information block of the successor program.

When continuity-data-area=0

When an action program terminates and the value in continuity-data-output-length is zero, no data is saved in the continuity data file.

3. Writing an Action Program

3.1. DIFFERENCES BETWEEN ACTION PROGRAMS AND NORMAL RPG II PROGRAMS

Using interface areas

In Section 2, we discussed rules for coding action programs. You'll recall that the major difference between action programs and a normal RPG II program is coding the interface areas. These areas are coded on the file description form. They handle all communication between IMS and the action program.

3.2. PURPOSE OF EXAMPLES

Scope of section

In this section, we present a series of action programming examples illustrating the coding principles described in Section 2. These examples are not complex and they emphasize the points you need to keep in mind when designing an action program. Let's summarize these points:

- **Entering a transaction code** signals the beginning of a transaction.

- **Action programs process** input messages and produce output messages.

- **Action programs depend upon IMS** for the handling of input and output messages.

Key features of action programs

- **Interface areas** – the program information block, input message area, output message area, and continuity data area – handle control data passing between your program and IMS. These areas are described in Section 2. How they are used is one of the major topics of this section.

- **To format output messages,** you have several options: screen format services; device independent control expressions (DICE); and, field control characters (FCCs). Using device independent expressions and field control characters is discussed in Appendix A. Screen format services is covered in Section 6.

3.3. HOW TRANSACTIONS ARE INITIATED

Entering a transaction code

A transaction begins when the operator enters a transaction code at the terminal. This code tells IMS what action program to schedule.

How action programs are scehduled

Each transaction code, and the action program that processes it, is specified at IMS configuration. Whenever a code is entered at a terminal, IMS checks the transaction table to determine if it's a valid code. IMS then checks to see what action program was configured to process this code. Once these steps are completed, if resources are available, IMS schedules the appropriate action program.

Example

In our example (Figure 3–1), when the operator keys in the word 'START', IMS checks the transaction table to verify the code and find the action program configured to process 'START'. The name of this program is RCMENU. If resources are available, IMS schedules RCMENU; if not, the transaction code START is queued until IMS can handle it.

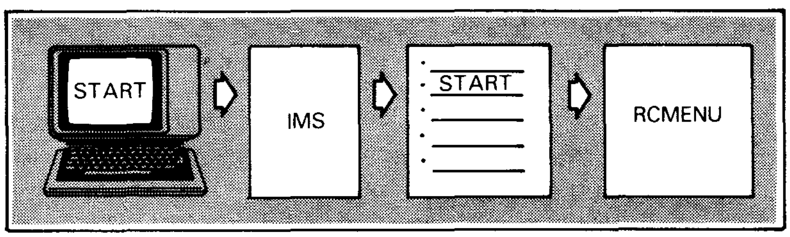


Figure 3–1. Transaction Code Initiates IMS Transaction

3.4. SAMPLE TRANSACTION (EXTERNAL SUCCESSION)

In this example, there are six action programs. The first program generates a menu. The other five programs allow a terminal operator to:

A sample transaction

- enter an order;
- update the customer file;
- update the order file;
- bill the customer; and
- terminate the transaction

Summary of processing

The first action program displays a menu on the terminal screen. The terminal operator selects the operation he wants to perform by entering the appropriate menu selection. The menu program validates the selection and displays a template on the terminal screen. The operator fills in the data requested and another action program uses the data to perform the requested operation, such as updating the customer file.

Programs RCMENU and RCCUST

We will describe the operation of two of the action programs, RCMENU and RCCUST. RCMENU displays the menu screen from which the terminal operator selects the operation (we assume 2 - CUSTOMER UPDATE is selected), and RCCUST updates the customer file. We will describe the operation in detail and show and explain the two action programs.

3.5. A DESCRIPTION OF WHAT THE SAMPLE TRANSACTION DOES

Structuring the transaction

Our sample transaction begins with the entry of the transaction code START at the terminal. The transaction consists of three actions. Therefore, there are three input messages entered at the terminal and three output messages generated by the action programs. Two programs process this transaction. They are RCMENU and RCCUST.

Execution of RCMENU

RCMENU is the first action program in this transaction. The transaction calls for two passes through this program, i.e., RCMENU will execute, be rescheduled, and execute a second time. Let's look at what happens in each pass.

RCMENU - Pass 1

On the first pass, RCMENU:

Processing on the first pass

1. Processes the input message coming from the terminal. On the first pass, the input message is the transaction code - START.

2. Creates an output message that is the menu screen.

3. Reschedules itself as successor program to validate the menu selection the terminal operator makes.

RCMENU – Pass 2

On the second pass, RCMENU:

Processing on the second pass

1. Processes the input message coming from the terminal. This time the input message is the number of the menu selection entered by the terminal operator. In our example, the selection made is 2 – CUSTOMER UPDATE.

2. Creates an output message that is the customer update screen. The screen generated relates to the menu selection made. In this case, it is a screen requesting data to update a customer account balance file.

3. Schedules the appropriate successor action program to process the data entered on the second output screen. In our example, the successor program is the customer update program RCCUST. If a different menu selection is made, RCMENU generates the appropriate screen as an output message and schedules the appropriate successor program to process it.

RCCUST

When RCMENU terminates after the second pass, RCCUST begins processing (we are assuming, of course, that the terminal operator chose menu selection 2). RCCUST:

Execution of RCCUST

1. Processes the data the terminal operator enters on the customer update screen generated as output by RCMENU on the second pass.

2. Computes a new balance for the customer account file.

3. Updates the customer account file.

4. Creates an output message containing the new customer balance to be sent to the terminal.

Figure 3-2 illustrates the processing for RCMENU and RCCUST.

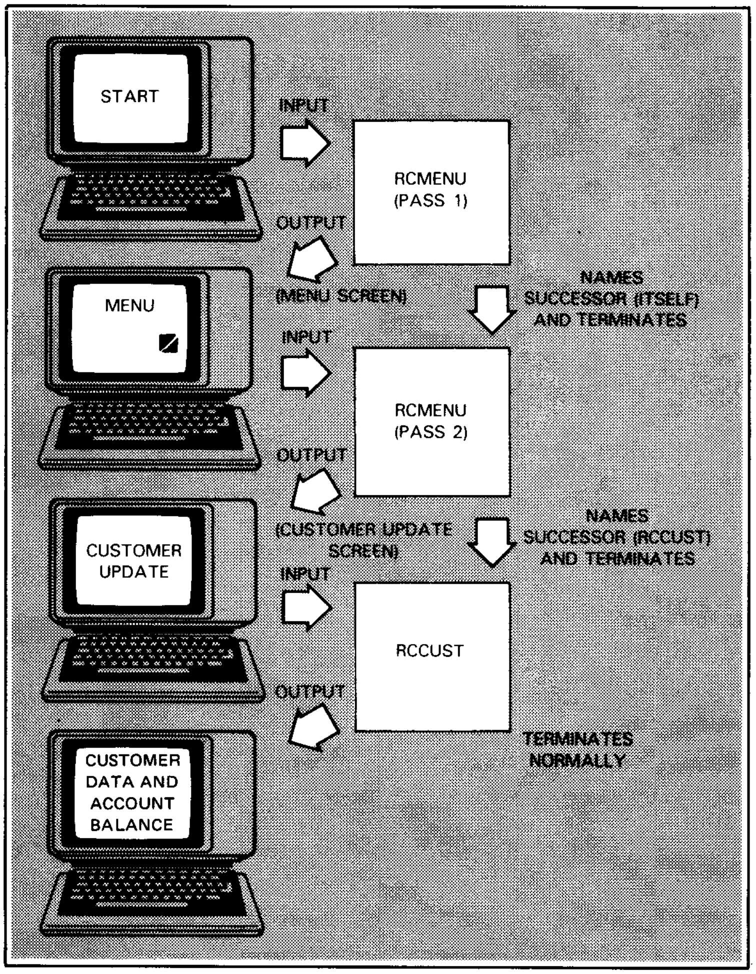


Figure 3-2. How RCMENU and RCCUST Process a Transaction

3.6. GENERAL OPERATION OF ACTION PROGRAMS

Action program design

Although the actual processing done by RCMENU and RCCUST differs somewhat, the activities involved are fundamentally the same. The terminal sends an input message. The action program processes it and generates an output message. The action program then schedules a successor program, if needed, and terminates.

Common characteristics

These activities are characteristic of action programming. Whether one or many action programs are involved, the basic design is the same. Action programs process input messages and generate output messages.

3.7. EXPLANATION OF THE CODING FOR RCMENU

With this general background, let's now look at the actual coding for this transaction, beginning with RCMENU. Figures 3-3 and 3-4 show compilation of the RCMENU and RCCUST action programs.

Formatting output

Note on the output form of both programs that a series of device independent control expressions and UTS 400 field control characters are used to format output messages sent to the terminal. To facilitate our discussion of the action programs themselves, we'll ignore these sequences for the time being. A discussion of device independent codes and field control characters can be found in Appendix A. Section 6 discusses how action programs use screen format services to format output messages.

```
          1   5 7 9   13   17   21   25   29   33   37   41   45   49   53   57   61   65   69   73   77
                                                  RCMENU
    UNIVAC OS/3 RPGII VERS 801007                 START                81/08/11 20.28          PAGE      1

                H                                                                          ARCMENU
                FIMA       IP  F      48             *IMA
                FOMA       O   F     500             *OMA
                FPIB       UD  F      70             *PIB
                IIMA       AA  20  17 CS  18 CT  19 CA
                I          AA  30  17 C1

                I          AA  40  17 C2

                I          AA  50  17 C3

                I          AA  60  17 C4

                I          AA  70  17 C5

                I          AA  99

                IPIB       NS  01

                I                                          5   10 SCRSIC
                I                                         11   11 TERMIC
                OOMA        D        20
                O         OR         99
                O                                        20 X'10UA0106'
                O                                        25 X'1F6CF96EF3'
                O                                        49 'MENU  SELECTION  PFOGRAM'
                O                                        54 X'1F50F96EF3'
                O                                        68 '1-ORDER  ENTRY'
                O                                        73 X'1F4EF96EF3'
                O                                        91 '2-CUSTOMER  UPDATE'
                O                                        96 X'1F60F96EF3'
                O                                       111 '3-ORDER  UPDATE'
                O                                       116 X'1F61F96EF3'
                O                                       125 '4-BILLING'
                O                                       130 X'1FF1F96EF3'
                O                                       136 '5-STOP'
                O                                       141 X'1FF4F96EF3'
                O                                       165 'ENTEP  YOUP  SELECTION  '
                O                                       170 X'1FF4D7F3F2'
                O                                       171 '_'
                O                                       176 X'1FF6F96EF3'
                O                                       197 'PLACE CURSOR HERE TO '
                O                                       206 'TRANSMIT '
                O                                       211 X'1FF6E6F3F0'
                O                                       212 '_'
                OOMA        D        40
                O                                        20 X'100A0106'
                O                                        25 X'1F6CF96EF3'
                O                                        42 'CUSTOMER  UPCATE '
                O                                        49 'PROGRAM'
                O                                        54 X'1F4DF96EF3'
                O                                        77 'ENTEP 5-DIGIT CUSTCMER '
                O                                        84 'NUMBER '
```

Figure 3–3. RCMENU Program (Part 1 of 2)

RCMENU CODING

```
 1   5 7 8   13   17   21   25   29   33   37   41   45   49   53   57   61   65   69   73   77 80
      UNIVAC OS/3 RPGII VERS 801007             START               81/08/11 20.28        PAGE     2

      046          0                                       89 X'1F4DE6F3F0'
      047          0                                       94 '_____'
      048          0                                       99 X'1F4DE06EF3'
      049          0                                      104 X'1F5CF96EF3'
      050          0                                      128 'ENTER + FOR PAYMENT MADE'
      051          0                                      133 X'1F4EF96EF3'
      052          0                                      157 'ENTER - FOR PAYMENT OWED'
      053          0                                      162 X'1F6BF96EF3'
      054          0                                      175 'ENTER + OR - '
      055          0                                      180 X'1F63C6F3F0'
      056          0                                      181 '_'
      057          0                                      186 X'1F6BC76EF3'
      058          0                                      191 X'1F4BF96EF3'
      059          0                                      204 'ENTER AMOUNT '
      060          0                                      209 X'1F4BC6F3F0'
      061          0                                      214 '_____'
      062          0          D          70
      063          0                                       14 X'0000'
      064          OPIB       D          20
      065          0       OR            99
      066          0                                       10 'RCMENU'
      067          0                                       11 'E'
      068          0          D          30
      069          0                                       10 'ORDENT'
      070          0                                       11 'E'
      071          0          D          40
      072          0                                       10 'RCCUST'
      073          0                                       11 'E'
      074          0          D          50
      075          0                                       10 'ORDRUP'
      076          0                                       11 'E'
      077          0          D          60
      078          0                                       10 'BILLS'
      079          0                                       11 'E'
      080          0          D          70
      081          0                                       11 'N'


                   PIB
                                          SYMBOL TABLES
 RESULTING INDICATORS

 ADDRESS RI     ADDRESS RI     ADDRESS RI     ADDRESS RI     ADDRESS RI     ADDRESS RI     ADDRESS RI

  000014 1P      000015 LR      000016 MR      000017 01      00002A 20      000034 30      00003E 40
  000048 50      000052 60      00005C 70      000079 99      00007A L0      000085 H0      000086 H1
  000087 H2      000088 H3      000089 H4      00008A H5      00008B H6      00008C H7      00008D H8
  00008E H9      00008F U1      000090 L2      000091 U3      000092 U4      000093 U5      000094 U6
  000095 U7      000096 U8

 FIELD NAMES

 ADDRESS FIELD        ADDRESS FIELD        ADDRESS FIELD        ADDRESS FIELD        ADDRESS FIELD
```

Figure 3–3. RCMENU Program (Part 2 of 2)

```
         1   5 7 9  13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
UNIVAC OS/3 RPGII VERS 801007         RCCUST                 81/08/11 20.30        PAGE      1

                    H                                                              ARCCUST
                    FIMA      IP  F 100 100            *IMA
                    FOMA      O   F 250 250            *OMA
                    FPIB      UD  F      70            *PIB
                    FCUSTFIL UC   F      80R SAI     1 DISK
                    IIMA      AA  01
                    I                                          17  21 CUST
                    I                                          22  22 SIGN           20
                    I                                          23  27 AMOUNT
                    ICUSTFIL BB   02
                    I                                           1   5 CUSTID
                    I                                           6  25 NAME
                    I                                          26  40 ADDR
                    I                                          41  55 CITY
                    I                                          56  60 ZIP
                    I                                        P 61  652BALDUE
                    IPIB      NS  01
                    I                                           5  10 SCRSID
                    I                                          11  11 TERMID
                    C             CUST       CHAINCUSTFIL             30
                    C   30                   GOTO END
                    C   N20       SIGN       COMP '+'                    41
                    C   N20N41    SIGN       COMP '-'                    42
                    C   N20N41N42            SETON                       50
                    C   50                   GOTO END
                    C                        TESTN          AMOUNT    61
                    C   N61                  SETON                    70
                    C   70                   GOTO END
                    C                        MOVE AMOUNT    AMT     52
                    C   20
                    COR 41        BALDUE     SUB  AMT       NEWBAL  92
                    C   42        BALDUE     ADD  AMT       NEWBAL
                    C             END        TAG
                    OOMA     D         N70N30N50
                    O        AND        01 02
                    O                                     20 X'100A0200'
                    O                                     30 'NAME     - '
                    O                           NAME      50
                    O                                     54 X'10010300'
                    O                                     64 'ADDRESS - '
                    O                           ADDR      79
                    O                                     83 X'100104C0'
                    O                                     93 'CITY-ST - '
                    O                           CITY     108
                    O                                    112 X'1001041E'
                    O                                    118 'ZIP - '
                    O                           ZIP      123
                    O                                    127 X'10040200'
                    O                                    141 'OLD BALANCE - '
                    O                           BALDUE   156 '  ,   , $0.  &-'
                    O                                    160 X'10040100'
                    O                                    174 'TRANSACTION - '
                    O                           AMT      183 '  $0.  -'
```

Figure 3-4. RCCUST Program (Part 1 of 2)

RCCUST CODING

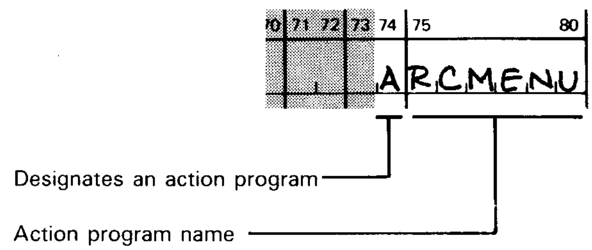
```
1   5 7 9  13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
    UNIVAC OS/3 RPGII VERS 801007          RCCUST               81/08/11 20.30      PAGE     2

  053              0                                  187 X'10040100'
  054              0                                  201 'NEW BALANCE - '
  055              0                         NEWBAL   219 ' ,    , $0.  &-'
  056              0                                  223 X'10010106'
  057              OCUSTFIL D        N70N30N50
  058              0        AND       01
  059              0                         NEWBAL    65P
  060              OOMA     D         30
  061              0                                   20 X'100A0200'
  062              0                                   39 'INVALID CUSTOMER ID'
  063              0                                   43 X'10010106'
  064              OOMA     D         50
  065              0                                   20 X'100A0200'
  066              0                                   32 'INVALID SIGN'
  067              0                                   36 X'10010106'
  068              OOMA     D         70
  069              0                                   20 X'100A0200'
  070              0                                   34 'INVALID AMOUNT'
  071              0                                   38 X'10010106'
  072              OPIB     D
  073              0                                   11 'A'


                   PIB                                                           NOTE  132
                                             SYMBOL TABLES

RESULTING INDICATORS

ADDRESS RI     ADDRESS RI     ADDRESS RI     ADDRESS RI     ADDRESS RI     ADDRESS RI     ADDRESS RI

 000014 1P      000015 LR      000016 00      000017 01      000018 02      00002A 20      000034 30
 00003F 41      000040 42      000048 50      000053 61      00005C 70      00007A L0      000085 H0
 000086 H1      000087 H2      000088 H3      000089 H4      00008A H5      00008B H6      00008C H7
 00008D H8      00008E H9      00008F U1      000090 U2      000091 U3      000092 U4      000093 U5
 000094 U6      000095 U7      000096 U8

FIELD NAMES

ADDRESS FIELD         ADDRESS FIELD         ADDRESS FIELD         ADDRESS FIELD         ADDRESS FIELD

0001B0  *ERROR        000210  CUST          000215  SIGN          000216  AMOUNT        000218  CUSTID
000220  NAME          000234  ADDR          000243  CITY          000252  ZIP           000257  BALDUE
00025C  SCRSID        000262  TERMID        00020C  END           000263  AMT           000266  NEWBAL


LITERALS

ADDRESS LITERAL                          ADDRESS LITERAL                          ADDRESS LITERAL

000268  +                                00026C  -                                00026D  X'100A0200'
000271  NAME     -                       00027B  X'10010300'                      00027F  ADDRESS -
000289  X'10010400'                      00028D  CITY-ST -                        000297  X'1001041E'
000298  ZIP -                            0002A1  X'10040200'                      0002A5  OLD BALANCE -
0002B3    -,---,--/.-- -                 0002C3  X'10040100'                      0002C7  TRANSACTION -
```

Figure 3-4. RCCUST Program (Part 2 of 2)

3.8. RCMENU – ASSIGNING A NAME TO THE PROGRAM

Every action program requires these entries on the control form:

Control form entries



3.9. RCMENU – DEFINING THE INTERFACE AREAS (IMA, OMA, and PIB)

Define only areas used

The file description form describes all interface areas your action program references. The action program defines only those areas it intends to use. We describe in detail the use of the interface areas in Section 2.

RCMENU uses three interface areas – the input message area (IMA), output message area (OMA), and program information block (PIB). Since RCMENU does no file processing, no user files are described; however, the interface areas are treated as files.

The following table describes the file description coding that defines the IMA, OMA, and PIB associated with RCMENU.

Defining the input message area (IMA)

Column	Entry	Description
7-13	IMA	User file name assigned to the input message area.
15-16	IP	Primary input file. As soon as IMS schedules RCMENU, and assigns its interface areas, it places all data entered at the terminal in RCMENU's input message area. When RCMENU begins executing, it immediately reads the data in the input message area into the program.
19	F	Required entry
24-27	48	This is the configured size of the input message area. You specify input message area size in the INSIZE parameter in the ACTION section of the IMS configuration. RCMENU isn't expecting a message larger than 48 characters. However, IMS does make allowances to accommodate larger messages.
40-46	*IMA	Required entry whenever defining the input message area.

RCMENU CODING DESCRIPTION

Defining the output message area (OMA)

Column	Entry	Description
7-13	OMA	User file name assigned to the output message area. You must define an output message area if the action program creates an output message. This area holds the output message that RCMENU creates.
15-16	O	Output file
19	F	Required entry
24-27	500	This is the maximum size of the output message RCMENU can generate. As coded, the program doesn't use all 500 characters. You specify output message area size in the OUTSIZE parameter in the ACTION section of the IMS configuration.
40-46	*OMA	Required entry when defining the output message area

Defining the program information block (PIB)

Column	Entry	Description
7-13	PIB	User file name assigned to the program information block. You only define this interface area if you intend to read it or read and update it in your action program. Whether or not you define it, RPG II checks the status and detailed status codes fields in the program information block after each I/O request and makes the values in these fields available to the action program. These codes inform the action program if the function request made to IMS was successful or not. If not, both the status- and detailed-status-code fields (1-4) in the program information block and *ERROR contain the reason for the failure.
15-16	UD	Update demand file. Since RCMENU updates the program information block, it must define it as an update demand file. At output, RCMENU moves values into the successor-id and termination-indicator fields. At action program termination, successor-id identifies to IMS the name of the successor action program. Termination-indicator identifies the type of termination for the current action program.
19	F	Required entry
24-27	70	This is the entire program information block area accessible to an action program. Other areas are for IMS use only. For a complete list of the program information block fields you can access in your program, see 2.5.
40-46	*PIB	Required entry whenever defining the program information block

3.10. CONTENTS OF MAIN STORAGE AFTER RCMENU IS SCHEDULED

When IMS schedules RCMENU, this is the way main storage looks. Notice in Figure 3-5 that the three interface areas defined by RCMENU are loaded with the action program.

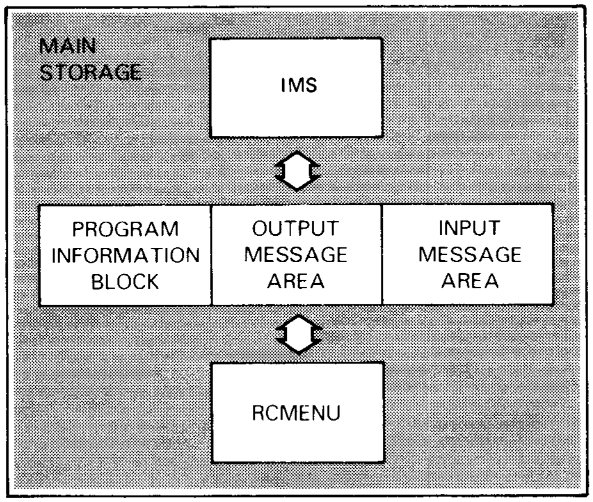


Figure 3-5. Main Storage when IMS Schedules RCMENU

3.11. HOW RCMENU USES THE INPUT MESSAGE AREA (PASS 1)

Reading the input message area

Only one input file is defined for RCMENU – IMA or input message area. When RCMENU begins executing, it reads the input message area. This area always contains a 16-character control header (see Table 2-10 for a description of the header) and the input message transmitted by the terminal operator. On the first pass through RCMENU, the input message is the word START. START is the transaction code that signals the beginning of the transaction and identifies to IMS the name of the first action program, RCMENU, to process this transaction.

Contents of the input message area – Pass 1

Once RCMENU reads the input message area, it compares positions 17, 18, and 19 to the characters S, T, and A. Remember to always allow positions 1-16 for the input message area header. Any input message (transaction code or other data) entered at the terminal always starts at position 17 or some position thereafter.

Characters match, RCMENU scheduled

In this example, the characters will match since S, T, and A are the first three letters of the transaction code that caused IMS to schedule RCMENU. When positions 17, 18, and 19 = S, T, A, indicator 20 is set on.

IMA contents

Figure 3-6 shows the contents of the input message area when RCMENU is scheduled.

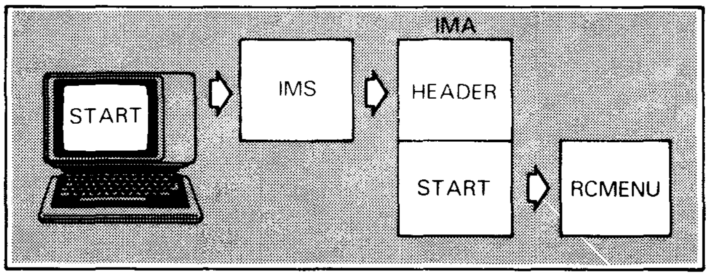


Figure 3-6. Contents of the Input Message Area – Pass 1

Menu screen sent to OMA

IMS handles I/O

Since there are no calculation specifications for this program, when indicator 20 is on, detail output is done. The output is the menu screen that goes to the output message area where it remains until RCMENU terminates. No output message generated by an action program, be it through exception, detail, or total time output, ever goes to the terminal before the program finishes all processing. IMS handles the actual input and output of messages.

Menu screen passed to terminal

In this example, when RCMENU generates the menu screen, processing is also complete. Consequently, the program terminates, rescheduling itself with external succession, and the menu screen is transmitted to the terminal.

Summary – RCMENU Pass 1

So, on the first pass RCMENU processes the transaction code START and produces a menu screen that IMS transmits to the terminal when RCMENU terminates.

3.12. HOW RCMENU USES THE INPUT MESSAGE AREA (PASS 2)

Processing operator menu choice

On the second pass through the program, position 17 of the input message area is matched to the character S. It doesn't match. The program then tries to match position 17 with the number 1,2,3,4, or 5. The numbers 1–5 represent possible menu choices the terminal operator can make.

Processing input message area contents – Pass 2

On the second pass, RCMENU is expecting one of these numbers in position 17 of the input message area. If the operator has followed directions correctly, this is what the program receives. If not, any other input entered from the terminal sets on indicator 99, which like indicator 20, retransmits the menu screen. The operator then has another chance to make the correct entry.

Figure 3–7 shows the contents of the input message area when the operator enters valid data.

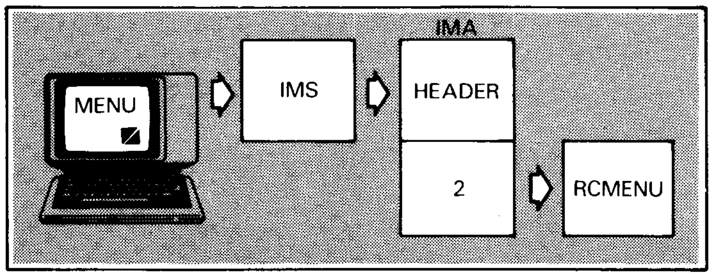


Figure 3–7. Contents of the Input Message Area – Pass 2

Indicator set on

When RPG II finds the number 1,2,3,4, or 5 in position 17 of the input message area, a specific indicator is set on and a specific type of detail output occurs. Once again, there are no calculations to be done. Table 3–1 summarizes the indicators set on and resulting output, based on the menu selection made.

Table 3–1. Indicators Set On During Second Pass through RCMENU and Resultant Output

Position 17	Indicator Set On	Output
S, T (18), A (19)	20	Menu screen
1	30	Order entry screen*
2	40	Customer update screen
3	50	Order update screen*
4	60	Billing screen*
5	70	Stop
None of the above	99	Menu screen

*Output coding not shown in example

3.13. HOW RCMENU USES THE OUTPUT MESSAGE AREA

RCMENU'S output at program termination

Output for action programs is defined the same as for any RPG II program, even the output message destined for the terminal. The important point, however, is that no output generated by an action program goes to the terminal until the program terminates.

Two output messages

Looking at the output form coding (Figure 3-3), you see that RCMENU generates two output messages destined for the terminal, one on each pass through the program.

Message formatting

All the hexadecimal sequences interspersed among the output fields format the message when it appears at the terminal. These sequences are discussed in Appendix A.

Generating the Output Message – Pass 1

Screen generated for Pass 1

Figure 3-8 shows the output message that goes to the terminal when RCMENU terminates after the first pass through the program:

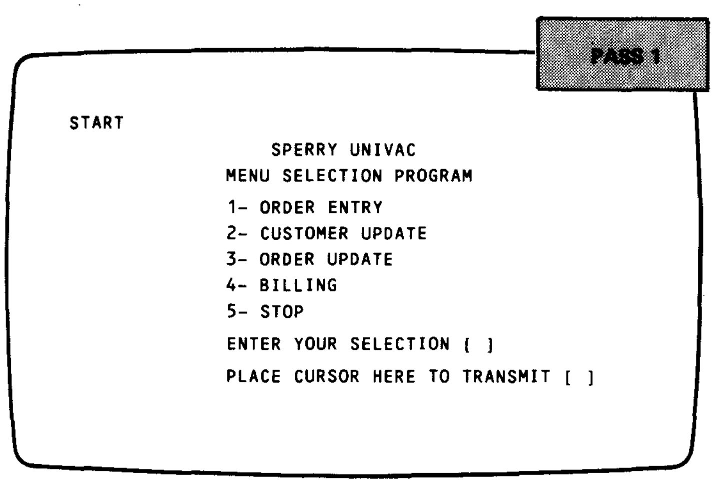


Figure 3-8. RCMENU's Output Message – Pass 1

Generating the Output Message – Pass 2

Screen generated for Pass 2

When the menu selection is 2-CUSTOMER UPDATE, indicator 40 is set on and RCMENU generates the output screen in Figure 3-9. This occurs on the second pass through RCMENU.

```
                                                        PASS 2
START
                  SPERRY  UNIVAC
             CUSTOMER UPDATE PROGRAM
             ENTER 5-DIGIT CUSTOMER NUMBER _____
             ENTER + FOR PAYMENT MADE
             ENTER - FOR PAYMENT OWED
             ENTER + OR - _
             ENTER AMOUNT _____
             PLACE CURSOR HERE TO TRANSMIT _
```

Figure 3-9. RCMENU's Output Message on Pass 2 for Menu Selection 2

Menu selections 1, 3, and 4

We have not included output message screens when indicators 30,50, and 60 are set on (menu selections 1,3, and 4). Such screens would be designed on the order of the customer update screen; however, they would request data relating to order entry (1), order updating (3), or billing (4).

When No Output Message is Generated

Ending the transaction

When indicator 70 is set on (menu selection 5), we move zeros into the text length field of the output message area. This causes IMS at program termination to send out a standard system message indicating that the IMS transaction is over. See Figure 3-10.

```
START
         TRANSACTION COMPLETE
```

Figure 3-10. RCMENU's Output Message when Menu Selection '5-STOP' Is Made

3.14. HOW RCMENU USES THE PROGRAM INFORMATION BLOCK

Updating the program information block

The only other output file described on the output form is the program information block. It shows what values RCMENU moves into successor-id and termination-indicator at output. Successor-id occupies positions 5–10 of the program information block and identifies to IMS the name of the successor action program. Termination-indicator occupies position 11 and indicates to IMS the type of termination for the current action program. The types of termination are normal, external, delayed, immediate, abnormal, and abnormal with snap dump. For more information on these termination types, see 1.4.

Defining the location of program information block fields

Whenever you define program information block fields in your action program, make sure that their beginning and end positions correspond exactly to their predefined location in the program information block. Table 2-6 defines these locations.

Indicating successor-id and termination type

Depending on what indicator is set on at output, the appropriate values are moved to successor-id and termination-indicator in the program information block. Table 3-2 summarizes the successor program name and termination type when a specific indicator is set on.

Table 3-2. Successor Programs and Type of Termination Corresponding to Each Indicator Set On

Indicator Set On	Successor Program	Type of Termination
20	RCMENU	External
30	ORDENT	External
40	RCCUST	External
50	ORDRUP	External
60	BILLS	External
70	No Successor	Normal (N)
99	RCMENU	External

IMS termination procedures

When output is complete, RCMENU terminates since there is no further processing to be done. IMS then checks the output message area and sends the message to the terminal. IMS also checks successor-id and termination-indicator to determine if further processing is required. When the terminal operator receives the output message and enters data to the screen, IMS then schedules the successor program to process it.

Determining successor program and type of termination

On the first pass through RCMENU, the successor is RCMENU. On the second pass, the successor corresponds to the menu selection made. In our example, the successor is RCCUST - the program that processes the customer update screen. RCMENU terminates with external succession. This means that IMS waits for an input message from the terminal before it schedules RCCUST. That input is the data entered by the terminal operator on the screen labeled SPERRY UNIVAC CUSTOMER UPDATE PROGRAM (Figure 3-9). When IMS receives the input message, it places it in a queue and schedules RCCUST as soon as resources are available.

3.15. EXPLANATION OF THE CODING FOR RCCUST

Earlier, we summarized what RCCUST does. To refresh your memory before examining the code, let's review its functions:

Processing for RCCUST

- **Accepts** input data entered on the customer update screen and validates it.
- **Computes** a new balance for the customer account.
- **Updates** the customer account file, CUSTFIL.
- **Creates** an output message to be sent to the terminal.

3.16. RCCUST – ASSIGNING A NAME TO THE PROGRAM

Control form entries

The control form entries are an **A** in column 74 and the program name in columns 75–80.

3.17. RCCUST – DEFINING THE INTERFACE AREAS (IMA, OMA, PIB)

Unique set of interface areas for RCMENU and RCCUST

The file description form defines the three interface areas and the one user file, CUSTFIL, referenced by RCCUST. The input message area (IMA) is defined as in RCMENU. The only difference is that the configured size is larger – 100 characters (columns 24–27) – to allow for a larger input message from the terminal. The output message area (OMA) and program information block (PIB) are defined exactly as they are in RCMENU. Remember, however, that although these areas are defined identically and that RCCUST directly follows RCMENU, RCCUST has its own unique interface areas assigned by IMS when the program is scheduled.

Using the same interface areas

The only time a successor program uses the same interface areas as the predecessor program is when I for immediate succession is specified in the termination-indicator field of the predecessor program.

User file – CUSTFIL

There is only one user file described for RCCUST, CUSTFIL. It is an indexed file that will be processed randomly using its 5-character key field.

3.18. DEFINING THE INPUT FIELDS

Defining input fields

The input form describes input fields for two files: the input message area (IMA) and the customer file, CUSTFIL. Like other RPG II programs, action programs only describe input fields they reference in the program.

Reading the input message area

When RCCUST begins executing, it reads the input message area. Indicator 01 is set on. The input message area contains the data entered by the terminal operator on the customer update screen. The fields defined as CUST, SIGN, and AMOUNT come into the program. These fields occupy positions 17 through 27 of the input message area. The first 16 positions contain the header. If the field SIGN contains a zero or a blank, indicator 20 is set on. Figure 3-11 shows the contents of the input message area when RCCUST begins processing.

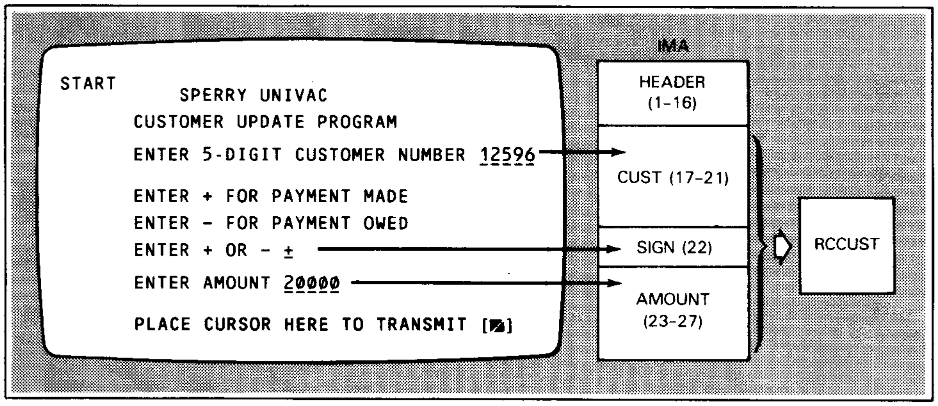


Figure 3-11. Input Message Coming into Program RCCUST

CUSTFIL fields

The input form also describes input fields for CUSTFIL. CUSTFIL is the user data file. Its key field CUSTID is a 5-character field that begins in position 1. The other five fields described for CUSTFIL occupy positions 6-65. Notice that the field BALDUE is a packed decimal field.

3.19. CALCULATIONS FOR RCCUST

Now let's look at the operations RCCUST performs.

Validating Input

Validating the customer number field

The 5-digit customer number entered as input at the terminal is used to chain into CUSTFIL. The customer number corresponds to the key field CUSTID in the input message area. If the number entered at the terminal doesn't match any of the keys in the index for CUSTFIL, indicator 30 is set on and detail output is done.

Validating the sign field

Next, RCCUST compares SIGN to '+' or '-'. If SIGN equals +, indicator 41 is set on. If SIGN equals - , indicator 42 is set on. If SIGN is not +, -, or blank (indicator 20 was set on), indicator 50 is set on and detail output is done.

Validating the amount field

Next, RCCUST tests AMOUNT to determine if it is numeric. If it is, indicator 60 is set on; if not, 70 is set on. When 70 is on, detail output is done. When AMOUNT is numeric (indicator 60 is set on), RCCUST moves AMOUNT to AMT, the result field.

Computing a New Account Balance

Once the input data is validated, the following calculations take place:

When a payment is made

When indicator 20 or 41 is on (SIGN = blank or +), AMT is subtracted from BALDUE. The result is NEWBAL. This means that the customer made a payment to his account. The SUB operation credits that amount to the customer's account and computes the new balance.

When a purchase is made

When indicator 42 is on (SIGN = -), AMT is added to BALDUE. The result is NEWBAL. This means that the customer made another purchase. The ADD operation adds the amount of the purchase to the existing balance and computes the new balance.

3.20. OUTPUT CODING FOR RCCUST

Output generated for RCCUST

Once calculations are complete, detail output occurs. Depending on what indicators are set on, RCCUST creates an output message. Table 3-3 shows the output message that goes to the terminal based on what indicators are set on.

Table 3-3. RCCUST Indicators Set On and Resulting Output

Output messages

Indicator Set On	Ouptut Message Created	Explanation
N70N30N50 01 02	NAME- SHANA GABRIEL ADDRESS- APPIAN WAY CITY-ST- GENEVA, OHIO 43727 OLD BALANCE - $586.25 TRANSACTION - $200.00 NEW BALANCE - $386.25	All data entered at the terminal was valid. In this case, the entry for SIGN was + indicating the customer made a $200.00 payment to her account. The SUB operation was performed and a new account balance computed.
30	INVALID CUSTOMER ID	The customer number entered at the terminal was invalid. It didn't match any of the keys in the index for CUSTFIL.
50	INVALID SIGN	The entry for SIGN wasn't +, - or blank.
70	INVALID AMOUNT	The entry for AMOUNT was either not numeric or was less than five digits. If the terminal operator entered more than five digits, RPG II truncates from the right.

Reinitiating the transaction

Line 054 repositions the cursor so that at the end of the transaction when the output message goes to the terminal, the cursor is at row 1, column 6. This positions it immediately after the word START, the transaction code, which is still displayed at the terminal. By simply pressing TRANSMIT, the transaction code START is retransmitted to IMS and the whole series begins again.

4. Writing a More Complex Action Program

4.1. GENERAL DESCRIPTION OF SAMPLE PROGRAM

More detailed examples

Now that we've developed some familiarity with the basic design of the action program in Section 3, we can study some more detailed examples. The structure of the action program discussed in this section is the same as before: it processes input messages and produces output messages. Now, however, the coding is somewhat more complex and introduces techniques that can be very useful to the applications programmer.

A sample transaction

As in the example discussed in Section 3, this transaction also begins with a menu program, JAMENU. Because of its similarity to the menu program described in detail in Section 3, we won't discuss JAMENU. Instead, we'll concentrate on its successor program, JAADD1. Since we've already given a good deal of attention to the basic coding of an action program in Section 3, we won't stress those same features here. Rather we'll concentrate on the new action programming tools it introduces and how they are used.

Let's begin by summarizing the new features JAADD1 introduces:

New features presented

- JAADDI uses the continuity data area to pass data between action programs.
- It also uses internal subroutines.
- It uses an error message file.
- And, it uses screen format services to format output messages.

4.2. A SUMMARY OF JAMENU'S PROCESSING

Figure 4-1 shows the output message screen JAMENU generates on the first pass through the program.

FIRST PASS

```
06/23/81      06:49:28          JA$MENU     02/09/81
                       ENTITLEMENT ACCOUNTING SYSTEM
SELECT ONE (1) OF THE FOLLOWING OPTIONS:
     1.  ADD A NEW CUSTOMER RECORD.
    *2.  UPDATE CUSTOMER NAME/ADDRESS INFORMATION.
    *3.  UPDATE BRANCH CUSTOMER INFORMATION.
    *4.  UPDATE CUSTOMER ENTITLEMENTS.
    *5.  DELETE A CUSTOMER RECORD.
    *6.  DISPLAY CUSTOMER INFORMATION.
     7.  LIST ALL ACCOUNTS (ON THE WORKSTATION).
     8.  ENTER WORKSTATION ACTIVITY RECORDS.
     9.  LOGOFF SYSTEM.
* INDICATES CUSTOMER NUMBER REQUIRED  ______
                        MENU SELECTION: __
    PLACE CURSOR TO TRANSMIT   [_]
```

Figure 4-1. Screen Generated by JAMENU

Processing the menu selection

Like RCMENU, JAMENU schedules itself as successor program and processes the menu selection entered on the screen. In our example, we assume the menu selection is *1. ADD A NEW CUSTOMER.* To process this menu selection, JAMENU moves the name JAADD1 to successor-id and I to termination-indicator. When JAMENU completes all processing, the program terminates. IMS checks the successor-id and termination-indicator fields and immediately schedules JAADD1.

4.3. A SUMMARY OF JAADD1, THE SAMPLE PROGRAM

The structure of the transaction

JAADD1 is the first of two action programs required to add a new customer and account record. JAADD1 validates the data used in the update. Its successor program, JAADD2 validates more data and does the actual file updating. We will discuss JAADD1 only since the two programs are very similar. However, we will include the coding for both programs to give you a fuller appreciation for the entire operation. The coding, output screen 1, and output screen 2 for JAADD1 are found in Figures 4-2, 4-3, and 4-4, respectively. Figure 4-5 is the coding for JAADD2.

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
ADD NEW CUSTOMER                                                 SOR    JAADD1
00001H                                                                  AJAADD1
00002F*----------------------------------------------------------------*JAADD1
00003F*    THIS PROGRAM IS THE 1ST OVERLAY TO BE CALLED IF THE MENU    *JAADD1
00004F*    SELECTION WAS TO ADD A CUSTOMER RECORD. THE 1ST TIME THRU,  *JAADD1
00005F*    THE 'JA$ADD1' SCREEN WILL BE PUT OUT AND THIS PROGRAM WILL  *JAADD1
00006F*    SUCCEED TO ITSELF. AFTER THE SCREEN IS TRANSMITTED BACK IN, *JAADD1
00007F*    THE CUSTOMER NUMBER AND ACCOUNT CODES ARE CHECKED FOR       *JAADD1
00008F*    VALIDITY (ASSUMING THE W/S OPERATOR DIDN'T WANT TO RETURN   *JAADD1
00009F*    TO THE MENU). IF THE DATA IS GOOD THE NEXT OVERLAY WILL     *JAADD1
00010F*    BE CALLED TO DO THE ACTUAL ADDING OF THE RECORDS.           *JAADD1
00011F*----------------------------------------------------------------*JAADD1
00012F*                                                                *JAADD1
00013F* I-  -C-------    F U N C T I O N   O F   I N D I C A T O R S   *JAADD1
00014F*                                                                *JAADD1
00015F*----------------------------------------------------------------*JAADD1
00016F* 01                 PROGRAM INFORMATION BLOCK                    JAADD1
00017F* 02                 INPUT MESSAGE AREA                           JAADD1
00018F* 03                 CONTINUITY DATA AREA                         JAADD1
00019F* 04                 CUSTOMER MASTER RECORD                       JAADD1
00020F* 05                 CUSTOMER RECORD DELETED                      JAADD1
00021F* 06                 ACCOUNT CROSS-REFFERENCE RECORD              JAADD1
00022F* 08                 SYSTEM CONTROL ERROR TEXT RECORD             JAADD1
00023F* 09                 ERROR TEXT RECORD DELETED                    JAADD1
00024F*       70           1ST TIME THRU PROGRAM (CALLED BY JAMENU)     JAADD1
00025F*       71           2ND TIME THRU PROGRAM                        JAADD1
00026F*       79           W/S OPERATOR CHOSE TO RETURN TO THE MENU     JAADD1
00027F*       80           GENERAL PURPOSE INDICATOR - LOCAL USAGE      JAADD1
00028F*       85           CUSTOMER NUMBER ZERO AND/OR ACCOUNT CODE BLANK JAADD1
00029F*       86           CUSTOMER NUMBER ALREADY EXISTS IN CUSTMST    JAADD1
00030F*       87           ACCOUNT CODE ALREADY EXISTS IN XREF1         JAADD1
00031F*       89           GENERAL ERROR INDICATOR                      JAADD1
00032F*       90           SYSTEM CONTROL RECORD NOT FOUND              JAADD1
00033F*                                                                 JAADD1
00034FPIB       UP  F       70          *PIB                            JAADD1
00035FIMA       IP  F      135          *IMA                            JAADD1
00036FOMA       O   F     4096          *OMA                            JAADD1
00037FCDA       UP  F      148          *CDA                            JAADD1
00038FCUSTMST IC    F 256  256R 6AI     9 DISC          S               JAADD1
00039FXREF1   IC    F  10   10R 4AI     7 DISC          S               JAADD1
00040FSYSCTL  IC    F  64   64R 6AI     1 DISC          S               JAADD1
00041IPIB       NS  01                                                  JAADD1
00042I                                          49  540PBDATE           JAADD1
00043I                                          55  600PBTIME           JAADD1
00044IIMA       NS  02                                                  JAADD1
00045I                                          17  20 IMACCT           JAADD1
00046I                                          21  260IMCUST           JAADD1
00047I                                          27  61 IMNAME           JAADD1
00048I                                          62  81 IMADR1           JAADD1
00049I                                          82 101 IMADR2           JAADD1
00050I                                         102 116 IMCITY           JAADD1
00051I                                         117 118 IMSTE            JAADD1
```

Figure 4–2. Action Program JAADD1 (Part 1 of 5)

JAADD1 CODING

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
00052I                                     119 1230IMZIP                  JAADD1
00053I                                     124 1260IMAREA                 JAADD1
00054I                                     127 1330IMPHON                 JAADD1
00055I                                     134 134 IMMENU                 JAADD1
00056I                                     135 135 IMXMIT                 JAADD1
00057ICDA      NS  07                                                     JAADD1
00058I                                       1   4 CDPSWD                 JAADD1
00059I                                       5  29 CDMSEL                 JAADD1
00060I                                      30  350CDCUST                 JAADD1
00061I                                      36  39 CDACCT                 JAADD1
00062I                                      40  400CDPASS                 JAADD1
00063I                                      41  41 CDSTAT                 JAADD1
00064I                                      42  47 CDCPGM                 JAADD1.
00065ICUSTMST  NS  04 256NCD                                              JAADD1
00066I       OR    05                                                     JAADD1
00067I                                       1   4 CMACCT                 JAADD1
00068IXREF1    NS  06                                                     JAADD1
00069I                                       1   60X1CUST                 JAADD1
00070ISYSCTL   NS  08  64NCD                                              JAADD1
00071I       OR    09                                                     JAADD1
00072I                                       7  56 SCERR                  JAADD1
00073C                     EXSR $ENTRY                   •WHICH PASS?     JAADD1
00074C   73                GOTO BUILD                    •1ST             JAADD1
00075C           IMMENU    COMP 'M'                    79•RETURN TO       JAADD1
00076C   79                GOTO RETURN                   •MENU?           JAADD1
00077C                     EXSR $CUST                    •CUST#/ACCT      JAADD1
00078C  N99                GOTO BUILD                    •VALID? YES      JAADD1
00079C                     EXSR $ERROR                   •NO GET MSG      JAADD1
00080C                     GOTO RETURN                                    JAADD1
00081C           BUILD     TAG                                            JAADD1
00082C                     READ PIB                                       JAADD1
00083C                     MOVE PBDATE    WRK6N                           JAADD1
00084C                     EXSR $REFDT                   •MAKE DATE MDY   JAADD1
00085C                     MOVE WRK6N     PBDATE                          JAADD1
00086C           RETURN    TAG                                            JAADD1
00087C                     EXCPT                                          JAADD1
00088C*                                                                   JAADD1
00089C* DEFINE WORK AREAS                                                 JAADD1
00090C*                                                                   JAADD1
00091CLRNLR                MOVE X'40'     WRK2    2                       JAADD1
00092CLRNLR                MOVE X'40'     WRK4    4                       JAADD1
00093CLRNLR                MOVE X'40'     WRK6    6                       JAADD1
00094CLRNLR                MOVE X'40'     WRK50  50                       JAADD1
00095CLRNLR                MOVE X'40'     BLNKS 256                       JAADD1
00096CLRNLR                MOVE X'F0'     WRK6N  60                       JAADD1
00097C*                                                                   JAADD1
00098C* CHECK CDA FOR NUMBER OF TIMES THRU THIS CODE                      JAADD1
00099C*                                                                   JAADD1
00100CSR         $ENTRY    BEGSR                                          JAADD1
00101CSR                   SETOF                     707189               JAADD1
00102CSR                   READ CDA                                       JAADD1
```

Figure 4–2. Action Program JAADD1 (Part 2 of 5)

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
00103CSR         CDPASS    COMP 0                        70.1ST TIME      JAADD1
00104CSR 70                MOVE 1         CDPASS                          JAADD1
00105CSR 70                GOTO $SENTEX                                   JAADD1
00106CSR         CDPASS    COMP 1                        71               JAADD1
00107CSR         $ENTEX    ENDSR                                          JAADD1
00108C*                                                                   JAADD1
00109C* CHECK CUSTOMER MASTER + ACCOUNT X-REFERENCE FOR DUPLICATES         JAADD1
00110C*                                                                   JAADD1
00111CSR         $CUST     BEGSR                                          JAADD1
00112CSR                   SETOF                    858687                JAADD1
00113CSR         IMACCT    COMP X'40'                    85.FIELDS        JAADD1
00114CSRN85      IMCUST    COMP 0                        85.MANDATORY     JAADD1
00115CSR 85                GOTO $CUSEX                                    JAADD1
00116CSR         IMCUST    CHAINCUSTMST             80    .DUPLICATE?     JAADD1
00117CSRN80                SETON                    86    .YES            JAADD1
00118CSR         IMACCT    CHAINXREF1               80    .ACCT CODE      JAADD1
00119CSRN80                SETON                    87    .DUPLICATE?     JAADD1
00120CSR         $CUSEX    TAG                                            JAADD1
00121CSR 85                                                               JAADD1
00122COR 86                                                               JAADD1
00123COR 87                SETON                    89    .ERROR?         JAADD1
00124CSR                   ENDSR                                          JAADD1
00125C*                                                                   JAADD1
00126C* GET ERROR MESSAGE FOR ERROR OVERLAY SCREEN                        JAADD1
00127C*                                                                   JAADD1
00128CSR         $ERROR    BEGSR                                          JAADD1
00129CSR                   MOVEL'EM'      WRK4                            JAADD1
00130CSR 85                MOVE '03'      WRK4                            JAADD1
00131CSR 86                MOVE '06'      WRK4                            JAADD1
00132CSR 87                MOVE '07'      WRK4                            JAADD1
00133CSR                   MOVELWRK4      WRK6                            JAADD1
00134CSR         WRK6      CHAINSYSCTL              90                    JAADD1
00135CSR                   MOVE X'0036'   OMTEXL  2       .TEXT LENGTH    JAADD1
00136CSR                   ENDSR                                          JAADD1
00137C*                                                                   JAADD1
00138C* REFORMAT DATE FIELD FROM YMD TO MDY                               JAADD1
00139C*                                                                   JAADD1
00140CSR         $REFDT    BEGSR                                          JAADD1
00141CSR                   MOVELWRK6N     WRK2            .SAVE YEAR      JAADD1
00142CSR         WRK6N     MULT 100       WRK6N           .SHIFT LEFT     JAADD1
00143CSR                   MOVE WRK2      WRK6N           .NOW MMDDYY     JAADD1
00144CSR                   ENDSR                                          JAADD1
00145C*                                                                   JAADD1
00146C* SOME ERROR HAS OCCURED - PUT OUT ERROR OVERLAY SCREEN             JAADD1
00147C*                                                                   JAADD1
00148OPIB      E         89                                               JAADD1
00149O                                   10 'JAMENU'                      JAADD1
00150O                                   11 'E'                           JAADD1
00151OOMA      E         89                                               JAADD1
00152O                                   K8 'JA$ERR'                      JAADD1
00153C* OVERRIDE TEXT LENGTH                                              JAADD1
```

Figure 4–2. Action Program JAADD1 (Part 3 of 5)

JAADD1 CODING

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
001540                          OMTEXL   14                                JAADD1
001550                        N90SCERR   66                                JAADD1
001560OCDA     E       89                                                  JAADD1
001570                                   40 '0'                            JAADD1
001580                                   41 X'FF'                          JAADD1
001590                                   47 'JAADD1'                       JAADD1
001600*                                                                    JAADD1
001610* 1ST PASS - CALL YOURSELF AND PUT OUT 1ST ADD SCREEN                JAADD1
001620*                                                                    JAADD1
001630OPIB     E       79                                                  JAADD1
001640                                   10 'JAADD1'                       JAADD1
001650                                   11 'E'                            JAADD1
001660OOMA     E       79                                                  JAADD1
001670                                   K8 'JA$ADD1'                      JAADD1
001680                          PBDATE   22                                JAADD1
001690                          PBTIME   28                                JAADD1
001700OCDA     E       79                                                  JAADD1
001710                          CDPASS   40                                JAADD1
001720                                   41 X'00'                          JAADD1
001730                                   47 'JAADD1'                       JAADD1
001740*                                                                    JAADD1
001750* I GUESS THE OPERATOR WANTS TO GO BACK TO THE MENU... OK, I WILL    JAADD1
001760*                                                                    JAADD1
001770OPIB     E       79                                                  JAADD1
001780                                   10 'JAMENU'                       JAADD1
001790                                   11 'I'                            JAADD1
001800OCDA     E       79                                                  JAADD1
001810                                   40 '0'                            JAADD1
001820                                   41 X'00'                          JAADD1
001830                                   47 'JAADD1'                       JAADD1
001840*                                                                    JAADD1
001850* EVERYTHING LOOKS OK - CALL NEXT OVERLAY                            JAADD1
001860*                                                                    JAADD1
001870OPIB     E       71N79N89                                            JAADD1
001880                                   10 'JAADD2'                       JAADD1
001890                                   11 'E'                            JAADD1
001900OOMA     E       71N79N89                                            JAADD1
001910                                   K8 'JA$ADD2'                      JAADD1
001920                          PBDATE   22                                JAADD1
001930                          PBTIME   28                                JAADD1
001940OCDA     E       71N79N89                                            JAADD1
001950                          IMCUST   35                                JAADD1
001960                          IMACCT   39                                JAADD1
001970                          CDPASS   40                                JAADD1
001980                                   41 X'00'                          JAADD1
001990                                   47 'JAADD1'                       JAADD1
002000                          IMNAME   82                                JAADD1
002010                          IMADR1  102                                JAADD1
002020                          IMADR2  122                                JAADD1
002030                          IMCITY  137                                JAADD1
002040                          IMSTE   139                                JAADD1
```

Figure 4–2. Action Program JAADD1 (Part 4 of 5)

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
002050                          IMZIP     142P                              JAADD1
002060                          IMAREA    144P                              JAADD1
002070                          IMPHON    148P                              JAADD1
```

Figure 4-2. Action Program JAADD1 (Part 5 of 5)

FIRST PASS

```
06/23/81     06:49:28              JA$ADD1     02/09/81
                 MENU SELECTION 1
         THIS SELECTION ADDS A CUSTOMER
         NAME AND ACCOUNT RECORDS.
ACCOUNT NUMBER:  ____
CUSTOMER NUMBER:  ______
NAME:  ____________________________________
ADDRESS (LINE 1):  ____________________
ADDRESS (LINE 2):  ____________________
CITY/STATE/ZIP:  _______________ __ _____
TELEPHONE NUMBER:  (___)___-_____
                        ENTER 'M' TO RETURN TO THE MENU:  _
                        PLACE CURSOR HERE TO TRANSMIT -->[_]
```

Figure 4-3. Output Generated by JAADD1 on First Pass

SECOND PASS

```
06/23/81    06:49:28        JA$ADD2     02/09/81
           MENU SELECTION 1
        THIS SELECTION ADDS CUSTOMER
        NAME AND ACCOUNT RECORDS
BRANCH NUMBER:
SALESMAN NUMBER:
PROJECT MANAGER:
ACCOUNT CONTACT:
                  DATES
CONTRACT    CONVERSION     PROPOSED      SYSTEM
 SIGNED      STARTED      COMPLETION    INSTALLED
__/__/__    __/__/__      __/__/__      __/__/__    __/__/__
                            PLACE CURSOR HERE TO TRANSMIT -->[_]
```

Figure 4-4. Output Generated by JAADD1 on Second Pass

JAADD2 CODING

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
ADD NEW CUSTOMER (PART 2)                                 SOP      JAADD2
00001H                                                                   AJAADD2
00002F*-----------------------------------------------------------------*JAADD2
00003F*   THIS PROGRAM IS CALLED BY JAADD1. IT TAKES THE DATA THAT WAS   *JAADD2
00004F*   ENTERED ON THE 'JA$ADD1' SCREEN FROM THE CDA PLUS ANY DATA     *JAADD2
00005F*   ENTERED ON THIS SCREEN ('JAADD2') AND ADDS A CUSTOMER MASTER   *JAADD2
00006F*   AND ACCOUNT CROSS REFERENCE RECORD. THE PROGRAM THEN CALLS     *JAADD2
00007F*   THE MENU OVERLAY.                                              *JAADD2
00008F*-----------------------------------------------------------------*JAADD2
00009F*                                                                  *JAADD2
00010F* I-  -C-------    F U N C T I O N   O F   I N D I C A T O R S    *JAADD2
00011F*                                                                  *JAADD2
00012F*-----------------------------------------------------------------*JAADD2
00013F* 01              PROGRAM INFORMATION BLOCK                         JAADD2
00014F* 02              INPUT MESSAGE AREA                                JAADD2
00015F* 03              CONTINUITY DATA AREA                              JAADD2
00016F* 04              CUSTOMER MASTER RECORD                            JAADD2
00017F* 05              CUSTOMER MASTER RECORD (DELETED)                  JAADD2
00018F* 06              ACCOUNT CROSS-REFERENCE RECORD                    JAADD2
00019F* 08              SYSTEM CONTROL RECORD                             JAADD2
00020F* 09              SYSTEM CONTROL RECORD (DELETED)                   JAADD2
00021F*     60          WRITE CUSTOMER MASTER RECORD                      JAADD2
00022F*     61          WRITE ACCOUNT CROSS-REFERENCE RECORD              JAADD2
00023F*     63          RETURN TO MENU AFTER ADDS                         JAADD2
00024F*     80          GENERAL PURPOSE INDICATOR - LOCAL USAGE           JAADD2
00025F*     86          CUSTOMER NUMBER ALREADY EXISTS IN CUSTMST         JAADD2
00026F*     87          ACCOUNT CODE ALREADY EXISTS IN XREF1              JAADD2
00027F*     89          GENERAL ERROR INDICATOR                           JAADD2
00028F*     90          SYSTEM CONTROL RECORD NOT FOUND                   JAADD2
00029F*                                                                   JAADD2
00030FPIB     UD  F      144          *PIB                                JAADD2
00031FIMA     IP  F      135          *IMA                                JAADD2
00032FOMA     O   F     4096          *OMA                                JAADD2
00033FCDA     UD  F      148          *CDA                                JAADD2
00034FCUSTMST UC  F 256  256R 6AI    9 DISC         S            A        JAADD2
00035FXREF1   UC  F  10   10R 4AI    7 DISC         S            A        JAADD2
00036FSYSCTL  IC  F  64   64R 6AI    1 DISC         S                     JAADD2
00037IPIB     NS  01                                                      JAADD2
00038I                                        49  540PBDATE               JAADD2
00039IIMA     NS  02                                                      JAADD2
00040I                                        17  200IMBRAN               JAADD2
00041I                                        21  260IMSLSM               JAADD2
00042I                                        27  51 IMPMGR               JAADD2
00043I                                        52  86 IMCONT               JAADD2
00044I                                        87  920IMSIGN               JAADD2
00045I                                        93  980IMCONV               JAADD2
00046I                                        99 1040IMCOMP               JAADD2
00047I                                       105 1100IMINST               JAADD2
00048I                                       111 1160IMRFU                JAADD2
00049I                                       117 117 IMXMIT               JAADD2
00050ICDA     NS  03                                                      JAADD2
00051I                                         1   4 CDPSWD               JAADD2
```

Figure 4–5. Action Program JAADD2 (Part 1 of 4)

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
00052I                                        5  29 CDMSEL                JAADD2
00053I                                       30  350CDCUST                JAADD2
00054I                                       36  39 CDACCT                JAADD2
00055I                                       40  400CDPASS                JAADD2
00056I                                       41  41 CDSTAT                JAADD2
00057I                                       42  47 CDCPGM                JAADD2
00058I* END OF STANDARD CDA FIELDS                                        JAADD2
00059I                                       48  82 CDNAME                JAADD2
00060I                                       83 102 CDADR1                JAADD2
00061I                                      103 122 CDADR2                JAADD2
00062I                                      123 137 CDCITY                JAADD2
00063I                                      138 139 CDSTE                 JAADD2
00064I                                    P 140 1420CDZIP                 JAADD2
00065I                                    P 143 1440CDAREA                JAADD2
00066I                                    P 145 1480CDPHON                JAADD2
00067ICUSTMST NS  04 256NCD                                               JAADD2
00068I        OR  05                                                      JAADD2
00069I                                        9  140CMCUST                JAADD2
00070IXREF1   NS  06                                                      JAADD2
00071I                                        1   6 XICUST                JAADD2
00072ISYSCTL  NS  08  64NCD                                               JAADD2
00073I        OR  09                                                      JAADD2
00074I                                        7  31 SCERR                 JAADD2
00075C                     READ PIB                                       JAADD2
00076C                     READ CDA                                       JAADD2
00077C                     EXSR $CUST                    •DUPLICATE       JAADD2
00078C   N60               GOTO ERROR                    •CUSTOMER#?      JAADD2
00079C                     EXSR $PUT                     •NO DMOUT        JAADD2
00080C                     EXSR $XREF1                   •NOW X-REF       JAADD2
00081C   N61               GOTO ERROR                                     JAADD2
00082C                     EXSR $PUT                                      JAADD2
00083C                     SETON                     63  •CALL MENU       JAADD2
00084C                     GOTO RETURN                                    JAADD2
00085C           ERROR     TAG                                            JAADD2
00086C                     SETON                     89                   JAADD2
00087C                     EXSR $ERROR                   •GET TEXT        JAADD2
00088C           RETURN    TAG                                            JAADD2
00089C                     EXCPT                                          JAADD2
00090C*                                                                   JAADD2
00091C* DEFINE WORK AREAS                                                 JAADD2
00092C*                                                                   JAADD2
00093CLRNLR                MOVE X'40'     WRK2    2                       JAADD2
00094CLRNLR                MOVE X'40'     WRK4    4                       JAADD2
00095CLRNLR                MOVE X'40'     WRK6    6                       JAADD2
00096CLRNLR                MOVE X'40'     BLNKS 256                       JAADD2
00097CLRNLR                MOVE X'F0'     WRK5N  50                       JAADD2
00098CLRNLR                MOVE X'F0'     WRK6N  60                       JAADD2
00099C*                                                                   JAADD2
00100C* CHECK CUSTOMER MASTER + ACCOUNT X-REFERENCE FOR DUPLICATES        JAADD2
00101C*                                                                   JAADD2
00102CSR         $CUST     BEGSR                                          JAADD2
```

Figure 4–5. Action Program JAADD2 (Part 2 of 4)

JAADD2 CODING

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
00103CSR         CDCUST    CHAINCUSTMST                 60                JAADD2
00104CSRN60                SETON                        86                JAADD2
00105CSR                   ENDSR                                          JAADD2
00106C*                                                                   JAADD2
00107CSR         $XREF1    BEGSR                                          JAADD2
00108CSR         CDACCT    CHAINXREF1                   61                JAADD2
00109CSRN61                SETON                        87                JAADD2
00110CSR                   ENDSR                                          JAADD2
00111C*                                                                   JAADD2
00112C* ADD CUSTOMER OR ACCOUNT CROSS-REFERENCE RECORD                    JAADD2
00113C*                                                                   JAADD2
00114CSR         $PUT      BEGSR                                          JAADD2
00115CSR                   EXCPT                                          JAADD2
00116CSR                   SETOF                        6061              JAADD2
00117CSR                   ENDSR                                          JAADD2
00118C*                                                                   JAADD2
00119C* GET ERROR MESSAGE FOR ERROR OVERLAY SCREEN                        JAADD2
00120C*                                                                   JAADD2
00121CSR         $ERROR    BEGSR                                          JAADD2
00122CSR                   MOVEL'EM'      WRK4                            JAADD2
00123CSR 86                MOVE '06'      WRK4                            JAADD2
00124CSR 87                MOVE '07'      WRK4                            JAADD2
00125CSR                   MOVELWRK4      WRK6                            JAADD2
00126CSR         WRK6      CHAINSYSCTL                   90               JAADD2
00127CSR                   MOVE X'0036'   OMTEXL  2          .TEXT LENGTH JAADD2
00128CSR                   ENDSR                                          JAADD2
001290*                                                                   JAADD2
001300* SOME ERROR HAS OCCURED - PUT OUT ERROR OVERLAY SCREEN             JAADD2
001310*                                                                   JAADD2
001320PIB      E         89                                               JAADD2
001330                                    10 'JAMENU'                     JAADD2
001340                                    11 'F'                          JAADD2
0013500MA      E         89                                               JAADD2
001360                                    K8 'JA$ERR'                     JAADD2
001370* OVERRIDE TEXT LENGTH                                              JAADD2
001380                        OMTEXL      14                              JAADD2
001390                      N90USCERR     66                              JAADD2
001400CDA      E         89                                               JAADD2
001410                                    40 '0'                          JAADD2
001420                                    41 X'FF'                        JAADD2
001430                                    47 'JAADD2'                     JAADD2
00144C*                                                                   JAADD2
001450* ADD CUSTOMER MASTER + X-REFERENCE RECORDS                         JAADD2
001460*                                                                   JAADD2
001470CUSTMST EADD       60                                               JAADD2
001480                        CDACCT       4                              JAADD2
001490                        IMPRAN       6                              JAADD2
001500                        CDCUST      14                              JAADD2
001510                        CDNAME      49                              JAADD2
001520                        CDADR1      69                              JAADD2
001530                        CDADR2      89                              JAADD2
```

Figure 4-5. Action Program JAADD2 (Part 3 of 4)

```
1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
001540                         CDCITY   104                               JAADD2
001550                         CDSTE    106                               JAADD2
001560                         CDZIP    109P                              JAADD2
001570                         CDAREA   111P                              JAADD2
001580                         CDPHON   115P                              JAADD2
001590                         IMCONT   150                               JAADD2
001600* POSITION 151 UNUSED                                               JAADD2
001610                         IMSLSM   155P                              JAADD2
001620                         IMPMGR   180                               JAADD2
001630                         IMSIGN   184P                              JAADD2
001640                         IMCONV   188P                              JAADD2
001650                         IMCOMP   192P                              JAADD2
001660                         IMINST   196P                              JAADD2
001670                         IMRFU    200P                              JAADD2
001680* ENTITLEMENTS                                                      JAADD2
001690                         WRK5N    203P                              JAADD2
001700                         WRK5N    206P                              JAADD2
001710                         WRK5N    209P                              JAADD2
001720                         WRK5N    212P                              JAADD2
001730                         WRK5N    215P                              JAADD2
001740                         WRK5N    218P                              JAADD2
001750                         WRK5N    221P                              JAADD2
001760                         WRK5N    224P                              JAADD2
001770                         WRK5N    227P                              JAADD2
001780                         WRK5N    230P                              JAADD2
001790                         WRK5N    233P                              JAADD2
001800                         WRK5N    236P                              JAADD2
001810* POSITIONS 237 - 251 UNUSED                                        JAADD2
001820                         PBDATE   255P                              JAADD2
001830                                  256 'A'                           JAADD2
001840XFEF1   EADD     61                                                 JAADD2
001850                         CDCUST     6                               JAADD2
001860                         CDACCT    10                               JAADD2
001870*                                                                   JAADD2
001880* RETURN TO THE MENU AFTER ADD                                      JAADD2
001890*                                                                   JAADD2
001900PIB     E        63                                                 JAADD2
001910                                   10 'JAMENU'                      JAADD2
001920                                   11 'I'                           JAADD2
001930CDA     E        63                                                 JAADD2
001940                                   40 '0'                           JAADD2
001950                                   41 X'00'                         JAADD2
001960                                   47 'JAADD2'                      JAADD2
```

Figure 4–5. Action Program JAADD2 (Part 4 of 4)

There are two passes through JAADD1. Let's summarize what happens in each pass.

Processing for the first pass

JAADD1 – Pass 1:

1. Reads data saved by JAMENU.
2. Reads the program information block for data.
3. Calls screen format services to create the output message screen, JA$ADD1.
4. Schedules itself as successor program.

Processing for the second pass

JAADD1 – Pass 2:

1. Reads data entered on the JA$ADD1 screen.
2. Reads data saved by JAADD1 on the first pass through the program.
3. Validates data entered on the JA$ADD1 screen and diagnoses errors.
4. Calls on screen format services to create the output message screen, JA$ADD2.
5. Schedules JAADD2 as successor program and passes data to it to do the actual adding of the customer and account records.

Designing IMS transactions

Once again you see the same basic design that we saw in Section 3 – a series of action programs all handling input, processing, and generating output. Perhaps you've also noticed that the action programs we're discussing are designed to accomplish one or two fundamental activities. It's better to link a series of action programs together to accomplish many small tasks than it is to try to incorporate all these tasks into a single program.

Objectives of IMS

In most user environments IMS is chosen for its interactive capabilities and fast throughput. To maintain speed and a conversational atmosphere, design your action programs to perform clearly defined tasks and to yield appropriate and quick responses.

You'll see that these goals of speed and conversational atmosphere are at the forefront in the design of all action programs presented in this manual.

4.4. USING THE CONTINUITY DATA AREA

Now let's focus our attention on the new features of action programming that JAADD1 introduces (Figure 4-2).

File Description Form (CDA)

JAADD1 uses four interface areas. We've already shown you ways to define the program information block, input message area, and output message area in Section 2; and in Section 3, we demonstrated how these areas are used. The use of the continuity data area, however, is new.

Purpose of continuity data area

An action program defines a continuity data area in order to read and/or update data saved there by the predecessor program or to pass data itself to a successor program. JAADD1 uses the continuity data area to read data saved by JAMENU, to update it, and to pass the updated data to the successor program.

Here is a description of how JAADD1 defines the continuity data area in order to use it in the ways we just described:

Defining the continuity data area

Column	Entry	Description
7-13	CDA	User name assigned to the continuity data area
15-16	UD	Update demand file. Since JAADD1 intends to read the continuity data area to get data passed by JAMENU and to update it, it must define it as an update demand file. There are many other ways to define the continuity data area depending on how you intend to use it. See 2.15 for a detailed discussion of the entries you can make in columns 15 and 16.
19	F	Required entry
24-27	148	This is the configured size of the continuity data area for JAADD1. JAADD1 can pass 148 characters of data to its successor program. An action program can increase the size of the continuity data area of the successor program by moving a new value into the field continuity-data-area-inc in the program information block at output time. See Section 2.
40-46	*CDA	Required entry whenever defining the continuity data area

Input Form Coding (CDA)

Seven fields contain data passed by JAMENU

As you would expect, the input form describes all input fields referenced by JAADD1. Notice, however, that for the continuity data area there are seven defined fields. They contain data passed by JAMENU.

Data passed by JAMENU

Table 4-1 lists the continuity data area fields passed by JAMENU to JAADD1 and their contents when JAADD1 begins processing.

Table 4-1. JAADD1 Continuity Data Area

CDA contents

Position	Field Name	Contents
1-4	CDPSWD	The transaction code that initiated the IMS transaction.
5-29	CDMSEL	The menu selection made by the terminal operator.
30-35	CDCUST	When JAADD1 begins executing, this field contains only zeros. JAADD1 uses this field on the second pass through the program.
36-39	CDACCT	When JAADD1 begins executing, this field contains only zeros. JAADD1 uses this field on the second pass through the program.
40-40	CDPASS	JAADD1 uses this field to determine which pass it is through the program.
41-41	CDSTAT	This field contains a zero when there is no error condition.
42-47	CDCPGM	This field contains the name of the current action program.

Defining the program information block input fields

The input form also defines fields for the program information block (PIB) and input message area (IMA). The two program information block fields defined correspond to transaction-date and time-of-day. For a complete listing of program information block fields, see Section 2. The fields defined for the input message area correspond to data entered on the JA$ADD1 screen and enter the program on the second pass.

Calculation Form (CDA)

Using $ENTRY to read CDA

Determining which pass through JAADD1

When JAADD1 begins processing, it calls upon subroutine $ENTRY. This subroutine reads the continuity data area. The continuity data area contains data saved by JAMENU. The purpose of reading the continuity data area first is to determine whether it is the first or second pass through the program. This information is contained in the field CDPASS. On the basis of whether CDPASS contains a zero (first pass) or 1 (second pass), all processing is determined.

Using the continuity data area to control processing

When CDPASS=0, indicator 70 is set on and 1 is moved to the field CDPASS. When CDPASS=1 initially, indicator 71 is set on. Indicator 70 triggers processing for the first pass through the program. Indicator 71 triggers processing for the second pass. The continuity data area is not used again until output is done.

Output Form (CDA)

Table 4-2 summarizes how JAADD1 updates the continuity data area when output occurs. All data saved in the continuity data area is passed to the successor program.

Updating the continuity data area at output

Table 4-2. Summary of JAADD1 Continuity Data Area Update at Output

Indicator Set On	Data Saved in Continuity Data Area	Description
70	CDPASS=1	Pass 1 through JAADD1 is complete.
	CDSTAT=0	No error condition occurred.
	CDCPGM=JAADD1	Name of the current program
71	CDPASS=0	Pass 2 through JAADD1 is complete. CDPASS is reinitialized to zero since all RPG II action programs are serially reusable.
	CDSTAT=0	No error condition occurred.
	CDCPGM=JAADD1	Same as for indicator 70
	All fields between lines 195 and 207 are written.	These fields contain the data entered on the JA$ADD1 screen and validated on the second pass through JAADD1. Notice that the location of IMCUST and IMACCT correspond to CDCUST and CDACCT described on the input form. This data is used in the updating of the CUSTMST and XREF1 files in program JAADD2.
79	CDPASS=0	Indicator 79 signifies the operator entered M on the JA$ADD1 screen. This means the operator wants to return to the menu. Consequently, CDPASS must be reinitialized to zero.

4.5. USING INTERNAL SUBROUTINES

Avoid repetitious code

We already briefly touched upon JAADD1's use of internal subroutines when we discussed $ENTRY. Using internal subroutines is a common tool of most RPG II programmers. It avoids tedious repetition of code. Action programs code internal subroutines in the same way as other RPG II programs.

JAADD1 uses four internal subroutines in all. We discussed $ENTRY, which reads the continuity data area and determines which pass it is through the program. The other three subroutines are $CUST, $ERROR, and $REFDT. Let's start with the last one first.

Subroutine $REFDT

Reading the program information block for data

Before talking about subroutine $REFDT, let's establish some necessary background information. In all the action programs we've discussed so far, we defined the program information block (PIB) as an update demand file on the file description form. We did this to move values into successor-id and termination-indicator when doing output. Other than that, the programs didn't use the program information block. JAADD1, however, does. That explains why the program information block is also defined on the input form. JAADD1 references the fields PBDATE and PBTIME. These fields correspond to transaction-date (positions 49–54) and time-of-day (positions 55–60) in the program information block.

Defining program information block size

On lines 082–084 of Figure 4-2, JAADD1 reads the program information block. This brings all program information block fields into the program. The reason all 70 positions of the program information block become available to JAADD1 is because they were defined on the file description form in record length.

Executing $REFDT

Now JAADD1 moves PBDATE to a field called WRK6N and executes subroutine $REFDT.

Reformatting a field

The purpose of this subroutine is to reformat transaction-date. Its present format in the program information block is yymmdd. The $REFDT subroutine moves the two leftmost characters (yy) in WRK6N to WRK2 (a 2-position field). It then multiplies WRK6N (containing mmdd) by 100 producing a result field mmdd00. The $REFDT subroutine then moves WRK2N (containing yy) back to WRK6N. The result is a reformatted date, mmddyy.

PIB is useful

There is nothing particularly unique about this subroutine. The reason we presented it is to point out that there is much data in the program information block that action programs can put to very good use. This was simply one example.

Subroutine $CUST

The second internal subroutine $CUST validates the data entered on the JA$ADD1 screen. Due to the conversational nature of IMS, there is a continual exchange of data taking place betweeen IMS and the terminal. As a result, there must be a means for checking the validity of the data the action program receives. Screen format services provides a certain amount of validation of terminal operator entries. However, if you aren't using screen format services or if your application requires special validation procedures, the action program must do it. JAADD1 uses the subroutine $CUST to do this. This subroutine executes only during the second pass through the program (when indicator 71 is set on).

Validating data

Use screen formats or action program

First, the values entered (at the terminal) in fields IMCUST and IMACCT are compared to zeros. If they don't contain zeros, the value IMCUST is checked against the index for user file CUSTMST, and the value IMACCT against the index for file XREF1. If no key is found for either value, processing continues. Otherwise, if IMCUST or IMACCT are zeros or if a key already exists with the same value as IMCUST or IMACCT, then indicators 85,86, or 87 are set on accordingly. Each of these indicators in turn sets on indicator 89, the general error indicator.

When data is invalid

Subroutine $ERROR

When errors occur

When indicator 89 is set on, before output takes place, a third internal subroutine takes control; it is $ERROR. Again, here is a little background information before discussing this subroutine.

Used to send error messages to terminal operator

Notice that on the file description form (line 040) we defined a user file, SYSCTL. This MIRAM file contains a series of user-created error messages to be sent to the terminal operator at program termination when an error condition occurs. In this way, terminal operators are kept aware of the status of their requests. The internal subroutine, $ERROR, uses the SYSCTL file.

4.6. USING AN ERROR MESSAGE FILE

Creating a user error file

When indicator 89 is set on, $ERROR takes control. Depending on which specific error indicator is set on (85,86,87), RPG II creates a key that is used to chain into the SYSCTL file. This file contains error messages related to specific errors that can occur during JAADD1's processing cycle.

Table 4-3 summarizes the error indicators that can be set on when JAADD1 is executing, the key that $ERROR creates, and the error message that goes to the terminal when the program terminates:

Table 4-3. Summary of Error Indicator and Error Messages for JAADD1

Error messages generated

Indicator Set On	Key	Error Message
85	EM0300	CUSTOMER NUMBER ZERO AND/OR ACCOUNT CODE BLANK. PLEASE ENTER AGAIN.
86	EM0600	CUSTOMER NUMBER ALREADY EXISTS IN CUSTOMER MASTER FILE. PLEASE ENTER AGAIN.
87	EM0700	ACCOUNT CODE ALREADY EXISTS IN X-REFERENCE FILE. PLEASE ENTER AGAIN.

Selecting error message

As we mentioned earlier, indicators 85, 86, and 87 all set on indicator 89. When output is done for indicator 89 (general error indicator), the error message identified by the $ERROR subroutine (Table 4-3) is sent to the terminal. These messages make it easy for the terminal operator to see the cause of the error and to correct the mistake and try again.

4.7. USING SCREEN FORMAT SERVICES

We have now talked about using the continuity data area, internal subroutines, an error file, and displaying error messages at the terminal. That leaves one other feature of JAADD1 to discuss – using screen format services.

No DICE or FCCs required

You've probably noticed that the output coding for JAADD1 contains none of the hexadecimal sequences so prevalent in RCMENU and RCCUST. JAADD1 formats all its output screens using screen format services. This is by far the easist way to format your output messages. The coding required is minimal.

Coding needed to build screens

Screen control

Lines 151–152, 166–167, 190–191 show the coding needed to build three different screens in the output message area. Which screen is built depends on which indicator is set on. When indicator 89 is set on, the error screen JA$ERR is built. When indicator 70 is set on, JA$ADD1 is built, and when indicator 71 is set on, JA$ADD2 is built. Figures 4–3 and 4–4 show the screens JA$ADD1 and JA$ADD2. Figure 4–6 illustrates a typical error screen when indicator 89 is set on.

```
06/23/81     06:49:28                JA$ADD1    02/09/81
                   MENU SELECTION 1
       THIS SELECTION ADDS A CUSTOMER
       NAME AND ACCOUNT RECORDS.
ACCOUNT NUMBER:  ____
CUSTOMER NUMBER:  ______
NAME:  ___________________________________
ADDRESS (LINE 1):  ____________________
ADDRESS (LINE 2):  ____________________
CITY/STATE/ZIP:  _______________ __ _____
TELEPHONE NUMBER:  (___)___-_____
                        ENTER 'M' TO RETURN TO THE MENU:  _
                        PLACE CURSOR HERE TO TRANSMIT -->[_]
      ACCOUNT CODE ALREADY EXISTS IN X-REFERENCE
      FILE. PLEASE ENTER AGAIN.
```

Figure 4–6. Error Screen Generated for Program JAADD1

Work area required

To use screen format services, you must configure a work area, although you don't define a work area in your action program. The work area is specified in the ACTION section of the IMS configuration (WORKSIZE=n).

Moving variable fields to work area

When an action program is ready to create a screen, RPG II moves all variable fields in the output message area to the work area before it calls upon screen format services to generate the screen. The screen format generator then uses the output message area to build the entire output screen. When the screen is complete, the variable fields are returned to the output message area to await program termination. At that point, the entire contents of the output message area (screen and variable fields) are transmitted to the terminal.

Coding for screen format services

To use screen format services, you must enter on the output form:

- a **K** in position 42;
- the **number of characters** in the screen format name in position 43; and
- the **format name** beginning in position 45.

Listing output fields in order expected by screen format generator

When listing the variable fields to be output to the screen, remember to list them in the order in which the screen format generator is expecting them – that is, in the order they are defined in the screen format. Also, the first variable field cannot occupy a position before position 17. The first 16 positions always contain the output message area header.

For a complete discussion of how action programs can use screen format services, see Section 6.

5. Special Types of Output Messages

Sections 3 and 4 (summarized)

Sections 3 and 4 presented several examples of action programs performing the fundamental processes of accepting input from the terminal, processing that input, and producing output. They showed convenient programming techniques for accomplishing these activities. After you've studied these examples, you should be able to write simple action programs.

5.1. DIFFERENT TYPES OF OUTPUT MESSAGES

In this section, we describe additional capabilities that IMS provides for generating output messages. As you become more experienced, you will find these capabilities very useful. They are the ability to:

Types of output

- **generate** multiple output messages;
- **send** uninterrupted output messages to a terminal or auxiliary device attached to the terminal (continuous output);
- **initiate** a transaction at a terminal other than the source terminal (output-for-input queueing); and
- **send** messages to another terminal (message switching).

5.2. GENERATING MULTIPLE OUTPUT MESSAGES

Definition

When an action program generates more than one output message, we call it multiple output.

Example

Program LSTLIM (Figure 5-1) demonstrates how an action program generates multiple output messages.

MULTIPLE OUTPUT MESSAGES

```
             1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
                H                                                                   ALSTLIM
                F*
                F*  LIST STOCKS BETWEEN LIMITS INPUT FROM IMS INPUT
                F*  TESTS MULTIPLE OUTPUT USING CALL SEND
                F*  AND LIMITS PROCESSING
                F*
        001     FINPUT   IP  F      30           *IMA
        002     FSTOCKS  ID  F      80L 3AI     1 DISK        S
        003     FOUTPUT  O   F     960           *OMA
        004     E                   A          10 80
        005     IINPUT   AA  01
        006     I                                       21  23 LOWLIM
        007     I                                       25  27 HGHLIM
        008     ISTOCKS  AA  02
        009     I                                        1   3 KEY
        010     I                                        1  80 RECORD
        011     C           LOWLIM    SETLLSTOCKS                    SET KEY TO LOW LIMIT
        012     C                     Z-ADD1         I        20     INITIALIZE ARRAY
                C*  DISPLAY SCREENS OF 10 RECORDS EACH
        013     C           LOOP      TAG
        014     C                     READ STOCKS                 20
        015     C   N20     KEY       COMP HGHLIM             20     ALSO SET IF HIGH
        016     C   N20               MOVE RECORD    A,I
        017     C     *     I         ADD  1         I
        018     C     *     I         COMP 11                     30
        019     C     * 30            EXCPT
        020     C     *               Z-ADD1         I
        021     C   N20               GOTO LOOP
                C*  DISPLAY RESIDUAL SCREEN
        022     C           I         COMP 1                      40
        023     C   N40               EXCPT
        024     OOUTPUT  E         02
        025     O       OR         20
        026     O                                  20 X'10030201'
        027     O                                  26 'SYMBOL'
        028     O                                  32 'NAME'
        029     O                                  55 'RANGE'
        030     O                                  65 'PRICE'
        031     O                                  75 'CHANGE'
        032     O                                  85 '% CHANGE'
        033     O                                  95 'EXCHANGE'
        034     O                                 104 X'10030301'
        035     O                        A       B 904

                                          SYMBOL TABLES

RESULTING INDICATORS

ADDRESS RI      ADDRESS RI      ADDRESS RI      ADDRESS RI      ADDRESS RI      ADDRESS RI      ADDRESS RI

 000019 1P       000015 LR       000016 00       000017 01       000018 02       00002A 20       000034 30
 00003E 40       00007A L0       000085 H0       000086 H1       000087 H2       000088 H3       000089 H4
```

Figure 5-1. Multiple Output Message Program (LSTLIM)

What LSTLIM does

LSTLIM sequentially processes an indexed file, STOCKS, containing stock records. The terminal operator enters as input low and high limit values that determine where processing of the file begins and ends. When LSTLIM receives these values, it begins reading STOCKS at the low limit and continues until the high limit is exceeded or the end of file is reached. When the program terminates, the records read are displayed at the terminal in groups of 10.

Coding the File Description Form

Lines 001-003 contain the file description form coding for the program. The operations performed are:

File definition

Line	Description
001	LSTLIM uses the input message area defined as the primary file, INPUT.
002	LSTLIM also uses an input demand file, STOCKS. STOCKS is an indexed file containing 80-character records on disk. L in column 28 means the file is processed sequentially within limits. The 3-character key (columns 29-30) is alphanumeric (column 31) and begins in position 1 (column 35-38).
003	LSTLIM uses the output message area defined as the output file, OUTPUT.

Coding the File Extension Form

Line 004 contains the file extension form coding for the program. The operation performed is:

Array definition

Line	Description
004	Array A, defined in column 27, holds the stock records processed. When full, the array contains ten 80-character records.

Coding the Input Form

Lines 005–010 contain the input form for coding the program. The operations performed are:

Line	Description
005-006	*Input field definition* — The LOWLIM field (positions 21-23 in the input message area) defines the lower limit used in processing the file, STOCKS. Positions 1-16 contain the input message area header; positions 17-19 contain the transaction code, STK; and positions 20 and 24 contain spaces.
007	*Operator entries* — The HGHLIM field (positions 25-27 in the input message area) defines the upper limit used in processing the file, STOCKS. When initiating the transaction, the operator enters the transaction code, the low limit, and the high limit. Since LOWLIM and HIGHLIM are the only input fields that LSTLIM references, they are the only ones defined on the input form.
008-010	*Key definition* — The STOCKS file contains 80-byte records that begin with a 3-byte key.

Coding the Calculations Form

Lines 011–023 contain the calculations for the program. The operations performed are:

Calculation form coding

Line	Description
011	LSTLIM uses the input field LOWLIM to set the lower limit for processing the file, STOCKS.
012	The array index (I) is set to 1.
013	The LOOP operation processes 10 STOCKS records before exception output is done.
014	RPG II begins reading STOCKS at the lower limit (LOWLIM). If end-of-file is reached, indicator 20 is set on and processing continues at line 022.
015	If the end-of-file condition is not met, the field KEY is compared to HGHLIM to determine if the high limit for file processing was exceeded. If KEY is greater than HGHLIM, indicator 20 is set on and processing continues at line 022.
016	If the end-of-file condition doesn't occur or high limit isn't exceeded, the record is moved to array, ARY.
017	The array index is incremented by 1.
018	The array index is compared to 11. If I equals 11, the array contains 10 records. Indicator 30 is set on.
019	When indicator 30 is set on, exception output is done. The 10 elements in the array are moved to the output message area. However, this output message doesn't go to the terminal until LSTLIM terminates. Once the contents of the array are moved to the output message area, the array is blanked out to allow it to receive another set of 10 records.
020	After exception output is done, processing resumes at line 020 and the array index is reinitialized to 1. Record processing begins to create another array of data.
021	When I is less than 11, indicator 30 is not set on. Processing returns to LOOP to read another record (line 013). This continues until the array is full, end-of-file condition is reached, or high limit is exceeded.
022-023	When indicator 20 is set on by the end-of-file condition or by exceeding the high limit for file processing, the array index is compared to 1. If it is greater than 1, exception output occurs.

Coding the Output Form

Lines 024–035 contain the output form coding for LSTLIM. The operations performed are:

Output form coding

Line	Description
024-025	The output message area is OUTPUT. Exception output to the output message area occurs when the array contains 10 records (indicator 02 is set on) or when the array is partially full and indicator 20 is set on.
026	The first output field is a 4-character device independent code (DICE) sequence ending in position 20. (The output message area header occupies the first 16 positions.) The DICE code sequence positions the cursor at line 2, position 1 on the terminal screen.
027-033	Heading data is displayed.
034	This DICE sequence repositions the cursor at line 3, position 1.
035	The 800-character array (10 records, 80-characters each) is displayed using *blank after*. Blank after reinitializes all elements of the array to zeros or blanks. This is needed because the array may be used many times during execution of the program depending on how many stock records are processed. When processing is complete, the array is again blanked out. This is needed because action programs are serially reusable.

LSTLIM generates as many messages as needed

As you can see, LSTLIM can generate as many output messages as needed. The low and high limits entered as input by the terminal operator are the sole determinants of the number of output messages – groups of 10 records each – that are generated.

5.3. HOW MULTIPLE OUTPUT MESSAGES ARE PROCESSED

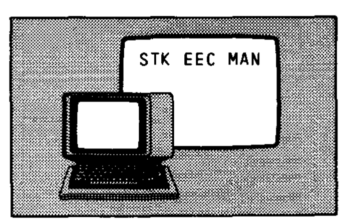
When messages are transmitted

The important point to remember regarding to multiple output messages, just as with any output message generated by an action program, is that none of the messages go to the terminal until the action program terminates. To understand what happens between the time these output messages are generated and when they actually appear on the terminal screen, let's use the action program LSTLIM once again and supply input data.

The input message entered is: STK EEC MAN

Terminal input

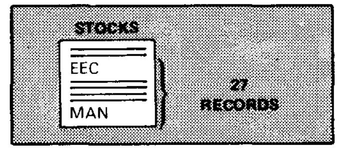
STK is the transaction code. It identifies to IMS the program LSTLIM that processes this transaction. The entries EEC and MAN define the lower and upper limits respectively, for processing the file, STOCKS.

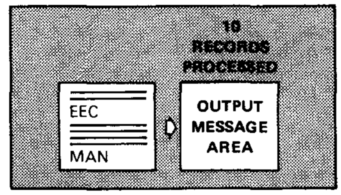


MULTIPLE OUTPUT MESSAGES

First output message

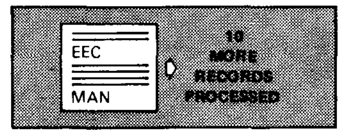
Let's assume that there are 27 records in STOCKS that fall between these limits. The first time LSTLIM does exception output, the 10 records processed are moved from the array to the output message area. Each time exception output is complete, the array is blanked.





Second output message

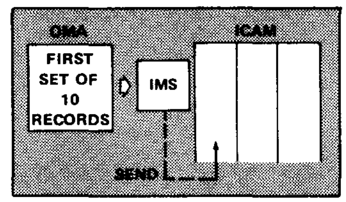
When control returns to the program, the program reinitializes the array index to 1 and processes 10 more records. Indicator 30 is set on, signaling more exception output.



CALL SEND

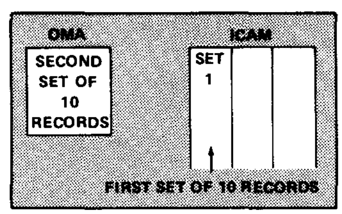
When the second request to do exception output is received and the output message area already contains data, RPG II issues a SEND function call. IMS takes the contents of the output message area and moves it to an ICAM (communications) queue. Note that the first set of 10 records was not sent to the terminal. The output message area is now free to receive the exception output. The second set of 10 records in the array is now moved to the output message area.

Moving array contents



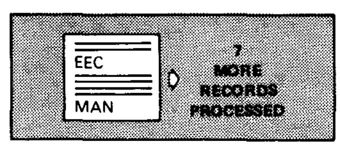
Queueing messages

Up to now, 20 records were processed. LSTLIM generated two output messages, neither of which was sent to the terminal. The second message is in the output message area; the first, in an ICAM queue.



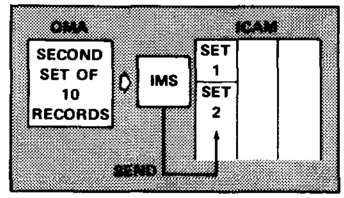
Reading last records in range

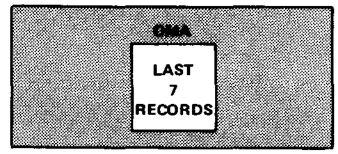
Once again your program reads STOCKS. After seven additional records are processed, indicator 20 is set on.



Final output message

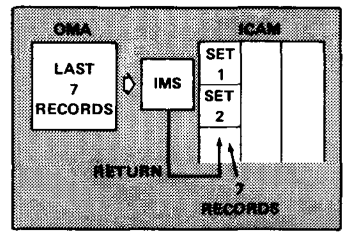
The high limit for file processing has been exceeded. The array index is compared to 1. It is 8. This signals more exception output. Again RPG II checks the output message area. It contains data. The SEND function call is repeated and the contents of the output message area (the second set of 10 records) is moved to the ICAM queue where the first set of 10 records is waiting. Now the output message area receives the seven records in the array.





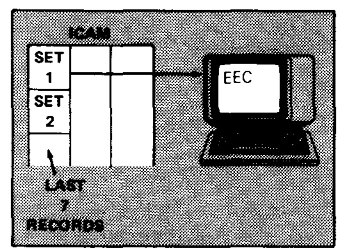
CALL RETURN

At this point, processing is complete. When the action program terminates, RPG II issues a call to the IMS RETURN function. IMS moves the last output message (the seven records) to the ICAM queue and begins transmitting output to the terminal.



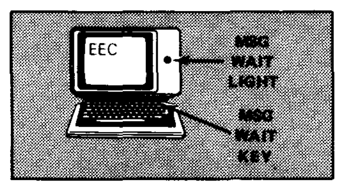
Output to the terminal

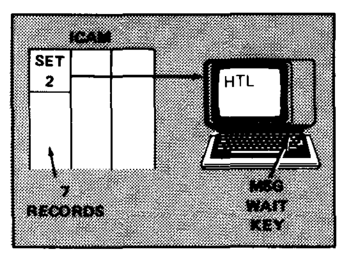
The data is sent to the terminal in the order that LSTLIM generated it – that is, the first screen of 10 records, the second screen of 10 records, and finally, the third screen of 7 records.

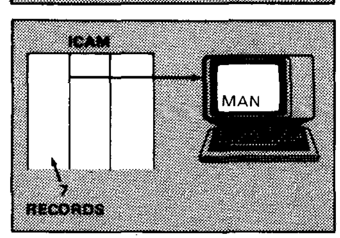


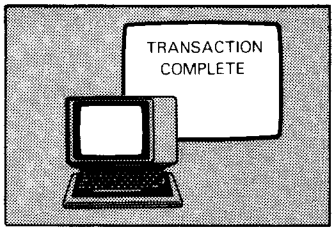
Operator action

After the first screen is transmitted, the message waiting light alerts the terminal operator that there is more output. When ready, the operator acknowledges the signal by pressing the **MSG WAIT** key and the next screen of 10 records is sent to the terminal. This process continues until all output generated by the program is sent to the terminal. The transmission of each output message after the first is preceded by the message waiting light and the operator pressing the **MSG WAIT** key.









Message handling

Whenever the action program creates more than one output message - using exception, detail, or total time output - RPG II and IMS handle the output in the manner just described. All output messages, except the final one, are transmitted using the SEND function. The last output message is always transmitted using a RETURN function when the program terminates.

Operator responses to multiple output

Table 5-1 shows how the terminal operator is informed of multiple output and how the operator acknowledges that output.

Table 5-1. Indicating and Accepting Multiple Output Messages

Terminal	Unsolicited Output Indicator	Response
Display (except IBM 3270)	Message waiting light	Press message waiting key.
Hard copy (except DCT 1000)	/CMW or other 4-character message*	Press CTRL/G, then press CTRL/C.
IBM 3270	Message waiting light or /CMW*	Press PA1 key.
DCT 1000	Message waiting light	Press CTRL/G, then XMIT.

*This message is defined by the MSGWAIT operand of the TERM macro in the ICAM network definition. The default is /CMW.

Requirement when using SEND function

Disk queueing

Message queueing

If the action programs you write use the SEND function, you must specify the UNSOL=YES parameter in the OPTIONS section of the IMS configuration. If the SEND function is used frequently, you should also include disk queueing for output messages when defining your communications network (ICAM). When you specify disk queueing, IMS queues output messages generated by an action program on disk each time the SEND function occurs. These messages are sent to the terminal when the program terminates. Disk queueing allows for more productive use of main storage.

Multiple output message limitations

If you want to examine each screen of data containing output, issuing multiple output messages is a good idea. You should not use it, however, as a substitute for obtaining lengthy output messages because the operator wastes considerable time pressing the **MSG WAIT** key to obtain the entire output. Instead, use the continuous output feature discussed in 5.4 through 5.12.

5.4. GENERATING CONTINUOUS OUTPUT

Definition

The second capability involving output messages is the ability to transmit a series of output messages to a terminal or more commonly to an auxiliary device attached to the terminal without operator intervention. This is called continuous output.

Useful for lengthy reports

This capability is very useful when you want to print lengthy reports at an interactive terminal.

Specifying continuous output in IMS configuration

To use continuous output, you must specify CONTOUT=YES in the OPTIONS section of your IMS configuration.

5.5. DEVICES THAT CAN RECEIVE CONTINUOUS OUTPUT

Terminals and auxiliary devices supported

Action programs can direct continuous output to hard copy terminals or to auxiliary devices (printer, tape cassette, or diskette) at display terminals. For a complete list of terminals and auxiliary devices supported by IMS, see the IMS system support functions user guide, UP-8364 (current version).

5.6. CODING FOR CONTINUOUS OUTPUT

Specifying continuous output in program

To distinguish a continuous output message from other output messages, an action program moves a special value to the aux-function field (position 15) of the output message area header. You move this value at the same time as you generate your output message. When the program terminates, IMS checks this field and recognizes that the message generated is a continuous output message.

Specifying continuous output to auxiliary devices

If that message is to go to an auxiliary device, as opposed to just going to the display terminal, the program also moves a value to the aux-device-no field (position 16) of the output message area header when generating the output message. This value informs IMS which device receives the continuous output message. You assign a unique number to each auxiliary device when you define your communications network.

Aux-function field settings

Table 5–2 summarizes the settings for the aux-function field when transmitting continuous output to a terminal or to an auxiliary device. You find those values in columns 6 and 7 of Table 5–2.

Table 5-2. Settings for Aux-Function Field of the Output Message Header (Print/Transfer Options)

Devices		Output Option			Contents of aux-function field			
					Continuous Output		No Continuous Output	
Primary	Auxiliary	Name	Space Suppression	Inhibit Space Suppression	Hexa-decimal	Character	Hexa-decimal	Character
X					C3	C	00	
	X	Print Mode	X		F3	3	F0	0
				X	F5	5	F2	2
		Print Transparent	X		F7	7	F4	4
				X	F9	9	F6	6
		Print Form (ESC H)	X		C1	A	D1	J
				X	C6	F	D6	O
		Transfer All (ESC G)	X		C2	B	D2	K
				X	C7	G	D7	P
		Transfer Variable (ESC F)	X		C4	D	D4	M
				X	C8	H	D8	Q
		Transfer Changed (ESC E)	X		C5	E	D5	N
				X	E8	Y	F8	8

Directing Continuous Output to a Terminal

Continuous output for the terminal

Looking at the columns labeled **Continuous Output** in Table 5-2, you notice that if you're sending continuous output to the terminal (primary device), you move the character C or a hexadecimal C3 to the aux-function field. Figure 5-2 shows how you code the output form to send continuous output to the terminal.

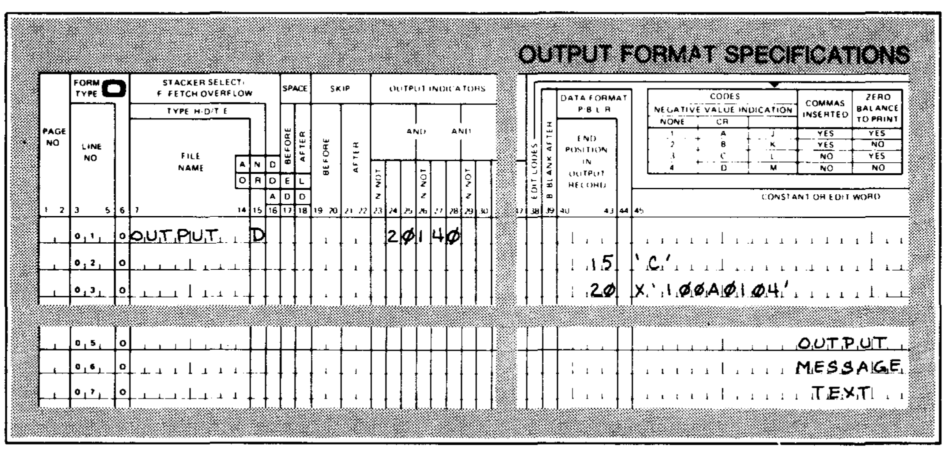


Figure 5-2. Coding a Continuous Output Message for the Terminal

Directing Continuous Output to an Auxiliary Device

Continuous output for an auxiliary device

When you are transmitting continuous output to a COP, TP, cassette, or diskette auxiliary device, Table 5-2 illustrates that there are numerous values you can move to the aux-function field. The value you choose depends on the print or transfer option you want.

Print and transfer options

Table 5-2 lists the print and transfer options you can select and their corresponding values. Table 5-3 further defines these options.

These print and transfer options can be used to transmit messages to auxiliary devices whether or not you're using the continuous output feature. Also, some auxiliary functions aren't allowed if you use screen format services. See Table 6-2.

Table 5-3. Print and Transfer Options

Option	Description
Print Mode	Message transmitted has the same format as the terminal screen. Cursor return sequences for the screen apply.
Print Transparent	Message transmitted is independent of the terminal screen format. Whatever format you include with your message applies.
Print Form (ESC H)	Message transmitted contains all unprotected characters from the start-of-entry (SOE or home position) to the cursor. Field control characters are suppressed.
Transfer All (ESC G)	Message transmitted to the auxiliary device contains all characters from the start-of-entry character to the cursor including field control character sequences.
Transfer Variable (ESC F)	Message transmitted to the auxiliary device contains only the unprotected characters between the start-of-entry character and the cursor including field control character sequences.
Transfer Changed (ESC E)	Message transmitted to the auxiliary device contains only the changed characters between the start-of-entry and the cursor including FCC sequences.

Definition of print transparent mode

One of the more commonly used options is **print-transparent** mode. In this mode, although the continuous output message generated goes through the logic of the primary device, its format is independent of the terminal screen format. Only device independent code (DICE) sequences and field control characters (FCCs) you use to format the continuous output message apply.

Using transparent mode

The cursor return characters normally inserted by the terminal are not transmitted. Thus, the length of the lines written to the auxiliary device is independent of the line length of the screen.

Print transparent mode with UNISCOPE 100

When using print-transparent mode with a UNISCOPE 100 display terminal, make sure that the output message generated doesn't exceed screen capacity. If it does, the excess lines wrap around and overlay the first few lines. Since the message on the screen is the message sent to the auxiliary device, the transmitted result is a message beginning with the excess lines

With other display terminals

instead of the original lines. The same consideration applies to all display terminals; however, the larger screen capacity of most terminals makes wraparound less likely.

Definition of print mode

In **print mode,** the continuous output message transmitted to the auxiliary device has the same format as the screen – that is, cursor return characters apply. For further details on print-mode and print-transparent mode, refer to the UNISCOPE programmer reference, UP-7807 (current version), and the UTS 400 programmer reference, UP-8359 (current version).

Space suppression

When you choose either print or transfer options, you can allow or inhibit space suppression (see Table 5-2). When you specify allow space suppression, the remote device handler suppresses all nonsignificant spaces in the output message. When you specify inhibit space suppression, the remote device handler changes all spaces to DC3 characters making it necessary to strap the printer to space when it receives a DC3 character in the output message text.

Identifying the auxiliary device

As we already noted, when you're transmitting continuous output to an auxiliary device, you must also move a value to the aux-device-no field. The value you move to the aux-device-no field identifies that auxiliary device. Each auxiliary device attached to a terminal has a specific number as defined in the communications network definition.

Example

Let's assume you want to transmit continuous output to a cassette using the transfer-all option. You would specify hexadecimal C2 or the character B in the aux-function field. In aux-device-no, you would put the value configured for the auxiliary device to which you are directing continuous output. Figure 5-3 shows how the coding might look:

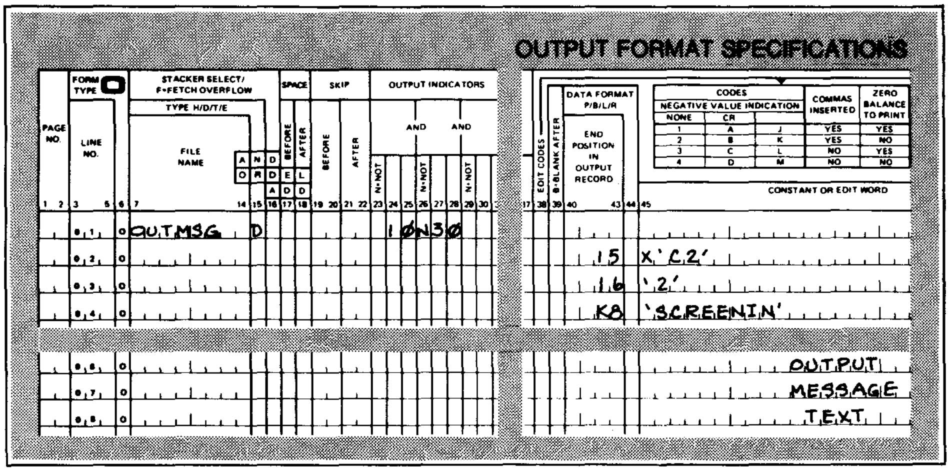


Figure 5-3. Coding a Continuous Output Message for an Auxiliary Device with the Transfer-All Option

5.7. WRITING A CONTINUOUS OUTPUT PROGRAM

You write an action program to generate continuous output as you would any action program. However, there are some special and very important considerations to take into account.

Using the aux-function field

First, as we described in 5.6, if you're transmitting continuous output to the terminal, on the output specifications form you must move hexadecimal C3 or the character C to the aux-function field (position 15) of the output message area header (see Figure 5-2). This informs IMS at action program termination that this program generated a continuous output message. It is not very common to direct continuous output to a terminal exclusively.

Using the aux-device-no field

Example

If you're transmitting the continuous output message to an auxiliary device attached to the terminal, you move a value to the aux-function field specifying the print or transfer option you select. Table 5-2 summarizes these options. In addition, you enter in the aux-device-no field (position 16) of the output message area header, the number configured for the auxiliary device. To illustrate these procedures, Figure 5-4 shows the output form coding to generate continuous output to a printer using the print transparent option with inhibit space suppression when the program terminates.

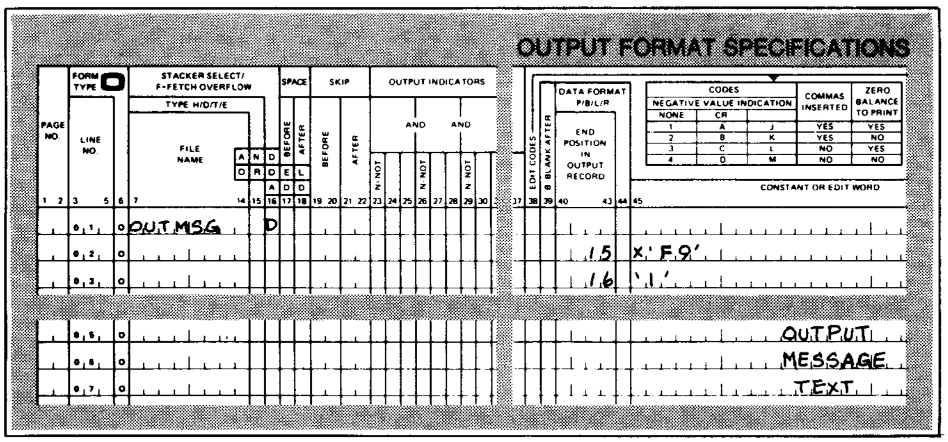


Figure 5-4. Coding a Continuous Output Message for a Printer with Print Transparent and Inhibit Space Suppression

Continuous output limitations

Effect of different terminal screen sizes

Second and most important, an action program can generate *only one* continuous output message. This message can be as large as the screen capacity of the terminal receiving the message will allow. Of course, this varies depending on the type of terminal or workstation you're using. Whether the message is destined for the terminal or for an auxiliary device, it always passes through the terminal screen first. If the message is larger than the screen, it wraps around, and when transmitted to the auxiliary device, the beginning of the message is lost. Consequently, the size restrictions for the terminal also apply to transmitting continuous output to an auxiliary device.

How to generate lengthy messages

The term continuous output, by its very nature, suggests lengthy output messages. If an action program can produce only one continuous output message and the largest message can only be the size of a screen, you're undoubtedly wondering how we generate long messages.

Continuous output and successor programs

That brings us to the third point: to continue generating continuous output, an action program must name a successor.

How a lengthy message is generated

The key is that the first program generates its continuous output message and names a successor program to continue generating continuous output. That program, in turn, names a successor and so on, and so forth. One program could reschedule itself numerous times or the successor program could be a different program.

No operator intervention

Once you identify an output message to IMS as continuous output, the message is transmitted to the terminal or auxiliary device and the successor program is scheduled to continue generating continuous output. There is no need for operator intervention. This is how lengthy reports can be printed at an interactive terminal.

Naming a successor program

To name a successor, the action program moves the successor's name to the successor-id field (positions 5–10) of the program information block when the program terminates. This is the same procedure any action program follows for naming a successor.

External succession required

The fourth consideration is that the action program must also move an E (for external succession) to the termination-indicator field (position 11) of the program information block when the program terminates in order to continue generating continuous output.

Input message to successor program

The reason for specifying external succession (E) as opposed to other types of termination is that when continuous output takes place, IMS generates a 5-character message that is sent as input to the successor program. This program must be prepared to accept that input. External succession means that the successor action program is ready to accept an input message.

Transmitting the continuous input message

The fifth and final point to remember when generating continuous output is that this message must be the final message the action program creates - that is, it must be transmitted using the IMS RETURN function when the action program terminates. You can't use the SEND function to transmit a continuous output message.

Other message types and continuous output

This does not mean, however, that an action program generating continuous output is restricted from using the SEND function altogether. The program can generate as many output messages as it chooses prior to creating the continuous output message. All the prior messages are transmitted using the SEND function. However, the continuous output message must be the last message generated and consequently, transmitted using the RETURN function.

Handling output messages

You recall that when an action program generates multiple output messages, all the messages except the last are transmitted using the SEND function. The last output message generated by an action program is always transmitted as a RETURN function. For more detailed information on how output messages are handled, see 5.3.

Summary

To summarize:

- An action program execution can generate one continuous output message only.
- The continuous output message can't exceed screen size.
- To continue generating continuous output, you specify a successor program and external succession.
- The continuous output message must be the final message the program generates.

5.8. THE IMS DELIVERY CODE

Identifies input message

Whenever an action program generates a continuous output message, its successor program receives from IMS a 5-character input message. The first four characters contain the value placed in the continuous-output-code field (positions 9–12) of the output message area header by the previous program. Placing a value in this field is optional. Generally, this code identifies the previous program in some way. If the program doesn't move a value to this field, then it contains binary zeros.

Defining the delivery code

The fifth character of the input message is the important one. It is a delivery code. The delivery code indicates whether ICAM successfully delivered the continuous output message to its destination or not.

Indicating a value in the continuity-output -code field

Figure 5–5 shows how you code to move a value to the continuous-output-code field, and Figure 5–6 demonstrates how IMS returns this value and the delivery code to the successor action program.

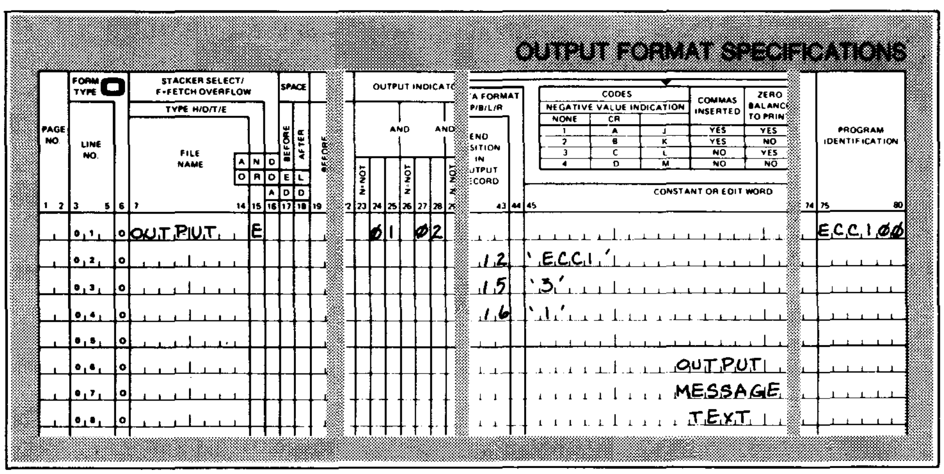


Figure 5–5. Coding to Move a Value to Continuous-Output-Code

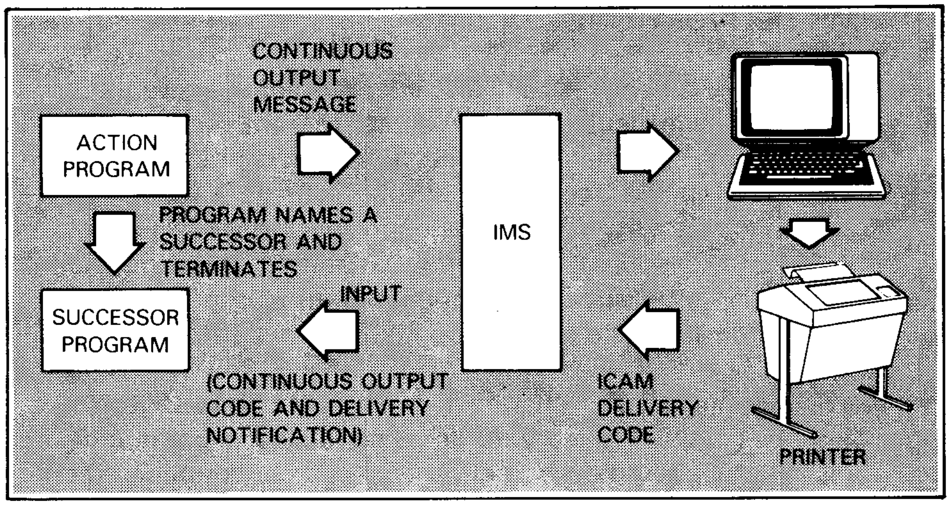


Figure 5-6. Input Message Returned to Successor Program in Continuous Output Transaction

How continuous-output-code field is used

Figure 5-5 shows that the value moved to continuous-output-code is ECC1. ECC1 identifies the program generating the message. When the action program terminates, the continuous output message generated is transmitted. When it is received and acknowledged by the destination terminal, IMS schedules the successor action program and the value ECC1 plus the delivery code acknowledgment from ICAM are sent as input to the successor program. The value ECC1 comes into the successor program in the input message area in positions 17-20. The delivery code comes into the program in position 21.

Specifying continuous output for auxiliary devices

The other two output fields coded in Figure 5-5, aux-function and aux-device-no, respectively, indicate that the continuous output message generated by this action program went to an auxiliary device attached to the terminal. The message is sent using print-mode with space suppression. The configured number for the auxiliary device is 3.

Continuous output status

Continuous output status codes

Obviously, the fifth character of the input message is the one of particular interest to the successor action program. It contains a value indicating the status of the continuous output message sent by the predecessor program. If the continuous output message was successfully delivered, the hexadecimal value 81 is returned to the successor action program. If the lowercase-to-uppercase translation option was specified for this action program at IMS configuration, the value 81 is translated to the character A. Any other value returned in the fifth character of the input message indicates the continuous output message was not successfully delivered. Tables 5-4 and 5-5 summarize the output delivery notice status codes that can be returned to an action program.

Output delivery notice status codes

Table 5-4. Output Delivery Notice Status Codes Returned by IMS

Condition	Primary Devices Addressed				Hexadecimal Value
	Polled		Nonpolled		
	UNISCOPE, UTS Devices and Workstations	DCT 1000	DCT 500	TTY	
Successful output completion	Yes	Yes	Yes, regardless of delivery	Yes, regardless of delivery	81①
Line down or disconnected. Message deleted by IMS.	Yes	Yes	Yes	Yes	11
Terminal marked down. Message deleted by IMS.②	Yes	Yes	No	No	12
Auxiliary device down. Message deleted by IMS. Output may be addressed to the primary device.	Yes	No	No	No	40
Missing or invalid destination or auxiliary specification in header	Yes	Yes	Yes	Yes	84①
No ICAM network buffer available③	Yes	Yes	Yes	Yes	85①
Disk error	Yes	Yes	Yes	Yes	86①
Invalid output buffer length	Yes	Yes	Yes	Yes	87①

NOTES:

① The hexadecimal value 81, indicating successful output completion, is translated to the character A if the lowercase-to-uppercase translate option is specified for messages input to the successor action. Similarly, the hexadecimal values 84 through 87, indicating error conditions, are translated to the characters D through G if the translate option is specified.

② When a terminal is marked down, input solicitation (polling) by ICAM continues automatically. When ICAM receives input from the down terminal, that terminal is marked up and the input is scheduled for IMS.

③ If this condition exists, a user action program can try to re-send the last continuous output message.

Auxiliary device condition codes

Table 5-5. UNISCOPE and UTS Auxiliary Device Condition Codes

Auxiliary Device Condition	Hexadecimal Value	UNISCOPE or UTS Auxiliary Status
Ready (good) status but COP/TP write function inoperative	41	1
Device out of paper, inoperative, or in test mode	42	2
Data error on TCS	43	3
Device is not responding; it may be disconnected, or a read of unwritten tape may have occurred.	44	4

5.9. RECOVERY CONSIDERATIONS WITH CONTINUOUS OUTPUT

Recovery and restart action program responsibilities

Recovery and restart processing are the responsibility of your action program. When the successor action program receives an unsuccessful delivery notice, it can continue processing continuous output or terminate the transaction. When the successor program continues processing, it can send a regular output message to the terminal requesting assistance and then terminate with external succession. Note that when a continuous output message is unsuccessfully sent to an auxiliary device, only that device is marked down. You can still send output to the primary device.

Operator reinitiates output after error correction

After the error condition is corrected, the terminal operator can send an input message to the successor program to reinitiate the continuous output transaction. In this case, the successor program must be prepared to accept input from the terminal when necessary, as well as the delivery notice returned by IMS. You should consider this possiblity when designing your action programs.

Program or operator can control output

Both operator-entered input and delivery notice input can cause attempts to schedule the successor continuous output program. If operator-entered input exists, IMS processes that input and discards the delivery notice. You should, therefore, code your action program to handle keyboard input that can end, temporarily break, and resume a continuous output transaction.

Function keys

The best way to interrupt continuous output is to use function keys as keyboard input. Function keys are faster to use because they are never locked.

Terminal type affects recovery

When a delivery attempt is unsuccessful, there are a number of recovery options. In planning recovery, however, it's important to realize the difference between polled and nonpolled devices with respect to unsuccessful delivery notices.

Polled device acknowledgment

The DCT 1000, UNISCOPE 100 and 200, and UTS terminals are polled devices and transmit an acknowledgment to ICAM after receiving a continuous output message; the nonpolled devices,

Nonpolled device acknowledgment

TELETYPE* and DCT 500 terminals, do not. For nonpolled devices, a delivery notice is automatically generated; it always indicates successful delivery regardless of whether or not the output message was successfully delivered. Only a line-down condition returns an unsuccessful delivery notice.

---

*Trademark of Teletype Corporation

Problem caused by nonpolled devices

Consequently, IMS almost always receives a successful completion status from ICAM when a message is delivered to a nonpolled device. IMS sends this delivery code to the successor action program which, in turn, generates more continuous output. As you can see, this is a situation to be avoided. So, in critical parts of continuous output applications, avoid using nonpolled devices.

Some errors not related to terminal type

Certain error conditions (the last four entries in Table 5–4) are detected by ICAM before the message is sent to the terminal. These errors return an unsuccessful delivery notice regardless of the device type.

5.10. A SAMPLE CONTINUOUS OUTPUT PROGRAM

Example

So far we have presented a great deal of information concerning continuous output. Now let's look at an action program that generates continuous output. The program we will use is the second in a series of three action programs that make up a continuous output transaction. Let's begin by summarizing what the first program, SALES 1, does:

What SALES1 does

- Updates a file, SLSST
- Saves data used in updating the file in the continuity data area
- Generates a continuous output message giving branch sales data
- Names a successor program to continue generating continuous output
- Terminates with external succession

The successor program is SALES2. Figure 5-7 contains the coding for SALES2. The SALES2 successor program:

What SALES2 does

- receives the 5-character input message generated by IMS;
- interrogates the fifth character of this message (delivery code);
- generates a continuous output message;
- names a successor program; and
- terminates with external succession.

CONTINUOUS OUTPUT PROGRAM SALES2

```
            1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
UNIVAC OS/3 RPGII VERS 801007           SALES2                  81/07/02 00.04

            H                                                                   ASALES2
            FPIB     UD  F       69             *PIB
            FIMA     IP  F      100             *IMA
            FCDA     ID  F      250             *CDA
            FOMA     O   F      600             *OMA
            IIMA     AA  01
            I                                         21  21 DELVCD
            ICDA     BB  02
            I                                          1  25 CNAME
            I                                         26  50 SRNAME
            I                                         51  70 CADDR
            I                                         71  85 CCITY
            I                                         86  90 CZIP
            I                                         91  94 BRNUM
            I                                         95 114 BRADDR
            I                                        115 129 BRCITY
            I                                        130 134 BRZIP
            I                                        135 140 INVCE
            I                                        141 146 DLDATE
            IPIB     CC  03
            I                                          1   2 STCODE
            I                                          3   4 DETCDE
            C           DELVCD    COMP X'81'                    20GOOD DELIVERY
            C           DELVCD    COMP X'41'                    30COP TURNED OFF
            C           DELVCD    COMP X'42'                    40COP OUT OF PAPER
            C  N20                GOTO END
            C                     READ CDA
            C           END       TAG
            OOMA     D         02 20
            O                                   K8 'PRINTOUT'
            O                                   12 'SAL1'
            O                                   15 '3'
            O                                   16 '1'
            O                         CNAME     41
            O                         CADDR     61
            O                         CCITY     76
            O                         CZIP      81
            O                         BRNUM     85
            O                         BRADDR   105
            O                         BRCITY   120
            O                         BRZIP    125
            O                         INVCE    131
            O                         DLDATE   137
            O                         SRNAME   162
            O        D         02N20
            O                                   K8 'ERRORFMT'
            O                  30               35 'TURN PRINTER ON    '
            O                  40               35 'RESET PAPER TO HOME'
            O                                   58 'HIT TRANSMIT TO RESTART'
            OPIB     D         03
            O                                   11 'E'
            O                 N20               10 'SALES1'
            O                  20               10 'SALES2'
```

Figure 5-7. Continuous Output Program SALES2

File Description Form Coding

Continuity data area

On the file description form coding, notice that there are four interface areas. These should be very familiar to you by now. The continuity data area contains data passed by SALES1, the predecessor action program. This is the data that SALES2 will use to generate its continuous output message when the program terminates. You recall that an action can generate only one continuous output message and that message cannot be larger than the terminal screen size. SALES1 generated one continuous output message; but there is still more data to transmit. So, it scheduled SALES2 as successor to continue generating continuous output.

Interaction between SALES1 and SALES2

Input Form Coding

This form defines input fields. Notice there are several fields defined for the continuity data area. These fields contain data passed by SALES1.

Describing the delivery code

In addition, there is one field defined for the input message area, DELV. DELV contains the delivery code returned by IMS. Whenever an action program generates continuous output (in this case the first program in the transaction, SALES1), IMS returns a 5-character code as input to the successor program. The fifth character or delivery code indicates whether the first continuous output was successfully delivered or not. Every successor program in a continuous output transaction must be prepared to receive this code.

Successful completion

Continuous-output-code field

IMA header

Delivery code position

In our example, the first four characters of the input message returned by IMS are SLS1 – the value moved to the continuous-output-code field (positions 9–12) of the output message area header by action program SALES1. This value comes into the input message area of SALES2 in positions 17–20. The input message area header occupies positions 1–16. The action program, SALES2, doesn't define positions 1–20 because these fields are not referenced in the program. However, it does define position 21 since this position contains the delivery code generated by IMS, indicating whether the continuous output message created by SALES1 was successfully delivered or not. Before SALES2 generates a continuous output message of its own, it must determine if the first message was transmitted successfully. It does this by interrogating the delivery code.

Calculation Form Coding

Interrogating the delivery code

On the calculation form, the three COMPARE operations interrogate the delivery code to determine what processing occurs next. When DELV equals hexadecimal 81, the first continuous output message was successfully delivered. When this value is returned to the program, indicator 20 is set on and SALES2 generates continuous output.

Unsuccessful delivery/ printer off

When DELV equals hexadecimal 41, the first continuous output message was not successfully delivered because the printer was not turned on. When this value is returned to the program, RPG II sets on indicator 30 and SALES2 does not generate continuous output.

Unsuccessul delivery/ printer out of paper

When DELV equals hexadecimal 42, once again the first continuous output messsage was not successfully delivered because the printer was out of paper. When this value is returned to the program, indicator 40 is set on and SALES2 does not generate a continuous output message.

Effect of printer inoperative delivery codes

To reiterate, when DELV equals hexadecimal 41 or 42, SALES2 does not generate continuous output since the initial continuous output message generated by SALES1 was not successfully delivered. Instead, SALES2 calls SALES1 as its successor program to attempt retransmitting the first continuous output message. You'll recall that the values 81, 41 and 42 were described in Tables 5-4 and 5-5.

Output Form Coding

Request for operator intervention

In addition, when the delivery code indicates an unsuccessful attempt to deliver the first continuous output message, SALES2 generates a regular output message (not continuous output) that is sent to the terminal operator. When indicator 40 is set on, the message sent is: RESET PAPER TO HOME. When indicator 30 is set on, the message sent is: TURN PRINTER ON. By doing this, SALES2 instructs the terminal operator to correct the situation that prevented the initial transmission of SALES1's continuous output message.

Continuous output message generation

When indicator 03 is set on, (SALES1's continuous output message was successfully delivered) and SALES2 generates a continuous output message of its own. This message is transmitted as a CALL RETURN when the program terminates.

Naming a successor program

SALES2 specifies its successor program, SALES3, by moving that name to the successor-id field (positions 5–10) of the program information block. SALES3, which is not presented in this manual, is designed similar to SALES2 and continues generating continuous output.

Passing the continuous output code

In addition, when indicator 10 is set on, RPG II moves the continuous output code SLS2 to positions 9–12 of the output message area header. This code is transmitted as input by IMS to the successor program (SALES3) in positions 17–20 of the input message area, along with the delivery code indicating whether SALES2's continuous output message was successfully delivered or not.

Print mode specification

The number 3 in position 15 (aux-function field) of the output message area indicates that this output message is transmitted as continuous output using the print-mode option. Print-mode means that the output message takes on the same format as the terminal screen, that is, cursor return characters for the screen apply.

Auxiliary device specification

The number 1 in position 16 (aux-device-no) of the output message area indicates the continuous output is sent to an auxiliary device attached to the terminal. In our example, that device is a COP printer. The number 1 identifies the device as it was defined in the communications network definition.

Termination

When SALES2 terminates with external succession (E in the termination-indicator field), the continuous output message is transmitted to the terminal. It is transmitted as a CALL RETURN by IMS.

Output to the printer

Figure 5-8 shows the continuous output message generated by SALES2 as it appears on the terminal screen before being transmitted to the printer.

```
6/26/81

                ::::::::::::::::::::::::::::
                :: CENTER CITY SUPPLY CO. ::
                :: 3572 FRANKLIN DRIVE    ::
                :: MONROE, NH  72480      ::
                ::::::::::::::::::::::::::::

BRANCH: 7531
        WASHINGTON LANE
        CUPERTINO CA  37121

INVOICE: 362418

DELIVERY DATE: 7/31/81

SALES REPRESENTATIVE: GRACE A. MICHELLI
```

Figure 5-8. Continuous Output Generated for SALES2

Terminal screen size limits message size

You may have noticed that the continuous output message generated by SALES2 is rather small. The reason for this is that the installation implementing this application program uses UNISCOPE 100 display terminals. Their relatively small screen size demanded small messages. In the following action program, you'll notice the continuous output messages generated are much longer. The installation using this application uses UNISCOPE 200 display terminals.

5.11. ANOTHER SAMPLE CONTINUOUS OUTPUT PROGRAM

Example

The second example of a continuous output program is NCSC. It is quite lengthy; but its design is very similar to the program SALES2 described in 5.10. For that reason, we will point out only the new coding features it introduces. The coding for NCSC is in Figure 5-9.

```
          1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
   UNIVAC OS/3 RPGII VERS 781109          COMPIMS            81/03/18 08.29        PAGE     1

                H                              S                                 ANCSC
001       01 010 FCRT     IP  F      80       *IMA
002       01 030 FOHEADER IC  F     505R 7AI     1 DISC      S
003       01 030 FCUSTMST IC  F     249R 5AI     1 DISC      S
004       01 032 FNSHPCPY ID  F     378L13AI     1 DISC      S
005       01 033 FPDETAIL UC  F     378R13AI     1 DISC      S
006       01 040 FPIB     UD  F      48           *PIB
007       01 050 FCDA     UD  F      36           *CDA
008       01 060 FOMA     O   F    1920           *OMA
          01 060 F* NORTH  CAROLINA SHOP COPY PROGRAM
009       02 010 E                    ITM          4  3
010       02 020 E                    QTY          4  3 0
011       02 030 E                    MOD          4 10
012       02 040 E                    DES          4 15
013       02 050 E                    MAT          4  1
014       03 010 E                    HP           4  4
015       03 020 E                    VLT          4  6 0
016       03 030 E                    TYP          4  5
017       03 040 E                    RPM          4  4 0
018       03 050 E                    PLT          4  5 0
019       04 010 E                    TAG          4 20
020       04 020 E                    MTR          4 10
021       04 030 E                    MP           4 10
022       04 040 E                    SP           4 10
023       04 050 E                    BLT          4 10
024       05 010 E                    SHF          4  6
025       05 020 E                    BOR          4  6
026       05 030 E                    AC1          4  4
027       05 040 E                    AC2          4  4
028       05 050 E                    AC3          4  4
029       06 010 E                    AC4          4  4
030       06 020 E                    AC5          4  4
031       06 030 E                    AC6          4  4
032       06 040 E                    AC7          4  4
033       06 050 E                    AC8          4  4
034       07 010 E                    AC9          4  4
035       07 020 E                    NOT          4 60
036       07 030 E                    DS           4 10
037       07 040 E                    SIZ          4  2
038       07 050 E                    SCH          4  4 0
039       07 060 E                    PRP          4 10
040       07 070 E                    LT           4  4
041       07 075 E                    CDE          4  2
042     S 02 070 E                    TABFRT  5    5  1   TABNME 12
043     S 01 230 E                    TABCRD  4   12  2   TABNAM 16
044       12 040 E                    TABOMT  6    6  2
045     S 08 010 ICRT     AA  01  17 CN  18 CC  19 CS
046       08 020 I                                          17  20 PROG
047       08 021 I                                          22  28 START
048       08 021 I        AB  07
049       08 025 I                                          21  21 DLVCD
050       09 010 IOHEADER CC  03
051       09 030 I                                           8  12 CUSTNO
```

Figure 5-9. Continuous Output Program NCSC (Part 1 of 9)

```
1   5 7 9   13   17   21   25   29   33   37   41   45   49   53   57   61   65   69   73   77 80
   UNIVAC OS/3 RPGII VERS 781109            COMPIMS              81/03/18 08.29        PAGE     2

      09 040 I                                        14  1900RDATE
      09 050 I                                        36   38 REPONE
      09 060 I                                        43   45 REPTWO
      09 070 I                                        50   55 REPORD
      09 080 I                                        56   56 FRT
      09 095 I                                        57   58 HDCRED
    S 09 090 I                                       113  127 CSTORD
      09 100 I                                       128  157 SNAME
      09 110 I                                       158  187 SADRS1
      09 120 I                                       188  217 SADRS2
      09 130 I                                       218  247 SCTY
      09 140 I                                       250  309 SNOTES
      09 150 I                                       310  369 MNOTES
      09 155 I                                       370  429 FNOTES
      09 156 I                                       430  454 REPNME
    S 09 065 I                                       455  469 REQRED
      09 067 I                                       470  484 VIA
      09 070 I                                       491  492 PH
      09 075 I                                       493  494 KY
      09 080 I                                       495  496 WV
      09 085 I                                       497  498 NC
      09 090 I                                       499  500 AS
      09 160 ICUSTMST DD   04
      09 170 I                                        11   32 NAME
      09 180 I                                        33   54 ADRES1
      09 190 I                                        55   76 ADRES2
      09 200 I                                        77   93 CTYSTA
      09 200 I                                       116  120 ZIPCDE
    S 08 040 INSHPCPY BB   02
      08 050 I                                         1   13 KEY
      08 050 I                                         1    7 ORDNO
      08 055 I                                         1    2 YR
      08 056 I                                         3    7 NO
      08 060 I                                         8   10 PRODCD
      08 065 I                                         1   10 ORDPRD
      08 070 I                                        11   13 ITEM
      08 080 I                                        14  170QUANT
      08 090 I                                        18   27 MODEL
      08 100 I                                        28   31 HRSPWR
      08 110 I                                        32   37 MTRCDE
      08 120 I                                        38   380PLANT
      08 130 I                                        39   41 SRTCDE
      08 140 I                                        42   450SCHDTE
      08 150 I                                        48   49 NOCODE
      09 085 I                                        97   98 CRCODE
    S 08 155 I                                       109  1100POINTS
      08 160 I                                       117  1220VOLT
      08 170 I                                       123  127 TYPE
      08 180 I                                       128  128 MATERL
      08 190 I                                       129  1320RPMS
    S 08 010 I                                       138  147 MOTPLY
      08 020 I                                       148  157 DRVPLY
      08 030 I                                       158  167 BELT
      08 040 I                                       168  173 SHAFT
```

Figure 5-9. Continuous Output Program NCSC (Part 2 of 9)

```
      1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
UNIVAC OS/3 RPGII VERS 781109          COMPIMS               81/03/18 08.29        PAGE    3

  08 050 I                                      174 183 MOTOR
  08 060 I                                      184 193 DSCODE
  08 070 I                                      194 197 ACC1
  08 080 I                                      198 201 ACC2
  08 090 I                                      202 205 ACC3
  08 100 I                                      206 209 ACC4
  08 110 I                                      210 213 ACC5
  08 120 I                                      214 217 ACC6
  08 130 I                                      218 221 ACC7
  08 140 I                                      222 225 ACC8
  08 145 I                                      226 229 ACC9
  08 150 I                                      230 249 TAGNO
  08 160 I                                      250 309 NOTES
  08 163 I                                      316 321 BORE
  08 165 I                                      330 344 DESCRP
  08 166 I                                      345 346 SIZE
  08 170 I                                      347 356 PROP
  08 180 I                                      357 360 LTCODE
  08 185 I                                      367 376 TDIM
  08 200 IPDETAIL GG  08
  08 210 I                                      361 3660RUNDTE
  10 010 IPIB     EE  05
  10 020 I                                        1   2 STAT
  10 030 I                                        3   4 DSTAT
  10 040 I                                       11  11 TERMD
  10 060 ICDA     FF  06
  10 070 I                                        1  13 KEY1
  10 080 I                                        1  10 ORD
  10 100 I                                       11  13 LINE
  10 110 I                                       14  14 HEAD
  10 115 I                                       15  16 LSTREC
  11 004 C                       SETOF                   656667
  11 005 C   01                  SETON                   10
S 10 090 C  N01       DLVCD      COMP X'81'                  10MESSAGE COMP.
  10 095 C    *N10               GOTO END
  11 005 C                       READ PIB
  11 010 C                       READ CDA
S 10 015 C            LSTREC     COMP 'EF'                   23END OF FILE ???
  10 016 C   23                  SETON                   24
  10 017 C   24                  GOTO END
  10 02  C   01                  MOVELSTART     KEY1
  10 03  C    *                  MOVE '       ' KEY1
  10 04  C    *                  MOVE ' '       HEAD
  11 020 C            KEY1       SETLLNSHPCPY
  11 030 C                       Z-ADD1         A      10
  11 040 C            HEAD       COMP '1'                656766
  11 050 C   67                  READ NSHPCPY                25MORE DATA ??
  11 060 C   67       ORDNO      CHAINOHEADER            6969
  11 064 C    *       FRT        LOKUPTABFRT    TABNME       35
  11 065 C    * 35               MOVE TABNME    FRTNME 12
  11 070 C   67N69    CUSTNO     CHAINCUSTMST            6868
  11 073 C    *       HDCRED     LOKUPTABOMT                 75NO ADDRESS??
  11 075 C    * 75    HDCRED     LOKUPTABCRD    TABNAM       60CRED STATUS??
  11 078 C    * 60               MOVE TABNAM    CREDIT
```

Figure 5-9. Continuous Output Program NCSC (Part 3 of 9)

CONTINUOUS OUTPUT PROGRAM NCSC

```
1   5 7 9   13   17   21   25   29   33   37   41   45   49   53   57   61   65   69   73   77 80
UNIVAC OS/3 RPGII VEPS 781109             COMPIMS                  81/03/18 08.29        PAGE      4

      11 080 C   67                  GOTO FIN
      11 110 C             LOOP      TAG
      11 120 C  N67                  READ NSHPCPY                     25MORE DATA ??
      11 123 C   *         ORD       COMP ORDPRD                      20MORE DATA ??
  S 07 115 C   * 20      ORDNO     CHAINOHEADER               6969
      07 120 C   *         CRCODE    LOKUPTABCRD    TABNAM            60CREDIT STATUS
      07 130 C   * 60                MOVE TABNAM    CREDIT 16
  S 02 050 C  N67 20     POINTS    MULT QUANT     TPTS    60
      02 060 C   *         TDIM      COMP '         '                 40
      02 070 C   *N40                MOVE TDIM      DSCODE
      11 125 C   25                  GOTO END
      11 140 C   20                  MOVE ITEM      ITM,A
      11 150 C   *                   MOVE QUANT     QTY,A
      11 160 C   *                   MOVE MODEL     MOD,A
      11 170 C   *                   MOVE DESCRP    DES,A
      11 180 C   *                   MOVE MATERL    MAT,A
      12 010 C   *                   MOVE HRSPWR    HP,A
      12 020 C   *                   MOVE VOLT      VLT,A
      12 030 C   *                   MOVE TYPE      TYP,A
      12 040 C   *                   MOVE RPMS      RPM,A
      12 050 C   *                   MOVE PLANT     PLT,A
      12 060 C   *                   MOVE TAGNO     TAG,A
      12 070 C   *                   MOVE MOTOR     MTR,A
      12 080 C   *                   MOVE MOTPLY    MP,A
      12 090 C   *                   MOVE DRVPLY    SP,A
      12 100 C   *                   MOVE BELT      BLT,A
      12 110 C   *                   MOVE SHAFT     SHF,A
      12 115 C   *                   MOVE BORE      BOR,A
      12 120 C   *                   MOVE DSCODE    DS,A
      12 130 C   *                   MOVE ACC1      AC1,A
      12 140 C   *                   MOVE ACC2      AC2,A
      12 150 C   *                   MOVE ACC3      AC3,A
      12 160 C   *                   MOVE ACC4      AC4,A
      12 170 C   *                   MOVE ACC5      AC5,A
      12 180 C   *                   MOVE ACC6      AC6,A
      12 190 C   *                   MOVE ACC7      AC7,A
  S JI I// C   *                   MOVE ACC8      AC8,A
      13 010 C   *                   MOVE ACC9      AC9,A
      13 020 C   *                   MOVE NOTES     NOT,A
      13 030 C   *                   MOVE SIZE      SIZ,A
      13 040 C   *                   MOVE SCHDTE    SCH,A
      13 045 C   *                   MOVE PROP      PRP,A
      13 050 C   *                   MOVE LTCODE    LT,A
      13 055 C   *                   MOVE NOCODE    CDE,A
      13 055 C   *         KEY       CHAINPDETAIL               9999
      13 056 C   *                   EXCPT
      13 060 C   *         A         COMP 4                       80
      13 061 C   * 80      A         ADD  1         A
      13 080 C   *                   GOTO LOOP
      13 090 C  20N80N67             READ NSHPCPY                     25MORE DATA ??
      13 100 C   *         ORD       COMP ORDPRD                      20MORE DATA ??
      13 110 C   25                  GOTO END
      13 150 C             FIN       TAG
  S 13 110 C                         MOVE KEY       NXTKEY 13        SAVE NEXT RECRD
```

Figure 5-9. Continuous Output Program NCSC (Part 4 of 9)

```
     1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
   UNIVAC OS/3 RPGII VERS 781109           COMPIMS              81/03/18 08.29        PAGE    5

214         13 115 C              END       TAG
215         13 160 C              ITM,1     COMP '  '                 2626
216         13 170 C              ITM,2     COMP '  '                 2727
217         13 180 C              ITM,3     COMP '  '                 2828
218         13 190 C              ITM,4     COMP '  '                 2929
219         14 000 OPDETAIL E          20N99
220         14 002 O                            UMONTH    362
221         14 004 O                            UDAY      364
222         14 006 O                            UYEAR     366
223         24 050 OOMA      D        N10 07
224      S  24 020 O                                       20 X'100A0102'
225         24 035 O                                       45 '***MESSAGE INCOMPLETE***'
226         24 037 O                                       49 X'10040102'
227         24 040 O                                       73 'DO ONE OF THE FOLLOWING '
228         24 043 O                                       97 'PRINTER CONDITIONS EXIST'
229         24 045 O                                      101 X'10040102'
230         24 050 O                                      125 '1) OUT OF FORMS         '
231         24 055 O                                      129 X'10040102'
232         24 060 O                                      153 '2) FORMS JAMMED         '
233         24 065 O                                      157 X'10040102'
234         24 070 O                                      181 '3) POWER NOT ON         '
235         24 075 O                                      185 X'10040102'
236         24 080 O                                      209 'CORRECT PROBLEM AND RE- '
237         24 085 O                                      218 'START JOB'
238      S  14 010 OOMA      D        67N24 10
239         14 011 O                                       16 '71'
240         14 020 O                                       20 X'100A0101'
241         14 030 O                            REPONE B   23
242         14 040 O                                       24 '/'
243         14 050 O                            REPTWO     27
244         14 060 O                            REPORD     35
245         14 070 O                            REPNME     62
246         14 080 O                            ORDATEY    72
247         14 083 O                            FRTNME     86
248         14 085 O                            YR         91
249         14 090 O                                       92 '-'
250         14 095 O                            NO         97
251         14 100 O                                      105 X'10040100'
252         14 110 O                            CSTORD    121
253         14 120 O                            VIA       140
254         14 130 O                            REQRED    167
255         14 133 O                                      174 '('
256         14 135 O                            PRODCD    177
257         14 137 O                                      178 ')'
258         14 140 O                                      182 X'10040101'
259         14 150 O                            SNOTES    242
260         14 160 O                                      246 X'10040201'
261         14 170 O                            MNOTES    306
262         14 180 O                                      310 X'10040201'
263         14 190 O                            FNOTES    370
264         14 200 O                                      374 X'10040302'
265      S   5 010 O                            NAME      396
266         15 020 O              N75           SNAME     442
267         15 025 O               75                     439 'CALL MIKE BOHRER BEFORE'
```

Figure 5-9. Continuous Output Program NCSC (Part 5 of 9)

CONTINUOUS OUTPUT PROGRAM NCSC

```
1   5 7 9   13   17   21   25   29   33   37   41   45   49   53   57   61   65   69   73   77 80
   UNIVAC OS/3 RPGII VERS 781109             COMPIMS                81/03/18 08.29         PAGE     6

     15 026 O                   75                448 'SHIPPING'
     15 030 O                                     452 X'10040002'
     15 040 O                            ADRES1   474
     15 050 O                  N75       SADRS1   520
     15 055 O                   75 60    CREDIT B 510
     15 060 O                                     530 X'10040002'
     15 070 O                            ADRES2   552
     15 080 O                  N75       SADRS2   598
     15 090 O                                     608 X'10040002'
     15 100 O                            CTYSTA   625
     15 105 O                            ZIPCDE   631
     15 110 O                  N75       SCTY     676
     15 120 O                                     686 X'10040400'
     16 01   OOMA      D        66N24 10
     16 011 O        OR         65N24 10
     16 011 O                                      16 '71'
     16 020 O                                      20 X'10060002'
     16 030 O                   26      ITM,1  B   24
     16 040 O                   26      QTY,1 ZB   29
     16 050 O                   26      MOD,1  B   40
     16 060 O                   26      DES,1  B   58
     16 070 O                   26      MAT,1  B   62
     16 080 O                   26      HP,1   B   68
     16 090 O                   26      VLT,1  B   77 '  0/ /  '
     16 100 O                   26      TYP,1  B   83
     16 110 O                   26      RPM,1 ZB   58
     16 115 O                   26      SCH,1 YB   96
     16 120 O                                     100 X'10040002'
     16 125 O                   26      LT,1   B  104
     16 130 O                   26      DS,1   B  115
     16 140 O                   26      MTR,1  B  127
     16 150 O                   26      MP,1   B  140
     16 155 O                   26                141 '-'
     16 160 O                   26      SHF,1  B  147
     16 170 O                   26      SP,1   B  159
     16 180 O                   26                160 '-'
     16 190 O                   26      BOR,1  B  166
     16 200 O                   26      BLT,1  B  176
     17 010 O                                     182 X'10040002'
     17 020 O                   26      SIZ,1  B  184
     17 030 O                   26      AC1,1  B  188
     17 040 O                   26      AC3,1  B  193
     17 050 O                   26      AC4,1  B  198
     17 060 O                   26      AC5,1  B  203
     17 070 O                   26      AC6,1  B  208
     17 080 O                   26      AC7,1  B  213
     17 090 O                   26      AC8,1  B  218
     17 100 O                   26      AC9,1  B  223
     17 105 O                   26      PRP,1  B  234
     17 110 O                   26                241 'TAG:'
     17 120 O                   26      TAG,1  B  261
     17 130 O                                     265 X'10040008'
     17 140 O                   26      NOT,1  B  325
     17 143 O                   26                329 '('
```

Figure 5-9. Continuous Output Program NCSC (Part 6 of 9)

```
|1   5 7 9   13   17   21   25   29   33   37   41   45   49   53   57   61   65   69   73   77 80|
 UNIVAC OS/3 RPGII VERS 781109               COMPIMS                 81/03/18 08.29        PAGE      7

322      17 145 0                    26      CDE,1   B 331
323      17 147 0                    26                332 ')'
324      17 150 0                                      337 X'10040002'
325      18 030 0                    27      ITM,2   B 341
326      18 040 0                    27      QTY,2  ZB 346
327      18 050 0                    27      MOD,2   B 357
328      18 060 0                    27      DES,2   B 375
329      18 070 0                    27      MAT,2   B 379
330      18 080 0                    27      HP,2    B 395
331      18 090 0                    27      VLT,2   B 394 '  0/ /  '
332      18 100 0                    27      TYP,2   B 400
333      18 110 0                    27      RPM,2  ZB 405
334      18 115 0                    27      SCH,2  YB 413
335      18 120 0                                      417 X'10040002'
336      18 125 0                    27      LT,2    B 421
337      18 130 0                    27      DS,2    B 432
338      18 140 0                    27      MTR,2   B 444
339      18 150 0                    27      MP,2    B 457
340      18 155 0                    27                456 '-'
341      18 160 0                    27      SHF,2   B 464
342      18 170 0                    27      SP,2    B 476
343      18 180 0                    27                477 '-'
344      18 190 0                    27      BOR,2   B 483
345      18 200 0                    27      BLT,2   B 493
346      19 010 0                                      499 X'10040002'
347      19 020 0                    27      SIZ,2   B 501
348      19 030 0                    27      AC1,2   B 505
349      19 040 0                    27      AC3,2   B 510
350      19 050 0                    27      AC4,2   B 515
351      19 060 0                    27      AC5,2   B 520
352      19 070 0                    27      AC6,2   B 525
353      19 080 0                    27      AC7,2   B 530
354      19 090 0                    27      AC8,2   B 535
355      19 100 0                    27      AC9,2   B 540
356      19 105 0                    27      PRP,2   B 551
357      19 110 0                    27                558 'TAG:'
358      19 120 0                    27      TAG,2   B 578
359      19 130 0                                      582 X'10040008'
360      19 140 0                    27      NOT,2   B 642
361      19 143 0                    27                646 '('
362      19 145 0                    27      CDE,2   B 648
363      19 147 0                    27                649 ')'
364      19 150 0                                      656 X'10040002'
365      20 030 0                    28      ITM,3   B 660
366      20 040 0                    28      QTY,3  ZB 665
367      20 050 0                    28      MOD,3   B 676
368      20 060 0                    28      DES,3   B 694
369      20 070 0                    28      MAT,3   B 698
370      20 080 0                    28      HP,3    B 704
371      20 090 0                    28      VLT,3   B 713 '  0/ /  '
372      20 100 0                    28      TYP,3   B 719
373      20 110 0                    28      RPM,3  ZB 724
374      20 115 0                    28      SCH,3  YB 732
375      20 120 0                                      736 X'10040002'
```

Figure 5–9. Continuous Output Program NCSC (Part 7 of 9)

CONTINUOUS OUTPUT PROGRAM NCSC

```
 UNIVAC OS/3 RPGII VERS 781109            COMPIMS                81/03/18 08.29        PAGE    8

     20 125 0               28        LT,3   B 740
     20 130 0               28        DS,3   B 751
     20 140 0               28        MTR,3  B 763
     20 150 0               28        MP,3   B 776
     20 155 0               28                 777 '-'
     20 160 0               28        SHF,3  B 783
     20 170 0               28        SP,3   B 795
     20 180 0               28                 796 '-'
     20 190 0               28        BOR,3  B 802
     20 200 0               28        BLT,3  B 812
     21 010 0                                  818 X'10040002'
     21 020 0               28        SIZ,3  B 820
     21 030 0               28        AC1,3  B 824
     21 040 0               28        AC3,3  B 829
     21 050 0               28        AC4,3  B 834
     21 060 0               28        AC5,3  B 839
     21 070 0               28        AC6,3  B 844
     21 080 0               28        AC7,3  B 849
     21 090 0               28        AC8,3  B 854
     21 100 0               28        AC9,3  B 859
     21 105 0               28        PRP,3  B 870
     21 110 0               28                 877 'TAG:'
     21 120 0               28        TAG,3  B 897
     21 130 0                                  901 X'10040008'
     21 140 0               28        NOT,3  B 961
     21 143 0               28                 965 '('
     21 145 0               28        CDE,3  B 967
     21 147 0               28                 968 ')'
     21 150 0                                  972 X'10040002'
     22 030 0               29        ITM,4  B 976
     22 040 0               29        QTY,4 ZB 981
     22 050 0               29        MOD,4  B 992
     22 060 0               29        DES,4  B1013
     22 070 0               29        MAT,4  B1014
     22 080 0               29        HP,4   B1020
     22 090 0               29        VLT,4  B1029 '  0/ /  '
     22 100 0               29        TYP,4  B1035
     22 110 0               29        RPM,4 ZB1040
     22 115 0               29        SCH,4 YB1048
     22 120 0                                 1052 X'10040002'
     22 125 0               29        LT,4   B1056
     22 130 0               29        DS,4   B1067
     22 140 0               29        MTR,4  B1079
     22 150 0               29        MP,4   B1092
     22 150 0               29                1093 '-'
     22 160 0               29        SHF,4  B1099
     22 170 0               29        SP,4   B1110
     22 180 0               29                1111 '-'
     22 190 0               29        BOR,4  B1117
     22 200 0               29        BLT,4  B1127
     23 010 0                                 1133 X'10040002'
     23 020 0               29        SIZ,4  B1135
     23 030 0               29        AC1,4  B1139
     23 040 0               29        AC3,4  B1144
```

Figure 5-9. Continuous Output Program NCSC (Part 8 of 9)

```
         1   5 7 9   13  17  21  25  29  33  37  41  45  49  53  57  61  65  69  73  77 80
  UNIVAC OS/3 RPGII VERS 781109             COMPIMS                   81/03/18 08.29         PAGE     9

430       23 050 0                29        AC4,4   B1149
431       23 060 0                29        AC5,4   B1154
432       23 070 0                29        AC6,4   B1159
433       23 080 0                29        AC7,4   B1164
434       23 090 0                29        AC8,4   B1169
435       23 100 0                29        AC9,4   B1174
436       23 105 0                29        PRP,4   B1185
437       23 110 0                29                 1192 'TAG:'
438       23 120 0                29        TAG,4   B1212
439       23 130 0                                   1216 X'10040008'
440       23 140 0                29        NOT,4   B1276
441       23 143 0                29                 1280 '('
442       23 145 0                29        CDE,4   B1282
443       23 147 0                29                 1283 ')'
444       23 150 0                20                 1287 X'10060001'
445       23 151 0                65N20              1287 X'10040202'
446       23 152 0                66N20              1287 X'10041202'
447       23 155 0                65N20              1363 'CREDIT  STATUS '
448       23 156 0                65N20              1367 X'10040002'
449       23 157 0                65N20     CREDIT  B1443
450       23 158 0                65N20              1447 X'10040002'
451       23 159 0                65N20              1523 'OTHER LOCATIONS'
452       23 160 0                65N20              1527 X'10040002'
453       23 162 0                65N20     PH      B1591
454       23 162 0                65N20     KY      B1594
455       23 162 0                65N20     WV      B1597
456       23 162 0                65N20     NC      B1600
457       23 165 0                66N20              1363 'CREDIT  STATUS '
458       23 166 0                66N20              1367 X'10040002'
459       23 167 0                66N20     CREDIT  B1443
460       23 168 0                66N20              1447 X'10040002'
461       23 169 0                66N20              1523 'OTHER LOCATIONS'
462       23 170 0                66N20              1527 X'10040002'
463       23 172 0                66N20     PH      B1591
464       23 172 0                66N20     KY      B1594
465       23 172 0                66N20     WV      B1597
466       23 172 0                66N20     NC      B1600
467       23 172 0                66N20     AS      B1603
468       24 010 OOMA      D      24
469       24 020 0                                     20 X'100A0102'
470       24 030 0                                     45 'END OF SHOP COPIES '
471       24 035 0                                     72 'LOAD BILLS OF LADING AND'
472       24 040 0                                     94 'ENTER PROGRAM (NCBL) '
473     S 19 060 OCDA      D      06 10
474       19 070 0                          NXTKEY     13
475       19 080 0               N67                   14 ' '
476       19 090 0                67                   14 '1'
477       19 100 0                20 66                14 '2'
478       19 105 0                25                   16 'EF'
479       19 110 OPIB      D      05
480       19 150 0               N24 10                11 'E'
481     S 12 120 0                24 10                11 'N'
482       12 130 0               N10                   11 'N'
```

Figure 5–9. Continuous Output Program NCSC (Part 9 of 9)

Output message returned by unsuccessful delivery

Issues error message to operator

Lines 223-237 of Figure 5-9 show the output message that action program NCSC generates when the delivery notice returned to the program indicates that the previous continuous output message was not successfully delivered. Notice that this message instructs the terminal operator to examine the printer for what could be causing the difficulty.

Different continuous output messages

Also notice that NCSC generates two different continuous output messages - lines 238-280 and 281-467 - depending on which indicators are set on or off, and that the continuous output messages created are quite lengthy. The only limitation on the size of the message is the screen size of the primary device to which the auxiliary is attached. These messages are being transmitted to a UNISCOPE 200 display terminal.

Saving the key of next record

Receiving control from previous action program

Note that the program uses the continuity data area to save the key of the next record to be processed (line 474) when the program succeeds to itself (line 480). This is a particularly useful tool when the continuous output being generated is producing a report that prints the contents of an entire file. When the successor program is scheduled, it reads the continuity data area. It then does a SETLL using the key saved in the continuity data area. In this way, the successor program begins processing the file at the point where the predecessor action program left off.

Here is an example of the printed output generated by NCSC:

FRANKLIN SUPPLY COMPANY
2552 2nd. Street / Baltimore, Md. (215) 762-8800

Rep's. No.	Rep. Ord. No.	Representative's Name:	Order Date:	Freight:	Penn. Order No.
ØØ1		PHILA. SALES	Ø3/Ø3/81	PREPAID	

Customer Order No.:	Ship Via:	Delivery Requested:
1ØØØØ	B/W	RUSH (DMX)

SPECIAL INSTRUCTIONS:

MARK FOR:

REMARKS:

SOLD TO:	CONSIGNED TO:
J. P. KRANTZ & SON 1662 MEADOWBROOK ROAD CARSON, DELAWARE 76248	J. P. KRANTZ & SON 1662 MEADOWBROOK ROAD CARSON, DELAWARE 76248

ITEM	QTY	MODEL	DESCRIPTION	MAT	H.P.	VOLT/PH/HZ	TYPE	RPM	SCHED. DATE
ACCESSORIES		DISCONNECT	MOTOR CODE	MOTOR PULLEY		SHAFT PULLEY TAG AND FAN NO.			BELT
NOTES									
L	2	BB45	DOMEX	A	1/6	115/1/6Ø		465	Ø3/3Ø
		DS1(A-1)	9A/F1	1VP25-1/2"		6.ØA-5/8"			4L24Ø
19BDD		ABS AP				TAG:EF-7,24			
NOTES		GREASE FITTINGS ON BEARINGS						( )	
M	1	BB45	DOMEX	A	1/6	115/1/6Ø		64Ø	Ø3/3Ø
		DS1(A-1)	9A/F1	1VP25-1/2"		6.ØA-5/8"			4L24Ø
19BDD		ABS AP				TAG:EF-1Ø			
NOTES		GREASE FITTINGS ON BEARINGS						( )	
N	2	BB45	DOMEX	A	1/6	115/1/6Ø		656	Ø3/3Ø
		DS1(A-1)	9A/F1	1VP25-1/2"		6.ØA-5/8"			4L24Ø
19BDD		ABS AP				TAG:EF-13,22			
NOTES		GREASE FITTINGS ON BEARINGS						( )	
P	1	BB45	DOMEX	A	1/6	115/1/6Ø		7Ø2	Ø3/3Ø
		DS1(A-1)	9A/F1	1VP25-1/2"		5.Ø-5/8"			4L22Ø
19BDD		ABS AP				TAG:EF-14			
NOTES		GREASE FITTINGS ON BEARINGS						( )	
Q	1	BB45	DOMEX	A	1/6	115/1/6Ø		495	Ø3/3Ø
		DS1(A-1)	9A/F1	1VP25-1/2"		6.ØA-5/8"			4L24Ø
19BDD		ABS AP				TAG:EF-15			
NOTES		GREASE FITTINGS ON BEARINGS						( )	
R	2	BB45	DOMEX	A	1/6	115/1/6Ø		726	Ø3/3Ø
		DS1(A-1)	9A/F1	1VP25-1/2"		5.ØA-5/8"			4L22Ø
19BDD		ABS AP				TAG:EF-16, 17			
NOTES		GREASE FITTINGS ON BEARINGS						( )	
S	1	BB531	DOMEX	A	1/3	115/1/6Ø		924	Ø3/3Ø
		DS1(A-1)	21A/F1	1VP34-1/2"		4.5A-5/8"			4L21Ø
19BDD		ABS AP				TAG:EF-21			
NOTES		GREASE FITTINGS ON BEARINGS						( )	
T	1	BB45	DOMEX	A	1/6	115/1/6Ø		46Ø	Ø3/3Ø
		DS1(A-1)	9A/F1	1VP25-1/2"		6.ØA-5/8"			4L24Ø
19BDD		ABS AP				TAG:EF-23			
NOTES		GREASE FITTINGS ON BEARINGS							

5.12. CONTINUOUS OUTPUT AND CASSETTE/DISKETTE USE

Functions available

Use

You can read and write, search, or position data on cassette and diskette auxiliary devices by using the continuous output feature. To do this, you move a value to the aux-function and aux-device-no fields of the output message area header just as you do when generating a continuous output message to an auxiliary device. Table 5-6 summarizes the settings for the aux-function field when reading from cassettes or diskettes. Print/transfer options in Table 5-2 also apply to cassette/ diskette.

Table 5-6. Settings for Aux-Function Field of Output Message Header (Read/Search Options)

Devices		Input/Output Options	Contents of aux-function Field			
			Continuous Output		No Continuous Output	
Primary	Auxiliary	Name	Hexa-decimal	Character	Hexa-decimal	Character
	X	Read	D9	R		
		Read Transparent	E2	S		
		Search and Read	E3	T		
		Search and Read Transparent	E5	V		
		Report Address	E6	W		
		Backward One Block	D3	L	E7	X
		Search and Position	E9	Z	E4	U

Most options used only with continuous output

Table 5-6 shows that all the options specified, except backward-one-block and search and position, must be used with the IMS continuous output feature. Backward-one-block and search and position can be used with continuous output and regular output by simply moving the appropriate value to the aux-function and aux-device-no fields.

Input options

There are four input options used with cassette/diskette: read, read transparent; search and read; and search and read transparent. The continuous output feature must be used with all these input options:

Read option

1. **The read option** reads a block of data from the cassette/diskette to the terminal screen. When you specify this option, don't put any message text in the output message area. Also, you must move the value 4 to the text-length field (positions 11–14) of the output message area header.

Read transparent option

2. **The read transparent option** reads a block of data from the cassette/diskette, and the remote device handler deletes the SOE cursor sequence, carriage return codes, and DICE codes.

Search and read option

3. **The search and read option** reads a block of data from the cassette/diskette only if a search argument specified in the message text of the output message area was satisfied. When the argument is satisfied, the block of data is moved to the terminal screen. Your search argument may be in one of three search and read modes. Table 5-7 shows the formats for these modes. When you use the search and read option, only the contents of the output message area message text should be the search argument in the mode you choose.

Search and read transparent option

4. **The search and read transparent option** performs the same function as the search and read option except that the remote device handler removes all DICE sequences, SOE cursor sequences, and carriage return characters from the input message.

Table 5-7. User Message Text for Searching Cassette/Diskette

Permissible search and read arguments

Search Argument Format	Search Type
Ataaaa or 1taaaa or ataaaa	Mode search to position the tape to a particular address and then read one block, where A, 1, or a is constant, and: t Is the track address (1 or 2). aaaa Is the address where the tape is to be positioned.
Btaaaa/c . . . c or 2taaaa/c . . . c btaaaa/c . . . c	Mode search to position the tape to a particular address, search for a specific character string, and read one block, where B, 2, or b is constant, t Is the track address (1 or 2). aaaa Is the block address. c . . . c Is the character string. Up to 16 characters can be specified.
Ct/c . . . c or 3t/c . . . c or ct/c . . . c	Mode search to find the specified character string, where C, 3, or c is constant, and: t Is the track address (1 or 2). c . . . c Is the character string. Up to 16 characters can be specified. The search starts at the present tape position.

Report address option

The report address option displays the address of the cassette/diskette device on the terminal screen. To use this option, you must use the continuous output feature and must specify the value 4 in the text-length field (positions 13–14) of the output message area header.

The two other options available for cassette/diskette are the search-and-position and backward-one-block options. Only these two options can be used with both continuous and regular output messages:

Search-and-position option

- The search-and-position option positions the cassette/diskette to the block requested in the search argument that your action program supplies in the output message text. Your output message text cannot contain any other entries.

Backward-one-block option

- The backward-one-block option repositions the cassette/diskette one block in reverse. The aux-device-no field must be set and the text-length field in the output message area must be 4.

Continuous output message identifier code

When performing these functions, you can also insert into the 4-character continuous-output-code field (positions 9–12) of the output message area header a code that identifies the continuous output message you generated. This code is, as you know from our discussion of IMS delivery codes (5.9), returned to the successor program as part of a 5-character input message. If you do not specify a code, the first four characters of the input message contain binary zeros.

If no code specified

Using the continuous-output-code field

Delivery notice only for unsuccessful transmission

The continuous-output-code field assumes special importance when you use any of the four input options or the report address option for cassettes and diskettes. When you specify one of these options in your action program, a delivery notice is returned to the successor program only if the message was *not* successfully delivered. Otherwise, there is no input to the successor program until a message is transmitted from the cassette/diskette via the terminal screen, or until the auto-transmit feature is set on to allow data to be transmitted from the cassette/diskette.

Screen bypass and the AUTO-TRANSMIT feature

Effect of not setting control page

When using a screen bypass terminal, you must first set the control page for that terminal to take advantage of the auto-transmit capability. If this is not done for any of these five options and a successful delivery notice is returned by the cassette/diskette device, the screen bypass terminal will stay in the interactive mode because no message is sent to IMS.

Importance of continuous output message code

Because a successor action program may receive as input either a delivery notice error or an input message from the cassette or diskette, the CONT-OUTPUT-CODE specified by the predecessor action program should be easily distinguished from the first four characters of any input message being read from the cassette or diskette. In this way, the successor program determines what type of input message it receives (i.e., delivery notice error or input message text) and processes it accordingly. In either case, the successor action program must be capable of handling both unsuccessful delivery notices and standard input messages.

5.13. INITIATING A TRANSACTION AT ANOTHER TERMINAL (OUTPUT-FOR-INPUT QUEUEING)

Definition

The third special capability of an output message generated by an action program is to initiate a transaction at another terminal. We call this output-for-input queueing. It means that an output message generated by that program is queued as input to a terminal other than the source terminal. This terminal is identified by the action program generating the output message. This output message is, in fact, a transaction code that intitiates a transaction at the distant terminal. Figure 5–10 illustrates how this happens.

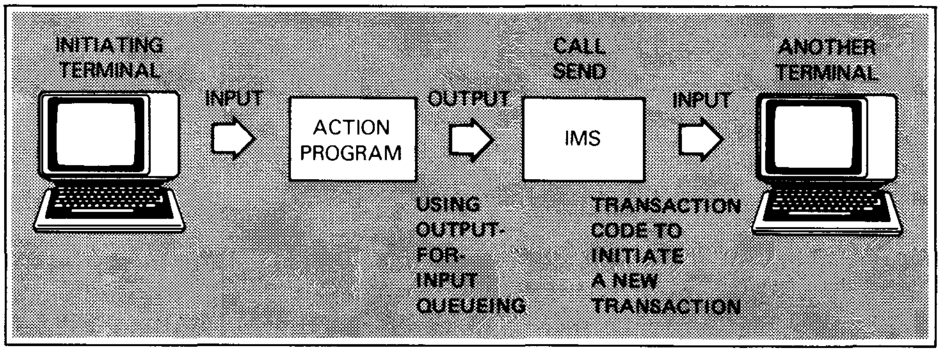


Figure 5–10. Generating Output Message Using Output-for-Input Queueing

Configuration requirement

To use output-for-input queueing, specify CONTOUT=YES in the OPTIONS section of the IMS configuration.

When you configure CONTOUT=YES, IMS automatically includes support for unsolicited output.

5.14. HOW YOU CODE USING OUTPUT-FOR-INPUT QUEUEING

Use CALL SEND to transmit output message

Identifying the terminal receiving output message

You must transmit any output message that initiates a transaction at a different terminal as a CALL SEND. In addition, your action program moves the hexadecimal value C9 or the character I to the aux-function field (position 15) of the output message area header. This value tells IMS to queue the output message generated as input to another terminal. You identify the terminal receiving the input by moving its configured value to the destination-terminal-id field (positions 1–4) of the output message area header. The configured value was specified during communications network definition. Figure 5–11 shows the coding required to accomplish these functions.

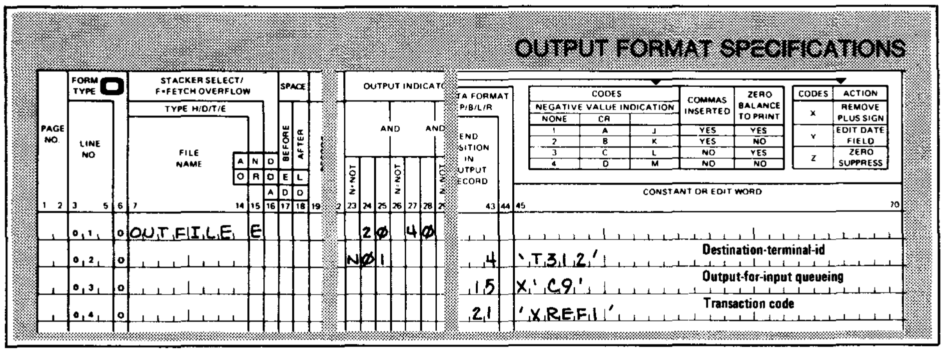


Figure 5–11. Coding an Output Message with Output-for-Input Queueing

Transaction code initiates new transaction

The only other requirement is that the output message contains the transaction code that initiates the new transaction at the destination terminal. This code, and any other output generated along with it, is queued immediately as input to the destination terminal.

Effect of abnormal termination

If, after issuing the CALL SEND using output-for-input queueing, the action program terminates abnormally, the new transaction is still generated at the destination terminal.

Effect of busy destination terminal status

If the destination terminal is in interactive mode when the SEND function is executed, that is, an IMS transaction is already in progress, or if it already has outstanding input messages queued for it, a new transaction can't be scheduled. In this case, the action program issuing the SEND function receives an unsuccessful status-code in the program information block. See 5.17.

Indicating errors to originating program

Reporting output message errors

When an action program generates an output message and requests that it be queued as input to another terminal, IMS validates the output message area header and the status of the destination terminal identified to receive the message. Validation errors are indicated to the originating action program by values returned to the status-code and detailed-status-code fields in the program information block. Any errors found while scheduling the next transaction are reported directly to the destination terminal. Errors found in the action program processing the new transaction at the destination terminal are reported to that action program. As a result, this program must be prepared to handle such error conditions, and if necessary, to report these conditions to the originating terminal.

Error codes

For a complete listing of error codes that IMS returns to the status-code and detailed-status-code fields of your action program following the SEND function, see Table 5-7.

Termination restrictions

Generally, a program that generates output using the output-for-input queueing option terminates with normal termination; however, it can specify external succession. It can't terminate with delayed succession.

5.15. OUTPUT-FOR-INPUT QUEUEING WITH CONTINUOUS OUTPUT

Create records at terminal – print them at another

It is fairly common to use the output-for-input queueing and continuous output options together. For instance, one transaction could create the records you want printed and write them to a MIRAM file. The last stage of this transaction generates an output message using output-for-input queueing at a destination terminal where the printing of the records is actually done. The transaction initiated at the destination terminal reads the MIRAM file and prints the message as continuous output.

5.16. OUTPUT-FOR-INPUT QUEUEING WITH A SCREEN BYPASS DEVICE

Screen bypass

Another situation where you can use the output-for-input queueing is with a screen bypass device on Universal Terminal System (UTS) terminals. This device is defined to the communications network (ICAM) as a logical terminal. However,

Only means of entering input

it has no physical medium for entering input. The only way to access a screen bypass device is to use the output-for-input queueing option. Another terminal in the IMS network generates (through an action program) an output message that initiates a transaction at the screen bypass device. This could be a continuous output transaction, and a report could be generated as output on a printer attached to the screen bypass device.

5.17. MESSAGE SWITCHING

SWTCH transaction

IMS provides a special action program that switches messages from one terminal to another. You need only to enter the transaction code SWTCH at any terminal in your IMS network, identify the destination terminal for the message, and key in the message itself. IMS handles the rest. For more information about this and other terminal commands, consult the IMS terminal users guide, UP-9208 (current version).

Action program initiated message switching

The message switching capability we're interested in here is one that operates from within your own user action program. For instance, an action program could direct error messages to the master terminal when the originating terminal is unable to handle the error. Or, take the case of an action program that initiates a transaction at a distant terminal. The distant terminal could send the originating terminal a message indicating the transaction was initiated or, as the case may be, successfully completed.

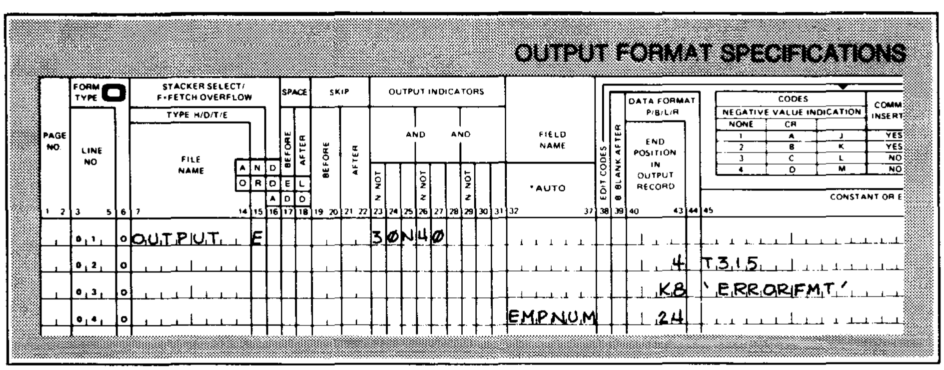
Required coding

To send messages to other terminals, an action program must move a value to the destination-terminal-id field (positions 1–4) in the output message area header. Figure 5–12 shows the coding to send a message to another terminal.

Sending messages to the console

Message size restriction

You can send a message to the system console or master workstation if console support is configured. To send a message to the console or master workstation, enter the name '1CNS' in the destination-terminal-id field. When you send a message to the console, your message may not exceed 120 characters. For more information about the system console and master workstation, see the IMS terminal users guide, UP-9208 (current version).


OUTPUT FORMAT SPECIFICATIONS

Line	Form Type	File Name	Type	Output Indicators	Field Name	End Position in Output Record	Constant or Edit Word
01	O	OUTPUT	E	3ØN4Ø			
02	O					4	T315
03	O					K8	'ERRORFMT'
04	O				EMPNUM	24	

Figure 5–12. Coding for Message Switching

How IMS handles message switching

IMS transmits the message destined for the distant terminal or console by using the SEND function. The message does not go to the destination terminal until the program terminates. In this respect, message switching is handled no differently by IMS than any other output message. (See Figure 5-13.)

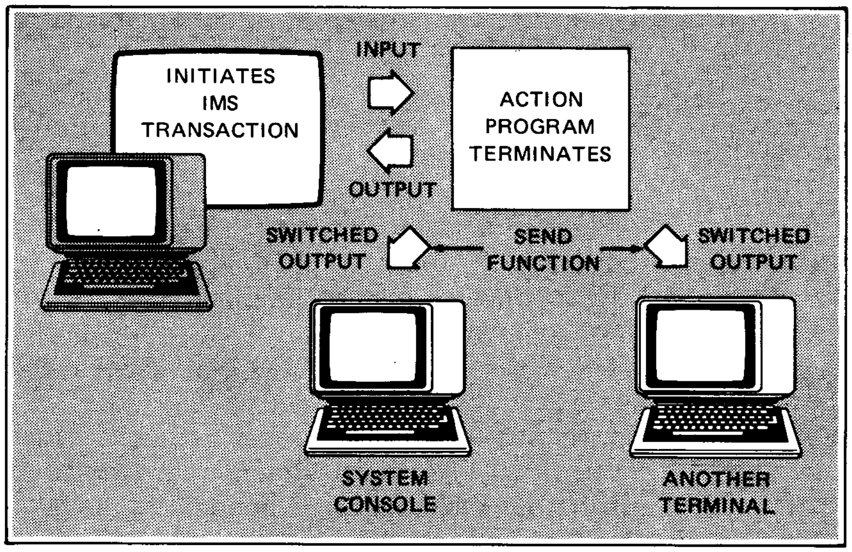


Figure 5-13. Generating Switched Output Message

When transaction terminates abnormally

If the transaction is terminated abnormally or canceled before the action program that generated the messages terminates, all output messages generated are deleted from the output message queue and no messages are delivered. IMS sends a message only to the originating terminal indicating the reason for termination.

Configuration requirements

As we previously mentioned when discussing the SEND function, you should specify disk queueing when generating your communications network if your action programs use the SEND function frequently. Also, you must specify the UNSOL=YES parameter in the OPTIONS section of the IMS configuration to use the SEND function.

5.18. THE IMS SEND FUNCTION AND IMS STATUS CODES

Selecting notification of successful SEND function

In this section, you have seen how many of the output messages generated are transmitted using the IMS SEND function. Whenever the SEND function takes place, if you have specified ERET=YES in the IMS configuration, then IMS notifies the action program whether or not the SEND function was successful. It does this by placing binary values in the status-code and detailed-status-code fields of the program information block. When control returns to the action program, you should interrogate these fields to determine the status of the CALL SEND.

PIB needed to determine SEND function result

To interrogate the status and detailed status code fields, you must define the program information block on the file description form. Also, you must define the two fields and their location on the input form. Status-code occupies positions 1-2 of the program information block; detailed-status-code occupies positions 3-4.

Action program checks SEND status

After the SEND function takes place, the program should read the status and detailed status code fields to determine whether or not the SEND was successful. These fields are extremely important to a programmer when debugging action programs. Debugging is discussed in detail in Section 7.

Result of not being notified of unsuccessful SEND function

If you don't specify ERET=YES, and the CALL SEND isn't successful, the action program does not regain control and IMS abnormally terminates your action program. We strongly recommend that you always configure ERET=YES.

Status codes

Table 5-8 lists the values that IMS can return after the SEND function takes place.

Trace values

IMS returns trace values to the status-code and detailed-status-code fields when ERET=YES is configured.

Table 5-8. Status Codes and Detailed Status Codes Returned Following the Send Function

Status Code (Decimal)	Detailed Status Code (Decimal)	Description
0		Successful
3	3	Parameter error
3	12	UNSOL=YES or CONTOUT=YES wasn't configured, or no process files were created in ICAM network definition.
6	2	Returned when output-for-input queueing is requested and: 1. destination terminal is in interactive mode; 2. destination terminal has an input message on queue; 3. ZZHLD or ZZDWN command was entered for destination terminal; 4. destination terminal is marked physically down to ICAM; or 5. IMS can't allocate a main storage buffer (multithread only); INBUFSIZ specification is inadequate.
6	3	Destination terminal physically or logically down; message queued
6	4	Invalid destination terminal, auxiliary device, or auxiliary function specified
6	5	No ICAM network buffer available
6	6	Disk error, or recoverable system error on output message to console
6	7	Invalid length specification

Detailed status code=2

IMS returns a status code of 6 and a detailed status code of 2 only when you use the SEND function to initiate a transaction at another terminal (output-for-input queueing). The conditions causing this error are not permanent. The output message header is valid, and you may be able to retransmit the same message successfully at a later time.

Detailed status code=3

Some of the conditions causing a detailed status code of 3 (with status code 6) are the same as those for a detailed status code of 2. However, this error is returned when you use the SEND function for message switching, not output-for-input queueing. In this case, the message sent is queued for the destination terminal and is automatically transmitted when the terminal is operational.

Detailed-status-code=4

On the other hand, when internal message control returns the detailed-status-code value 4 after the SEND function, this means that the contents of the output message area header are not valid. Any effort to retransmit the same message is unsuccessful.

When this value is returned, check your action program for one of the following errors:

- **The value in destination-terminal-id** (positions 1–4) of the output message area header is not a valid configured terminal identification.

- **The value in auxiliary-device-id** (position 16) of the output message area header is invalid.

- **The value in aux-function** (position 15) of the output message area header contains the hexadecimal value C3, F3, or F7, indicating that the program attempted to generate continuous output. You cannot transmit continuous output as a CALL SEND; it must always be transmitted as a CALL RETURN when the program terminates (5.7). If the message was addressed to the system console (destination-terminal-id 1CNS), only the hexadecimal values 00 or C9 are acceptable.

5.19. DISCONNECTING A LINE FROM AN ACTION PROGRAM

Purpose

Configuration requirements

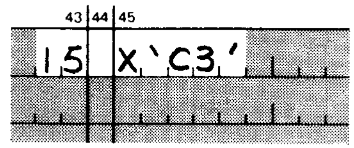
Available only for dedicated networks

The line disconnect feature allows an action program to disconnect a single-station dial-in line following the delivery of its output message to enable another terminal to dial in on the same line. To use the line disconnect feature, you must include the continuous output capability in your configuration by specifying CONTOUT=YES in the OPTIONS section. The line disconnect feature is available only in a dedicated ICAM network, not a global network.

To disconnect a line after message transmission, the action program must:

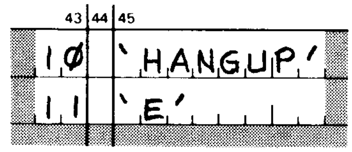
Aux-function value, X'C3'

- place a continuous output flag (X'C3') in the aux-function byte (position 15) of the output message header; and



Use external succession and HANGUP successor-id

- specify external succession with 'HANGUP' as the successor by setting the termination-indicator (position 11) in the program information block to E and the successor-id (position 5) to 'HANGUP'.



HANGUP, IMS action program

HANGUP is an action program supplied by IMS that terminates with a special code causing IMS to issue a line release/line request sequence to ICAM to disconnect the line.

Delivery notice before scheduling

After the output message is sent, no further input is required from the terminal operator. IMS waits for ICAM notification of message delivery before scheduling the external successor, HANGUP. In this way, delivery of the message prior to the line disconnect is ensured.

Figure 5-14 shows the output specification form coding used to disconnect a line from an action program.

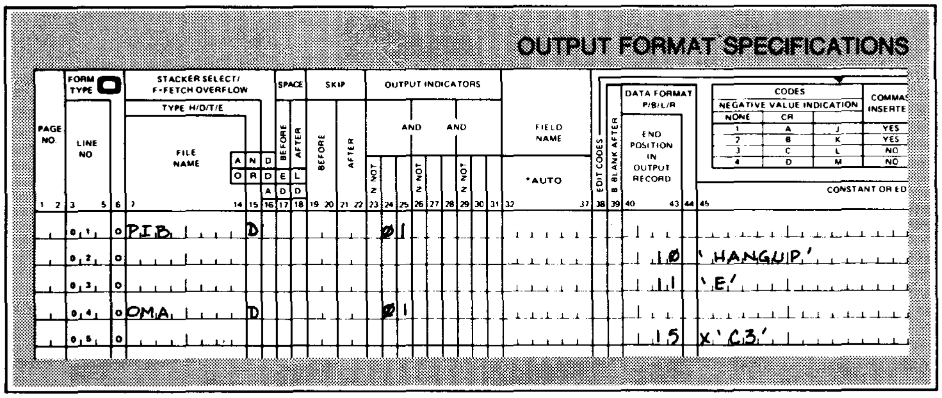


Figure 5-14. Coding a Line Disconnect from an Action Program

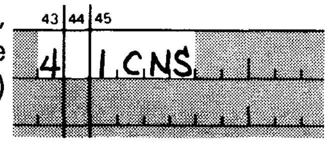
5.20. SENDING MESSAGES TO THE SYSTEM CONSOLE

Configuring console support

Your action program can send output messages to the system console if console support is configured. You configure console support by specifying OPCOM=YES in the OPTIONS section of the IMS configuration or by not specifying a master terminal in any TERMINAL section.

Terminal-id is 1CNS

To send output to the system console, place the terminal-id 1CNS in the destination-terminal-id field (positions 1–4) of the output message header.



When IMS session has a master workstation

Sometimes an IMS session has a master workstation associated with it. A master workstation is a workstation from which the IMS start-up job control stream is entered, or it may be defined in the job control stream. When there is a master workstation and you use the destination-terminal-id 1CNS, your output message goes to the master workstation instead of the console. When the master workstation logs off or is disabled, then the message goes to the console.

Types of output you can send

You can send normal output, multiple output, switched output, continuous output, and output-for-input queueing messages to the system console. However, there are certain restrictions on output to the console:

Auxiliary devices not supported

- You cannot send output to an auxiliary device at the system console. The only auxiliary function settings you can use are hexadecimal 00, C3 (continuous output), or C9 (output-for-input queueing).

Message length restriction

- The maximum length of the output message is 120 characters, not including the output message header. Additional characters are truncated.

No screen formats

- Because of the message length restriction, you cannot output a screen format to the console.

Messages not edited

- Output messages are not edited. DICE functions, FCCs, and other control characters appear as blanks, or in a few cases as printable characters.

No message waiting signal

- There is no message waiting signal. Switched and multiple output messages are sent out immediately.

Error Returns on Output to the Console

Auxiliary device error

IMS returns a status code of 6 and detailed status code of 4 when you attempt to send output to an auxiliary device at the system console. These are the same codes IMS returns when you have an invalid destination-terminal-id, auxiliary device, or auxiliary function specification on output messages to regular terminals.

When console is down

When your output message can't be delivered because the console is physically or logically down, the action IMS takes depends on the type of output message.

Switched and continuous output messages

- With a switched message, IMS returns a status code of 6 and detailed status code of 6. With a continuous output message, IMS returns a delivery notice status of X'86'. These codes indicate recoverable system errors.

Other output messages

- With other types of output messages (such as normal output in response to input from the console), IMS returns a successful status code of 0. The reason IMS does this is that an error status would cause a ''TRANSACTION CANCELLED'' message to be sent to the console, and this could cause an abnormal termination of the IMS session.

6. Using Screen Format Services To Format Messages

6.1. DISPLAYED FORMATTED SCREENS

In Section 4, we briefly discussed using screen format services to format output messages. The sample action program JAADD1 used screen format services to generate its output screens.

Saves programming effort

With screen format services, generating output screens is easy because the screens are predefined using the screen format generator. You don't have to include device control characters in your action program. In addition, screen format services does validity checking of input data, thereby reducing the amount of input validation you must do in your action program.

6.2. DEVICES SUPPORTING SCREEN FORMAT SERVICES

Terminals supporting screen formats

You can direct screen formats to any display terminal supported by IMS except the IBM 3270 terminal, and also to auxiliary devices attached to display terminals. UNISCOPE 100 and UNISCOPE 200 terminals must have the screen protection feature, and UTS 400 terminals operating in native mode must have the **PROTECT/FCC** switch set to **FCC** and the control page set to **XMIT VAR.** For local workstations, specify a line buffer length of at least 900 words on the LBL option of the ICAM network definition.

Local workstation consideration

6.3. GENERATING SCREEN FORMATS

Screen formats generated offline

You define your screen formats offline from IMS by executing the screen format generator. (See the screen format services concepts and facilities, UP-8802 (current version).) When you create each screen format, you assign a unique name to it. The screen format generator stores the formats in the system screen format library YFMT or other MIRAM disk file. The screen formats for an IMS session may reside in one or two screen format files.

Formats stored for later use

NOTE:

To use screen format services, you must generate a supervisor in consolidated data management (CDM) or mixed mode. However, you can configure IMS in either CDM or DTF mode. See the IMS system support functions user guide, UP-8364 (current version).

6.4. CONFIGURATION REQUIREMENTS

When using screen format services, you must give special consideration to four parameters at IMS configuration:

Affected parameters

1. the **SFS=n** parameter;
2. the **RESFMT=n** parameter;
3. the **WORKSIZE=n** parameter; and
4. the **OUTSIZE=n** parameter.

SFS Parameter

Number of terminals using screen formats

You must include the SFS parameter in the OPTIONS section of your IMS configuration. With this parameter, you specify the maximum number of terminals that will use screen formats at the same time. Be sure to specify a large enough number of terminals. A screen format is considered in use at a terminal from the time the operator requests it until the format is displayed, input entered, and the input acknowledged.

RESFMT Parameter

Number of resident screen formats

With the RESFMT parameter, also in the OPTIONS section, specify the number of screen formats you want retained in main storage between calls to screen format services. The default is 1 for single-thread IMS and 3 for multithread.

The WORKSIZE Parameter

Work area required

Determining size

You must configure a work area for each action program using screen format services. The RPG II action program itself does not use this area, but the compiler does. You include the WORKSIZE parameter in the ACTION section of the configuration. Its format is WORKSIZE=n. The n denotes work area size. The size you specify must be large enough to accommodate all variable output data generated by the action program plus 99 bytes for the RPG II indicators.

The OUTSIZE Parameter

Maximum OMA size

Specify the OUTSIZE parameter in the ACTION section of the configuration (OUTSIZE=n). The n denotes the maximum size of the output message area for a particular action.

Where the screen format is built

Using output message area

When you request a screen format in your action program, you have it built in the output message area or in dynamic main storage. If you use the output message area, it must be large enough to handle the screen format buffer constructed by the screen format coordinator. This buffer contains all variable output data, display constants, and device control characters. See the IMS system support functions user guide, UP-8364 (current version) for information on calculating the size of the output message area.

Using dynamic main storage

The advantage of building the screen format in dynamic main storage is that you don't have to calculate the size needed for the format buffer. You must still allocate an output message area large enough to contain the output message header and your variable data fields. The OUTSIZE=STAN specification will give you an adequate output message area size.

When OUTSIZE is insufficient

When the action program requests a screen format and the output message area is not large enough to contain the format buffer, IMS returns an error code in the status fields of the program information block. IMS also places the output message area size required in the text-length field (positions 13–14) of the output message area header.

6.5. REQUIREMENTS AT IMS START-UP

Device assignment sets

LFD names

When using screen format services, you must include a device assignment set for each screen format file in the job control stream at IMS start-up. Use the LFD name TC01FMTF for the primary file and TC02FMTF for the secondary file, if there is one.

Figure 6–1 illustrates the steps required to create and use screen formats with IMS.

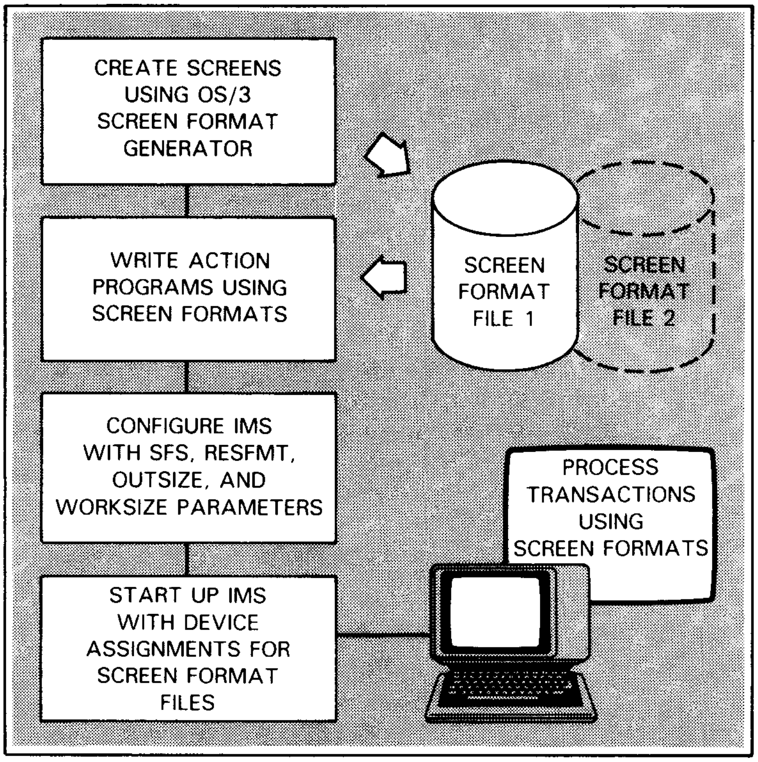


Figure 6–1. Creating and Using Screen Formats

6.6. HOW IMS HANDLES SCREEN FORMATTED MESSAGES

Retrieves screen format

When your action program requests a particular screen format, IMS retrieves the format from the screen format file and places it in the output message area or in dynamic main storage. (When you assign two screen format files, IMS checks TC01FMTF first, then TC02FMTF.)

Variables moved to work area

The variables in the output message area are moved to the work area defined at configuration. The variables remain there for as long as it takes the screen format coordinator to construct the screen in the buffer area.

Display contents moved to screen buffer

The screen format coordinator places the output display constants of the format into their respective locations within the screen buffer. These constants are always protected.

Variables moved to screen buffer

When the screen is built, the screen format coordinator inserts the variable data from the work area into the appropriate locations in the screen buffer.

Screen displayed on terminal

When the program terminates, the screen format and variable data are transmitted to the terminal.

Example

Figure 6-2 shows an output screen containing display constants and variable data. Underlines represent input fields.

```
                    PERSONAL CREDIT REPORT
NAME:JOHN DOE
ADDR:1552 MAIN ST.        STATE:PA            ZIP:19140
ACCOUNT NO:193-A564
BALANCE:350.00
PAYMENT:______          DATE:__/__/__
```

Figure 6-2. Output Screen Format with Display Constants, Variable Data, and Input Fields

Using input and output screens

Example

Any field you define as input, or both input and output, in your action program is an unprotected field. This means that the terminal operator is free to change that field when making entries on the screen format. It is protected if you define a variable data field as output only when you build a screen buffer. In Figure 6-3, the terminal operator has changed the address field and entered a payment amount and date.

```
                        PERSONAL CREDIT REPORT
NAME:JOHN DOE
ADDR:224 PINE ST.       STATE:PA          ZIP:19102
ACCOUNT NO:193-A564
BALANCE:350.00
PAYMENT:25.00          DATE:12/23/80
```

Figure 6-3. Input Screen Format with Display Constants and Changed Input Fields

Output-only screens required for: delayed succession continuous output message switching

When your action program terminates with delayed succession or uses continuous output, IMS forces the screen format to be output only. Also, you must use an output-only format for any screen formatted output message switched to a different terminal.

Function keys cancel screens

The **message wait** key and function keys cancel any screen format currently effective at the terminal.

When multiple screens are generated

An action program may send multiple formatted messages to the originating terminal; however, only the last format may be used for entering data as input to the successor program.

6.7. USING FORMATTED SCREENS FOR INPUT

Checking input for terminal commands

When the terminal operator fills in input data, the data is validated before IMS passes control to the successor program. IMS checks the message for terminal command input before requesting the screen format coordinator to validate the entries.

All commands cancel screens except ZZRSD

If the input message contains a terminal command other than ZZRSD, IMS processes it accordingly and cancels any screen format currently effective at the terminal.

Results when ZZRSD is entered

Normally, ZZRSD causes the last output message to be sent again, thus retaining the current screen format. However, if the screen format is built in dynamic main storage instead of the output message area, it can't be sent again and the screen format is canceled. The terminal operator receives a NO MSG IN QUEUE message and can't enter input on the formatted screen.

When an invalid transaction code is entered

When the input message contains a transaction code, IMS verifies the code and if it is invalid, sends the message back to the terminal and blinks the transaction code. This does not cancel the screen format currently effective at the terminal.

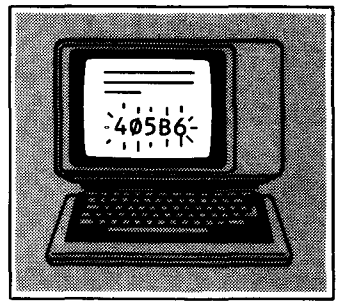
Validating input data

No other editing performed on input

When the input message does not contain a terminal command or invalid transaction code, IMS requests the format coordinator to validate the message. If the input data filled in by the terminal operator is valid, IMS places only that data into the input message area of the successor action program. IMS does not perform any other editing on this input. Your action program then begins processing.

When input data is invalid

When some of the input data is invalid, the screen format coordinator blinks the invalid fields. The terminal operator can correct the input until the retry count specified at screen format generation time is exhausted. (See screen format concepts and facilities, UP-8802.)



Error codes returned for invalid data

Once the retry count is exhausted, the successor program receives control. At that point, the program information block contains a status code of 7 and a detailed status code of 0. (See Table 6-1 for a description of error codes returned when using screen format services.)

Specifying type of termination

In order for the successor program to receive this data, the predecessor action program must specify E in the termination-indicator field (position 11) of the program information block. If that program terminated with normal termination (N in the termination-indicator field), the first input field entered on the screen format must be a valid transaction code that will schedule the appropriate action program to process the input data.

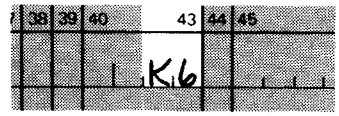
6.8. CODING REQUIRED TO USE SCREEN FORMAT SERVICES

Output form coding

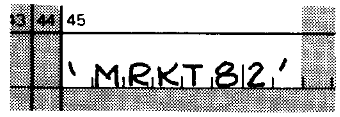
To use a formatted screen, you make the following entries on the output form:

Required entries

- The character **K** in column 42
- **Length** of the screen format name in column 43

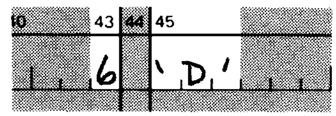


- **Screen format name** itself in columns 45–70



To build screen in dynamic main storage

To build the screen format in dynamic main storage, move 'D' to the SFS-location field (position 6) of the output message header.



Example

Figure 6–4 illustrates how you code the output form to build a screen format containing variable data in dynamic main storage.

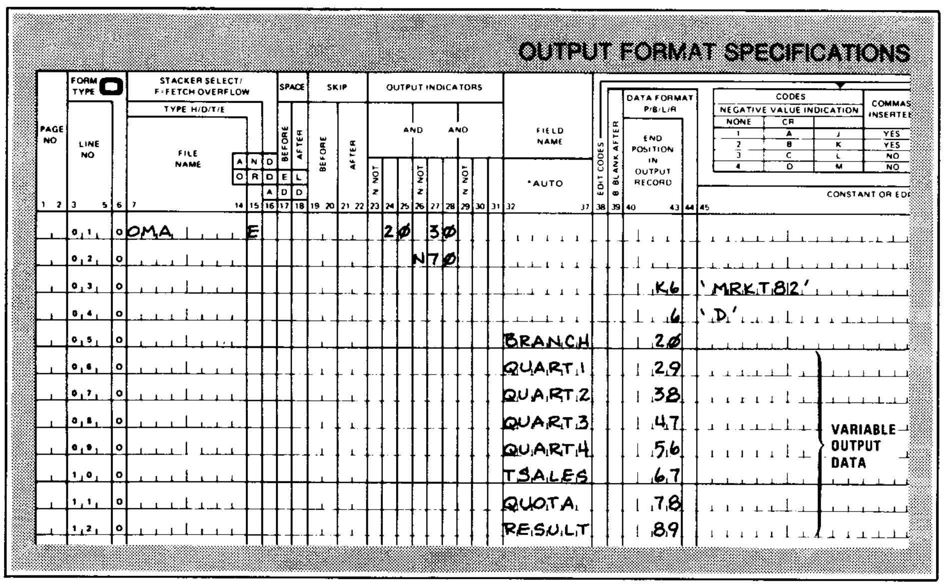


Figure 6–4. Coding the Output Form to Use Screen Format Services

Defining the screen format

You indicate that you are using a screen format on the first field description for the output file. Only one screen format is allowed for each output record. In Figure 6-4, the output file is OMA. As you notice, the screen format is the first field description for the file. The character K in column 42 indicates you are using screen format services. The number 6 in column 43 is the length of the format name. MRKT82 is the format name as it was defined at screen format generation.

List variable output data in receiving order

You must list the variable fields in the order that the screen format expects to receive them. The first field always begins after position 16. You must allow 16 positions for the output message area header.

Figure 6-5 shows the screen format described in Figure 6-4 as it appears at the terminal.

```
                      MARKETING SUMMARY '82
                   COLONIAL STEEL CORPORATION
                           BRANCH: 7Ø18
                          SALES SUMMARY
QUARTER 1:  $345,678,721       QUARTER 3:  $322,628,456
QUARTER 2:  $299,799,838       QUARTER 4:  $349,798,951
            TOTAL SALES:       $1,317,905,966
            YEARLY QUOTA:      $1,288,988,955
            RESULTS:           $28,916,971 +
```

Figure 6–5. Output Screen Display for Figure 6–4.

Handling screen formatted output

IMS handles output messages that use screen format services just like any other output message. They can be transmitted using the SEND or RETURN function. However, they do not appear at the terminal until the action program terminates. The terminal operator may then enter data, which is verified and stored in the successor program's input message area.

6.9. GENERATING AN OUTPUT SCREEN WITH NO VARIABLE DATA

When there is no variable output data

When an action program generates an output screen with no variable fields, such as an error message screen, you must move zeros to the text-length field of the output message area header before specifying the screen format. Figure 6-6 shows how you code the output form to do this.

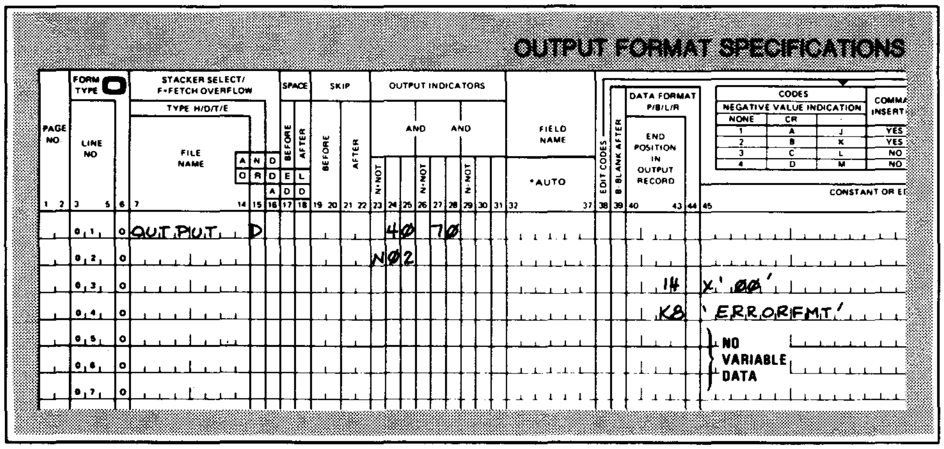


Figure 6–6. Coding for a Formatted Screen without Variable Output Data

6.10. ERROR CODES RETURNED BY IMS

Errors return status codes to PIB

When IMS encounters a problem while using screen format services, it returns values to the status-code and detailed-status-code fields of the program information block. Table 6-1 lists and describes these values.

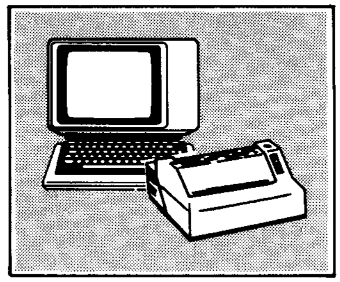
Table 6-1. Error Codes Returned by IMS when Using Screen Format Services

Status Code (Decimal)	Detailed Status-Code (Decimal)	Description
1	-	Named format can't be found
3	12	Screen format services not configured
6	4	Invalid terminal name or type
7	0	Validation error; all error fields within variable data area are replaced by hexadecimal F's.
7	1	Format area not large enough. The OUTSIZE=n specification wasn't large enough to handle screen format, variable data, and device control characters. IMS returns the correct output message area size to the text-length field (positions 13–14) of the output message area header.
7	2	Variable data area not large enough. The WORKSIZE=n specification wasn't large enough to handle the variable data plus the 99 bytes for RPG II indicators.
7	3	Insufficient number of terminals was configured.
7	4	Variable data specified for input format is invalid.
7	5	Format width is greater than screen width.
7	6	Fatal error (I/O error)
7	10	Screen format incorrectly generated
7	11	System error
7	16	Inadequate main storage available in system; or format contains protected fields and terminal doesn't have protect feature or isn't in protect mode.
7	17	Screen format services error
7	18	Action program processing DDP transaction attempted to send screen format to initiating action program.

See Appendix C for a complete listing of status and detailed status codes in hexadecimal.

6.11. TRANSMITTING FORMATTED SCREENS TO AN AUXILIARY DEVICE

You can output a screen format to an auxiliary device – printer, cassette, or diskette – attached to a display terminal.



Setting output message header fields

To output a screen format to an auxiliary device, you move a value to the aux-function (position 15) and the aux-device-no (position 16) of the output message area header before specifying the screen format required.

Aux-function field entries

Table 6-2 lists the values you move to the aux-function field to accomplish this. Different values are specified for the aux-function field depending on whether the action program is using continuous output or not.

Example

Figure 6-7 shows the coding to transmit a formatted screen to a printer attached to a UTS 400 display terminal using print mode with space suppression. The action program involved is not generating continuous output.

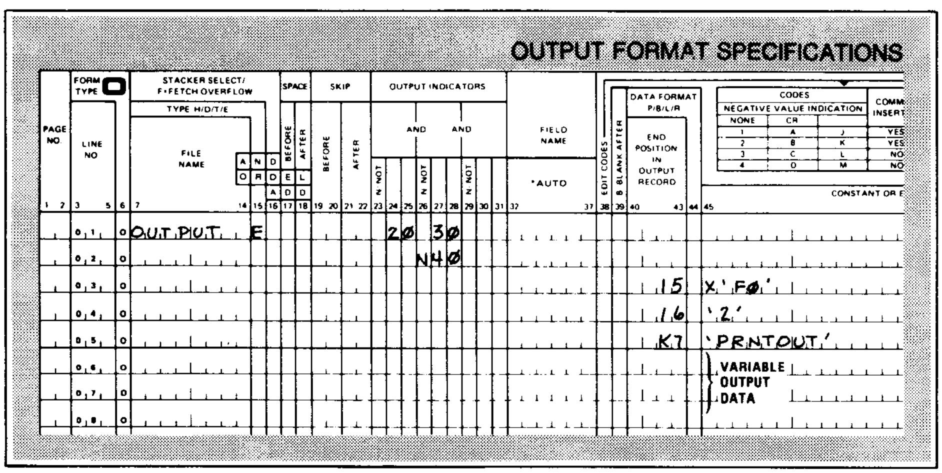


Figure 6-7. Coding to Transmit Formatted Screen to a Printer

NOTE:

When you build a screen in dynamic main storage, all values, including auxiliary device numbers and functions, must be present in the output message header before the call is issued to screen format services. If any header values (except SFS-options) are changed after the call to screen format services, the new values are ignored.

Table 6-2. Print/Transfer Options for Writing of Screen Formats to Auxiliary Devices

Input/Output Options			Contents of aux-function Field				Auxiliary Devices			
			Continuous Output		No Continuous Output		UTS 400		UNISCOPE 100/200	
Name	Suppression	Inhibit Space Suppression	Hex	Character	Hex	Character	Supported	Not Supported	Supported	Not Supported
Print Mode	X		F3	3	F0	0	X (recommended) ①③		X (recommended) ①	
		X	F5	5	F2	2	X (recommended) ①③			X (unpredictable output at screen and auxiliary device)
Print Transparent	X		F7	7	F4	4	X ②③		X ②	
		X	F9	9	F6	6	X ②③			X (unpredictable output at screen and auxiliary device)
Print Form (ESC H)	X		C1	A	D1	J	X ④			X ⑥
		X	C6	F	D6	0	X ④			X ⑥
Transfer All (ESC G)	X		C2	B	D2	K	X (recommended) ⑤			X ⑥
		X	C7	G	D7	P	X ⑤			X ⑥
Transfer Variable (ESC F)	X		C4	D	D4	M	X ④			X ⑥
		X	C8	H	D8	Q	X ④			X ⑥
Transfer Changed (ESC E)	X		C5	E	D5	N		X (field control characters not supported)		X ⑥
		X	E8	Y	F8	8		X (field control characters not supported)		X ⑥

LEGEND:

① Printer - same format as screen

② Printer - same information as screen; no carriage returns

③ Cassette/diskette - same format as screen; no field control characters

④ Cassette/diskette - same format as screen; only records unprotected fields

⑤ Cassette/diskette - same format as screen; records all fields and all field control characters

⑥ Cassette/diskette - not available

7. Action Programming in a Distributed Data Processing Environment

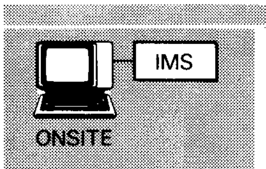
7.1. BASIC DDP REQUIREMENTS AND TERMINOLOGY

DDP requirements

IMS handles distributed data processing (DDP) transactions through the IMS transaction facility. To use distributed data processing with IMS, you must include the IMS transaction facility in your software at each OS/3 system and must configure multithread IMS at each system. Also, you must define a global ICAM network that supports distributed data processing and include a LOCAP section in the IMS configuration for each IMS system where you want to route transactions or which will route transactions to you. Consult the IMS system support functions user guide, UP-8364 (current version) for configuration and network definition requirements.

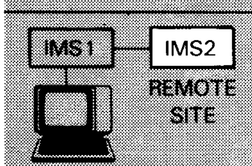
DDP terminology

Let's define some terms we'll be using throughout the discussion of DDP transaction processing:



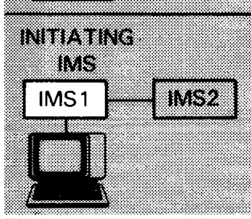
<u>LOCAL TRANSACTION</u>

Transaction that is processed at the same IMS system where it is initiated



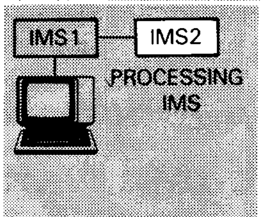
<u>REMOTE TRANSACTION</u>

Transaction that is initiated at one IMS system and processed at another



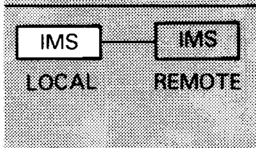
<u>PRIMARY IMS</u>

IMS system where a remote transaction is initiated. In our illustrations we call this system IMS1.



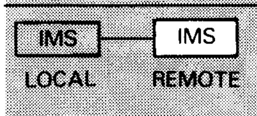
SECONDARY IMS

IMS system where a remote transaction is processed. The action programs processing the transaction and any files they access are located here. In our illustrations we call this system IMS2.



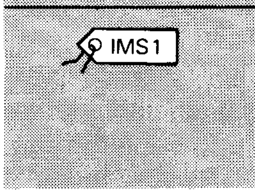
LOCAL IMS

Your IMS system, regardless of whether your system is primary or secondary for a particular transaction



REMOTE IMS

IMS system at another computer



LOCAP-NAME

The 4-character label of a LOCAP macroinstruction in your ICAM network definition, identifying a local or remote IMS system

7.2. HOW IMS ROUTES REMOTE TRANSACTIONS

Transaction routing types

There are three different ways in which the primary IMS can route a transaction to a secondary system:

ROUTING A TRANSACTION TO SECONDARY SYSTEM

1. Directory routing

The terminal operator enters a transaction code that identifies a transaction at a secondary system. The transaction code is defined in the configurator TRANSACT section.

2. Operator routing

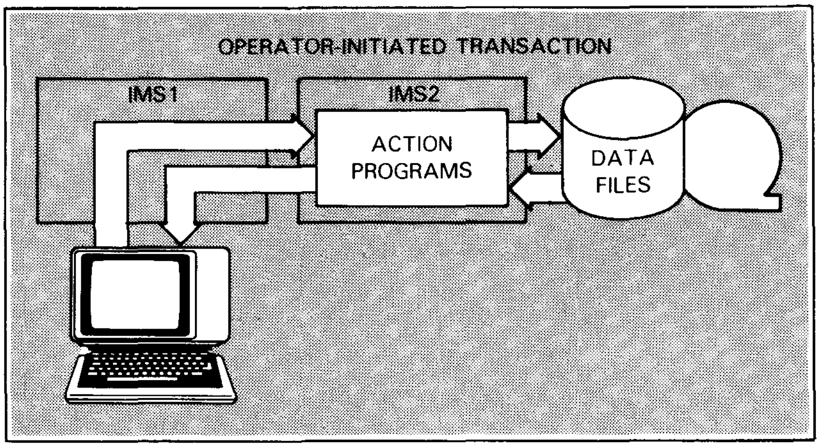
The terminal operator prefixes the transaction code with a route character (followed by a period) that routes the transaction to a secondary system. This route character is defined in the configurator LOCAP section or in a PARAM job control statement at IMS start-up.

3. Action program routing

The terminal operator enters a transaction code that initiates a transaction at the primary system. The action program processing this local transaction issues an ACTIVATE function call to initiate a transaction at a secondary system. Action programs initiating remote transactions are written in COBOL or basic assembly language (BAL).

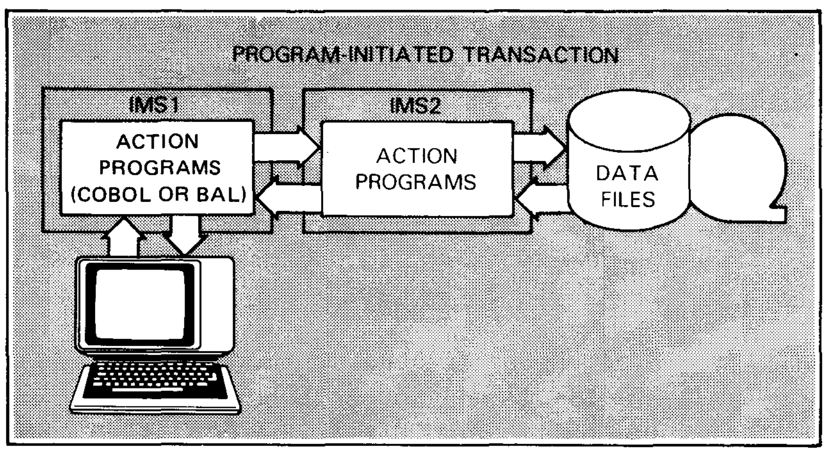
Operator-initiated transaction

From the programmer's viewpoint, directory and operator routing are the same, because they are both initiated by a terminal operator. Once the transaction is routed to the secondary system, an action program or series of action programs at that system interacts with the terminal operator the same way as in a local transaction. No action programs are involved at the primary system.



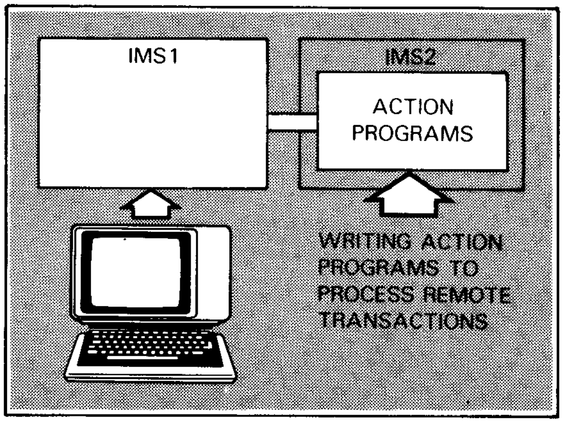
Program-routed transaction

With action program routing, action programs at the secondary system don't interact directly with the terminal operator. They return a message to the initiating action program or its successor, which in turn, outputs a message to the terminal operator.



7.3. PROCESSING A REMOTE TRANSACTION

As an RPG II programmer, you may be writing action programs at a secondary IMS to process transactions initiated by an operator or an action program at a primary IMS system.



Similar to processing local transaction

There is little difference between the way you process a remote transaction and the way you process a local transaction. You can probably use the same action programs to process both local and remote transactions.

Receiving input message

When the transaction begins, you receive an input message starting with a 1- to 8-character transaction code, just as with a local transaction.

Determining input message source

You can determine the source of the input message by testing the DDP-mode field of the program information block and the source-terminal-id field of the input message header.

DDP-mode field

The DDP-mode field (position 70 of the program information block) contains the value 'R' when the transaction is operator-initiated (either directory routing or operator routing). It contains the value 'A' when the transaction is initiated by an action program. When a transaction is local, the DDP-mode field contains zeros. This field has other possible values but they apply to action programs at the primary IMS system.

Source-terminal-id field

When an action is scheduled to process a transaction at a secondary IMS, the source-terminal-id field (positions 1–4 of the input message header) contains the locap-name of the IMS system originating the transaction rather than a terminal-id. You can't test for the actual terminal initiating a remote transaction.

General restrictions

There are a few general restrictions on processing remote transactions. (There are several additional restrictions for program-initiated remote transactions, which we'll discuss a little later.)

SEND function restriction

- You can't use the SEND function to output a message to the originating terminal (or any terminal at the remote IMS). However, you can use the SEND function to output a message to a terminal at your local IMS. (See 5.17.) Afterwards, clear the destination-terminal-id field (positions 1–4 of the ouput message header) or move the source locap-name to that field before sending an output message to the originating terminal.

Continuous output restriction

- You can't send continuous output to the originating terminal. Again, you can use the SEND function to initiate continuous output at a local terminal using output-for-input queueing.

Auxiliary device restriction

- You can't send output to an auxiliary device attached to the originating terminal. However, you can send output to an auxiliary device at a local terminal using the SEND function.

7.4. PROCESSING AN OPERATOR-INITIATED REMOTE TRANSACTION

With the few exceptions we've already mentioned, you process an operator-initiated remote transaction the same way as a local transaction.

Action program succession

You can use any type of action program succession with operator-initiated transactions. Once the transaction begins, the IMS transaction facility establishes a communications link which stays in effect until the transaction ends. When you use external succession, the terminal operator receives and responds to your output messages without entering any additional codes.

Figure 7-1 illustrates a remote dialog transaction, using both internal (either immediate or delayed) and external succession.

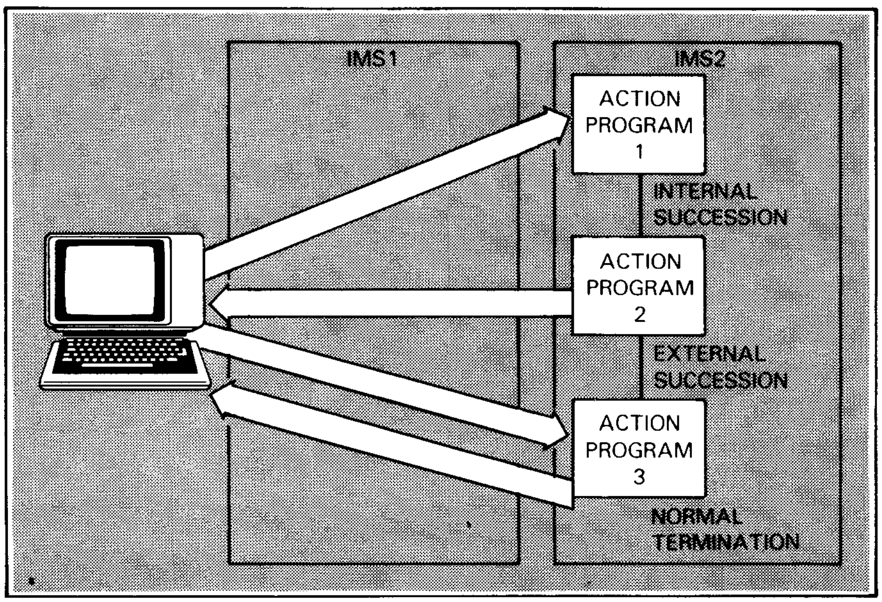


Figure 7-1. Processing an Operator-Initiated Remote Dialog Transaction

Screen format services in DDP

You can use screen format services with operator-initiated remote transactions. (See 7.6.)

7.5. PROCESSING A PROGRAM-INITIATED REMOTE TRANSACTION

When a remote transaction is initiated by an action program, you send an output message back to the originating action program's successor. That action program in turn outputs a message to the terminal operator.

Considerations and restrictions

Because your output message goes to an action program rather than to a terminal, there are a few additional considerations and restrictions:

Output message formatting

1. **You may want to format your output message differently;** you don't need control characters. Of course, you may want to use the same output message for either operator- or program-initiated transactions. In this case, the action program receiving your message must be prepared to receive your control characters.

Screen formatting restriction

2. **You can't use a screen format for the output message** you return to the originating action program or its successor. However, you can use the SEND function to display a screen format at a local terminal.

Allowable termination types

3. **You must use normal termination when** you return an output message to the originating action program's successor. You can't use external succession. You can, however, use immediate or delayed internal succession and have your successor program return the output message (Figure 7-2).

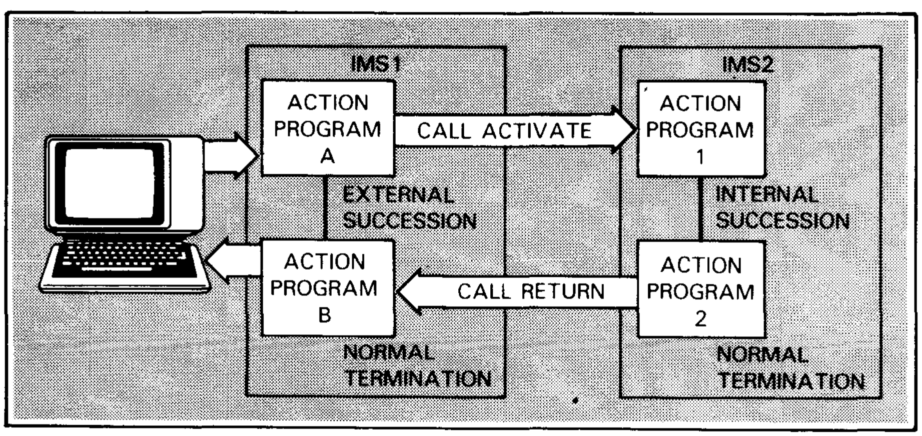
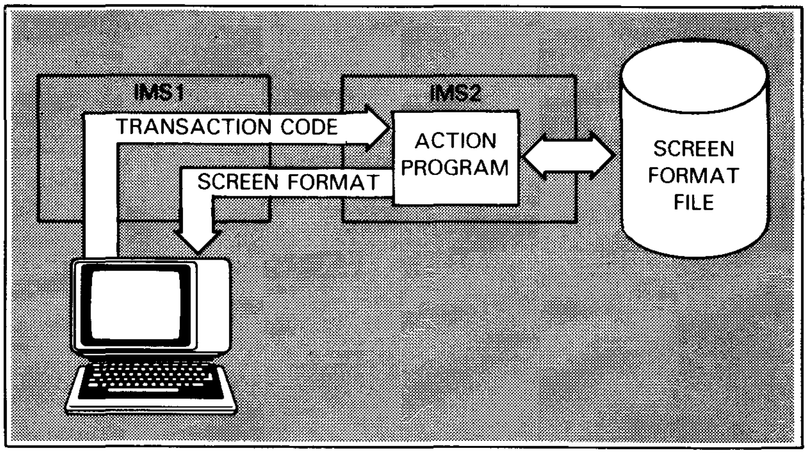


Figure 7-2. Processing a Program-Initiated Remote Transaction

7.6. USING SCREEN FORMAT SERVICES TO PROCESS REMOTE TRANSACTIONS

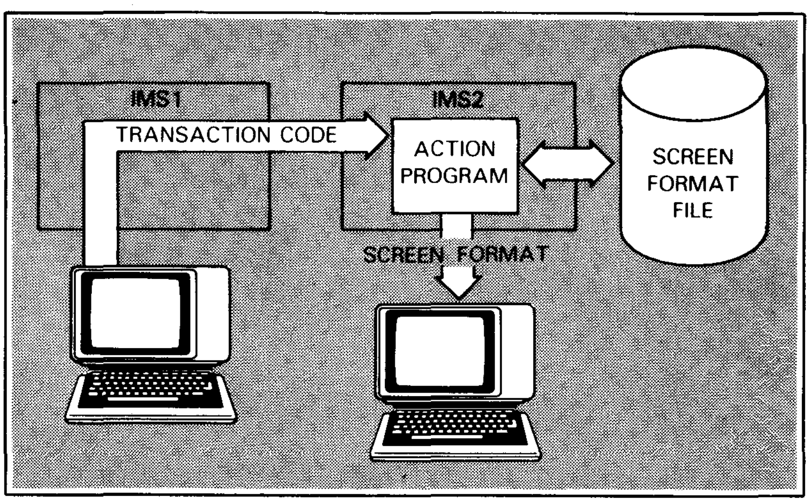
Displaying screen format at initiating terminal

When your action program processes an operator-initiated remote transaction, you can use screen format services to display a screen format at the initiating terminal (or at an auxiliary device attached to that terminal).



Displaying screen format at local terminal

Whether the remote transaction is operator-initiated or program-initiated, you can use the SEND function to display a screen format at a terminal (or auxiliary device) attached to your local IMS system.



identifying local terminal

To display a screen at a terminal attached to your local IMS system, move the terminal-id to the destination-terminal-id field (positions 1-4 of the output message header). Remember, you can display only an output format when you use the SEND function. Afterwards, clear the destination-terminal-id field or move the locap-name of the primary IMS to that field before sending an output message to the source terminal.

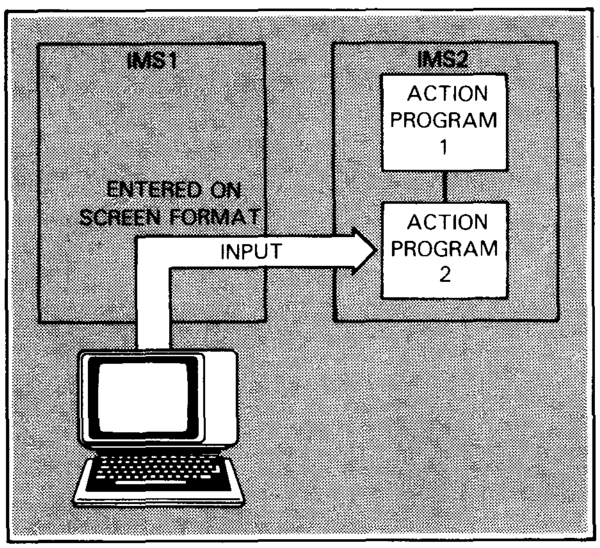
Limitations of SEND function

Termination types allowed

When you display an input/output screen format at the source terminal (at the remote system), you can terminate your program normally or with external succession. We recommend external succession.

Receiving formatted input

When the terminal operator at the remote system enters input on the screen format, the successor program you name at your local IMS system (which could be the same action program) takes control and receives the input.



8. Compiling, Linking, and Storing Action Programs

8.1. PREPARING ACTION PROGRAMS FOR ONLINE PROCESSING

After you write an action program,

DO the following:

What you must do

1. Compile the action program (8.1).

2. Link edit the program to create a load module (8.2).

3. Store the program in the appropriate load library (8.3).

4. Identify the program to IMS in a PROGRAM section of the configuration. (See the IMS system support functions user guide, UP-8364 (current version).)

5. Identify the load library in the job control stream at IMS start up, unless programs are stored in the system load library, YLOD. (See UP-8364.)

Scope of section

This section tells you how to compile and link your action programs and where to store them for use during the online IMS session. For additional information on the job control statements and procedures shown in the examples, refer to the current versions of the job control user guide, UP-8065, and the RPG II user guide, UP-8067.

8.2. COMPILING ACTION PROGRAMS

Action programs compiled like any other program

You compile action programs the same way as other RPG II programs, using the RPG job control procedure (jproc) or the EXEC RPGII job control statement. Don't use the RPGL jproc to compile and link an action program.

Using RPG jproc with embedded input

Figures 8-1 and 8-2 show two ways of compiling an action program using the RPG jproc. In Figure 8-1, the source program is embedded in the job control stream.

```
// JOB PROG1
// RPG
/$
    .
    .    source program
    .
/*
/&
// FIN
```

Figure 8-1. Compiling an Action Program Using Jproc and Embedded Source Program

Using RPG jproc with filed source program

In Figure 8-2, the source program, MYPROG, is filed in the system source library, YSRC. When the source program is filed in a library, you identify the module name in the label field of the RPG jproc. The IN parameter gives the location of the source module - in this case, the system source library.

```
// JOB PROG2
//MYPROG RPG IN=(RES)
/&
// FIN
```

Figure 8-2. Compiling an Action Program Using Jproc and Filed Source Program

Usng standard job control with embedded input

Figure 8-3 uses the EXEC RPGII job control statement and takes source input from the job control stream. You must allocate a printer and two work files for the compiler.

```
// JOB PROG3
// DVC 2Ø   // LFD PRNTR
// WORK1
// WORK2
// EXEC RPGII
/$
    .
    .    source program
    .
/*
/&
// FIN
```

Figure 8-3. Compiling an Action Program Using Standard Job Control and Embedded Source Program

Using standard job control with filed source program

Figure 8-4 also uses the EXEC RPGII job control statement. In this case, the source program is filed in a user source library, SRCIN. You identify the source module and library in a PARAM statement and must also include a device assignment set for the source library.

```
// JOB PROG4
// DVC 2Ø   // LFD PRNTR
// DVC 5Ø   // VOL DISKØ1   // LBL SRCLIB   // LFD SRCIN
// WORK1
// WORK2
// EXEC RPGII
// PARAM IN=MYPROG/SRCIN
/&
// FIN
```

Figure 8-4. Compiling an Action Program Using Standard Job Control and Filed Source Program

8.3. LINK EDITING ACTION PROGRAMS

After you obtain a clean action program compilation, you must link edit the program and store it in the appropriate load library. We discuss load libraries in 8.4.

You can use the LINK job control procedure or the EXEC LINKEDT job control statement. On the LINK jproc, you must specify the OUT parameter to store the action program in a load library:

LINK jproc format

```
// LINK action-program-name, OUT={(vol-ser-no,label)}
                                 {(RES,$Y$LOD)      }
```

For example:

```
// LINK MYPROG,OUT=(RES,$Y$LOD)
```

If you want to give your action program load module a different name than the object module, use this format:

Format for naming load module

```
//load-module-name LINK object-module-name, OUT={(vol-ser-no,label)}
                                                {(RES,$Y$LOD)      }
```

LINK jproc example

Figure 8-5 uses the jproc to link edit an object module called MYPROG and create a load module called CREDIT. Output is to LOADLIB. You don't need a device assignment for LOADLIB because the LINK jproc generates it from your OUT specification.

```
// JOB LINK
//CREDIT LINK MYPROG,OUT=(IMSVOL,LOADLIB)
/&
// FIN
```

Figure 8-5. Link Editing an Action Program Using Jproc

Using standard job control

When you execute the linkage editor using standard job control, you need a LOADM statement to name the load module and INCLUDE statements for the action program object module and the IMS link module, ZF#LINK.

Example using EXEC LNKEDT

Figure 8-6 shows a standard job control stream for the linkage editor. The linkage editor requires a printer file and one work file. You can omit the printer file if you assigned one to the compiler in the same job control stream. Output is to the system load library, YLOD; a device assignment is not needed for this file.

```
// JOB LNKEDT
// DVC 20   // LFD PRNTR
// WORK1
// EXEC LNKEDT
// PARAM OUT=$Y$LOD
/$
   LOADM CREDIT
   INCLUDE MYPROG
   INCLUDE ZF#LINK,$Y$OBJ
/*
/&
// FIN
```

Figure 8-6. Link Editing an Action Program Using Standard Job Control

Compile and link example using jprocs

Figure 8-7 shows a job control stream for compiling and linking an action program, using both the RPG and LINK jprocs. The action program is stored in the LOAD action program library (see 8.4). The LINK jproc generates a device assignment for the load library.

```
// JOB RPGL1
//MYPROG RPG IN=(RES)
//CREDIT LINK MYPROG,OUT=(IMSVOL,LOAD)
/&
// FIN
```

Figure 8-7. Compiling and Linking an Action Program Using Jprocs

Compile and link example using standard job control

Figure 8-8 shows a job control stream for compiling and linking an action program, using standard job control. A device assignment set is required for the output file, LOADLIB.

```
// JOB RPGL2
// DVC 20  // LFD PRNTR
// DVC 50  // VOL IMSVOL  // LBL LOADLIB  // LFD LOADLIB
// WORK1
// WORK2
// EXEC RPGII
    .
    .  source program
    .
/*
// WORK1
// EXEC LNKEDT
// PARAM OUT=LOADLIB
/$
   LOADM CREDIT
   INCLUDE MYPROG
   INCLUDE ZF#LINK,$Y$OBJ
/*
/&
// FIN
```

Figure 8–8. Compiling and Linking an Action Program Using Standard Job Control

8.4. STORING ACTION PROGRAMS IN A LOAD LIBRARY

When you link edit an action program, you must specify the load library where you want it stored. IMS has specific requirements for storing action programs.

One library for action programs

The first requirement is that all your action programs must reside in the same load library.

When you use fast load feature

The load library you choose depends on whether or not you configure the fast load feature by specifying FASTLOAD=YES in the OPTIONS section of your IMS configuration. (See the IMS system support functions user guide, UP-8364 (current version).)

Improves performance

The fast load feature improves online performance in applications with large action programs or frequent action program loading.

Fast loading requires LOAD library

If you configure fast loading, place all action programs in a separate action program load library in unblocked format. You assign this library at IMS start-up with the LFD-name LOAD. At start-up, you also assign the fast load file, LDPFILE. The first time a transaction calls on a particular action program, IMS

Action programs loaded from fast load file

copies the program from LOAD to the LDPFILE. After that, action programs are loaded from LDPFILE.

When you do not use fast load feature

If you don't want fast loading, store your action programs in either of two libraries (but all in the same library):

1. the system load library, YLOD; or
2. the library containing your online IMS load module. This library is identified at configuration time by the LIBL parameter of the IMSCONF jproc.

8.5. REPLACING ACTION PROGRAMS IN THE LOAD LIBRARY DURING ONLINE PROCESSING

You can replace action programs in the load library while IMS is online, whether or not you use the fast load feature.

How to replace programs

You replace an action program in the YLOD, LOAD, or other load library by recompiling and relinking or by applying a patch (COR). For an explanation of the COR function, see the system service programs user guide, UP-8062 (current version).

Fast load requirement

When you use the fast load feature, you must insert the statement:

```
// DD ACCESS=EXCR
```

in the device assignment set for the LOAD library in the compile and link or COR job control stream.

Recompile and link example

The job control stream in Figure 8-9 recompiles and links an action program for output to the LOAD file. This example assumes you use the fast load feature.

```
// JOB RECOMP
// DVC 50  // VOL IMSVOL  // DD ACCESS=EXCR  // LBL LOAD  // LFD LOAD
//MYPROG RPG IN=(RES)
//CREDIT LINK MYPROG,OUT=(IMSVOL,LOAD)
/&
// FIN
```

Figure 8-9. Recompiling and Linking an Action Program During Online Processing

ZZPCH command

After replacing the action program in the load library, issue the ZZPCH master terminal command. The next time a transaction calls on the action program, IMS loads the new version from the load library. When you use the fast load feature, IMS copies the new version to the LDPFILE. The ZZPCH master terminal command is described in the IMS terminal users guide, UP-9208 (current version).

Adding action program to library

Follow the same procedure to add an action program to the load library that is missing at start-up. Of course, the program must be defined in a PROGRAM section of the IMS configuration.

ALTER statement restricted when using fast loading

When you use the fast load feature, do not use ALTER statements in the job control stream at IMS start-up. When you do not use fast loading, you can insert ALTER statements in the start-up job control stream to make temporary changes to action programs.

9. Debugging an Action Program

As often as we might wish that nothing would ever go wrong with our programs, in reality that never seems to be the case. Since action programs can't use the generate-debug capability available to other RPG II programs, it is important to be able to debug your action program using the snap dump feature provided by IMS.

9.1. CONDITIONS FOR A SNAP DUMP

What causes a snap dump

IMS provides a snap dump under three conditions:

- An action program voluntarily terminates abnormally by moving S to the termination-indicator field (position 11) in the program information block.
- An action program terminates abnormally due to a program check.
- An action program terminates abnormally due to a timer-check (time-out due to a loop in the action program).

9.2. TYPES OF SNAP DUMPS

Edited and unedited snap dumps

IMS provides both edited and unedited snap dumps. In single-thread IMS, an edited snap dump is a standard feature. Multithread IMS users must specify SNAPED=YES in the OPTIONS section of the IMS configuration to obtain an edited snap dump. The configurator then includes the module ZG#SNAPM that provides the edited directory for the snap dump.

9.3. LAYOUT OF A SNAP DUMP

Snap dump layout

Figure 9-1 illustrates the general layout of an IMS snap dump. This same general layout applies to both single-thread and multithread IMS.

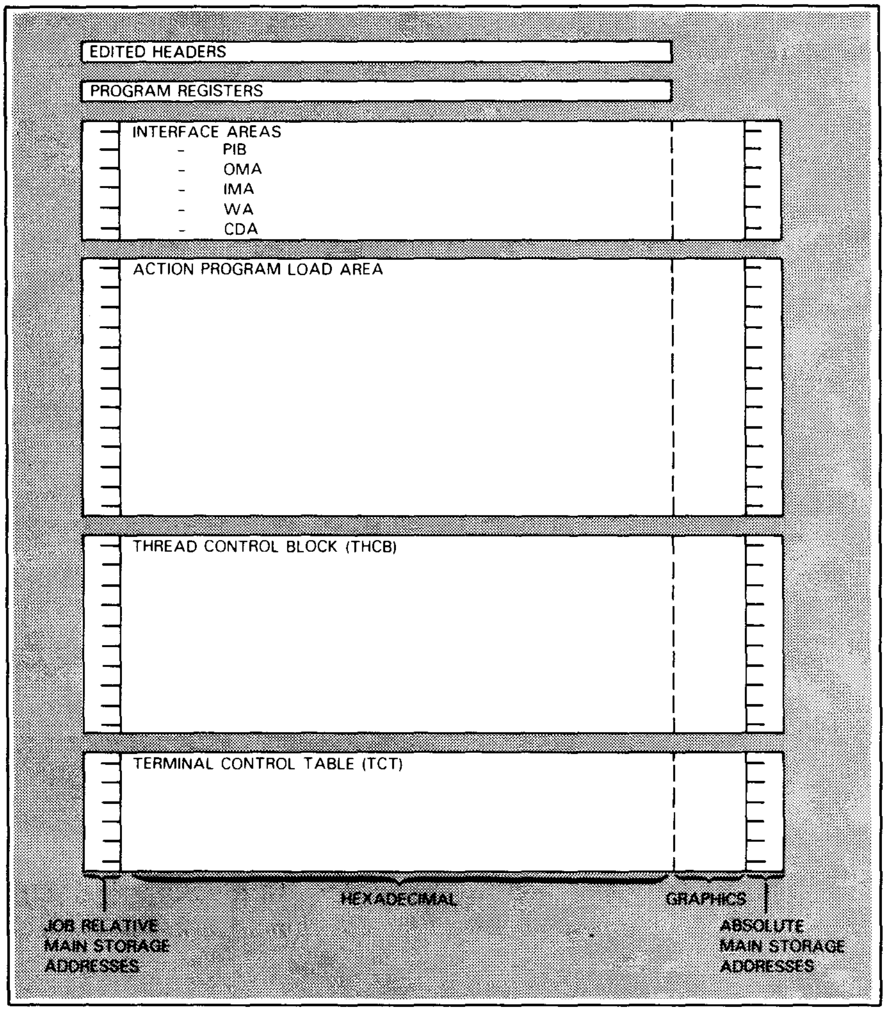


Figure 9-1. Layout of a Snap Dump

Snap dump general areas

As you can see, a snap dump is broken down into six general areas: edited headers, IMS and action program registers, interface areas, action program load area, thread control block (THCB), and terminal control table (TCT).

Header data

Edited header areas contain: (1) data about which action program was running at the time of the snap; (2) an allocation map that provides the relative addresses of areas of interest within the snap dump; and, (3) a general statement of why the snap dump occurred – e.g., ACTION PROGRAM REQUESTED ABNORMAL TERMINATION.

Register section

The next section contains registers. There's one or two sets of registers depending on the reason for the snap dump.

Registers saved by a voluntary snap

If you voluntarily terminated your action program by moving S to the termination-indicator field of the program information block, the snap dump contains one set of registers. These are IMS registers. They are of little use to an IMS action programmer. To find the registers belonging to your action program, you must go to relative location PIB + $4C_{16}$, which contains a full word forward pointer. This word is the address of the SAVE area that contains your action program's registers. Go to this address and advance three full words. The next full word is register 14, then 15, then registers 0–12. Figure 9–3 illustrates these fields.

Registers saved by an involuntary snap

If, on the other hand, IMS terminated your action program due to a program check or time-out, the snap dump contains two sets of registers, IMS and user action program registers. The user registers are labeled so they are easily identifiable. In addition, a duplicate set of user registers can be found at location PIB + 44_{16}. At this location in the program information block, you'll find the 16-byte program status word indicating the address of the instruction immediately following the one that caused the abnormal termination. Also, right after the progam status word are the action program's 16 registers (0–F).

Interface areas

Following the register section, you find the interface areas – program information block, output message area, input message area, work area, continuity data area, and defined record area.

Program area

The next section of the snap dump is the action program load area. It contains the executable load module that was output by the OS/3 linker.

Thread control block

Following the action program area is a section used for the action program's thread control block. In the thread control block, most pointers and flags required to control the user environment are stored for use by IMS and indirectly by the user action program.

Figure 9-2 illustrates the relationship between the IMS thread control block and the user interface areas for both single-thread and multithread IMS.

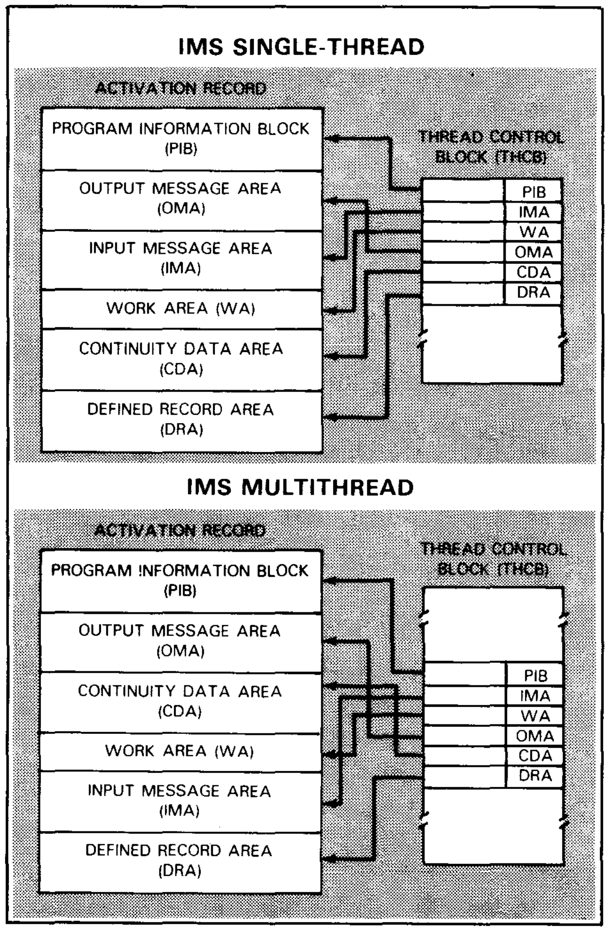


Figure 9-2. Relation between THCB and Interface Areas

Single and multithread main storage layout differences

You will notice that there are pointers within the thread control block that point to each interface area. The differences between single-thread and multithread IMS in this area are only in the location of these pointers and in the relative order of the interface areas themselves.

Terminal control table

The last section in the snap dump is the terminal control table. The data in this area is relevant to the terminal that initiated the action and is the least useful section of the dump to the IMS programmer.

9.4. ANALYZING A SNAP DUMP

Now we'll discuss in detail Figure 9-3, which is a sample RPG II snap dump.

Allocation map addresses

The action name is RCCUST and the current program processing that action is also RCCUST. The term-id (terminal identification) for this transaction is WS1. This is the way the workstation that initiated the transaction was defined in the communications network definition. The allocation map that follows contains the beginning and end locations as well as the lengths of user interface areas, and other areas included in the snap dump. The locations refer to relative addresses. Relative addresses are printed on the far left side of the snap dump.

No work area or continuity data area

The directory in Figure 9-3 shows that there are no addresses for the work area (WA) or continuity data area (CDA). The reason for this is that these areas were not given values in the configuration.

THCB addresses

Location of interface areas

If you aren't using an edited snap dump, that is, if it contains no directory listing, it's still quite easy to locate all the action program's interface areas. Go directly to the thread control block, which is at location $D0_{16}$. The first five full words (40 bytes) contain the relative addresses of the program information block, input message area, work area, output message area, continuity data area, and action program load area, in that order.

Reason for snamp dump

Following the allocation map on Figure 9-3 is the reason for the snap dump: ACTION PROGRAM REQUESTED ABNORMAL TERMINATION. Voluntary termination results when an action program moves S to the termination-indicator field (position 11) of the program information block.

One set of registers

The register section contains only one set of registers because the action program terminated voluntarily. These are IMS registers. To find RCCUST's registers, go to relative location PIB + $4C_{16}$. At that location, you find a full-word address of RCCUST'S save area. The save area contains the action program registers.

SAVE area

The save address is $B484_{16}$. Once at this address, which is in the action program load area, advance three full words. At location $B490_{16}$ you will find register 14, and in the subsequent full words, registers 15 and 0-12, respectively.

ANALYZING A SNAP DUMP (FIGURE 9-3)

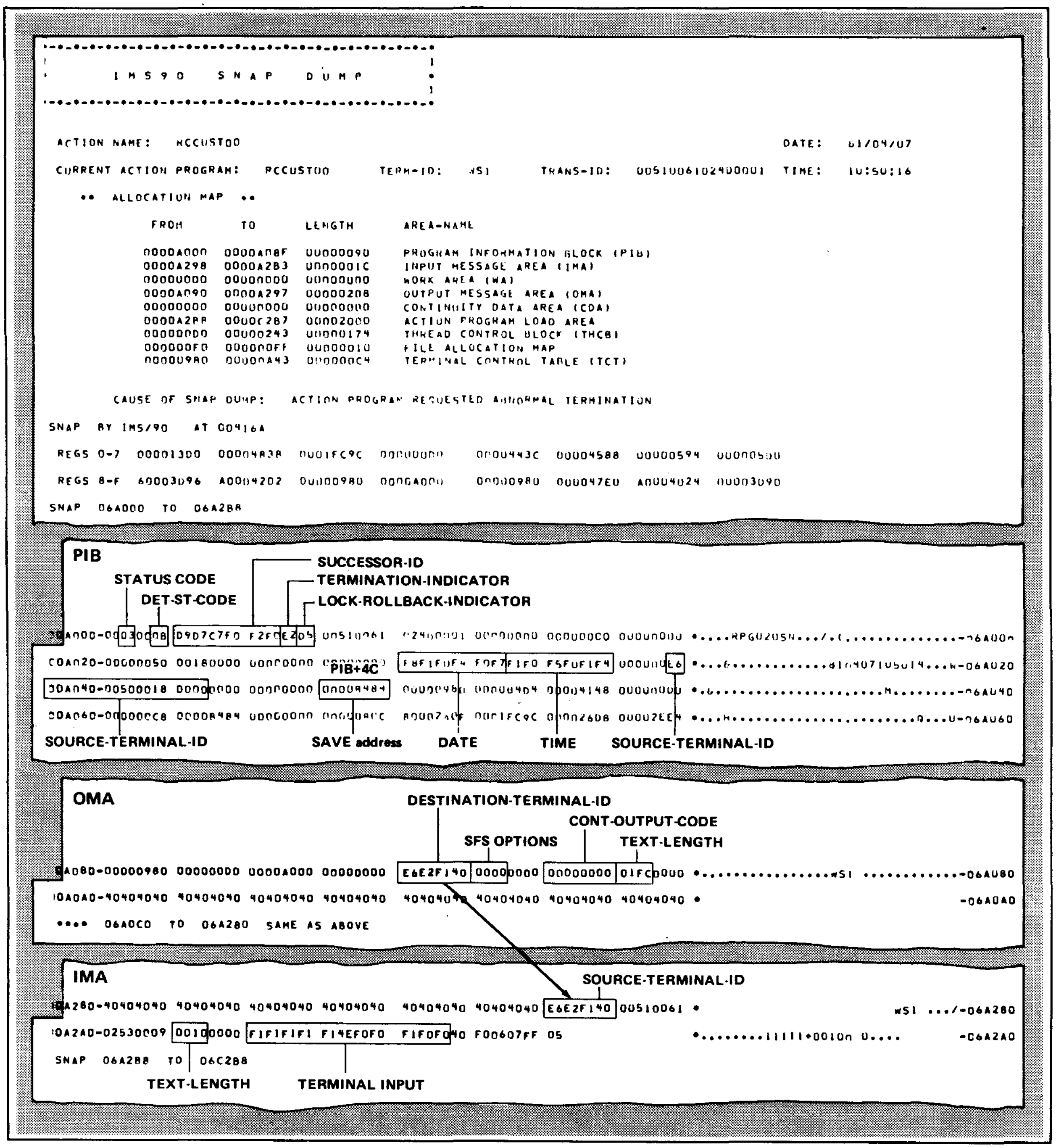


Figure 9-3. Sample RPG II Snap Dump (Part 1 of 2)

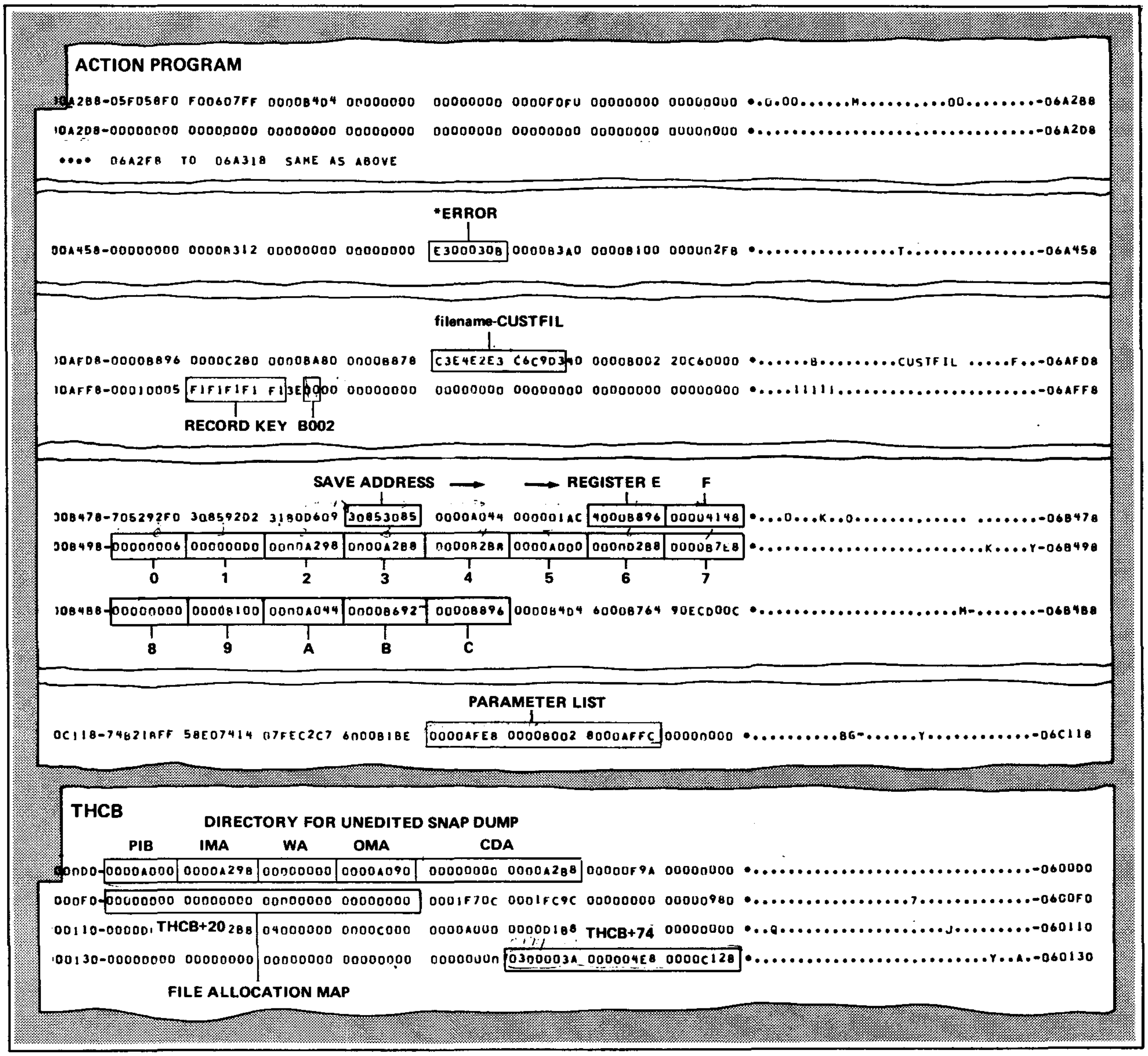


Figure 9-3. Sample RPG II Snap Dump (Part 2 of 2)

9.5. THE PROGRAM INFORMATION BLOCK (PIB)

Finding Your Error

Status codes

Locating the status codes

The program information block begins at address 0A00. The first word (4 bytes) contains the status-code and det-status-code fields. IMS returns values to these fields indicating the result of action program function calls. If the function call is successful, these fields contain zeros. In Figure 9-3, however, you see that the function call made to IMS was not successful. The value 03_{16} in status-code indicates the action program made an invalid request. The $0B_{16}$ in det-status-code indicates that the file requested in the function call was not assigned to this action at IMS configuration. To find out exactly which file is involved, you must consult the parameter list address in the thread control block. We will discuss how this is done very shortly.

For a complete listing of the values IMS returns in the status-code and det-status-code fields, see 2.6.

Successor-id

Locating the successor-id field

Looking further into the snap dump at relative location PIB + 4_{16}, you find the successor-id field. Notice that this field contains 'RPG020'. Whenever RPG II encounters an error, it places the appropriate error code in the successor-id field prior to requesting the snap. RPG020 indicates an indexed file error. For a complete listing and description of error codes, consult OS/3 system messages, UP-8076 (current version).

*ERROR

*Locating *ERROR*

A further statement of the error condition can be found in the field, *ERROR. RCCUST's link relative location or link-org is 0 and *ERROR is displaced $1B0_{16}$ into it. To locate *ERROR, we take the start location for the action program load area that the allocation map tells us is $A2B8_{16}$ and add $1B0_{16}$ to it. This gives us location $A468_{16}$ or *ERROR. At this location in the snap, we find $E3_{16}$ in the first byte and 03_{16} and $0B_{16}$, respectively, in the third and fourth bytes. You will recognize 03_{16} and $0B_{16}$ as the status-code and det-status-code fields. The $E3_{16}$ (character T) can be found in OS/3 system messages, UP-8076 (current version) and is defined as an RPG020 error.

Interpreting error codes

At this point, it's obvious that the wrong file name was used for I/O or the file requested is not available to this action program. In our example, the file CUSTFIL to which the function call was made wasn't configured for use by action RCCUST.

Finding Other Data in the Program Information Block

Termination indicator

Locating the termination-indicator field

Still in the program information block, at relative location PIB + A_{16} is the field termination-indicator. It contains an $E2_{16}$ (character S) for snap dump. The value in this and any other program information block field varies depending on the action program and whether the program terminated voluntarily or involuntarily.

Lock-rollback-indicator

Locating the lock-rollback-indicator field

Relative location PIB $+B_{16}$ is the lock-rollback-indicator field. It contains $D5_{16}$ (character N), which is the default value. The value N establishes a new rollback point in the audit file (before-images of records to be updated) and releases all locks for this transaction.

Locating other PIB fields

By comparing the program information block fields listed in Table 2-6 to the program information block area of the snap dump, you can see exactly what values all these fields contained when the dump occurred. For your convenience, we have noted a few of these fields in Figure 9-3: transaction-date (810407), time-of-day (105014), and source-terminal-type ($E6_{16}$ or W for workstation).

Entire PIB displayed

All 145-character positions of the program information block are displayed. Remember, however, that only the first 70 positions are accessible to your action program.

9.6. THE OUTPUT MESSAGE AREA

Destination-terminal-id

Locating the destination-terminal-id field

Using the allocation map, we see the output message area begins at address $A090_{16}$. This area contains the 16-byte header and the output message generated by the action program. Since RCCUST terminated abnormally before generating an output message, the output message area contains spaces. However, the header data is displayed. The first word contains the destination-terminal-id field. This indicates the destination of the output message had the program not terminated abnormally. Note that this value is the same as the value in source-terminal-id, which occupies the first word of the input message area.

Message-length

Locating the message-length field

Also, in the output message area at location $A09C_{16}$ or OMA + C_{16} is the 2-byte message-length field. This field indicates the size of the output message to be generated.

Since RCCUST doesn't use screen format services and it isn't a continuous output program, relative locations $A094_{16}$ and $A098_{16}$, respectively, contain zeros.

9.7. THE INPUT MESSAGE AREA

Locating the input message

The input message area begins at relative address $A298_{16}$. Its contents include the input message area header (16 bytes) and the input data entered by the terminal operator. The terminal input starts at IMA + 11 or $A2A8_{16}$. The terminal operator entered the customer number 11111 (F1F1F1F1F1), a plus (+) sign (4E), and AMOUNT $1.00 (F0F0F1F0F0). These entries correspond to the data requested by the screen format shown in Figure 3-11.

9.8. ACTION PROGRAM LOAD AREA

Largest section of dump

Since there is no continuity data area, work area, or defined record area for this particular action program, we will now discuss the program load area. This is by far the lengthiest section of the snap dump. Since data contained in the thread control block is essential to interpreting the program area, we will discuss the two areas at the same time.

Thread control block

Using the thread control block

In this example, the thread control block is at location $D0_{16}$. It contains the addresses of all the interface areas and the action program load area. This data is of value only if you're using an unedited dump. However, the thread control block does contain other information very useful to the IMS programmer.

File allocation map

Locating the file allocation map

At THCB + 20_{16} or in our example location $F0_{16}$, there are four full words used for a file allocation bit map. To use this bit map, you must realize that four full words contain 128 bits. IMS uses these bits to indicate which specific files a user action program can access – one file per bit. The file allocation map for multithread IMS is 8 full words long (256 bits).

When bits are set off

In Figure 9-3, no bits are on at location F0. Consequently, RCCUST could not access any files. If you recall, the det-status-code field already informed us that the file wasn't defined at IMS configuration. However, in cases where this same problem doesn't exist, the file allocation map can be very valuable in determining exactly which files are being accessed by an action program.

When bits are set on

For example, if the high order bit was on, the action program could access one file – the first file configured. If additional bits were on, additional files could be accessed. These bits are maintained in the same relative order as the actual files were configured.

THCB + 74

Moving to relative location 144 or THCB + 74_{16}, we find three words that in most instances are very useful for debugging purposes:

0300003A 000004E8 0000C128

CALL function

Determining the last function call

The first of these words needs to be broken down into individual bytes. Byte 0 (03) indicates the number of parameters passed on the last CALL function made by the action program. Bytes 1 and 2 are not used. Byte 3 (3A) indicates what CALL function was issued. In this case, it was a GETUP function with three parameters passed.

Although the RPG II action program appears to access files normally without issuing function calls, RPG II is in fact, issuing these calls to IMS.

Table 9-1 lists all the IMS function calls and their corresponding hexadecimal values for use in debugging your action program.

Hexadecimal equivalent for function calls

Table 9-1. Hexadecimal Equivalents for Function Calls

Hexadecimal	Function Call
06	RETURN
0A	SEND
26	ESETL
2A	SETL
2E	INSERT
32	DELETE
36	PUT
3A	GETUP
3E	GET
4A	SNAP
8E	SUBPROGRAM

DTF

Locating the DTF

The second word of this 3-word group is the relative address of the DTF referenced by the function call if it was an I/O function. This address is not within the range of the user snap dump and is useful only when a job dump is available.

Parameter List

Locating the parameter list

The last word of the group is the address of the parameter list that was passed for the function. In our example, the relative address of the parameter list is $C128_{16}$. This address is in the action program load area. Since three parameters were passed in the call, the next three full words are the addresses of those parameters. The first address is the file name. It's at location $AFE8_{16}$ in the program area. At this location, we find a 7-byte constant, CUSTFIL, which was the file RCCUST attempted to access. The second and third addresses are $B002_{16}$ and $AFFC_{16}$, respectively. Address $B002_{16}$ points to the location into which the CUSTFIL record was to be read. As you can see, there is no record in this location since the GETUP was never accomplished. The third address, $AFFC_{16}$, points to the location that contains the record key, F1F1F1F1F1. Both of these locations are in the user program area.

9.9. SINGLE AND MULTITHREAD SNAPS

Order of interface areas

There are two major differences between single-thread and multithread snap dumps. First, the order of the interface areas is different. In single-thread, it is: program information block; output message area; input message area; work area; continuity data area; and, defined record area. On multithread, it is: program information block; output message area; continuity data area; work area; input message area; and, defined record area. Since the allocation map in an edited dump points directly to these areas, there should be no difficulty in locating them in either single or multithread IMS.

Different DSECTs

The second major difference concerns the thread control block. The format for single-thread and multithread is totally different. Figures 9-4 and 9-5 provide listings of the thread control block DSECTs for both single-thread and multithread IMS. You will see by examining these figures that although the format is different, the data they contain is basically the same.

SINGLE-THREAD THREAD CONTROL BLOCK

```
LOC.          LINE    SOURCE STATEMENT

              A9979+           ZM#DTHCB
000000        B9980+ZT#DTHCB DSECT
              B9981+*
              B9982+*   THREAD CONTROL BLOCK / SYSTEM INFORMATION BLOCK
              B9983+*
              B9984+*     THREAD CONTROL SECTION
              B9985+*
              B9986+*
              B9987+*          INSERTED EQU'S TO MATCH OS/7 NAMES
              B9988+*
000000        B9989+ZT#TPIBA EQU    *
000000        B9990+ZT#HPIBA DS     A PROGRAM INFORMATION BLOCK ADDR
000004        B9991+ZT#TIMA  EQU    *
000004        B9992+ZT#HIMA  DS     A INPUT MESSAGE AREA ADDR
000008        B9993+ZT#TWA   EQU    *
000008        B9994+ZT#HWA   DS     A WORK AREA ADDR
00000C        B9995+ZT#TOMA  EQU    *
00000C        B9996+ZT#HOMA  DS     A OUTPUT MESSAGE AREA ADDR
000010        B9997+ZT#TCDA  EQU    *
000010        B9998+ZT#HCDA  DS     A CONTINUITY DATA AREA ADDR
000014        B9999+ZT#TDRMA EQU    *
000014        B0000+ZT#HDRA  DS     A DEFINED RECORD AREA ADDR
000018        B0001+ZT#DDREC EQU    *
000018        B0002+ZT#HDDRA DS     F DATA DEFINITION RECORD ADDR
00001C        B0003+ZT#SUBFL EQU    *
00001C        B0004+ZT#HDFA  DS     F DEFINED FILE/SUBFILE PKT ADDR
000020        B0005+ZT#TFAM  EQU    *
000020        B0006+ZT#HFAM  DS     4F FILE ALLOCATION MAP
000010        B0007+ZT#HNUMF EQU    *-ZT#HFAM FILE ALLOCATION MAP LENGTH
000030        B0008+ZT#TATA  EQU    *
000030        B0009+ZT#HATA  DS     F ACTION CONTROL REC PTR
000034        B0010+ZT#TPTA  EQU    *
000034        B0011+ZT#HPTA  DS     F PROG CONTROL TABLE REC PTR
000038        B0012+ZT#TPTA1 DS     F
00003C        B0013+ZT#TTTA  EQU    *
00003C        B0014+ZT#HTTA  DS     F TERM CONTROL TAB REC PTR
000040        B0015+ZT#HIOAV DS     F START OF VARIABLE I/O AREA
000044        B0016+ZT#HPLA  DS     F PROGRAM LOAD AREA ADDRESS
000048        B0017+ZT#HBIQP DS     F BYPASS INTERRUPT QUEUE PTR
              B0018+*
              B0019+*     EQUATES FOR 1ST BYTE OF ZT#HBIQP
000008        B0020+ZB#SOLSH EQU    X'08' SHUTDOWN IN PROCESS
000004        B0021+ZB#SOLAS EQU    X'04' AUTOMATIC STATUS
000002        B0022+ZB#SOLCO EQU    X'02' ZZUP/ZZDWN COMMAND OUTSTANDING
000001        B0023+ZB#SOLST EQU    X'01' SHUTDOWN TIMER
              B0024+*
00004C        B0025+ZT#HBIQL DS     XL1 BYPASSED INTERRUPT QUEUE LENGTH
00004D        B0026+ZA#USER  EQU    *
00004D 00     B0027+ZT#USER  DC     X'0' . USER FLAG
              B0028+*
              B0029+*               MUST ALWAYS BE ON ODD BYTE BOUNDARY
```

Figure 9–4. Single-thread Thread Control Block (Part 1 of 4)

```
 LOC.     LINE   SOURCE STATEMENT
        80030+*
        80031+*                                        80 - I/O HAS OCCURRED
        80032+*                                        40 - INITIAL SETTING FOR USER
        80033+*                                        00 - IMS ACTIVE
        80034+*                                           - COUNT FOR TOTAL TIME
00004E  80035+ZT#TIND  EQU   *
00004E  80036+ZT#HIND  DS    XL1 CONTROL INDICATORS
        80037+*
        80038+*     EQUATES FOR ZT#HIND
        80039+*
000080  80040+ZT#HINSP EQU   X'80' SNAP INDICATOR
000040  80041+ZT#HINER EQU   X'40' ERROR RETURN
000020  80042+ZT#HINDI EQU   X'20' DELAYED INTERNAL SUCCESSION
000010  80043+ZT#HINEO EQU   X'10' EXPLICIT OUTPUT
000008  80044+ZT#HINEX EQU   X'08' EXTERNAL SUCCESSION
000004  80045+ZT#HINCN EQU   X'04' CANCELLED
000002  80046+ZT#HINIR EQU   X'02' INTERNAL REQUEST TO FILE MGMT
000001  80047+ZT#HINUP EQU   X'01' UPDATE PERFORMED BY THIS ACTION
        80048+*
00004F  80049+ZT#SYIND DS    XL1 CONTROL INDICATORS
000080  80050+ZT#ILIST EQU   X'80' INTERRUPT LIST IF SET
000040  80051+ZT#TOMRD EQU   X'40' . IF ON INDICATES READ FROM TOMFOLE
000020  80052+ZT#TRSD  EQU   X'20' . RESEND = NO
000010  80053+ZT#UTOUT EQU   X'10' USER TIME OUT
000008  80054+ZT#ESETL EQU   X'08'
000004  80055+ZT#USETX EQU   X'04' USE THE TEXT IN OMA ALTHOUGH TRANS WAS CNC
000002  80056+ZT#ZZOPN EQU   X'02' INDICATES TO WRITE ZZOPN TERM. RECORD
000050  80057+ZT#PSSK  DS    9F
        80058+*
        80059+*     FILE MANAGEMENT ENTRIES
        80060+*
000074  80061+ZT#TFC   EQU   *
000074  80062+ZT#HFC   DS    F BYTE 0 :# OF PARAMS
        80063+*                                    BYTE 3 : FUNCTION CODE
000078  80064+ZT#TUPDA EQU   *
000078  80065+ZT#HUPDA DS    F UNPROTECTED DTF ADDR
00007C  80066+ZT#TCR   EQU   *
00007C  80067+ZT#HRPLA DS    F PARAM LIST ADDR
000080  80068+ZT#TFWA  EQU   *
000080  80069+ZT#HFWA  DS    3A FILE MGMT WORK AREA
00008C  80070+ZT#DMSL  DS    A TCT ADDR OF DMS RUN-UNIT
000090  80071+ZT#DMCA  DS    A DMS - DMCA ADDRESS
        80072+*
        80073+*     SAVE AREAS
        80074+*
        80075+*
        80076+*
000094  80077+ZT#HSADM DS    18F DATA MANAGEMENT SAVE AREA
0000DC  80078+ZT#HSAIR DS    18F INTERNAL REQUEST SAVE AREA
        80079+*
        80080+*     SYSTEM INFORMATION SECTION
        80081+*
```

Figure 9-4. Single-thread Thread Control Block (Part 2 of 4)

SINGLE-THREAD THREAD CONTROL BLOCK

```
 LOC.    LINE   SOURCE STATEMENT
000124  80082+ZB#STIDT DS     F TRANSACTION CODE TABLE
000128  80083+ZB#SACT  DS     F ACTION CONTROL TABLE
00012C  80084+ZB#SPCT  DS     F PROGRAM CONTROL TABLE
000130  80085+ZB#SFCTI DS     F FILE CONTROL TABLE INDEX
000134  80086+ZB#STERM DS     F TERMINAL CNTL TBL ADDR
000138  80087+ZB#SDCTI DS     F DEF FILE CONTROL TABLE
00013C  80088+ZB#SFADR DS     F IMS LOAD ADDRESS
000140  80089+ZB#SAVAL DS     F AVAILABLE LIST ADDRESS
000144  80090+ZB#STCS  DS     F TERM. CONTROL SECTION
000148  80091+ZB#SIMB  DS     F INPUT MESSAGE BUFFER
00014C  80092+ZB#SIOAE DS     F I/O AREA END ADDR
000150  80093+ZB#SFSAD DS     A ADDR IMS SESSION STATISTICS
000154  80094+ZB#LOUTM DS     H LARGEST OUTPUT MSG.
000156  80095+ZB#LINM  DS     H LARGEST INPUT MSG.
000158  80096+ZB#LOMTI DS     4C LARGEST OUTPUT MSG.-TERM ID. NAME
00015C  80097+ZB#LIMTI DS     4C LARGEST INPUT  MSG.-TERM ID. NAME
000160  80098+ZB#SMLL  DS     H STANDARD MESSAGE LINE LENGTH
000162  80099+ZB#SMNL  DS     H STANDARD MESSAGE NUMBER OF LINES
000164  80100+ZB#SIMBL DS     H INPUT MESSAGE BUFFER LENGTH
000166  80101+ZB#TMCCA DS     H NUMBER OF TERMS IN ICAM CCA
000168  80102+ZB#STOF  DS     XL1 . USER TIMEOUT FLAG
000169  80103+ZB#SOLOF DS     XL1 CONTROL INDICATORS FOR AUDIT
        80104+*
        80105+*    EQUATES FOR ZB#SOLOF
000080  80106+ZB#SOLUP EQU    X'80' UPDATING PERMITTED
000040  80107+ZB#SOLAI EQU    X'40' AUDIT MODULE INCLUDED
        80108+*                                 (BEF IMAGES, TR FILES)
000020  80109+ZB#SOLRD EQU    X'20' ROLLBACK PROGRAM / FILE DOWN
000010  80110+ZB#SOLSU EQU    X'10' SUPPRESS UPDATES
000008  80111+ZB#SOLTB EQU    X'08' BEFORE IMAGES TRACED
000004  80112+ZB#SOLTA EQU    X'04' AFTER IMAGES TRACED
000002  80113+ZB#SOLTI EQU    X'02' INPUT MESSAGES TRACED
000001  80114+ZB#SOLTE EQU    X'01' I/O ERROR TRACE FILE
        80115+*
00016C  80116+         DS     0F
        80117+*
00016C  80118+ZB#FLG1  DS     X . FLAG1 OF STARTUP
000080  80119+ZB#STRIN EQU    X'80' . STARTUP ACTIVE
000040  80120+ZB#TCRSH EQU    X'40' .*TRCFILE=CRASH
000020  80121+ZB#TEXT  EQU    X'20' .*TRCFILE=EXT
00016D  80122+ZB#FLG2  DS     X .FLAG FOR TOMFILE
000080  80123+ZB#TOMUP EQU    X'80' . TOMFILE CONFIGURED
000001  80124+ZB#TOMER EQU    X'01' . ERROR ON TOM FILE
000002  80125+ZB#TOMNT EQU    X'02' . DO NOT TRACE TOMFILE
00016E  80126+ZB#FLG3  DS     X .FLAG FOR TYPE OF RESTART
000001  80127+ZB#INDCL EQU    X'01' .START=CLEAN
000002  80128+ZB#INDWA EQU    X'02' .START=WARM
000004  80129+ZB#INDCO EQU    X'04' .START=COLD
00016F  80130+ZB#FLG4  DS     X DMS FLAG BYTE
000080  80131+ZB#IMSDM EQU    X'80' IMS HAS MADE A REQUEST TO DMS
000040  80132+ZB#DMSDO EQU    X'40' DMS HAS TERMINATED
000020  80133+ZB#DMSRU EQU    X'20' DMS RUN-UNIT EXISTS
```

Figure 9–4. Single-thread Thread Control Block (Part 3 of 4)

```
 LOC.    LINE    SOURCE STATEMENT
000010  80134+ZB#IMSNA EQU   X'10' IMS NOT ALLOWED ACCESS TO DMS
000008  80135+ZB#DMSNA EQU   X'08' DMS IS NOT THERE
000170  80136+ZB#FLG5  DS    XL1
000080  80137+ZB#KAT   EQU   X'80' KATAKANA CONFIGURED
000040  80138+ZB#STATS EQU   X'40' STATISTICS AT SHUTDOWN
000020  80139+ZB#SFSEN EQU   X'20' SFS ENABLED
000008  80140+ZB#GLB   EQU   X'08' GLOBAL NETWORK
000004  80141+ZB#DED   EQU   X'04' DEDICATED NETWORK
000171  80142+         DS    XL3 UNUSED
000174  80143+ZB#LPCT  DS    F LAST PCT ADDRESS
000178  80144+ZB#LACT  DS    F LAST ACT ADDRESS
00017C  80145+ZB#LAD   DS    F LAST LOAD AREA ADDRESS
000180  80146+ZB#NLST  DS    H INTLIST=N VALUE
000182  80147+         DS    XL2 UNUSED
000184  80148+ZC#CCA   DS    F CCA NAME
000188  80149+ZC#LOCAP DS    F LOCAP NAME
00018C  80150+ZB#MDICE DS    F DICE-SCREEN CLEAR/MSG POSITION
000190  80151+ZB#UNDEF DS    A POINTER TO TRIDT TO PROCESS UNDEF.TRANS.CODES
000194  80152+ZB#DATE  DS    F TODAY'S DATE
000198  80153+ZB#SESLN DS    F LENGTH-SESSION TABLE-ZSTAT
00019C  80154+ZQ#THFIN DS    OF . THIS TAG MUST STAY AT END
00019C  80155+ZT#HLEN  EQU   *-ZT#DTHCB LENGTH OF THCB
00019C  80156+ZT#TLEN  EQU   ZT#HLEN
000000  80157+ZC#IIP   CSECT
```

Figure 9–4. Single-thread Thread Control Block (Part 4 of 4)

MULTITHREAD THREAD CONTROL BLOCK

```
 LOC.     LINE    SOURCE STATEMENT

          2628            PRINT GEN
          2629            ZM#DTHCB
000000   A2630+ZT#DTHCB DSECT
000000   A2631+ZT#THQPT DS     F . NEXT THREAD IN QUEUE POINTER
000004   A2632+ZT#NTHCB DS     F . NEXT THREAD FOR SCHEDULING
000008   A2633+ZT#THURF DS     X . URGENT FLAG   0 - ROUTINE
000009   A2634+ZT#THRDF DS     X . THREAD READY FLAG  1 - READY
00000A   A2635+ZT#DWAIT DS     0X BIT 0 INITIAL THREAD WAIT FLAG - WAIT
00000A   A2636+ZT#REGRS DS     X BIT 7 RESTORE REGISTER FLAG  0 - YES
00000B   A2637+ZT#IECB3 DS     X BIT 0 CANCEL FLAG  1 - CANCEL
         A2638+*                      BIT 2 OUTPUT MESSAGE GENERATED BY ZG#MTMSO
         A2639+*                      BIT 3 INTERNAL CANCEL INITIATED
         A2640+*                      BIT 7 IECB FLAG    1 - 3WORD
00000C   A2641+ZT#THSVR DS     F . THREAD SAVE AREA REGISTER
000010   A2642+ZT#THRAD DS     F . THREAD RETURN ADDRESS
000014   A2643+ZT#TPIBA DS     A PROGRAM INFORMATION BLOCK ADDR
000018   A2644+ZT#TIMA  DS     A INPUT MESSAGE AREA ADDR
00001C   A2645+ZT#TWA   DS     A WORK AREA ADDR
000020   A2646+ZT#TOMA  DS     A OUTPUT MESSAGE AREA ADDR
000024   A2647+ZT#TCDA  DS     A CONTINUITY DATA AREA ADDR
000028   A2648+ZT#TDRMA DS     A DEFINED RECORD AREA ADDR
00002C   A2649+ZT#DDREC DS     A DATA DEFINITION RECORD ADDR
000030   A2650+ZT#SUBFL DS     A DEFINED FILE SUB-FILE DESC ADDR
000034   A2651+ZT#TFAM  DS     8F FILE ALLOCATION MAP
000020   A2652+ZT#TNUMF EQU    *-ZT#TFAM FILE ALLOCATION MAP LENGTH
000054   A2653+ZT#TATA  DS     A ACTION CONTROL TABLE RECORD ADDR
000058   A2654+ZT#TPTA  DS     A PROGRAM CONTROL TABLE RECORD ADDR
00005C   A2655+ZT#TPTA1 DS     F
000060   A2656+ZT#TTTA  DS     A TERMINAL CONTROL TABLE RECORD ADDR
000064   A2657+ZT#TIMB  DS     A INPUT MSG BUFFER ADDR
000068   A2658+ZT#TEDIT DS     A EDIT TABLE ADDR
00006C   A2659+ZT#TRID  DS     CL8 TRANSACTION ID
000074   A2660+ZT#TIND  DS     XL1 CONTROL INDICATORS
         A2661+*                      BIT 0     TERMINATION TYPE    0    NORMAL
         A2662+*                                                    1    ABNORMAL
         A2663+*                      BIT 2     ERROR RETURN        0    NO
         A2664+*                                                    1    YES
         A2665+*                      BIT 3-4   INTERNAL MESSAGE CONTROL:
         A2666+*                                00   END ACTION OR END TRANSACTION
         A2667+*                                01   EXPLICIT OUTPUT
         A2668+*                                10   DELAYED INTERNAL SUCCESSION
         A2669+*                                11   CANCELLED
         A2670+*                      BIT 5     INTERNAL REQUEST INDIC FOR FM
         A2671+*                                                    0    NO
         A2672+*                                                    1    YES
         A2673+*                      BIT 6  OUTPUT IN PROCESS
         A2674+*                      BIT 7  OUTPUT WAITED
000075   A2675+ZT#TER#  DS     X ERROR CODE NUMBER
000076   A2676+ZT#TES   DS     H RELATIVE ACT RECORD ADDR
000078   A2677+ZC#SFSSC DS     H INPUT STATUS BYTE COUNT
00007A   A2678+ZC#ITLN  DS     XL1 XTION FLD LEN CTR-INVALID TRANSACTION
```

Figure 9–5. Multithread Thread Control Block (Part 1 of 2)

```
 LOC.     LINE   SOURCE STATEMENT
C0007B   A2679+ZC#SFSID DS     CL6 SUCCESSOR-ID FOR REBUILD
         A2680+* FILE MANAGEMENT ENTRIES
         A2681+*     PARAMETER LIST FOR SUBTASK
C00084   A2682+ZT#TBA   DS     A BEGIN ADDR
000088   A2683+ZT#TRPLA DS     A REQUEST PARAM LIST ADDR
C0008C   A2684+ZT#TFC   DS     A BYTE 0 - # OF PARAMS IN LIST
         A2685+*                                  BYTE 3 - FUNCTION CODE
000090   A2686+ZT#TUPDA DS     A UNPROTECTED DTF ADDR
000094   A2687+ZT#TCR   DS     A COVER REG
         A2688+*       OTHER
000098   A2689+ZT#TFWA  DS     3A WORK AREA
0000A4   A2690+ZT#TSAV1 DS     11A SAVE AREA 1
0000D0   A2691+ZT#TSAV2 DS     11A
0000D0   A2692+ZT#SAV5  EQU    ZT#TSAV2 SAVE AREA 5
0000F8   A2693+ZT#SAVE6 EQU    ZT#SAV5+40
0000FC   A2694+         DS     7F'0'
000118   A2695+ZT#TSAV4 DS     18A SAVE AREA 4
000160   A2696+ZT#TSAV3 DS     11A SAVE AREA 3
00018C   A2697+ZA#PSSK  DS     9F
0001B0   A2698+ZT#TFLA  DS     F REQUIRED BY IRAM
0001B4   A2699+ZT#TF1   DS     F APPL.MANAG.
0001B8   A2700+ZT#TF2   DS     F FLAG BYTE
0001B8   A2701+ZT#SYIND EQU    ZT#TF2 FLAGS
000040   A2702+ZT#TOMRD EQU    X'40' INDICATES TOM READ
000004   A2703+ZT#ZZOPN EQU    X'04' INDICATES TO WRITE ZZOPN TERM. RECORD
000001   A2704+ZT#RDF   EQU    X'01' MIRAM RE-READ FLAG
0001BC   A2705+ZT#UDMCA DS     A USER PROGRAM DMCA ADDRESS
0001C0   A2706+ZT#IDMCA DS     A IMS INTERNAL DMCA ADDRESS
0001C4   A2707+ZT#SIBA  DS     F SIB ADDRESS
0001C8   A2708+         DS     0F
0001C8   A2709+ZT#TLEN  EQU    *-ZT#DTHCB LENGTH OF CONTROL BLOCK
000000   A2710+Z0#OUTMT CSECT
```

Figure 9-5. Multithread Thread Control Block (Part 2 of 2)

Terminal control table

The terminal control table for single and multithread IMS is also a valuable debugging aid. Figure 9-6 shows this table.

```
  LOC.     LINE    SOURCE STATEMENT

           2712            ZM#DTCT
000000     A2713+ZC#DTCT   DSECT   **** TERMINAL CONTROL TABLE RECORD ****
           A2714+*
000000     A2715+ZC#LINK   DS       F ACT LINK TO NEXT TCT IN QUEUE
000004     A2716+ZC#TID    DS       XL4 TERMINAL ID
000008     A2717+ZC#TAL    DS       F REL ADDR SOURCE TCT  (OS/3)
00000C     A2718+ZC#TALT   DS       F REL ADDR ALTERNATE TCT (OS/3)
000010     A2719+ZC#TTTA   DS       F CORRESPONDING TTT ADDRESS
000014     A2720+ZC#TESR   DS       F SUCC ACT REL ADDR - ROLLBACK
000018     A2721+ZC#TCDL   DS       H CONTINUITY DATA LENGTH
00001A     A2722+ZC#TLN    DS       XL1 LINE NUMBER
00001B     A2723+ZC#TTST   DS       XL7 STATUS BYTES
00001B     A2724+ZC#TST    EQU      ZC#TTST
           A2725+*
           A2726+*   EQUATES FOR ZC#TTST/ZC#TST
           A2727+*
000080     A2728+ZC#TTLST EQU      X'80' LAST TCT
000040     A2729+ZC#TTTMD EQU      X'40' TEST MODE
000020     A2730+ZC#TTUM   EQU      X'20' URGENT MESSAGE, ACTION
000010     A2731+ZC#TTDWN EQU      X'10' TERMINAL DOWN
000008     A2732+ZC#TTHLD EQU      X'08' HOLD TERMINAL
000004     A2733+ZC#TTUT   EQU      X'04' URGENT TERMINAL
000002     A2734+ZC#TMWR   EQU      X'02' MSG WAIT (FOR ZZTST) RECEIVED
000001     A2735+ZC#TMTC   EQU      X'01' MWRITE FOR ZZTST (SINLGE THREAD)
000001     A2736+ZC#TOMW   EQU      X'01' OUTSTANDING MWRITE (MULTI THREAD)
           A2737+*
00001C     A2738+ZC#TST1   EQU      ZC#TST+1,1
           A2739+*
           A2740+*   EQUATES FOR ZC#TST1
           A2741+*
000080     A2742+ZC#TTIM   EQU      X'80' INTERACTIVE MODE
000040     A2743+ZC#TTMT   EQU      X'40' MASTER TERMINAL
000020     A2744+ZC#TALTS EQU      X'20' ALTERNATE TERM SPECIFIED
000010     A2745+ZC#TTRC   EQU      X'10' ROLLBACK COMPLETE
000008     A2746+ZC#TTMWS EQU      X'08' IMS SENT MSG WAIT
000004     A2747+ZC#TTBTH EQU      X'04' BATCH TERMINAL
000002     A2748+ZC#TTRP   EQU      X'02' ROLLBACK IN PROCESS
000001     A2749+ZC#TTMS   EQU      X'01' MSG TO ORIG TERM SENT
           A2750+*
00001D     A2751+ZC#TST2   EQU      ZC#TST1+1,1
00001D     A2752+ZC#TPRSF EQU      ZC#TST2
           A2753+*
           A2754+*   EQUATES FOR ZC#TST2
           A2755+*
000080     A2756+ZC#TTUNS EQU      X'80' MWRITE ISSUED FROM ZO#UNSMT MODULE
000040     A2757+ZC#TTREL EQU      X'40' RELEASE BUFFER AT MWRITE COMPL
000020     A2758+ZC#TPRMQ EQU      X'20' MSG IN QUEUE
000010     A2759+ZC#TPRMP EQU      X'10' MSG IN PROCESS
000008     A2760+ZC#TTSTA EQU      X'08' SEND AUTO STATUS MESSAGE
000004     A2761+ZC#TCONT EQU      X'04' CONTINUOUS OUTPUT REQUESTED
000002     A2762+ZC#TDELN EQU      X'02' DEL NOTICE - ACTION TO BE SCHED
```

Figure 9-6. Single-thread and Multithread Terminal Control Table (Part 1 of 5)

```
LOC.     LINE   SOURCE STATEMENT
000001   A2763+ZC#TOIQ  EQU   X'01' OUTPUT GENERATED FOR INPUT QUEUING
         A2764+*
00001E   A2765+ZC#TST3  EQU   ZC#TST2+1,1
         A2766+*
         A2767+*     EQUATES FOR ZC#TST3
         A2768+*
000080   A2769+ZC#TTDR  EQU   X'80' DISCONNECT REQUESTED (S/T)
000040   A2770+ZC#TTQNE EQU   X'40' TERMINAL'S LOW QUEUE NOT EMPTY
000020   A2771+ZC#THDRS EQU   X'20' OUTPUT HEADER SAVED
000010   A2772+ZC#TIDN  EQU   X'10' INTERNAL DELIVERY NOTICE
000008   A2773+ZC#TIGM  EQU   X'08' IMS GENERATED ERROR MSG
000004   A2774+ZC#COIP  EQU   X'04' CONTINUOUS OUTPUT IN PROCESS (M/T)
000002   A2775+ZC#TNRDY EQU   X'02' NO IMS READY MSG TO THIS TERMINAL
000001   A2776+ZC#TUNAC EQU   X'01' SEND UNSOLICITED OUTPUT INDICATOR
         A2777+*                          FOR SWITCHED MESSAGES AT ACTION END
         A2778+*
00001F   A2779+ZC#TST4  EQU   ZC#TST3+1,1
         A2780+*
         A2781+*     EQUATES FOR ZC#TST4
         A2782+*
000080   A2783+ZC#ERMEX EQU   X'80' A/M GENERATED ERROR MSG.
000040   A2784+ZC#SFSRB EQU   X'40' REBUILD ALLOWED BY A/P
000020   A2785+ZC#ABTDY EQU   X'20' ABORT DYNAMIC SESSION
000010   A2786+ZC#DYTWD EQU   X'10' ABORT TERM WINDOW
000008   A2787+ZC#SIGN  EQU   X'08' SIGN ON FOR DYNAMIC SESSION
000004   A2788+ZC#ATTRI EQU   X'04' TERM HAS CONFIG. ATTRIBUTES
000002   A2789+ZC#CONSL EQU   X'02' CONSOLE TERMINAL
000001   A2790+ZC#CNTRD EQU   X'01' OUTSTANDING TCS/DISKETTE READ FUNCTION
         A2791+*
000020   A2792+ZC#TST5  EQU   ZC#TST4+1,1 DMS FLAGS
         A2793+*
         A2794+*     EQUATES FOR ZC#TST5
         A2795+*
000080   A2796+ZC#IMPRT EQU   X'80' ISSUED IMPACT FOR ACTION
000040   A2797+ZC#DEPND EQU   X'40' DEPART PENDING
000040   A2798+ZC#DEPRT EQU   X'40' ACTION ISSUED DEPART
000020   A2799+ZC#DMSUP EQU   X'20' ISSUED DSM OPEN FOR UPDATE
000020   A2800+ZC#BND   EQU   X'20' BOUND/UNBOUND STATE
000010   A2801+ZC#UBPND EQU   X'10' UNBIND PENDING
000008   A2802+ZC#DMSRO EQU   X'08' DMS FORCED DEPART WITH ROLLBACK
000004   A2803+ZC#DMSUB EQU   X'04' DMS RUN UNIT UNBOUND
000008   A2804+ZC#UPDRU EQU   X'08' OPENED FOR UPDATE IN THIS RUN-UNIT
000004   A2805+ZC#UPDTD EQU   X'04' UPDATING RUN-UNIT IN THIS SUCCESS UNIT
000002   A2806+ZC#TCALL EQU   X'02' FUNCTION CALL/TERMINATION CALL
000001   A2807+ZC#DMSDR EQU   X'01' DMS REQUEST VIA D.R.M.
         A2808+*
000021   A2809+ZC#TST6  EQU   ZC#TST5+1,1 DMS FLAGS EXTENSION
         A2810+*
         A2811+*     EQUATES FOR ZC#TST6
         A2812+*
000080   A2813+ZC#DMSER EQU   X'80' DMS ERROR IN RUN-UNIT
000040   A2814+ZC#WRK1  EQU   X'40' TEMPORARY FLAG #1
```

Figure 9–6. Single-thread and Multithread Terminal Control Table (Part 2 of 5)

```
 LOC.    LINE   SOURCE STATEMENT
000020  A2815+ZC#WRK2  EQU   X'20' TEMPORARY FLAG #2
000010  A2816+ZC#TTMDF EQU   X'10' MDEFER ISSUED FOR THIS TERMINAL
        A2817+*  THE FOLLOWING STATUS  BYTE TAGS ARE NOT CLEARED WHEN A GLOBAL
        A2818+*  NETWORK DYNAMIC TERMINAL DOES A $$SOFF
        A2819+*          ZC#TTLST
        A2820+*          ZC#TTUT
        A2821+*          ZC#TTMT
        A2822+*          ZC#TNRDY
        A2823+*          ZC#TUNAC
        A2824+*          ZC#ATTRI
        A2825+*
        A2826+*
000022  A2827+ZC#DDPST DS    X DDP STATUS BYTE
        A2828+*
        A2829+*    EQUATES FOR ZC#DDPST
        A2830+*
000080  A2831+ZC#REMTR EQU   X'80' REMOTE TRANS
000040  A2832+ZC#FSOUT EQU   X'40' FIND SESSION OUTSTANDING
000020  A2833+ZC#PSEDO EQU   X'20' PSEUDO TCT
000010  A2834+ZC#DDPOT EQU   X'10' MWRITE FOR DDP
        A2835+*
000023  A2836+ZC#DDPMD DS    X DDP MODE
        A2837+*
        A2838+*    EQUATES FOR ZC#DDP MODE
        A2839+*
0000D9  A2840+ZC#DTR   EQU   C'R' DIRECTORY TRANS. ROUTING
0000C1  A2841+ZC#PTRA  EQU   C'A' PROGRAM   TRANS. ROUTING - ACTIVATE
0000C3  A2842+ZC#PTRC  EQU   C'C' PROGRAM   TRANS. ROUTING - ABORT/CANCEL
0000C5  A2843+ZC#PTRE  EQU   C'E' PROGRAM   TRANS. ROUTING - END
        A2844+*
000024  A2845+ZC#SFLAG DS    XL1 GENERAL SFS FLAG  BYTE
        A2846+*
        A2847+*    EQUATES FOR ZC#SFLAG
        A2848+*
000080  A2849+ZC#INFMT EQU   X'80' INPUT FORMAT
000040  A2850+ZC#DYNM  EQU   X'40' DYNAMIC MEMORY
000020  A2851+ZC#SFBT1 EQU   X'20' SFS FLAG 1
000010  A2852+ZC#ITCF  EQU   X'10' INVALID XTION
000008  A2853+ZC#SFBT2 EQU   X'08' SFS FLAG 2
        A2854+*
000025  A2855+ZC#SFIRC DS    XL1 SFS INPUT RETRY COUNT
        A2856+*
000026  A2857+         DS    XL2 UNUSED
000028  A2858+ZC#TRCTA DS    A TRCT ADDR
00002C  A2859+ZC#TQE   DS    F CANCEL LINK
000030  A2860+ZC#PRFT  DS    F DISPL TO PROCESS FILE TABLE
000034  A2861+ZC#PQCNT DS    H PROCESS QUEUE COUNT
000036  A2862+ZC#MQCNT DS    XL1 LAST ICAM SVC
000037  A2863+ZC#TDELS DS    XL1 DELIVERY NOTICE STATUS
000038  A2864+ZC#LQCNT DS    H LOW QUEUE COUNT
00003A  A2865+ZC#TIN   DS    H TOTAL INPUT COUNT
00003C  A2866+ZC#TINT  DS    H TRANS. INPUT COUNT
```

Figure 9-6. Single-thread and Multithread Terminal Control Table (Part 3 of 5)

```
 LOC.     LINE   SOURCE STATEMENT
00003E   A2867+ZC#TTCM   DS     H TERM COMMAND COUNT
000040   A2868+ZC#TINCH  DS     F TOTAL NO. INPUT  CHARS.
000044   A2869+ZC#TOTCH  DS     F TOTAL NO. OUTPUT  CHARS.
000048   A2870+ZC#TOC    DS     H TOTAL OUTPUT COUNT
00004A   A2871+ZC#TOMSZ  DS     H SOURCE TERM O/P MSG. SIZE
00004C   A2872+ZC#TON    DS     F TIMER LINK
000050   A2873+ZC#IML    DS     H INPUT MESSAGE LENGTH
000052   A2874+ZC#OML    DS     H OUTPUT MESSAGE LENGTH
000054   A2875+ZC#TML    DS     H TIMER MESSAGE LENGTH (OS/3 M.T.)
         A2876+*    OS/3 S.T. USES ZC#COSEQ INSTEAD OF ZC#TML
000054   A2877+ZC#COSEQ  EQU    ZC#TML C/O SEQ COUNT (OS/3 S.T. ONLY)
000056   A2878+ZC#DML    DS     H DDP MSG. LENGTH
000058   A2879+ZC#IBF    DS     A INPUT  BUFFER ADDR
00005C   A2880+ZC#OBF    DS     A OUTPUT BUFFER ADDR
000060   A2881+ZC#TBF    DS     A TIMER  BUFFER ADDR
000064   A2882+ZC#DBF    DS     A DDP    BUFFER ADDR
000068   A2883+ZC#DPREL  DS     A DDP BUFFER RELEASE ADDR
00006C   A2884+ZC#TDELC  DS     XL4 USER CONTINUOUS OUTPUT CODE
000070   A2885+ZC#SFSTC  DS     A SFS TERMINAL CLASS ENTRY ADDR
000074   A2886+ZC#SFSFN  DS     CL8 SFS FORMAT NAME
00007C   A2887+ZC#SESAD  DS     A SESSION STAT TABLE ADDR
000080   A2888+ZC#SESID  DS     F SESSION ID
000084   A2889+ZC#TDMEM  DS     F SFS DYNAMIC MEMORY ADDR
000088   A2890+ZC#TTRID  DS     CL8 TRANS ID (INITIA, DATE/TIME)
000088   A2891+ZC#TRID   EQU    ZC#TTRID OS/4 TAG
000090   A2892+ZC#DLCNT  DS     H IMC DEADLOCK DETECTION COUNT
000092   A2893+          DS     H UNUSED
000094   A2894+ZC#TCB    DS     A THREAD CONTROL BLOCK ADDR
000098   A2895+ZC#TLI    DS     8F TRANS LOCK INDICATOR
0000B8   A2896+ZC#TAUM   DS     8F AUDITED UPDATE MAP
         A2897+*** ZC#TLI AND ZC#TAUM MUST AGREE WITH ZT#TNUMF IN THE THCB
0000D8   A2898+ZC#TTEXT  DS     CL8 TRANSLATED TERM CMD/TRANS CODE
0000D8   A2899+ZC#TCODE  EQU    ZC#TTEXT OS/4 TAG
0000E0   A2900+ZC#TDDRC  DS     CL1 DDR NAME ID CHAR (HIGH BYTE = X'FD')
         A2901+*** THE ABOVE FIELD IS DEFINED IN OS/4 BUT NOT TAGGED
0000E1   A2902+ZC#TDDRN  DS     CL7 DATA DEF REC NAME
0000E8   A2903+ZC#TDFN   DS     CL7 DEFINED FILE NAME
0000EF   A2904+          DS     X UNUSED
0000F0   A2905+ZC#TES    DS     F SUCC ACT RECORD RELATIVE ADDR
         A2906+*     MULTI-THREAD SYSTEMS USE ZC#ES & ZC#CDC IN PLACE OF ZC#TES
0000F0   A2907+          ORG    ZC#TES
0000F0   A2908+ZC#ES     DS     H SUCC ACT RECORD RELATIVE ADDR
0000F2   A2909+ZC#CDL    DS     H CONTINUITY DATA LENGTH
         A2910+*
0000F4   A2911+ZC#WAI    DS     H WORK AREA INC
0000F6   A2912+ZC#CDI    DS     H CONTINUITY DATA AREA INC
0000F8   A2913+ZC#TTTN   DS     XL1 TCT RECORD NUMBER
0000F9   A2914+          DS     XL1 UNUSED
0000FA   A2915+          DS     H UNUSED
         A2916+*    MULTI-THREAD USES ZC#CDR & ZC#CES INSTEAD OF ZC#TTTN & ZC#TINT
0000F8   A2917+          ORG    ZC#TTTN
0000F8   A2918+ZC#CDR    DS     H TCT RECORD NUMBER
```

Figure 9–6. Single-thread and Multithread Terminal Control Table (Part 4 of 5)

```
 LOC.    LINE   SOURCE STATEMENT
0000FA  A2919+ZC#CES    DS    H SUCC ACT REL ADDR _ ROLLBACK
0000FC  A2920+ZC#SCFR   DS    XL4 COUNT FIELD FOR ROLLBACK
        A2921+*
000100  A2922+ZC#TTIR   DS    XL1 TERM IND FOR ACTION PROG USING ROLLBACK
000100  A2923+ZC#TIR    EQU   ZC#TTIR OS/4 TAG
000100  A2924+          ORG   ZC#TIR
000100  A2925+ZC#TRWA   DS    F TRACE WORK AREA
000104  A2926+ZC#FBPA   DS    H * FIRST BLOCK OF PARTITION
000106  A2927+ZC#CBPA   DS    H * CURRENTLY ACCESSED BLOCK
000108  A2928+ZC#LBPA   DS    H * LAST BLOCK OF PARTITION
00010A  A2929+ZC#NRBCB  DS    H *# OF REM.BYTES IN CURR. BLOCK
        A2930+*
00010C  A2931+ZC#TLNAM  DS    CL4 LINE NAME
000110  A2932+ZC#TCHAR  DS    CL4 TERMINAL CHARACTERISTICS
000110  A2933+ZC#TTSL   EQU   ZC#TCHAR SCREEN LENGTH
000111  A2934+ZC#TTSW   EQU   ZC#TTSL+1 SCREEN WIDTH
000112  A2935+ZC#TTTYP  EQU   ZC#TTSW+1 TERMINAL TYPE
        A2936+*
        A2937+*   EQUATES FOR ZC#TTTYP
        A2938+*
000000  A2939+ZC#TTNFC  EQU   X'00' U100/U200/UTS1n/TTY
000080  A2940+ZC#TT4PR  EQU   X'80' UTS400 PR
000040  A2941+ZC#TT4U2  EQU   X'40' UTS400 CP (U2 MODE)
000020  A2942+ZC#TT4U4  EQU   X'20' UTS400 CP (U4 MODE) OR UTS400
000010  A2943+ZC#TT327  EQU   X'10' IBM 3271
000008  A2944+ZC#TTU40  EQU   X'08' UTS40
000004  A2945+ZC#TTU20  EQU   X'04' UTS20
000002  A2946+ZC#TT40T  EQU   X'02' UTS400 TEXT EDITOR
        A2947+*
000113  A2948+ZC#TTATT  EQU   ZC#TTTYP+1 TERMINAL ATTRIBUTES
        A2949+*
        A2950+*   EQUATES FOR ZC#TTATT
        A2951+*
000080  A2952+ZC#TTKAN  EQU   X'80' KATAKANA
000040  A2953+ZC#TTNVI  EQU   X'40' NON-VIDEO
000020  A2954+ZC#TTSBT  EQU   X'20' SCREEN BYPASS
000010  A2955+ZC#TTPKT  EQU   X'10' PACKET PDN TERMINAL
000008  A2956+ZC#TTCST  EQU   X'08' CIRCUIT SWITCH PDN TERMINAL
000004  A2957+ZC#TTCCT  EQU   X'04' TERMINAL ON CLUSTER CONTROLLER
        A2958+*
000114  A2959+ZC#TINER  DS    F SFS ERROR FIELD
000118  A2960+ZC#TRIDA  DS    A PTR TO TRIDT ENTRY FOR CURRENT TRANSACTION
00011C  A2961+ZC#ALTID  DS    F ALTERNATE TERM ID
000120  A2962+ZC#TFIN   DS    0F THIS MUST ALWAYS BE AT END
000120  A2963+ZC#TLEN   EQU   *-ZC#DTCT
000000  A2964+ZO#OUTMT  CSECT
```

Figure 9–6. Single-thread and Multithread Terminal Control Table (Part 5 of 5)

9.10. OTHER DEBUGGING RESOURCES

Link map and symbol table

To find the cause of an action program snap dump requires the use of both the snap dump and user action program compile and link. Very briefly, we'd like to point out data in the link map and symbol table of your action program useful in debugging. Figures 9-7 and 9-8 show the link map and symbol table for RCCUST.

Module RCCUST and P?IMS000

Looking at Figure 9-7, the first object module is RCCUST and its lnk-org is 0. Following RCCUST is P?IMS000. This object module handles initiation and termination procedures for the action program. It also handles communication between the program and the interface areas. Its lnk-org is 12A0.

Module ZF#LINK

The third object module is ZF#LINK. This module provides the interface between action program function calls and IMS. Its lnk-org is 14F8.

Module P?SERIAL

The object module P?SERIAL is responsible for making the RPG II action program serially reusable. It clears all switches and indicators prior to an action program getting control. However, the RPG II programmer must reset all fields and arrays prior to program execution. The important point to remember is that RPG II action programs must be serially reusable since IMS doesn't reload a program if it's already in main storage.

Module P?SPL000 and P?IMSIX0

The next two object modules included in Figure 9-7 are P?SPL000 and P?IMSIX0. They provide I/O interfaces between IMS and the RPG II action program. P?SPL000 handles all general I/O interface needs and P?IMSIX0 handles all requests to indexed files.

Module P?IMSEQ0 and P?IMSD00

Two other object modules not present in Figure 9-7 but which could have been included are: P?IMSEQ0, which handles sequential file requests; and P?IMSD00, which handles DAM file requests. Which modules are actually included depends upon the specific I/O design of the action program.

**ERROR field*

Figure 9-8 shows the symbol table for RCCUST. The important data it contains is the location of *ERROR at relative location 1B0.

LINK MAP

```
UNIVAC SYSTEM OS/3  LINKAGE EDITOR                                                                                   VER800403
DATE- 81/04/07 TIME- 10.36

CONTROL STREAM ENCOUNTERED AND PROCESSED AS FOLLOWS-

            // PARAM OUT=LOD
            /$
                        LOADM RCCUST
                        INCLUDE RCCUST
*EMBEDDED*              INCLUDE P7IMS000,$Y$OBJ
*EMBEDDED*              INCLUDE ZF#LINK,$Y$OBJ
*EMBEDDED*              INCLUDE P7SERIAL,$Y$OBJ
*EMBEDDED*              INCLUDE P7SPL000,$Y$OBJ
*EMBEDDED*              INCLUDE P7IMSIX0,$Y$OBJ
            /*


                                                 *DEFINITIONS DICTIONARY*

SYMBOL.   TYPE.  PHASE. ADDRESS.        SYMBOL.   TYPE.  PHASE. ADDRESS.        SYMBOL.   TYPE.  PHASE. ADDRESS.

ADDKY     ENTRY  ROOT   000018A4        ARETURN   ENTRY  ROOT   000018E8        BUILD     ENTRY  ROOT   0000188C
CLOSE     ENTRY  ROOT   0000191C        CMDRB     ENTRY  ROOT   000018E0        DELETE    ENTRY  ROOT   000018FC
DELKY     ENTRY  ROOT   000018A8        DLADR     ENTRY  ROOT   000018FC        DLKCP     ENTRY  ROOT   000018B0
ENDCRL    ENTRY  ROOT   000018DC        ESETL     ENTRY  ROOT   00001908        ESLMT     ENTRY  ROOT   00001908
FIND      ENTRY  ROOT   00001920        FREE      ENTRY  ROOT   0000190C        GET       ENTRY  ROOT   000018F0
GETLOAD   ENTRY  ROOT   00001898        GETUP     ENTRY  ROOT   000018F4        GTADR     ENTRY  ROOT   00001900
IMA       ENTRY  ROOT   00000018        INSERT    ENTRY  ROOT   00001900        KE$ALP    ENTRY  ABS    00001F6A
KE$RES    ENTRY  ABS    00001F6A        LNKCP     ENTRY  ROOT   000018AC        OMA       ENTRY  ROOT   00000020
OPEN      ENTRY  ROOT   00001918        OPENF     ENTRY  ROOT   00001918        P7CDA     ENTRY  ROOT   00001796
P7IMA     ENTRY  ROOT   00001764        P7IMSIX0  CSECT  ROOT   00001A58        P7IMS000  CSECT  ROOT   00001530
P7OMA     ENTRY  ROOT   0000150E        P7PIB     ENTRY  ROOT   000017C8        P7SERIAL  CSECT  ROOT   00001978
P7SPL000  CSECT  ROOT   000019E0        PIB       ENTRY  ROOT   00000028        PUT       ENTRY  ROOT   000018F8
RCCUST    CSECT  ROOT   00000000        RDID      ENTRY  ROOT   000018F0        RDIDC     ENTRY  ROOT   00001924
RDIDCL    ENTRY  ROOT   00001920        RDIDL     ENTRY  ROOT   000018F4        RDKEY     ENTRY  ROOT   000018B4
RDKEYC    ENTRY  ROOT   00001890        RDKEYCL   ENTRY  ROOT   0000188C        RDKEYL    ENTRY  ROOT   00001888
RDKYI     ENTRY  ROOT   0000188C        RDKYIC    ENTRY  ROOT   00001888        RDSQ      ENTRY  ROOT   000018C8
RDSQC     ENTRY  ROOT   0000191C        RDSQCL    ENTRY  ROOT   00001914        RDSQI     ENTRY  ROOT   000018CC
RDSQIC    ENTRY  ROOT   000018E4        RDSQL     ENTRY  ROOT   0000190C        RDSR      ENTRY  ROOT   000018C0
RDSRC     ENTRY  ROOT   00001910        RDSRCL    ENTRY  ROOT   000018E8        RDSRL     ENTRY  ROOT   000018C4
REBUILD   ENTRY  ROOT   00001890        RELREC    ENTRY  ROOT   00001910        RETURN    ENTRY  ROOT   00001928
SEND      ENTRY  ROOT   00001924        SETL      ENTRY  ROOT   00001904        SETLOAD   ENTRY  ROOT   0000189C
SNAP      ENTRY  ROOT   000018E4        SSLOCK    ENTRY  ROOT   000018D0        SSUNLK    ENTRY  ROOT   000018D4
STCRL     ENTRY  ROOT   000018D8        STLMT     ENTRY  ROOT   00001909        SUB       ENTRY  ROOT   000018A0
SUBPROG   ENTRY  ROOT   000018A0        UNLOCK    ENTRY  ROOT   00001914        WRID      ENTRY  ROOT   000018F8
XR3IMS    ENTRY  ROOT   00001894        ZF#LINK   CSECT  ROOT   00001888


                                                  ** ALLOCATION MAP **

                        LOAD MODULE -    RCCUST          SIZE -    00001F6A

 PHASE NAME  TRANS ADDR    FLAG        LABEL      TYPE        ESID          LNK ORG      HIADDR      LENGTH        OBJ ORG
 RCCUST00       NODE - ROOT                                                 00000000     00001F69    00001F6A
 *** START OF AUTO-INCLUDED ELEMENTS -
 *** END OF AUTO-INCLUDED ELEMENTS -
        - 81/04/07 10.35 -            RCCUST     OBJ
                           L          RCCUST     CSECT       01            00000000                 DEFERREn      00000000
                                      IMA          ENTRY     01            00000018                               00000018
                                      OMA          ENTRY     01            00000020                               00000020
                                      PIB          ENTRY     01            00000028                               00000028
                           L                       TRF       01                                     **0000153n**  00000000
        - 80/05/28 14.42 -            P7IMS000   OBJ
                                      P7IMS000   CSECT       01            00001530     00001883    00000354      00000000
                                      P7OMA        ENTRY     01            0000150E                               000000AE
                                      P7IMA        ENTRY     01            00001764                               00000234
                                      P7CDA        ENTRY     01            00001796                               00000266
                                      P7PIB        ENTRY     01            000017C8                               00000298
        - 80/02/09 23.03 -            ZF#LINK    OBJ
                                      ZF#LINK    CSECT       01            00001888     00001977    000000Fn      00000000
                                      XR3IMS       ENTRY     01            00001894                               0000000C
                                      BUILD        ENTRY     01            0000188C                               00000004
                                      REBUILD      ENTRY     01            00001890                               00000008
                                      GET          ENTRY     01            000018F0                               00000068
                                      GETUP        ENTRY     01            000018F4                               0000006C
                                      PUT          ENTRY     01            000018F8                               00000070
                                      DELETE       ENTRY     01            000018FC                               00000074
                                      INSERT       ENTRY     01            00001900                               00000078
                                      SETL         ENTRY     01            00001904                               0000007C
                                      ESETL        ENTRY     01            00001908                               00000080
                                      FREE         ENTRY     01            0000190C                               00000084
                                      RELREC       ENTRY     01            00001910                               00000088
                                      UNLOCK       ENTRY     01            00001914                               0000008C
                                      OPEN         ENTRY     01            00001918                               00000090
                                      CLOSE        ENTRY     01            0000191C                               00000094
                                      FIND         ENTRY     01            00001920                               00000098
                                      SEND         ENTRY     01            00001924                               0000009C
                                      RETURN       ENTRY     01            00001928                               000000A0
                                      ARETURN      ENTRY     01            000018E8                               00000060
                                      SNAP         ENTRY     01            000018E4                               0000005C
```

Figure 9–7. Link Map for RCCUST (Part 1 of 2)

```
PHASE NAME  TRANS ADDR    FLAG      LABEL      TYPE      ESID        LNK ORG      HIADDR      LENGTH       OBJ ORG
                                    SUB         ENTRY     01          000018A0                               00000018
                                    RDSQL       ENTRY     01          0000190C                               00000084
                                    RDIDC       ENTRY     01          00001924                               0000009C
                                    RDIDCL      ENTRY     01          00001920                               00000098
                                    RDSQC       ENTRY     01          0000191C                               00000094
                                    RDSQCL      ENTRY     01          00001914                               0000008C
                                    RDSRC       ENTRY     01          00001910                               00000088
                                    RDSRCL      ENTRY     01          000018E8                               00000060
                                    RDSQIC      ENTRY     01          000018E4                               0000005C
                                    RDKEYC      ENTRY     01          00001890                               00000008
                                    RDKEYCL     ENTRY     01          0000188C                               00000004
                                    RDKYIC      ENTRY     01          00001888                               00000000
                                    GTADR       ENTRY     01          00001900                               00000078
                                    DLADR       ENTRY     01          000018FC                               00000074
                                    ADDKY       ENTRY     01          000018A4                               0000001C
                                    DELKY       ENTRY     01          000018A8                               00000020
                                    LNKCP       ENTRY     01          000018AC                               00000024
                                    DLKCP       ENTRY     01          00001880                               00000028
                                    WRID        ENTRY     01          000018F8                               00000070
                                    RDID        ENTRY     01          000018F0                               00000068
                                    RDIDL       ENTRY     01          000018F4                               0000006C
                                    RDKEY       ENTRY     01          00001884                               0000002C
                                    RDKEYL      ENTRY     01          00001888                               00000030
                                    RDKYI       ENTRY     01          0000188C                               00000034
                                    RDSR        ENTRY     01          000018C0                               00000038
                                    RDSRL       ENTRY     01          000018C4                               0000003C
                                    RDSQ        ENTRY     01          000018C8                               00000040
                                    RDSQI       ENTRY     01          000018CC                               00000044
                                    STLMT       ENTRY     01          00001904                               0000007C
                                    ESLMT       ENTRY     01          00001908                               00000080
                                    SSLOCK      ENTRY     01          000018D0                               00000048
                                    SSUNLK      ENTRY     01          000018D4                               0000004C
                                    STCRL       ENTRY     01          000018D8                               00000050
                                    ENDCRL      ENTRY     01          000018DC                               00000054
                                    CMDRB       ENTRY     01          000018E0                               00000058
                                    OPENF       ENTRY     01          00001918                               00000090
                                    SUBPROG     ENTRY     01          000018A0                               00000018
                                    SETLOAD     ENTRY     01          0000189C                               00000014
                                    GETLOAD     ENTRY     01          00001898                               00000010
           - 79/08/08 18.03 -       P?SERIAL    OBJ
                                    P?SERIAL  CSECT       01          00001978    000019DF    0000006A       00000000
           - 79/08/08 17.58 -       P?SPL000    OBJ
                                    P?SPL000  CSECT       01          000019E0    00001A57    0000007A       00000000
           - 80/03/21 16.50 -       P?IMSIXO    OBJ
                                    P?IMSIXO  CSECT       01          00001A58    00001F69    00000512       00000000
                00000000


                                            FLAG CODES -
B - BLK DATA CSECT     D - AUTO-DELETED       E - EXCLUSIVE 'A' REF   G - GENERATED EXTRN    I - INCLUSIVE 'V' REF
L - DEFERRED LENGTH    M - MULTIPLY DEFINED   N - NOT INCLUDED        P - PROMOTED COMMON    R - SHARED REC PRODUCED
S - SHARED ITEM        U - UNDEFINED REF      V - VCON ITEM
*ANY OTHER CODES REPRESENT PROCESS ERRORS*

LINK EDIT OF 'RCCUST'   COMPLETED
DATE- 81/04/07 TIME- 10.37
ERRORS ENCOUNTERED- 0000  UPSI- X'40'
```

Figure 9–7. Link Map for RCCUST (Part 2 of 2)

SYMBOL TABLE

```
                 PIB                                                                              NOTE  132
                                                    SYMBOL TABLES

RESULTING INDICATORS

ADDRESS RI       ADDRESS RI       ADDRESS RI       ADDRESS RI       ADDRESS RI       ADDRESS RI       ADDRESS RI

 000014 1P        000015 LR        000016 00        000017 01        000018 02        00002A 20        000034 30
 00003F 41        000040 42        000048 50        000053 61        00005C 70        00007A LO        000085 H0
 000086 H1        000087 H2        000088 H3        000089 H4        00008A H5        00008B H6        00008C H7
 00008D H8        00008E H9        00008F U1        000090 U2        000091 U3        000092 U4        000093 U5
 000094 U6        000095 U7        000096 U8

FIELD NAMES

ADDRESS FIELD           ADDRESS FIELD           ADDRESS FIELD           ADDRESS FIELD           ADDRESS FIELD

0001B0  *ERROR          000210  CUST            000215  SIGN            000216  AMOUNT          000218  CUSTID
000220  NAME            000234  ADDR            000243  CITY            000252  ZIP             000257  BALDUE
00020C  END             00025C  AMT             00025F  NEWBAL

LITERALS

ADDRESS LITERAL                             ADDRESS LITERAL                             ADDRESS LITERAL

000265  +                                   000266  -                                   000267  X'00D3'
000269  X'100A0200'                         00026D  NAME     -                          000277  X'10010300'
00027B  ADDRESS -                           000285  X'10010400'                         000289  CITY-ST -
000293  X'1001041E'                         000297  ZIP -                               00029D  X'10040200'
0002A1  OLD BALANCE -                       0002AF    -,---,--/.-- -                    0002BF  X'10040100'
0002C3  TRANSACTION -                       0002D1    -/.----                           0002DA  NEW BALANCE -
0002E8    ,---,---,--/.-- -
     010                                                 CUSTID                                   NOTE  205
     000                                                                                          NOTE  115


LITERALS

ADDRESS LITERAL                             ADDRESS LITERAL                             ADDRESS LITERAL

000388  X'10010106'                         00038C  RCMENU                              000392  D
000393  INVALID CUSTOMER ID                 0003A6  INVALID SIGN                        0003B2  INVALID AMOUNT

NOTE  115  THE NUMBER OF SYMBOLS USED IN THIS PROGRAM CAUSED THE COMPILER TO RUN LESS
           EFFICIENTLY THAN IF AN INCREASED MEMORY SIZE WERE ALLOCATED.

NOTE  132  NO INPUT AND/OR OUTPUT SPECIFICATIONS FOUND FOR THIS FILE.

NOTE  205  WARNING:  FIELD NAME IS UNREFERENCED.
```

Figure 9-8. Symbol Table for RCCUST

Appendix A. Using Device Independent Control Expressions and Field Control Characters

A.1. GENERAL

Using DICE for formatting

You use device independent control expressions (DICE) to format input and output messages handled by action programs. These codes control various operations, such as cursor positioning and carriage return, on the terminal screen.

Scope of section

This appendix supplies all DICE sequences and their interpretations, and describes how to use them in formatting messages. In addition, it presents limited information concerning the use of field control characters.

A.2. FORMATTING MESSSAGES

Output Messages

There are numerous methods for formatting output messages. The action program can use:

Other ways to format messages

- Screen format services. For a complete discussion of how to use screen format services, see Section 6.
- Device independent control expressions
- Format control expressions with UNISCOPE 100 and 200 display terminals
- Field control characters (FCCs) with workstations and Universal Terminal System terminals

DICE and FCCs

Format control expressions

This appendix supplies information on DICE sequences and how to use them. We will also include limited information concerning field control characters since one program, RCMENU, presented in Section 3 of this manual uses this type of formatting. For detailed information concerning format control expressions, consult the UNISCOPE display terminal programmer reference, UP-7807 (current version).

Use of format control expressions

When a program uses format control expressions, it must include a different formatting routine for each type of terminal receiving the output. Figure A-1 illustrates this.

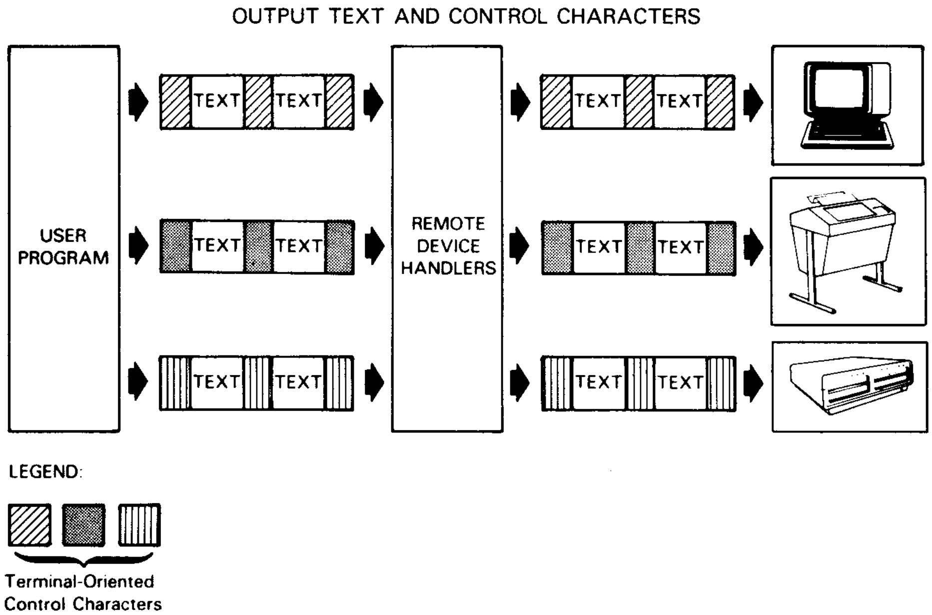


Figure A-1. Using Terminal-Oriented Control Characters to Format Messages

Handling DICE sequences

Using DICE sequences to format messages eliminates this problem. The remote device handler converts DICE sequences to control characters for each destination terminal, regardless of type. Some of the control character functions are:

Functions performed

- **Line feed** - cursor movement to the first space of a new line
- **Form feed** - cursor to the home position of a new page
- **Carriage return** - cursor to the beginning of the same line
- **Cursor movement** to a specific row and column on a display

DICE placement

You can place DICE sequences anywhere in a message. As you can see in Figure A-2, DICE sequences simplify message formatting.

Coding with DICE

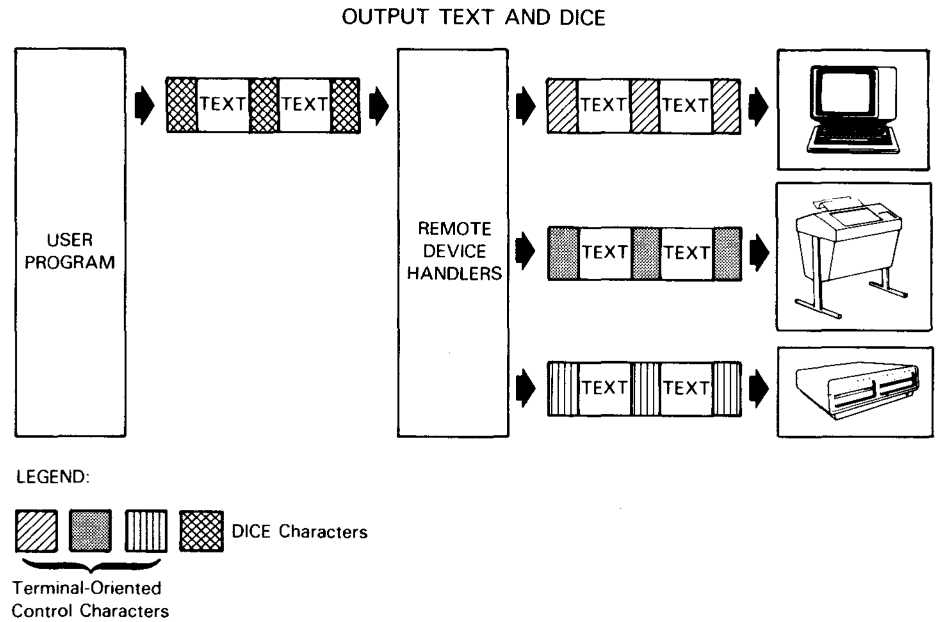


Figure A-2. Using Dice Sequences to Format Messages

Input Messages

Using input DICE

For input, control characters received in a message are converted into DICE sequences by the remote device handler. For certain terminals, your program can analyze these sequences to determine cursor position. In addition, input DICE is handy for message switch applications because control characters in each input message are converted to DICE sequences. The remote device handler converts these sequences into the appropriate control characters for the destination terminal.

Stripping DICE

When you specify EDIT=c or EDIT=tablename in the ACTION section of the IMS configuration, input DICE is stripped from your input message. You should specify EDIT=c or EDIT=tablename in your IMS configuration. (Specify EDIT=tablename only when you generate an edit table for the action. See Appendix B.)

A.3. DICE AND ICAM

Defining DICE at network definition

You can turn DICE on or off when you define your communications network with the DICE operand of the TERM macroinstruction.

```
DICE=({ON })
      {OFF}
```

where:

DICE=(ON)
: The remote device handler creates input DICE according to your input terminal cursor movements: DICEs are created automatically.

DICE=(OFF)
: The remote device handler doesn't create input DICE.

DICE=(ON) is recommended

The default is DICE=(ON). We recommend that you specify DICE=(ON) or omit this operand because many IMS features require the use of input DICE. Certain terminal commands and IMS transaction codes aren't available when you specify DICE=(OFF).

See ICAM concepts and faciltiies, UP-8194 (current version), for a detailed explanation of input DICE creation, and the IMS system support functions user guide, UP-8364 (current version), for specific IMS considerations.

A.4. THE FORMAT OF DICE SEQUENCES

DICE format

The format of a DICE sequence is:

select character	function code	m field	n field

where:

select character
: Is a hexadecimal character (10) designating the start of a DICE sequence.

function code
: Defines the device control sequence that is recognized by the remote device handlers on input. On output, this code is a 1-character field defining the operation to be performed on the text message. DICE function codes are listed in Table A-1.

m field and n field
: These fields are treated as parameters to the DICE function code. Their actual definition varies and is determined by the individual DICE macroinstruction. Generally, m relates to vertical positioning and n refers to horizontal positioning.

Text message alignment

Cursor movement

These fields may be expressed in absolute values (m_a and n_a) or relative displacement values (m_r and n_r). The absolute values align the text message to the actual location (row and column) on a page or screen. The relative displacement values give a relative location from the present position of the cursor, that is, move cursor two rows down and one column to the right. All values are expressed in hexadecimal notation.

DICE INTERPRETATION

DICE commands

Table A-1 illustrates all the possible DICE input/output commands and their explanation.

Table A-1. DICE Input/Output Commands, Codes, and Device Interpretation (Part 1 of 4)

DICE Macro-Instruction	Function	Function Code Value	I/O	m	n	Character-oriented Devices (1)	CRT Devices	Page Printing Devices (n is Not Interpreted)	Communications Output Printer (COP) or Terminal Printer (TP) (1)
ZO#BEG	Beginning of current line control	08_{16}	INPUT	00	00	Carriage return	Not used	Not used	Not used
			OUTPUT	m_r	n_r	Carriage return followed by m line feeds and n spaces to the right	Move cursor to beginning of current line. Then move cursor m lines down and n columns to the right.	Advance m lines.	m line feeds and n spaces to the right.
ZO#TABS	Set tab stop at an absolute position 4	09_{16}	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_a	n_a	No line feed, space to right.	Set tab stop at row m and column n.	Advance m lines.	Not used
ZO#FORMA	Forms control with clear; protected/unprotected data	$0A_{16}$	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_a	n_a	Action is optional. (2)	Move cursor to row m and column n and clear protected/unprotected data to end of screen.	Action is optional. (2)	Action is optional. (2)
ZO#ERSLN	Erase to end of line	$0B_{16}$	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_a	n_a	No action	Cursor does not move. Unprotected data to the end of a line or to the end of the first unprotected field is cleared, whichever comes first.	Advance 0 lines.	Not used
ZO#POS	New line control	04_{16}	INPUT	00	00	Carriage return, line feed	Cursor return	Not used	Not used
			OUTPUT	m_r	n_r	Carriage return, line feed, followed by m line feeds and n spaces to the right.	Move cursor to beginning of next line. Then move cursor m lines down and n columns to the right	Advance (m+1) lines.	Line feed, followed by m line feeds and n spaces to the right.

Table A-1. DICE Input/Output Commands, Codes, and Device Interpretation (Part 2 of 4)

DICE Macro-instruction	Function	Function Code Value	I/O	m	n	Character-oriented Devices (1)	CRT Devices	Page Printing Devices (n is Not Interpreted)	Communications Output Printer (COP) or Terminal Printer (TP) (1)
ZO#POSC	New line control with clear	05_{16}	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_r	n_r	Carriage return, line feed, followed by m line feeds and n spaces to the right	Same as 04_{16} except area between start and end positions is cleared.	Advance (m+1) lines.	Line feed, followed by m line feeds and n spaces to the right
ZO#CUR	Current position control	06_{16}	INPUT	01	00	Line feed	Line feed	End of input card	Not used
			OUTPUT	m_r	n_r	m line feeds and n spaces to the right	Move cursor m lines down and n columns to the right.	Advance m lines.	Insert n spaces if nonsignificant space suppression is allowed. If not, insert n DC3 characters; m is not interpreted. (3)
ZO#CURC	Current position control with clear	07_{16}	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_r	n_r	m line feeds and n spaces to the right	Insert n spaces if nonsignificant space suppression is allowed. If not, insert n DC3 characters; m is not interpreted. (3)	Advance m lines.	Insert n spaces if nonsignificant space suppression is allowed. If not, insert n DC3 characters; m is not interpreted. (3)
ZQ#COORD	Set coordinates	01_{16}	INPUT	m	n	Not used	m and n represent the start-of-entry (SOE) cursor coordinates.	Not used	Not used
			OUTPUT	m_a	n_a	Action is optional. (2)	Move cursor to row m and column n.	Action is optional. (2)	Action is optional. (2)
ZO#FORM	Forms control	02_{16}	INPUT	01	01	Form feed	Form feed	Not used	Not used
			OUTPUT	m_a	n_a	Form feed, carriage return, and advance to line m and column n (m-1 line feeds and n—1 spaces to the right)	Move cursor to row m and column n.	Top of form and advance to line m (m-1 line feeds)	Form feed, line feed, and advance to line m and column n (m—1 line feeds and n—1 spaces to the right)

Table A-1. DICE Input/Output Commands, Codes, and Device Interpretation (Part 3 of 4)

DICE Macro-instruction	Function	Function Code Value	I/O	m	n	Character-oriented Devices ①	CRT Devices	Page Printing Devices (n is Not Interpreted)	Communications Output Printer (COP) or Terminal Printer (TP) ①
ZO#FORMC	Forms control with clear unprotected data	03_{16}	INPUT	—	—	Not used	Not used	Not used	Not used
			OUTPUT	m_a	n_a	Action is optional. ②	Move cursor to row m and column n, and clear unprotected data to end of screen.	Action is optional ②	Action is optional. ②

NOTES:

① Most character-oriented terminals can be strapped to handle the carriage return (CR) character and the line feed (LF) character as follows:

- CR
  1. print mechanism moves to beginning of the same line; or
  2. print mechanism moves to the beginning of the same line followed by a line feed.
- LF
  1. line feed (no column change); or
  2. line feed followed by return of the print mechanism to the beginning of the new line.

To achieve device independence between terminal types, the character-oriented terminals must use the first option for CR and the first option for LF if the device macroinstruction is ZO#CUR or ZO#BEG.

Use the first option when the character-oriented terminals are a part of a message switch environment.

Certain terminals do not have a form feed capability (i.e., some teletypewriters). For these terminals, the DICE expressions that specify form feed will line feed.

② The set coordinates macroinstruction (ZO#COORD) or the forms control with clear macroinstruction (ZO#FORMC), when acted upon by character-oriented or page-printing terminals, will vary in its action, depending on the usage of the DICE keyword parameter of the TERM macroinstruction at network definition time:

TERM ...,DICE? FORMS ,...

When FORMS is specified, the set coordinates macroinstruction is interpreted as the forms control macroinstruction.

When NEWLINE is specified, the set coordinates macroinstruction and the forms control with clear macroinstruction result in a carriage return, line feed for character-oriented terminals, or advance one line for page-oriented terminals; m and n are not interpreted.

When the DICE parameter is not specified, the default option is NEWLINE.

Table A-1. DICE Input/Output Commands, Codes, and Device Interpretation (Part 4 of 4)

③ The UNISCOPE display terminal suppresses nonsignificant spaces on each line (except for the line containing the cursor) when text is transmitted to the processor or printed locally on the COP or TP.

Your program may send data to the UNISCOPE screen containing significant blank segments that include the last column of the screen. If this data is transmitted from the terminal to the processor or is printed locally on the COP or TP, the blank segments must consist of nonspace characters that are nondisplayable. The DC3 character meets these qualifications. The ICAM interface provides your program with the capability to prevent nonsignificant space suppression on the UNISCOPE display terminal. The "current position control with clear" is the only DICE macroinstruction that can perform a clear function if your program is preventing nonsignificant space suppression.

NOTE:

The ASCII-to-EBCDIC translation table is modified so that the DC3 character is translated to space 40_{16} for input from the UNISCOPE display terminal.

④ Using DICE function code 09_{16} for setting a tab stop, m=0 and n=0 results in a tab stop being placed at the current cursor location (no cursor positioning is performed). This applies to UNISCOPE and UTS devices only. For teletypewriters and DCT 500 terminals, a space character is inserted.

When m or n is greater than the maximum allowable m or n, action varies depending on the remote terminal:

- UNISCOPE display terminals - wraparound occurs on the screen.
- Character-oriented terminals - gives different results depending on device characteristics.

A.5. INTERPRETING DICE SEQUENCES

Device independent

When using DICE, your program doesn't need to be aware of the terminal type. A particular DICE denotes the same positioning on any terminal. There are some exceptions that result from terminal limitations.

Factors controlling interpretation of DICE sequences

The interpretation of a DICE by the remote device handler is controlled by:

- DICE function code
- DICE m and n fields
- The terminal involved
- The particular device on the terminal being used

Terminals supporting DICE

The remote device handlers currently provide device-independent support for three classes of remote terminal devices:

Hard copy, character-oriented devices

1. Hard copy character-oriented devices, such as the SPERRY UNIVAC Data Communications Terminal 475 (DCT 475), Data Communications Terminal 500 (DCT 500), Data Communications Terminal 524 (DCT 524), and Data Communications Terminal 1000 (DCT 1000), and TELETYPE teletypewriter models 28, 32, 33, 35, 37.

Hard copy, page printer devices

2. Hard copy page printer type device, such as the SPERRY UNIVAC 1004 Card Processor System, Data Communications Terminal 2000 (DCT 2000), and 9200/9300 Systems, and the IBM 2780.

CRT terminals

3. CRT-type terminals, such as the UNISCOPE 100 and 200 and the UTS 400 Display Terminals.

Table A-2 defines the primary output device and the primary input device for each terminal type.

Table A-2. DICE Primary Devices

DICE primary devices

Terminal Type	Primary Output Device	Primary Input Device
Character-oriented terminals	Printer	Keyboard
Page printing terminals	Printer	Card reader
CRT terminals	Screen	Keyboard

Auxiliary devices supported

In addition to the specified primary devices, each terminal has the ability to support one or more auxiliary devices. The auxiliary devices suggested by each terminal are listed in Table A-3.

Table A-3. DICE Usage for Auxiliary Devices

Remote Terminals	Auxiliary Device	DICE Usage
UNISCOPE	■ Tape cassette (TCS) ■ Communications output printer (COP) ■ 800 terminal printer (TP)	DICE is applied to the COP. ①
DCT 1000	■ Card reader/card punch ■ Paper tape reader/punch	DICE is applied as if the output/input is to/from the primary device, even though it is for the auxiliary device. ②
DCT 500/TTY	■ Paper tape reader/punch	
DCT 524	■ Tape cassette (TCS) in paper tape read and write only	
Batch terminals	■ Punch	DICE is used for end of network buffer sentinel. No forms control action is taken.

NOTES:

① When the print transparent option is not used, DICE is applied to the UNISCOPE screen even though the output is sent to an auxiliary device of the UNISCOPE terminal. In this case, the format of the data printed on the COP or TP is identical to the screen format. Nonsignificant space suppression by the UNISCOPE terminal may have to be prevented to keep the formats identical.

The full capability of DICE cannot be applied to the COP because of hardware characteristics. All data to a UNISCOPE auxiliary device passes through the UNISCOPE terminal. When DICE is applied to the COP, the use of print transparent mode means that no carriage returns are transferred to the COP. Line feeds and form feeds take a storage position in the UNISCOPE storage and are nondisplayable. These characters are passed to the COP where:

- an LF causes a line feed followed by return of the print mechanism to the beginning of the new line; and

- an FF causes a page eject and positioning of the print mechanism at the beginning of the first line of the form.

The COP has no tabbing capability.

These characteristics are reflected in the interpretation of DICE output function codes for the COP as shown in Table A-2.

For messages sent to a UNISCOPE auxiliary device with transparent transfer, the cursor to home (ESC e) sequence is inserted at the beginning of the text by the RDH.

② The control characters that are generated from the DICE macroinstructions are always created for the primary device of a character-oriented device, even though your program is sending to an auxiliary device. The message and these control characters (carriage returns, line feeds, form feeds, and spaces) will be punched/written by the output auxiliary device that was specified by your program or was switch-selected by the terminal operator. If the punched/written data is later read by the terminal's input auxiliary device, the carriage returns, line feeds, and form feeds are converted to input DICE as specified in Table A-1.

A.6. USING DICE IN AN RPG II ACTION PROGRAM

Coding DICE sequences

To format an output message, you enter DICE sequences on the output form in columns 45–70, along with the message text. The remote device handler takes the DICE sequence and converts it into the form required by the destination terminal. The 4-character DICE sequence determines how the output message looks when it appears at the terminal. The DICE sequences themselves never appear on the terminal screen.

Example

Figure A-3 shows how an action program generates a formatted output message using DICE sequences. Figure A-4 shows how the message looks when it appears at the terminal.

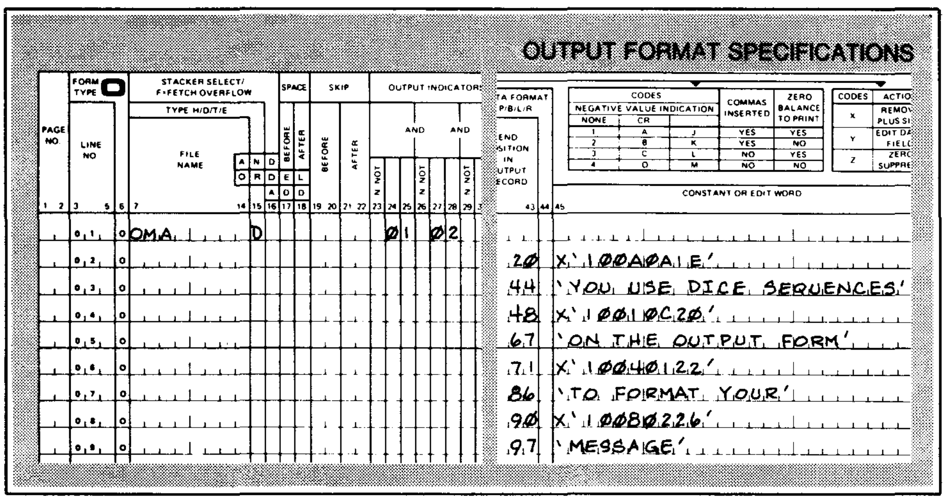


Figure A-3. Using DICE to Format an Output Message

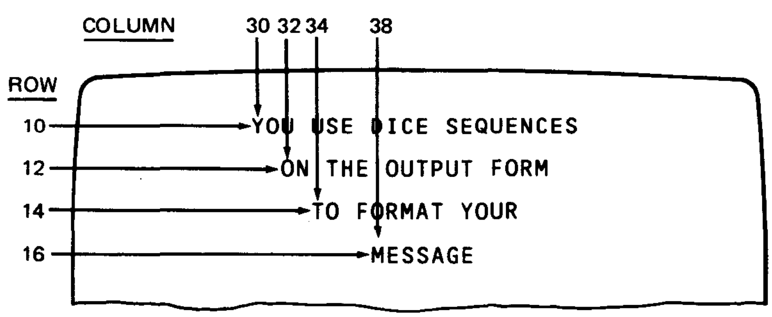


Figure A-4. How DICE Formatted Message in Figure A-3 Appears at the Screen

Here is a brief description of the DICE sequences used in Figure A-3.

Description of DICE sequences (Fig. A-3)

DICE Sequence	Description
100A0A1E	The select character 10 signals the start of the DICE sequence. The function code (0A) clears all protected and unprotected data from the terminal screen. The m field (0A) and the n field (1E) position the cursor to row 10, column 30. Notice that the end position for the DICE sequence is 20. Remember that the DICE sequence is a 4-character code and that the output message area header occupies positions 1-16.
10010C20	The select character 10 is always the same and signals the start of the DICE sequence. The function code (01) sets coordinates as directed by the m and n fields of the DICE sequence. The m field (0C) and the n field (20) position the cursor at row 12, column 32.
10040122	The select character is the same as before. The function code (04) moves the cursor to the beginning of the next line and then sets the coordinates as directed by the m and n fields. The m field (01) and the n field (22) position the cursor one row below where it presently is and in column 34.
10080226	The select character is again the same. The function code (08) returns the cursor to the beginning of the current line. The m field (02) and the n field (26) position the cursor two lines below the current line and in column 38.

A.7. USING FIELD CONTROL CHARACTERS

The FCC sequence format is:

Field control character format

```
            FCC SEQUENCE
TEXT | US | R | C | M | N | TEXT
```

Characters in the FCC sequence are defined as follows:

Defining FCC characters

US

Is the control character that signals the start of an FCC sequence. It corresponds to a hexadecimal 1F.

R

Is the number of the row in which the FCC is placed.

C

Is the number of the column in which the FCC is placed.

M

Is the hexadecimal value placed in the sequence to define bits 4, 5, 6, and 7 of the FCC. Table A-4 lists the hexadecimal codes you can use.

N

Is the hexadecimal value placed in the sequence to define bits 0, 1, 2, and 3 of the FCC. Table A-5 lists the hexadecimal codes you can use.

Table A-4. Hexadecimal Codes Used as M in the FCC Sequence (Part 1 of 2)

ASCII Character	Hexadecimal Code	Field Characteristics
0	30	Tab stop, normal intensity, changed field*
1	31	Tab stop, display off (no intensity), changed field*
2	32	Tab stop, low intensity, changed field*
3	33	Tab stop, blinking display, changed field*
4	34	Tab stop, normal intensity
5	35	Tab stop, display off (no intensity)
6	36	Tab stop, low intensity
7	37	Tab stop, blinking display
8	38	Not tab stop, normal intensity, changed field*

Table A-4. Hexadecimal Codes Used as M in the FCC Sequence (Part 2 of 2)

ASCII Character	Hexadecimal Code	Field Characteristics
9	39	Not tab stop, display off (no intensity), changed field*
:	3A	Not tab stop, low intensity, changed field*
;	3B	Not tab stop, blinking display, changed field*
<	3C	Not tab stop, normal intensity
=	3D	Not tab stop, display off (no intensity)
>	3E	Not tab stop, low intensity
?	3F	Not tab stop, blinking display

* Normally, when an FCC is generated by the host processor, the changed-field designator is cleared. However, the host processor can generate individual FCCs with the changed-field designator set; this capability may be used for selective transfer or transmission of fields which were not in fact changed by the terminal operator. By sending an ESC u code to the terminal in a text message, the host processor can clear the changed-field designators in all FCCs without regenerating each FCC and without altering the data within the fields.

Table A-5. Hexadecimal Codes Used as N in the FCC Sequence

ASCII Character	Hexadecimal Code	Field Characteristics
0	30	Any input allowed
1	31	Alpha only allowed
2	32	Numeric only allowed
3	33	Protected (no entries and no changes allowed)
4	34	Any input allowed, right-justified
5	35	Alpha only allowed, right-justified
6	36	Numeric only allowed, right-justified

For detailed information on using field control characters, consult the UTS 400 programmer reference, UP-8359 (current version).

Appendix B. Generating Edit Tables

B.1. PURPOSE

The edit table generator offers a convenient means for converting unformatted input received from terminal operators into fixed formats required by action programs and checking this input for types of data, value ranges, and presence of required fields.

Edit table generator output

The output of the edit table generator is written to the named record file (NAMEREC). From there it is loaded at the appropriate time by IMS. Each edit table is associated with a particular action at configuration time via the EDIT parameter in an ACTION section. The edit table utility can be run either before or after configuration, but the NAMEREC file must be previously initialized.

B.2. STATEMENT CONVENTIONS AND CODING RULES FOR EDIT TABLE GENERATOR INPUT

Edit table generator input parameters

Input to the edit table generator is in the form of keyword parameters that define the edit table, the fields you want edited, and the edit criteria for each field.

Statement conventions

In the format for edit table parameters, these conventions are observed:

A	Capital letters represent entries that must be coded exactly as shown.
a	Lowercase words are generic terms representing data that you must supply.
{ }	Entries within braces represent choices, of which you select one.

[]	Data within brackets represents optional entries.
▢	Shaded entries are default values.

To code input to the edit table generator, apply the following rules:

Sequence numbers

1. Input entries must contain sequence numbers in columns 77 through 80, in ascending order. The lowest permissible sequence number is 0001.

Where to code parameters

2. Parameters can be coded in any column between 1 and 76. Blanks are ignored and are permitted anywhere in the edit table definition.

Example:

```
1                                                         77   80
SEP=;ETAB=ETABTST;KEY=1;POS=Ø;MAN=Y;LEN=5;                    Ø1ØØ
KEY=2;FIL= ;JUS=L;LEN=15;MAN=Y;TYP=A;POS=5;                   Ø2ØØ
KEY=3;FIL= ;JUS=L;LEN=20;POS=20;TYP=M;;                       Ø3ØØ
```

Spanning lines

3. Specifications for an edit table and for each field can span more than one line. However, a keyword and its value must be contained on one line.

Example:

INCORRECT	CORRECT
SEP=;ETAB=ETABTST;KEY=1;POS= Ø1ØØ Ø;MAN=Y;LEN=5;MAN=Y;LEN=5;; Ø2ØØ KEYWORD AND VALUE NOT ON SAME LINE	SEP=;ETAB=ETABTST;KEY=1;POS=0; Ø1ØØ

New line

4. A new edit table specification must start on a new line. Each field need not begin on a new line.

Example:

INCORRECT

```
SEP=;ETAB=ETABTST;KEY=1;POS=0;  0100
MAN=Y;LEN=5;                    0200
KEY=2;FIL= ;JUS=L;LEN=15;MAN=Y; 0300
TYP=A;POS=5;;SEP=,ETAB=TABL1,   0400
KEY=1,LEN=20,POS=20,,           0500
```

NEW EDIT TABLE NOT SPECIFIED ON NEW LINE

CORRECT

```
SEP=;ETAB=ETABTST;KEY=1;POS=0;      0100
MAN=Y;LEN=5;KEY=2;FIL= ;JUS=L;      0200
LEN=15;MAN=Y;TYP=A;POS=5;;          0300
SEP=,ETAB=TABL1,KEY=1,LEN=20,       0400
POS=20,,                            0500
```

NEW FIELD NEED NOT START ON NEW LINE

Field separator character

Changing separator character

5. The field separator character specified by the SEP keyword parameter must be used as the field separator throughout the edit table specification, as well as the input message to be edited. Double separator characters indicate the end of the edit definition. A new edit table can establish a different separator character.

Example:

INCORRECT

```
SEP=;ETAB=ETABTST,KEY=1,POS=0;  0100
MAN=Y;LEN=5;                    0200
```

END OF EDIT DEFINITION NEEDS DOUBLE SEPARATOR

SAME FIELD SEPARATOR NOT USED THROUGHOUT EDIT TABLE DEFINITION

CORRECT

```
SEP=;ETAB=ETABTST;KEY=1;POS=0;      0100
MAN=Y;LEN=5;;                       0200
SEP=.ETAB=TABL4.KEY=1.POS=0.        0300
MAN=Y.LEN=5..                       0400
```

ESTABLISHES A NEW SEPARATOR CHARACTER

Order of parameters

6. The SEP, ETAB, and KEY parameters must be coded in the prescribed order; the remaining keyword parameters can be specified in any order. SEP and ETAB are coded once for each edit table. The remaining parameters are repeated for each field in the input message to be edited.

INCORRECT	CORRECT

`SEP=;POS=Ø;LEN=5;KEY=1;        Ø1ØØ`
`ETAB=ETABTST;;                 Ø2ØØ`
 	
ETAB AND KEY PARAMETERS DON'T IMMEDIATELY FOLLOW SEP
	`SEP=;ETAB=ETABTST;KEY=1;POS=Ø;   Ø1ØØ`
	`MAN=Y;LEN=5;;                   Ø2ØØ`

Numeric values

7. Numeric values are positive unless preceded by a minus sign (–). The plus sign (+) is not permitted in numeric values.

Example:

INCORRECT	CORRECT

`SEP=;ETAB=TABL1;KEY=1;LEN=5;     Ø1ØØ`
`POS=Ø;MAX=+2ØØØØØ;MIN=-1;;       Ø2ØØ`
 	
PLUS SIGN NOT ALLOWED
 	
NUMBER OF CHARACTERS EXCEEDS LENGTH GIVEN IN LEN PARAMETER
	`SEP=;ETAB=TABL1;KEY=1;LEN=5;     Ø1ØØ`
	`POS=Ø;MAX=2ØØØØ;MIN=-1;;        Ø2ØØ`

B.3. EDIT TABLE GENERATOR PARAMETERS

Input parameter format

The input parameters you give to the edit table generator should follow this format:

```
SEP=separator-character
ETAB=tablename
KEY={keyword }
    {position}
LEN=field-length
POS=starting-position
[FIL=fill-character]
[JUS={L}]
[    {R}]
[MAN={N}]
[    {Y}]
[MAX=maximum-value]
[MIN=minimum-value]
[TYP={A}]
[    {B}]
[    {M}]
[    {N}]
[    {P}]
```

Separator Character (SEP)

Separator character (SEP)

The separator parameter specifies the field separator character for both the edit table definition and the input message to be edited. It cannot be a blank, equal sign, or minus sign. This parameter is required, must be the first entry on the first line of the edit table definition, and can be specified only once per edit table.

Edit Table Name (ETAB)

Edit table name (ETAB)

The edit table name parameter names the edit table and must immediately follow the SEP parameter. This specification associates the edit table with an action at configuration, via the EDIT=tablename option in the ACTION section.

Key Field Identification (KEY)

Key field identification (KEY)

The key field parameter identifies the input message field for which edit criteria are specified in subsequent parameters and must be the first parameter specified for each field. The edit table generator associates all subsequent specifications with this field until it encounters another KEY parameter. Input fields can be positional or keyword. Positional fields precede keyword fields.

Positional fields

KEY=position specifies the relative position of the field as it appears in the input message. Positional fields must be defined in numeric order, starting with 1.

Keyword fields

KEY=keyword specifies a 1- to 3-character alphanumeric identification. The first character must be alphabetic for a keyword field in the input. The terminal operator enters keyword fields in the form *keyword=data*. For example, when you specify KEY=OLD, the terminal operator might enter OLD=57500 for this field. Once a keyword field is identified in the edit table definition, all subsequent fields must be defined as keyword fields.

Figure B-1 shows the correct coding for positional and keyword parameters to the edit table generator.

```
                                    POSITIONAL
                                        |
                SEP=;ETAB=TABL1;KEY=1;POS=0;MAN=Y;LEN=5;                 0100
POSITIONAL-->   KEY=2;FIL= ;JUS=L;LEN=15;MAN=Y;TYP=A;POS=5;             0200
KEYWORD-->     {KEY=NEW;FIL= ;JUS=L;LEN=10;POS=20;TYP=M;                0300
               {KEY=OLD;FIL= ;JUS=L;LEN=10;POS=30;TYP=M;;               0400
```

Figure B-1. Edit Table Parameter Description with Positional and Keyword Parameters

Edited Field Length (LEN)

Edited field length (LEN)

The length parameter specifies the length of the edited field and is a required parameter. You may specify a maximum of 255 characters for alphanumeric fields and four characters for binary fields. Ten characters is the maximum length for numeric fields unless you specify both MIN and MAX parameters for this field. If you identify a numeric field in the action program as packed decimal, you can specify up to 16 characters in the LEN parameter.

NOTES:

Field-length longer than screen width

1. *If the field-length is larger than the width of the screen on which data is to be entered, IMS removes the DICE code at the end of each line of terminal input and replaces it with a blank character. You must provide for these additional blank characters in the action program and include them in the field-length specified by the LEN parameter.*

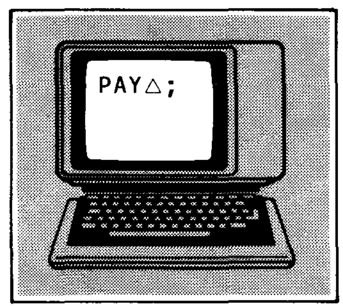
Binary and packed field lengths

2. *The length specified for binary (TYP=B) and packed (TYP=P) fields is the maximum length for the field in the input message, not the length of the field in your program. For example, if a field is defined as packed with a LEN=3, the largest number the terminal operator can key in is 999, even though 1000 may be represented in a packed field in 3 bytes.*

Transaction codes under five characters

3. *If the transaction code (the first field in the input message) is less than five characters, the terminal operator must key in a space before entering the separator character for the next field. You must include the space in the field-length specified by the LEN parameter.*

TRANSACTION CODE IS PAY
SO OPERATOR ENTERS



Transaction code field larger than five characters

The length of the first field can be greater than five characters, but only the first five characters are used in the transaction code. The LEN parameter should specify the actual length of the field.

Field Starting Position (POS)

Field starting position (POS)

The starting position parameter specifies the starting position of this field as it appears in the edited message and is a required parameter. The first field starts at 0.

Fill Character Identification (FIL)

Fill character identification (FIL)

The fill character parameter optionally specifies the fill character inserted in the edited field when the data the terminal operator enters as input is shorter than the field-length specified by the LEN parameter. The default fill character is 0. If you want to fill with spaces (X'40'), code either FIL= or FIL=△; i.e., you can include or omit a space before the separator character for the next field. Binary fields are always filled with binary zeros; therefore, this parameter is ignored if specified for a binary field.

Field Justification (JUS)

Field justification (JUS)

JUS=L left-justifies this field in the edited message. Binary and packed fields are always right-justified; therefore, this parameter is ignored if specified for binary or packed fields.

JUS=R right-justifies this field in the edited message and is the default assumed.

Mandatory Field (MAN)

Mandatory field (MAN)

MAN=N indicates that this field is not mandatory in the edited message for input to be acceptable.

MAN=Y indicates that this field is mandatory in the edited message.

Input Field Value Limitations (MAX and MIN)

Maximum value limitation (MAX)

The maximum value parameter specifies the maximum value allowed for the field in the input message. This parameter applies only to numeric fields. The highest value allowed is 2 to the thirty-first power minus 1 (2^{31}-1). The number of characters in this value must not exceed the length specified by the LEN parameter.

Minimum value limitation (MIN)

The minimum value parameter specifies the minimum value allowed for the field in the input message. This parameter applies only to numeric fields. The lowest value allowed is minus 2 to the thirty-first power minus 1 (-(2^{31}-1)). The number of characters in this value must not exceed the length specified by the LEN parameter.

Data Type (TYP)

Data type (TYP)

The type parameter describes the type of data to be contained in the edited field.

TYP=A specifies alphabetic data. A field defined to the editor as alphabetic is treated as an alphanumeric field.

TYP=B specifies binary data.

TYP=M specifies alphanumeric data and is the default value.

TYP=N specifies numeric data.

TYP=P specifies packed decimal data.

B.4. EXECUTING THE EDIT TABLE GENERATOR

Job control stream

Once you code input parameters describing the edit table format and the NAMEREC file is initialized, you can execute the ZH#EDT edit table generator using the control stream illustrated in Figure B-2.

```
// JOB ADDEDT,,A000
// DVC 20 // LFD PRNTR
// OPTION DUMP
// DVC 50 // VOL DS9999 // LBL NAMEREC,DS9999 // LFD NAMEREC
// EXEC ZH#EDT
/$
      input parameters
          .        .
          .        .
          .        .
/*
/&
// FIN
```

Figure B-2. Sample Execution of Edit Table Generator

When execution is successful

If the input definition is acceptable, the generated edit table is written to the NAMEREC file and the following message is issued;

```
tablename ADDED
```

Duplicate edit table name

If the edit table has the same name as a table already existing in the NAMEREC file, the new edit table replaces the existing table, and the following message is issued:

```
TABLE ADDED, DUPLICATE DELETED
```

Errors in edit table generation

If errors cause rejection of the edit table, the following message is issued:

```
tablename REJECTED
```

UPSI byte values

Another way to determine edit table errors is to look at the UPSI byte. The following UPSI byte values pertain to the edit table error status:

UPSI Byte Contents	Meaning
00	No errors
40	Warning. ZH#EDT continues processing edit table input parameters but no edit table is built.
80	Fatal error. Edit table processing terminates.

B.5. ERROR PROCESSING

Warning errors

When the edit table generator encounters a file I/O error or certain types of input errors, it terminates and prints a message in the output listing. The resulting value in the UPSI byte is 80. Most types of input errors do not cause termination. Processing and validation continues, but an error message is printed and the edit table is rejected. Input specifications for the edit table generator are not printed in the output listing. This type of error results in an UPSI byte value of 40.

Fatal errors

If an I/O error occurs while reading input to the edit table generator, the following message is issued, and the program terminates with an UPSI byte value of 80:

```
INPUT READ ERROR, SCAN TERMINATED
```

If an error occurs while opening, reading, or closing the named record file, the following error message is issued and the program terminates with an UPSI byte value of 80:

```
FILE ERROR, SCAN TERMINATED
```

Error message format

Errors in the input statements are reported in the following format:

```
nnnn cc error-message-text
```

where:

nnnn
: Is the sequence number in columns 77 through 80 of the card containing the error.

cc
: Is the column number of the beginning of the input text that is in error. This column number is suppressed if the error is detected during final validation of all parameters for a given field.

error-message-text
: Is the description of the error as listed in Table B-1.

Error message example

An example of an input statement error and the resultant error message follows:

Input:

```
SEP=,ETAB=EDIT1,KEY=1,LEN=5,POS=Ø,JUS=X,MAN=Y,        ØØØ2
```

Error message:

```
ØØØ2 39 JUSTIFICATION ILLEGAL
```

Table B-1 lists alphabetically the message texts inserted into the input statement error message. In each case, processing continues, unless otherwise indicated in the explanation column.

Table B-1. Edit Table Diagnostic Messages (Part 1 of 2)

Error Message	Explanation
B TYPE LENGTH GR THAN 4	Four characters (one full word) is maximum
CARDS NOT IN SEQUENCE	Scan terminated, run aborted*
DOUBLE SEPARATOR MISSING	Warning only; end-of-file encountered while searching for separator
DUPLICATE NAME	Duplicate name for nonpositional field
FIELD NOT ACCEPTED, KEYS STARTED	Positional parameters not allowed after nonpositionals started
FIELD NOT IN SEQUENCE	Positional parameters must be in sequence
FILLER MUST BE SINGLE CHARACTER	Self-explanatory
ILLEGAL FIELD TYPE	Only A, B, M, N, or P accepted
INVALID MAN SPECIFICATION	Only Y or N accepted
INVALID NAME	Name too long or contains invalid characters
INVALID SEPARATOR	Scan terminated, run aborted; = and - are not allowed as separators*
JUSTIFICATION ILLEGAL	Only R or L accepted
KEYWORD ETAB MISSING	Self-explanatory
KEYWORD INVALID	Self-explanatory
KEYWORD KEY= MISSING	Self-explanatory
KEYWORD SEP= MISSING	Scan terminated, run aborted*
LEN OR POS EXCEEDS MAX	Maximum length is 255; maximum position is 32,767

Table B–1. Edit Table Diagnostic Messages (Part 2 of 2)

Error Message	Explanation
LEN OR POS MISSING	Required parameters
LEN ZERO	Length must be at least 1
MAX OR MIN ABSOLUTE VALUE TOO LARGE	$2^{31}-1$ is largest absolute value allowed
N TYPE LENGTH GR THAN 10	Ten characters is maximum unless MAX and MIN both specified
NO DEFAULT FOR THIS FIELD	Parameter value must be specified
NO FIELDS DEFINED	Empty table not allowed
P TYPE LENGTH GR THAN 16	Sixteen characters maximum for packed decimal field
REPEATED FIELD	Parameter already specified
SEPARATOR CHARACTER MISSING	Self-explanatory
SEQUENCE NUMBER NOT NUMERIC	Scan terminated, run aborted*
= SIGN MUST FOLLOW KEYWORD	Self-explanatory
TOO MANY FIELDS	Scan terminated, run aborted; output buffer overflow*
xxx OVERLAPS yyy	Warning only; overlapping fields permitted

* These errors set the UPSI byte to 80; all other errors in this table result in an UPSI byte value of 40.

B.6. ENTERING INPUT MESSAGES FROM TERMINAL

When the terminal operator enters an input message for which you've generated an edit table, an IMS component called the expanded input editor processes it. The following considerations apply when entering input messages from the terminal:

Transaction code first

- When an input message contains a transaction code, the transaction code must always be the first field. If the transaction code is less than five characters, enter a space before keying in the separator character.

Beginning positional fields

Omitting positional fields

- Positional fields begin with the first nonblank character and extend to the next separator. Positional fields must appear in the same order as specified in the edit table definition. If you omit a positional field, enter an additional separator character in its position. A positional field entered as input may not contain an equal sign.

Keyword fields

- Keywords must be followed by an equal sign with no intervening blanks. Data starts immediately after the equal sign and extends to the next field separator.

Invalid plus sign

- Numeric values are positive unless preceded by a minus sign. The plus sign (+) is an invalid character.

Error messages screen placement

- Error messages are displayed on the first line of the display terminal; therefore, we recommend that you start input messages on the second line so that the input is not erased by an error message.

Continuing fields

- If you continue fields from one line to another, IMS removes the DICE code at the end of each line and replaces it with a blank character, which it sends to the action program as part of the data. Always enter on one line fields that don't exceed the width of the screen. If a field exceeds the screen width and must be continued from one line to another, avoid splitting a word between lines.

Ending input with positional parameters

- If the terminal input ends with a positional parameter (no keyword parameters are specified), enter a separator character at the end of the input message; otherwise, the input message could be partially deleted. A correct terminal entry is:

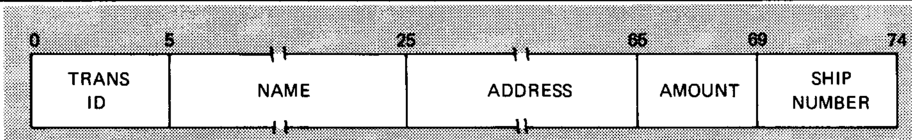
```
INFOR,BIOLOGY,CLASS2,MARY J. BLISS,
```

When terminal input ends with a keyword parameter, this is not necessary.

B.7. SAMPLE EDIT TABLE APPLICATION USING POSITIONAL AND KEYWORD PARAMETERS

Example edit table input

Figure B-3 and Table B-2 describe sample input to the edit table generator for an accounts receivable application and the format in which the edited input is delivered to the action program.




```
                                                                        77 80
SEP=,ETAB=EDIT1,KEY=1,LEN=5, POS=0, MAN=Y,                              0001
KEY=2,LEN=20, POS=5, FIL=,JUS=L,MAN=Y,                                  0002
KEY=3,LEN=40,POS=25,FIL= ,JUS=1,                                        0003
KEY=AMT,LEN=4, POS=65, MIN=1000, TYP=B,MAN=Y,FIL=0,JUS=R,               0004
KEY=SN, LEN=6,POS=69,FIL=,JUS=R,,                                       0005
```

Figure B-3. Sample Input to Edit Table Generator and Format of Input Delivered to Action Program

Table B-2. Description of Sample Input to Edit Table Generator (Part 1 of 2)

Line	Parameter	Explanation
1	SEP=,	The field separator is a comma for both the edit specification and input from the terminal.
	ETAB=EDIT1	The edit table name is EDIT1.
	KEY=1	The first field described is positional. It must be the first field in the input message.
	LEN=5	The edited field is five characters long.
	POS=0	In the edited message the field begins in position 0.
	MAN=Y	The field must be present for the message to be acceptable.
2	KEY=2	The field is positional. It must be the second field in the input message.
	LEN=20	The edited field is 20 characters long.
	POS=5	In the edited message the field begins in position 5.
	FIL=	The field is to be blank filled in the edited message.
	JUS=L	The field is to be left-justified in the edited message.
	MAN=Y	The field must be present for the message to be acceptable.

Table B-2. Description of Sample Input to Edit Table Generator (Part 2 of 2)

Line	Parameter	Explanation
3	KEY=3	The field is positional. It must be the third field in the input message.
	LEN=40	The edited field is 40 characters long.
	POS=25	In the edited message the field begins in position 25.
	FIL=	The field is to be blank filled in the edited message.
	JUS=L	The field is to be left-justified in the edited message.
4	KEY=AMT	The field is a keyword field. AMT=n must be specified in the input message.
	LEN=4	The edited field is four characters long.
	POS=65	In the edited message the field begins in position 65.
	MIN=1000	The minimum level allowed for the message to be acceptable is $10.00 (entered as 1000).
	TYP=B	In the edited message the field is to be converted to binary.
	MAN=Y	The field must be present for the message to be acceptable.
	FIL=0	The field is to be zero filled in the edit message. (This parameter could have been omitted.)
	JUS=R	The field is to be right-justified in the edited message. (This parameter could have been omitted.)
5	KEY=SN	The field is a keyword field.
	LEN=6	The edited field is six characters long.
	POS=69	In the edited message, the field begins in position 69.
	FIL=	The field is to be blank filled in the edited message.
	JUS=R	The field is to be right-justified in the edited message. (This parameter could have been omitted.)
	,,	End of edit definition.

The following examples show freeform input from the terminal and the resulting messages sent to the action program in accordance with the edit table specifications or, in case of error, the output message displayed at the terminal. Note that in the edited messages, the 4-character binary field specified for the AMT entry is represented by an underlined, 4-hexadecimal-digit field. Spaces between each delimiter and the first character of the next field are ignored.

Example 1:

Terminal input

```
PAYMT, JOHN D. SMITH,1112 BREEZE DR. PHILA.PA. 19160,
AMT=2500,SN=123456
```

Edited message received by action program

```
PAYMTJOHN△D.△SMITH△△△△△△△1112△BREEZE△DR.△PHILA.△PA.
△19160△△△△△△△△09C4123456
```

Example 2:

Terminal input

```
PAYMT,JOHN D. SMITH,,SN=123456,AMT=2500
```

Edited message received by action program

```
PAYMTJOHN△D.△SMITH△△△△△△△△△△△△△△△△△△△△△△△△△△△
△△△△△△△△△△△△△△△△△△△△09C4123456
```

Explanation

The address field wasn't specified as mandatory in the edit table input and is omitted here; an additional comma is coded in its position. The AMT and SN fields are keyword fields and need not be entered in the order defined in the edit table input.

Example 3:

Terminal input

```
PAYMT ,JOHN D. SMITH,1112 BREEZE DR. PHILA. PA. 19160,
AMT=2500,SN=123456
```

Output message

```
ILLEGAL INPUT
```

Explanation

The transaction code field is longer than the LEN specification.

Example 4:

Terminal input

```
PAYMT,JOHN D. SMITH,1112 BREEZE DR. PHILA. PA.19160,
AMT=700,SN=123456
```

Output message

```
AMT IS BELOW MIN
```

Explanation

Edit table specifies AMT must be at least 1000.

Example 5:

Terminal input

```
PAYMT, JOHN D. SMITH,1112 BREEZE DR. PHILA. PA. 19160,
SN=123456
```

Output message

```
AMT MISSING
```

Explanation

AMT was specified as mandatory.

Appendix C. Summary of IMS Error Codes

This appendix presents all the error codes returned by IMS as a result of function calls made by action programs.

Completion status codes

Table C-1 lists and defines the values returned to the status-code field of the program information block. This value indicates the completion status of the function request.

Defined record management status codes

Table C-2 lists and describes values returned to the detailed-status-code field with status code 1 (invalid key) when errors occur on a defined file.

Invalid request status codes

Table C-3 lists and describes values returned to the detailed-status-code field when the status code returned is 3 (invalid request).

Internal message control status codes

Table C-4 lists and describes values returned to the detailed-status-code field when the status code returned is 6 (internal message control error).

Screen formatting status codes

Table C-5 lists and describes values returned to the detailed-status-code field when the status code is 7 (screen format services error).

Table C-1. Values Returned to the Status-Code Field after Function Requests

Status	Value
Successful	0
Invalid key or record number	1
End of file or unallocated optional file	2
Invalid request	3
I/O error	4
Violation of data definition	5
Internal message control error	6
Screen format error	7

Table C-2. Detailed Status Codes for Defined Record Management Errors (Invalid Key – Status Code 1)

Detailed Status Code (Hexadecimal)	Description	Meaning
E1	No identifier supplied	Insert an IDENTIFIER statement in the item definition.
E2	Identifier too long	Identifier may be from 1 to 30 alphanumeric characters long.
E4	Identifier out of range	Value entered at terminal isn't in range of VALUE clause specified.

INVALID REQUEST ERROR CODES

Table C-3. Detailed Status Codes for Invalid Requests (Part 1 of 2)

Detailed Status Code (Hexadecimal)	Description	Meaning
01	Incorrect number of parameters	Please submit a software user report (SUR) or contact your Sperry Univac representative.
02	Function code out of legal range	Please submit a SUR or contact your Sperry Univac representative.
03	Incorrect parameter value	Please submit a SUR or contact your Sperry Univac representative.
04	Shared record not in use by this transaction	This code does not apply to user action program requests.
05	File not defined	A file named in a request to IMS was not defined at configuration.
06	File not open	A file named in a request to IMS was closed by the master terminal (ZZCLS) or by data management as the result of an unrecoverable error.
07	Function invalid for type of file	The function specified in a request to IMS is not valid for the type of file named. For example, a SETLL for a nonindexed file.
08	Record(s) not locked	Please submit a SUR or contact your Sperry Univac representative.
09	Function sequence for an update operation is invalid	Input did not precede output.
0A	Illegal function requested	The requested function is not consistent with the DTF or RIB parameters in the configuration.
0B	File not assigned to this action	Same as code 05
0C	Required module not included in configuration	A request was made to IMS that required a module not included in the IMS load module at configuration.
0D	Capacity exceeded on ADD operation	A request was made to add a record to a MIRAM or ISAM file, but there wasn't sufficient space.
0E	Insufficient space in main storage	User must allocate more main storage space.
0F	Update not permitted in configuration	A request was made to perform some update function, but this update was disallowed at configuration.

Table C-3. Detailed Status Codes for Invalid Requests (Part 2 of 2)

Detailed Status Code (Hexadecimal)	Description	Meaning
10	Update suppressed for files	The requested update is not permitted because of an I/O error in the audit file, a file used for online recovery.
11	Trace file down	File recovery is not operational; no updates are allowed. Only file displays are allowed.
12	Record locked by another transaction (single-thread only)	Under single-thread, an action tried to add or update a record, but the record was already locked by another transaction.

Table C-4. Detailed Status Codes for Internal Message Control Errors (Status Code 6)

Detailed Status Code (Hexadecimal)	Description	Meaning
02	Destination terminal busy, on hold, or down	Output-for-input queueing was requested and: 1. destination terminal is in interactive mode; 2. destination terminal has an input message on queue; 3. ZZHLD or ZZDWN command was entered for destination terminal; 4. destination terminal is marked physically down to ICAM; or 5. IMS cannot allocate main storage buffer (multithread only; INBUFSIZ specification inadequate.
03	Destination terminal physically or logically down; message queued	SEND function was issued for message switching. Message is queued at destination terminal and is transmitted when terminal becomes operational.
04	Invalid specification in output message header	Invalid destination terminal-id or auxiliary-device-id; or, aux-function field contains X'C3', X'F3', or X'F7' (not valid with SEND function).
05	No ICAM network buffer available	Insufficient buffer space was allocated in ICAM network definition.
06	Disk error, or recoverable system error on output to console	Output error occurred on attempt to write a message to disk; error was passed to IMS by ICAM. On output to console, this error occurs when console is physically or logically down.
07	Invalid length specification	In delayed internal succession or output-for-input queueing, output message length was larger than the input buffer pool.

Table C-5. Detailed Status Codes for Screen Formatting Errors (Status Code 7)

Detailed Status Code (Hexadecimal)	Description	Meaning
00	Validation error; all error fields in variable data area are replaced by hexadecimal F's.	Variable data fields don't match specifications at screen format generation.
01	Format area not large enough; IMS places actual length required for format in the text-length field	OUTSIZE=n specification in ACTION section of configuration isn't large enough.
02	Variable data area not large enough	WORKSIZE=n specification in ACTION section of configuration isn't large enough.
03	Screen format can't be displayed because no terminal slots are available	SFS=n specification in OPTIONS section of configuration isn't large enough.
04	Variable fields specified for input-only format	Screen format was designed for input only.
05	Format dimensions are greater than screen dimensions	Screen format is larger than source terminal screen.
06	Fatal error; I/O error reading format file	Get DM error message from console; refer to system messages programmer/reference, UP-8076 (current version).
0A	Data description in action program doesn't match screen format generation	Screen format was incorrectly generated.
0B	SFS failed	System error; take dump and write software user report (SUR). Can also occur if format contains protected fields and terminal doesn't have protect feature or isn't in protect mode.
10	SFS failed during input conversion	Inadequate main storage in system; or format contains protected fields and terminal doesn't have protect feature or isn't in protect mode.
11	IMS error	Take IMS job dump and submit SUR.
12	Screen format can't be transmitted because this is a program-initiated DDP transaction.	Action program processing DDP transaction attempted to send screen format to initiating action program.

Appendix D. Action Program Coding Restrictions

General coding restrictions

Table D-1 is a summary of coding restrictions for all the RPG II coding forms.

Coding restrictions for random files

Table D-2 summarizes allowable entries on the file description form for random access, MIRAM, ISAM, DAM, and defined files.

Coding restrictions for sequential files

Table D-3 summarizes allowable entries on the file description form for sequential MIRAM and SAM files.

Table D-1. IMS Restrictions for RPG II Coding

Form Type	Specifications Form	Column	Description
H	Control card specifications	8 9 15 41	Error analysis dump Operator control Generate debug code First page forms alignment
F	File description specifications	15 16 20-23 32	File type (C and D not allowed) Table and array file designation (T) ① Block length (Same as record length) ② File organization: ADDROUT (D) ① Record address (blank) ① Additional I/O areas ② Sequential MIRAM and SAM tape/disk input files ISAM and indexed MIRAM output files
		40-46	Device: CTLRDR READER CRP PUNCH CONSOLE PRINTER WORKSTATION REMOTE FILES
		53 54-59 60-65 66 67 68-69 70 71-72	Labels ② Name of label exit option ② Size of ISAM index entry ② Unordered load Cylinder overflow space percentage ② Number of extents ② Tape rewind ② File conditioners (U1-U8)
E	Extension specifications ①	9-10	Chaining (C1-C9) tables or arrays
I	Input format specifications	19-20 42	Spread card feature (TR) Stacker select
C	Calculation specifications	28-32	Display operation (DSPLY)
O	Output format specifications	16	Stacker select
T	Telecommunications specifications	-	-

NOTES:

① Used only with nonindexed MIRAM and DAM files.

② Ignored by RPG II compiler; must be specified in IMS configuration.

Table D-2. Allowable File Description Specifications for ISAM, MIRAM, DAM, and Defined Files

File description form entries for ISAM, MIRAM, DAM, and defined files

Column Title and Number	Specification
Form Type (Column 6)	F
File Name (Column 7-13)	User-defined name
File Type (Column 15)	I, U, or O
File Desgination (Column 16)	S, R, C, D, or P
Format (Column 19)	F
Record Length (Column 24-27)	User's record size
Mode of Processing (Column 28)	L, R, or blank
Key Field Length (Column 29-30)	01-99 ①
Record Address Type (Column 31)	A or P ① R ② Blank ③
File Organization (Column 32)	I ① D ② Blank ③
Key Field Start Position (Column 35-38)	0001-9999 ①
Device (Column 40-46)	Must be disk device
File Addition (Column 66)	Blank or A

NOTES:

① Indexed files

② Nonindexed (relative) files

③ Sequential processing

ALLOWABLE FILE DESCRIPTION SPECIFICATIONS

File description form entries for sequential MIRAM and SAM files

Table D-3. Allowable File Description Specifications for Sequential MIRAM and SAM Output Files

Column Title and Number	Specification
Form Type (Column 6)	F
File Name (Column 7-13)	User-defined name
File Type (Column 15)	O
Format (Column 19)	F
Record Length (Column 24-27)	User's record size
Overflow Indicator (Column 33-34)	May be specified for line counter files
Line Counter (Column 39)	Blank or L
Device (Column 40-46)	Must be disk or tape device

Index

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(Document Title)

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

Cut along line.

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

FOLD

CUT

SPERRY UNIVAC

USER COMMENT SHEET

Your comments concerning this document will be welcomed by Sperry Univac for use in improving subsequent editions.

Please note: This form is not intended to be used as an order blank.

(Document Title)

(Document No.) *(Revision No.)* *(Update No.)*

Comments:

From:

(Name of User)

(Business Address)

Fold on dotted lines, and mail. (No postage stamp is necessary if mailed in the U.S.A.)
Thank you for your cooperation

Cut along line.

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 21 BLUE BELL, PA.

POSTAGE WILL BE PAID BY ADDRESSEE

SPERRY UNIVAC

ATTN.: SYSTEMS PUBLICATIONS

P.O. BOX 500
BLUE BELL, PENNSYLVANIA 19424

FOLD

CUT