Check Processing Control System
International MVS/ESA

Installation Guide

Release 1

GC31-2942-01

Check Processing Control System
International MVS/ESA

Installation Guide

Release 1

GC31-2942-01

Note!

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

| Second Edition (September 1998)

This edition applies to Release 1 Modification 0 of the IBM Check Processing Control System International MVS/ESA licensed program (Program No. 5799-FKT).

Information in this manual is subject to change from time to time. Before using this publication in connection with the operation of IBM systems, consult your IBM representative to be sure you have the latest edition and any Technical Newsletters.

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

A form for reader’s comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Department 58G, MG34/204, 8501 IBM Drive, Charlotte, NC 28262-8563, U.S.A. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Part 2. Sample Problems

Figures

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service can be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectable rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact: IBM Corporation, Department MG39/201, 8501 IBM Drive, Charlotte, NC 28262-8563, U.S.A. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, New York 10504-1785, U.S.A.

Trademarks

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

ACF/VTAM
COBOL/370
High Performance Transaction System
ImagePlus
MVS/ESA
RACF
System/370
3090
AD/Cycle
ES/9000
IBM
MVS/DFP
MVS/SP
SAA
Systems Application Architecture

Any terms denoted by two asterisks (**) are trademarks or service marks of other companies.

About This Book

This book tells you how to install the IBM Check Processing Control System International MVS/ESA (CPCS-I) and how to run the sample problems. The sample problem exercises are included to help you verify correct operation after you install the software. You can also use this book and the *CPCS-I Customization Guide* when you are modifying and tuning the operation of CPCS-I. To ensure you have the most current information, read the *Program Directory* you receive with the product tape before you begin installation of the software.

Who Should Read This Book?

This book is for people who are responsible for installing and testing the CPCS-I. The sample problems in Part 2 are helpful for training new users of CPCS-I.

How Is This Book Organized?

This book contains the following sections:

Part 1. Installation

- Chapter 1, “Preparing to Install CPCS-I,” describes installation tape contents and the prerequisite hardware and software you need to install and operate CPCS-I.
- Chapter 2, “Loading CPCS-I on Your System,” describes how to use the System Modification Program Extended (SMP/E) procedures to allocate your CPCS-I data sets and load the source code. It also includes special information to consider before using the SMP/E procedures.
- Chapter 3, “Installing CPCS-I,” describes the tasks you must perform to install CPCS-I software on your system, including the allocation of required data sets and the generation of program modules.

Part 2. Sample Problems

- Chapter 4, “Overview of the Sample Problems,” contains a summary of the sample problems and the general setup instructions.
- Chapter 5, “Sample Problem 1: Concurrent Processing” provides instructions for using concurrent processing functions, including multiple-subset with concurrent data capture, distribution, kill/remittance, and reporting. Additional information is included to test the tracers-in-kill-pockets option.
- Chapter 6, “Sample Problem 2: Enhanced Reject Processing” provides instructions for CPCS-I processing of alternate reject pockets and consolidated reject processing for various pockets to meet different transit deadlines.
- Chapter 7, “Sample Problem 3: Unqualified Data” provides instructions for CPCS-I processing of entries with blank amount fields. The online reject re-entry task is used to simulate a key entry system for inscribing the correct amounts.
- Chapter 8, “Sample Problem 4: Divider Resynchronization” provides instructions for using divider documents to resynchronize data during a subsequent pass.

- Chapter 9, “Sample Problem 5: Base Functions with Expanded MDS” provides instructions for CPCS-I processing with an expanded MDS record.
- Chapter 10, “Sample Problem 6: Basic Functions” provides instructions for performing the most common types of CPCS-I processing. Separate instructions are included to run the sample problem with high speed reject reentry (HSRR) processing.

Appendixes

- Appendix A, “Recommended Enhanced System Manager Workflows for DCVR,” describes the Enhanced System Manager workflows recommended for DCVR.

This book also contains a glossary, a bibliography, and an index.

Related Publications

The following publications contain information that relates to Check Processing Control System International MVS/ESA (CPCS-I). For an additional list of relevant publications, see the “Bibliography.”

- *IBM Check Processing Control System International MVS/ESA: General Information*, GC31-2944
  Short Title: *CPCS-I General Information*

  This publication gives a general introduction to CPCS-I. It describes various features and advantages of CPCS-I and the hardware and software requirements for operating this system. It also discusses CPCS-I support of the IBM 3890 Document Processor and the IBM 3890/XP Series document processors, along with some of the features of these processors.
- *IBM Check Processing Control System International MVS/ESA: Installation Guide*, GC31-2942
  Short Title: *CPCS-I Installation Guide*

  This guide describes the steps necessary for using the IBM System Modification Program Extended (SMP/E) procedures to install CPCS-I software. It also provides installation procedures for generating CPCS-I modules and creating operational data sets. It provides data for sample problems to test and verify operations after CPCS-I installation.
- *IBM Check Processing Control System International MVS/ESA: Terminal Operations Guide*, SC31-2946
  Short Title: *CPCS-I Terminal Operations Guide*

  This guide explains how to perform CPCS-I tasks and is written for the CPCS-I operators. Included in this guide are terminal operations for the MICR restart procedures and sample reports.
- *IBM Check Processing Control System International MVS/ESA: Programming Guide*, SC31-2948
  Short Title: *CPCS-I Programming Guide*

  This guide contains guidelines for CPCS-I programmers. It includes information about application-program processing, problem analysis and documentation procedures.
- *IBM Check Processing Control System International MVS/ESA:*

Programming Reference, GC31-3997
Short Title: *CPCS-I Programming Reference*

This publication gives a structured view of the CPCS-I interfaces, specifically application programming, Assembler macros, subroutines, and some control block information.

- *IBM Check Processing Control System International MVS/ESA: Customization Guide*, SC31-2943
  Short Title: *CPCS-I Customization Guide*

  This guide provides customization information for CPCS-I programmers. It also includes system programming information, and generation and installation procedures.

- *IBM Check Processing Control System International MVS/ESA: Messages and Codes*, SC31-3981
  Short Title: *CPCS-I Messages and Codes*

  This book describes console and supervisor messages, as well as program return and exit codes.

- *IBM Check Processing Control System International MVS/ESA: Propagation of Adjustments*, SC31-3994
  Short Title: *CPCS-I Propagation of Adjustments Guide*

  This guide contains the guidelines for the CPCS-I personnel who use the Propagation of Adjustments (PRAD) feature. It includes functional descriptions and information about terminal operations, programming, and application output.

- *IBM Check Processing Control System International MVS/ESA: Master Index*, SC31-3980
  Short Title: *CPCS-I Master Index*

  This reference combines the index entries for all the publications in the CPCS-I library.

Part 1. Installation

Chapter 1. Preparing to Install CPCS-I

Review the information in this chapter and in Chapter 2, “Loading CPCS-I on Your System,” before beginning the installation process. See the *CPCS-I Program Directory* for additional information on the IBM System Modification Program Extended (SMP/E). For an overview of CPCS-I operations, concepts, and terminology, see the *CPCS-I General Information* manual.

SMP/E loads on your system the files delivered on the product tape. After SMP/E loads the files in the SMP/E libraries, you can begin the process of allocating data sets and generating program modules for CPCS-I. When the CPCS-I installation process is complete, you can run the sample problems described in Part 2 of this book.

Note: Before you install CPCS-I, contact your IBM representative about CPCS-I preventive service planning (PSP) information that might have been added after the printing of this book.

If you are using the IBM ImagePlus* High Performance Transaction System*, you must identify its applications to CPCS-I in the DKNBLDL table. You can use the Resource Access Control Facility (RACF*) to manage security requirements for High Performance Transaction System transactions. For information about adding application tasks to DKNBLDL, see the *CPCS-I Customization Guide*. For more information about installation requirements for the High Performance Transaction System, see the *ImagePlus High Performance Transaction System Planning Guide* and the *ImagePlus High Performance Transaction System Central Installation Guide*.

Important!

The installation procedures described in this book create data sets with CPCSI.V01R01 as the default high-level qualifier. This is the qualifier used throughout this book.

You can choose another qualifier based on the data set naming conventions for your system.

Customizing CPCS-I

CPCS-I is delivered from IBM, with the Logging subsystem inactive, and with a standard mass data set (MDS) record length. The installation procedures described in Chapter 3, “Installing CPCS-I,” create a basic system that is capable of running the sample problems. Additional customization is required to meet the needs of your particular financial institution.

For details about customizing and tailoring CPCS-I, see the *CPCS-I Customization Guide*.

* Trademark of IBM

For installation procedures for the Enhanced System Manager feature, see the *CPCS Enhanced System Manager User's Guide*.

Chapter 2. Loading CPCS-I on Your System

This chapter describes how to allocate the data sets used by the System Modification Program Extended (SMP/E) and how to use SMP/E to load the CPCS-I into those data sets.

The following sections describe special considerations both for first-time installation of CPCS-I and for installing this release with an existing system. They also describe the SMP/E functions (Receive, Apply, and Accept) that you use to copy the CPCS-I files from the product tape to your system.

Using SMP/E

CPCS-I is packaged for installation with SMP/E. The sample SMP/E JCL included in the *CPCS-I Program Directory* and on the CPCS-I product tape is only for demonstration purposes. You are responsible for creating the correct procedures to complete the SMP/E installation of CPCS-I on your system.

Installing with an Existing CPCS-I System

If you have never used SMP/E to install CPCS-I software and you have an existing CPCS-I system, follow the instructions in the *CPCS-I Program Directory* for copying sample SMP/E JCL on the CPCS-I product tape. The JCL provides examples of SMP/E procedures that you can use to establish the required libraries, catalogues, and data sets. You will need to use these procedures, with some changes, to prepare your system for the SMP/E installation process.

Warning: SMP/E installation is not designed to delete the previous release of CPCS-I. You must decide either to maintain CPCS-I and a previous release of CPCS or to replace the existing release.

Coexisting with a Previous Release of CPCS

Allocate and initialize a separate set of SMP/E libraries before you install CPCS-I to coexist with a previous release of CPCS. Otherwise, existing macro and module names may cause SMP/E installation to fail. You must also allocate a separate set of CPCS-I libraries before you start the SMP/E installation process. You can copy examples of JCL that you can use to allocate and initialize the SMP/E data sets from the CPCS-I product tape. See the *CPCS-I Program Directory* for additional instructions.

Deleting a Previous Release of CPCS

Figure 2-1 on page 2-2 is an example of an SMP/E job for deleting a previous release of CPCS that was installed using SMP/E.

```
//JOB1 JOB ....
//RECEIVE EXEC SMPEPROC
//SMPPTFIN DD *
++FUNCTION (USERMODFMID).          (SEE NOTE BELOW)
++VER (Z038) DELETE(HMRXXXX).      (SEE NOTE BELOW)
//SMPCNTL DD *
    SET BOUNDARY(GLOBAL).
    RECEIVE.       (NOTE: CONDITION CODE 4 IS NORMAL)
/*

//JOB2 JOB ....
//APPLY EXEC SMPEPROC
//SMPCNTL  DD *
    SET BOUNDARY(TARGET1).
   APPLY S(USERMODFMID) C(ALL) DIS(WRITE).
/*                     OREMOVE 'DIS(WRITE)'
//JOB3 JOB ....
//ACCEPT EXEC SMPEPROC
//SMPCNTL  DD *
    SET BOUNDARY(DLIB1).
   ACCEPT S(USERMODFMID) C(ALL) DIS(WRITE).
/*                      OREMOVE 'DIS(WRITE)'
NOTE:  USERMODFMID = SOME UNIQUE USER DEFINED FMID
       HMRXXXX     = FMID OF EARLIER RELEASE BEING DELETED
```

Figure 2-1. Sample JCL to Delete a Previous Release of CPCS

Installing CPCS-I for the First Time

If you are installing CPCS-I for the first time, allocate and initialize a set of SMP/E libraries to be used by SMP/E. You can use an existing set of SMP/E libraries, or you can allocate a set of libraries specifically for this product. You can copy examples of JCL that you can use to allocate and initialize these libraries from the CPCS-I product tape. See the *CPCS-I Program Directory* for additional instructions.

Installing User Modifications Using SMP/E

For an explanation of how to convert the installation of user modifications, see the *System Modification Program Extended (SMP/E) Program Packaging Guide*.

Copying the CPCS-I Files from the Product Tape

You use the following SMP/E functions to copy the CPCS-I files from the product tape to your system:

- Receive
- Apply
- Accept

For a sample of the JCL for each of these functions, see the *CPCS-I Program Directory* that you received with the CPCS-I product tape.

Chapter 3. Installing CPCS-I

The SMP/E procedures described on page 2-2 tell you how to unload the files from the tape and place them on your system. The installation procedures described in this chapter tell you how to allocate the required CPCS-I data sets and assemble, or compile, and link-edit the program modules.

The steps for installing CPCS-I are:

1. Allocate the CPCS-I data sets.
2. Allocate the duplex data sets.
3. Allocate the spool data sets.
4. Define the VSAM data sets.
5. Copy the endpoint tables to the DKNEP data set.
6. Copy the sort-pattern definitions to the DKNSPDEF data set.
7. Copy the system profile definitions to the GENSYSTP data set.
8. Run the MDX macro to create SDKNMAC1.
9. Generate COBOL/370* options.
10. Assemble and link-edit the called assembler source.
11. Modify the Feature Generation Options (FEATOPTS) copy module.
12. Assemble and link-edit the base assembler source.
13. Compile and link-edit the COBOL source.
14. Assemble and link-edit the DKNMICR assembler source.
15. Generate the system help file.
16. Assemble and link-edit the DKNMICR module.
17. Assemble and link-edit the DKNMTASK module.
18. Assemble and Link-edit the DKNBLDL module.
19. Assemble and Link-edit the DKNDSAT module.
20. Assemble, compile, and link-edit the user-exits and SCI programs.
21. Allocate the DEFT VSAM data sets.
22. Compile and link-edit the DEFT source code.
23. Load the Application Profile data set
24. Compile and link-edit DKNCSBU, DKNXSRV, and DKNCSBUM.
25. Compile and link-edit OLMS source code.
26. Compile and link-edit FSCN source code.
27. Compile, assemble, and link-edit the HCDM source code.
28. Compile and link-edit the UK English language source code.
29. Create the ABA and BCF files.
30. Initialize the item-sequence-number data set.
31. Create the system message file.
32. Assemble and link-edit the sample concurrent sort module.
33. Add the CPCS-I load library to the APF list.
34. Add DKNMTASK to the program property table.
35. Install Enhanced System Manager.

Note: The installation procedures described in this chapter create a system that you can use to run the sample problems described in Part 2, “Sample Problems.” Additional changes and customization are required to create a system that meets the needs of your financial institution. For information about modifying CPCS-I operation, see the *CPCS-I Customization Guide*.

* Trademark of IBM

Step 1: Allocate the CPCS-I Data Sets

In this step, you allocate the data sets needed to run CPCS-I. The JCL is in the member ALLOCDS of CPCSI.V01R01.SDKNSAM1. Before you tailor the JCL, see the description of MDEF macro parameters BLKSIZE, BLKSEG, SEGDEV, and MAXDA in the *CPCS-I Customization Guide*.

This JCL allocates all the required CPCS-I data sets, using the following default assumptions:

- They do not already exist.
- They are all put on volume *xxxxxx*.
  (Use a volume label defined for your installation.)
- They are all assigned a high-level qualifier of CPCSI.V01R01.

Important!

You can install CPCS-I with the default qualifier of CPCSI.V01R01, or you can change the qualifier to meet the requirements of your installation. If you change the qualifier, you must review all the JCL members referenced to ensure that you are using the correct qualifier before you run the jobs.

The following files are allocated:

- Checkpoint data set
- Cycle data set (primary)
- Endpoint file
- Application/temporary work files
- Tracer-group data set (primary)
- Mass data set and index
- Scroll file
- Application-profile data set
- Sort-pattern definition file
- Application-posting files (ICRE, MCRE, ECYC)

Step 2: Allocate the Duplex Data Sets

If you are not using data-set duplexing, continue with “Step 4: Define the VSAM Data Sets” on page 3-3.

In this step, you allocate the CPCS-I duplex data sets. The JCL is in the member ALLOCDUP of CPCSI.V01R01.SDKNSAM1. These data sets are necessary for recovery of critical operating information if a disk failure occurs.

The JCL allocates the CPCS-I duplex data sets, using the following assumptions:

- They do not already exist.
- They are all on volume YYYYYY.

  Warning: Do not use the same volume for the duplex data sets and for standard CPCS-I data sets allocated by ALLOCDS. If they are on the same volume and a DASD failure occurs, you will lose both data sets.
- They are all assigned a high-level qualifier of CPCSI.V01R01. (You can change CPCSI.V01R01 to a qualifier selected for your system.)

The following duplex files are allocated:

- Cycle data set
- Kill-bundle file
- Microfilm file
- Tracer-group data set

Step 3: Allocate the Spool Data Sets

In this step, you allocate the CPCS-I spool data sets. The JCL is in the member ALLOCSPL of CPCSI.V01R01.SDKNSAM1. These data sets are necessary for the recovery of report information if a disk failure occurs.

The JCL allocates the CPCS-I spool data sets, using the following assumptions:

- They do not already exist.
- They are all on volume *xxxxxx*.

Step 4: Define the VSAM Data Sets

In this step, you define the VSAM data sets for CPCS-I. The JCL is in the member ALLOCVS of CPCSI.V01R01.SDKNSAM1. This JCL allocates the VSAM data sets, using the following assumptions:

- They already exist.
- They are placed on volume *xxxxxx*.
- They are assigned a high-level qualifier of CPCSI.V01R01.

The following files are allocated:

- Divider resynchronization file
- Electronic mail file
- Electronic tracer-group assignment file
- Microfilm file
- Kill bundle file

The data-set names allocated are:

DIVM.FILE
DIVM.DATA
DIVM.INDX
MAIL.FILE
MAIL.DATA
MAIL.INDX
TGSS.FILE
TGSS.DATA
TGSS.INDX
DKNMF.FILE
DKNMF.DATA
DKNMF.INDX
DKNKB.FILE

Step 5: Copy the Endpoint Tables to the DKNEP Data Set

In this step, you set up the endpoint tables for CPCS-I. The JCL is in the member GENEPT of CPCSI.V01R01.SDKNSAM1.

The EPT001 data set is copied from CPCSI.V01R01.SDKNSAM2 to CPCSI.V01R01.DKNEP.

Step 6: Copy the Sort-Pattern Definitions to the DKNSPDEF Data Set

In this step, you set up the sort-pattern definitions used during CPCS-I operation. The JCL is in the member GENSPDEF of CPCSI.V01R01.SDKNSAM1.

Step 7: Copy the system profile definitions to the GENSYSTP data set.

In this step, you set up the system profile definitions used during CPCS-I operation. The JCL is in the member GENSYSTP of CPCSI.V01R01.SDKNSAM1.

Step 8: Run the MDX Macro to Create SDKNMAC1

This step is required. It allocates CPCSI.V01R01.SDKNMAC1 and updates it with members that have the indicated field sizes. The JCL is in member CMPL0010 of CPCSI.V01R01.SDKNSAM1.

This JCL creates the standard expansion configuration. If you are expanding the mass data set from the standard 50-byte size, include MDX operands for the fields you want to change. For details about MDX operands, see the *CPCS-I Customization Guide*.

Step 9: Generate COBOL/370 options

Note: You should customize the ASMPROC that you are using to assemble and link-edit.

In this step, assemble and link-edit CEEUOPT from CPCSI.V01R01.SDKNSAM2. The JCL is CMPL0020 in CPCSI.V01R01.SDKNSAM1. This JCL creates the CPCS-I COBOL/370 runtime parameters.

Step 10: Assemble and Link-Edit the Called Assembler Source

In this step, you assemble and link-edit the assembler source modules you will use during the link-edit in “Step 12: Assemble and Link-Edit the Base Assembler Source.” The JCL is in the member CMPL0030 of CPCSI.V01R01.SDKNSAM1. Load modules are placed in CPCSI.V01R01.SDKNMOD1.

This step must run before any other assembly or compile procedures.

For user-exit interface descriptions and programming requirements, see the *CPCS-I Customization Guide*. No changes to the user exits are required to run the sample problems.

Step 11: Modify the CPCS-I FEATOPTS Copy Module

This step is required if you need to change the default high-level qualifier of CPCSI.V01.R01 on the CPCS-I data-set names in the DSAT table. It enables you to modify the entries in the CPCS-I FEATOPTS copy module, which is located in CPCSI.V01R01.SDKNSRC1.

Step 12: Assemble and Link-Edit the Base Assembler Source

In this step, you assemble and link-edit assembler source modules used for this release of CPCS-I. The JCL is in the member CMPL0040 of CPCSI.V01R01.SDKNSAM1. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

Note: If RACF is not installed, DKNSECRR will not successfully assemble.

Step 13: Compile and Link-Edit the COBOL Source

In this step, you compile and link-edit all COBOL source modules for this release of CPCS-I. The JCL is in the member CMPL003 of CPCSI.V01R01.SDKNSAM1. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

You can change the COBOL compiler name in the //STEPLIB DD statement of DKNJPROC to match the name used on your system.

Step 14: Assemble and Link-Edit the DKNMICR Assembler Source

Note: You should customize the ASMPROC to assemble Assembler programs. You should customize COBPROC to compile the COBOL programs used in the following steps.

In this step, you assemble and link-edit assembler source modules needed to run DKNMICR in this release of CPCS-I. The JCL is in the member CMPL0060 of CPCSI.V01R01.SDKNSAM1. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

For user-exit interface descriptions and programming requirements, see the *CPCS-I Customization Guide*. No changes to the user exits are required to run the sample problems.

Step 15: Generate the System Help Facility

In this step, you generate the screen help data. The JCL is in the member DKNGHELP in CPCSI.V01R01.SDKNSAM1.

This job allocates the HELP data set and loads it with help screen data from selected members in the CPCSI.V01R01.SDKNSAM2 data set.

For US installations, the members are DKNINEHU and DKNISMH1; for UK installations, DKNIHENG and DKNISMH2. Be sure to uncomment the member pairs appropriate to your installation before you submit DKNGHELP.

Step 16: Assemble and Link-Edit the DKNMICR Module

In this step, you assemble MICRDATA to generate the DKNMICR module for this release of CPCS-I. The JCL is in the member GENMICR of CPCSI.V01R01.SDKNSAM1. The load module is placed in CPCSI.V01R01.SDKNMOD1.

Note: Return code 0004 is valid for step MGEN.

For more information about parameters for the CPCSRDR macro, see the *CPCS-I Customization Guide*.

Step 17: Assemble and Link-Edit the DKNMTASK Module

In this step, you assemble the MDEFDATA to build the DKNMTASK module for this release of CPCS-I. The JCL is in member GENMTASK of CPCSI.V01R01.SDKNSAM1. The load module is placed in CPCSI.V01R01.SDKNMOD1.

For a description of the MDEF macro parameters, see the *CPCS-I Customization Guide*.

Step 18: Assemble and Link-Edit the DKNBLDL Module

In this step, you assemble and link-edit the DKNBLDL module used for this release of CPCS-I.

The JCL is in the member CMPLBLDL of CPCSI.V01R01.SDKNSAM1. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

Step 19: Assemble and Link-Edit the DKNDSAT Module

In this step, you assemble and link-edit the DKNDSAT module used for this release of CPCS-I.

The JCL is in the member CMPLDSAT of CPCSI.V01R01.SDKNSAM1. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

Step 20: Assemble, Compile, and Link-Edit the User-Exits and SCI Programs

In this step, you assemble and link-edit the user exits and SCI programs used in the sample problems.

The JCL for user exits is in the member CMPLUXIT in CPCSI.V01R01.SDKNSAM1. The JCL for SCI programs is in DKNCMPLU for US codelines, DKNCMPLG for UK codelines. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

For user-exit interface descriptions and programming requirements, see the *CPCS-I Customization Guide*. No changes to the user exits are required to run the sample problems.

Step 21: Allocate the DEFT VSAM Data Sets

In this step, you allocate the DEFT processing file. The JCL is in the member ALLOCDFT in CPCSI.V01R01.SDKNSAM1. This job allocates all the required DEFT data sets, using the following default assumptions:

- The DEFT VSAM data sets already exist.
- All the DEFT data sets are placed on Volume *nnnnnn*. Use a volume label defined for your installation.
- All the DEFT data sets are assigned a high-level qualifier of CPCSI.V01R01.

Important!

You can install DEFT with the default qualifier of CPCSI.V01R01, or you can change the qualifier to meet the requirements of your installation. If you change the qualifier, you must review all the JCL members referenced to ensure you are using the correct qualifier before you run the jobs. Also, this job deletes any previous DEFT data sets.

The following data sets are allocated:

- DEFTD.FILE
- DEFTD.DATA
- DEFTD.INDX

Step 22: Compile and Link-Edit the DEFT Source Code

In this step, you compile and link-edit all the DEFT shipped modules. The JCL is in member DKNJDEFT of CPCSI.V01R01.SDKNSAM1. User exits are also compiled; their source is taken from CPCSI.V01R01.SDKNSAM2.

Step 23: Load the Application Profile Data Set

In this step, you load the Application Profile data set, DKNAPPL. The JCL member is DKNGAPPL.

Step 24: Compile and Link-Edit DKNCSBU, DKNXSRV, and DKNCSBUM

In this step, you copy DKNCSBU, DKNXSRV, DKNCSBUM, DKNCSBUP, and DKNCSBUZ into the source library. The source library for these parts is CPCSI.V01R01.SDKNSRC1. Compile and link-edit DKNCSBU, DKNXSRV, and DKNCSBUM with attributes ASM and LNK=RENT. The compile and link-edit JCL is in member CMPLCSBU of CPCSI.V01R01.SDKNSAM1.

Step 25: Compile and Link-Edit OLMS Source Code

In this step, you compile and link-edit all OLMS modules. The JCL is in member CMPLOLMS of CPCSI.V01R01.SDKNSAM1. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

Step 26: Compile and Link-Edit FSCN Source Code

In this step, you compile and link-edit all FSCN modules. The JCL is in member CMPLFSCN of CPCSI.V01R01.SDKNSAM1. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

Step 27: Compile, Assemble, and Link-Edit the HCDM Source Code

You compile and link-edit all HCDM shipped modules. The JCL is in the member CMPLHCDM of CPCSI.V01R01.SDKNSAM1. The load modules are placed in CPCSI.V01R01.SDKNMOD1.

Step 28: Compile and Link-Edit the UK English Language Source Code

You compile and link-edit the country-specific source code in CPCSI.V01R01.SDKNSAM1. If you want U.S. English, use DKNCMPLU, if you want U.K. English, use DKNCMPLG.

Step 29: Create ABA and BCF Files

In this step, you allocate and load the ABA file (DKNAB) and the Bank Control File (DKNBCF). The JCL for the ABA file is in the member GENAB of CPCSI.V01R01.SDKNSAM1. The JCL for the BCF file is in the member GENBCF of CPCSI.V01R01.SDKNSAM1.

The GENAB step runs the program DKNLOAD (compiled in a previous step) and uses as input the member ABAFILE of CPCSI.V01R01.SDKNSAM2. The result is a direct access data set (DKNAB) that contains destination information organized by endpoint ID. GENAB assumes that the DKNAB file already exists.

The GENBCF step runs the program DKNBCFLD (compiled in a previous step) and uses as input the member BCFDATA of CPCSI.V01R01.SDKNSAM2. The result is a partitioned data set (DKNBCF) that contains bank processing information.

Step 30: Initialize the Item-Sequence-Number Data Set

In this step, you use DKNIISEQ to format and allocate the item-sequence-number VSAM data set, CPCSI.V01R01.ISEQ.FILE. This data set is used to ensure unique sequence numbers during CPCS-I processing. The JCL is in the member GENISEQ. GENISEQ assumes that the ISEQ file already exists in CPCSI.V01R01.SDKNSAM1.

For more information about the item-sequence-number data set, see the *CPCS-I Customization Guide*. Information about using item-sequence numbers in CPCS-I processing is in the *CPCS-I Customization Guide*.

Step 31: Create the System Message File

In this step, you allocate and load the system message file. The JCL is in member DKNGSMSG of CPCSI.V01R01.SDKNSAM1. The CPCS-I system message VSAM file is first allocated with this JCL and is then loaded with the system message data from selected members of the CPCSI.V01R01.SDKNSAM2 data set.

For US installations, the members are DKNIMENU and DKNSMMG1; for UK installations, DKNIMENG and DKNSMMG2. Be sure to uncomment the member pairs appropriate to your installation before you submit DKNGSMSG. The resulting file is named CPCSI.V01R01.DKNSMSG.FILE.

Step 32: Assemble and Link-Edit the Sample Concurrent Sort Module

In this step, you assemble and link-edit the sample concurrent sort module. The input is in the member GENSORT of CPCSI.V01R01.SDKNSAM2. The resulting load module lets CPCS-I tasks that require a sort to run concurrently. The MDEFDATA parameter “MAXSORT” specifies the number of concurrent sorts (in MDEFDATA default, this value is 3).

Step 33: Add the CPCS-I Load Library to the APF List

Note: This step and “Step 34: Add DKNMTASK to the Program Property Table” are performed by your system programmer.

This step authorizes DKNMTASK. Add an entry for the load library data set, CPCSI.V01R01.SDKNMOD1, to member IEAAPF00 of SYS1.PARMLIB. An error can occur for modules in SYS1.CSSLIB if the location of the MVS Callable Service Library is unknown. CPCS-I runs from authorized libraries and the MVS Callable Service Library must reside in the link list or in the link pact area (LPA). IPL the system to make this change effective.

For more information about APF list changes, see the *MVS/ESA Initialization and Tuning Reference*.

Step 34: Add DKNMTASK to the Program Property Table

When CPCS-I is running, you must ensure that DKNMTASK remains in real storage. To make DKNMTASK a storage-resident routine, and make the CPCS-I region nonswappable, do the following:

1. Add DKNMTASK to the Program Property Table (PPT), specifying X'20' in the program property attribute.
2. Add an entry in member SCHED00 of SYS1.PARMLIB for DKNMTASK, as shown in Figure 3-1 on page 3-10.

Note: This does not prevent the region from being paged.

```
COL1|  COL8|
    |      |
    PPT    PROGNAME(DKNMTASK)
           NOSWAP
```

Figure 3-1. Example of SCHED00 Entry for DKNMTASK

For more information about using nonswappable tasks, see the *MVS/ESA Initialization and Tuning Reference*.

You can now run the sample problems to verify the CPCS-I installation.

If you are installing the logging subsystem or are expanding the MDS record length, follow the procedures described in the *CPCS-I Customization Guide*.

Step 35: Install Enhanced System Manager

See the installation steps for Enhanced System Manager in the *CPCS Enhanced System Manager User's Guide*.

Part 2. Sample Problems

Chapter 4. Overview of the Sample Problems

The CPCS-I sample problems run on a base system that you create when you follow the installation steps that are described in Chapter 2 and Chapter 3. The sample problems help you verify that you have correctly installed CPCS-I, show several functions available with CPCS-I, and provide new users with a way to learn the processing flow of CPCS-I.

The 3890/XP MVS Support Program supplies a simulated document processor facility that lets you run both the prime and subsequent pass entries without a physical document processor. The simulator supports high-speed reject re-entry (HSRR) as well as online reject re-entry (OLRR) to let you correct data errors. It also supports both standard and XF (expanded format) sorts.

Each sample problem is a complete entity and can be used separately. The sample problems include a description of the following:

- The macro definition necessary to generate DKNMTASK.
- The macro definitions necessary to generate DKNMICR.
- Any required SCI or OLRR modules.
- The sort-pattern definitions necessary to show the functions specific to the problem.
- Data for the specific problem.

  To demonstrate the reporting and balancing capabilities of CPCS-I, the data provided contains deliberate errors. You correct these errors as part of the sample problem exercise.

  Note: You can use the data supplied with the simulator as is, or you can encode it for use with a physical document processor. If you encode it, do not include the periods (.) shown in the records.
- Operating instructions to guide you through the sample exercises.

  For more help with CPCS-I commands, see the *CPCS-I Terminal Operations Guide*.
- Output listings from the steps run in the problems.

  These listings let you verify the operation of the system after installation.

Sample Problem 1: Concurrent Processing

This problem shows the capabilities of concurrent processing: concurrent data capture, distribution, kill/remittance, and reporting. The data consists of three subsets. Some kill/remittance bundles are contained within a subset and others span multiple subsets.

The problem includes:

1. Prime-pass entry with multiple rehandle pockets
2. Automatic start-up of the distribution and concurrent kill/remittance functions
3. Online reject re-entry for multiple subsets
4. Merge for multiple repair strings using DKNMRGE
5. Subsequent-pass processing for multiple subsets
6. Report generation for multiple subsets
7. Backup of MDS data and completion of the cycle.

The prime-pass entry shows the processing of multiple subsets. The problem does not pause between subsets.

The OLRR exercise shows the potential for multiple operators performing simultaneous repair. The DKNMRGE operations exhibit the variety of merge functions available with CPCS-I. The subsequent-pass processing shows concurrent distribution and kill/remittance functions.

Sample Problem 2: Enhanced Reject Processing

This problem shows the advantages of using multiple reject pockets. It simulates a production environment from a banker's perspective, showing how you can selectively process various pockets in a timely manner to meet different transit deadlines. The problem includes the following functions:

1. Prime-pass entry with multiple alternate reject pockets and kill pockets that contain bad items
2. Production of kill/remittance bundles and cash letters/docket control vouchers (DCVs) for a first deadline
3. Subsequent pass of the rehandle data, including distribution, kill/remittance, and cash letter/DCV production for a second deadline
4. Repair of the transit rehandle pocket, including distribution, kill/remittance, and cash letter/DCV production for a third deadline
5. Repair and distribution of the on-us reject data
6. Creation of a consolidated reject string from the bad data in good pockets and repair of these items
7. Repair of the error items in the system reject pocket
8. Merge of all corrected data to a final M-string
9. Distribution of the final M-string to produce revised kill/remittance lists and cash letters/DCVs
10. Backup of MDS data and completion of the cycle.

Sample Problem 3: Unqualified Data

This problem processes entries that have the amount field left blank. It uses OLRR to simulate a key entry system for inscribing the amount. The problem shows the 3892/XP Document Processor power encoding function, under the following conditions:

- None of the sample items contain an amount field.
- The SCI program distributes the items to several alternate reject pockets.
- You use OLRR to repair this data by adding the amounts.
- You simulate power encoding of the repaired items during the subsequent pass.

Sample Problem 4: Divider Resynchronization

This problem shows how you can use divider documents to resynchronize the data during a subsequent pass. Divider resynchronization helps to greatly reduce the effects of *hand swapping*.

Note: This problem cannot be run using the simulator. You must use a physical document processor and encode the data on actual test documents.

Sample Problem 5: Base Functions with Expanded MDS Data

This problem is similar to Sample Problem 1, with the following exceptions:

- The expanded process control field (field 6) contains data to show its potential use in return item processing.
- The MDS record has two new fields, 9 and 12.
- The amount field (field 1) contains 16 digits.
- The SCI program sorts amounts over one million dollars to a special high-dollar pocket.

The problem includes the following functions:

- Prime pass and distribution
- OLRR error correction and merge of corrected data
- Production of kill/remittance bundles and cash letters/DCVs
- Subsequent-pass processing
- End-of-cycle processing.

Sample Problem 6: Basic Functions

This problem shows the basic functions of CPCS-I. It includes data and instructions for performing the following runs:

1. Proof-of-deposit (POD)
2. High speed reject re-entry (HSRR)
3. XF sort POD
4. XF sort HSRR.

The proof-of-deposit (POD) exercise shows the basic flow of work through the CPCS-I system:

- Prime-pass entry with rehandle pocket
- Item distribution reports
- Online reject re-entry
- Balancing reports
- Subsequent-pass entry
- Data generation for back-end application processing.

The HSRR case shows HSRR in the context of the POD sample case.

The XF sort POD and XF sort HSRR exercises show the expanded capture feature. The sort pattern provided does not capture any extended data but does show the MICR XF screens and the MICR XF messages logged in the scroll data set.

Chapter 5. Sample Problem 1: Concurrent Processing

This sample problem shows the capabilities of concurrent processing. You can also use it to test I-string subsets without concurrent processing.

The data for this sample problem also supports multiple-subset processing with concurrent data capture, distribution, kill, and reporting. This includes kill bundles that are contained completely within a subset and kill bundles that span one or more subsets.

Preparing to Run the Sample Problem

The test material contains:

- MDEF source for MTASK generation
- GENMICR source for concurrent processing
- SPDEF definition for concurrent processing options
- Data containing additional tracer slips to support subset processing
- SCI and OLRR changes for concurrent processing.

DKNMTASK Generation Options

The member SMDEFUS in CPCSI.V01R01.SDKNSAM2 contains the options for DKNMTASK generation.

The Enhanced System Manager workflows must be generated to make DKNDIST automatically start KILL/RMIT for each I-string.

DKNMICR Generation Options

The Enhanced System Manager workflows must be generated to automatically start kill.

```
         TITLE 'CPCS - MICR TASK GENERATION'
*********************************************************************
*        READER SORTER 1
*
RDR1     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER01,MFILM=YES,MODEL=A,   -
               ENDORSE=YES,INF=YES,ATTACH=SIM,CODELINE=US
*
*********************************************************************
*        READER SORTER 2
*
RDR2     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER02,MODEL=B,             -
               MFILM=YES,ATTACH=SIM,CODELINE=US
*
*********************************************************************
*        READER SORTER 3
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,            -
               MFILM=YES,ATTACH=SIM,PEND=YES,IMAGE=YES,CODELINE=US
*
*********************************************************************
*        READER SORTER 4
*
RDR4     CPCSRDR TYPE=3892-6,DDRDRIN=SORTER04,MODEL=XP,            -
               MFILM=YES,ATTACH=SIM,POWER=YES,PEND=YES,CODELINE=US
*
*********************************************************************
*        READER SORTER 5
*
RDR5     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER05,MODEL=XP,            -
               MFILM=YES,ATTACH=CHANNEL,PEND=YES,CODELINE=US
*
*********************************************************************
*        READER SORTER 6
*
RDR6     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER06,MODEL=XP,          -
               MFILM=YES,ATTACH=SIM,PEND=YES,CODELINE=US
*
*********************************************************************
*        READER SORTER 7
* -
RDR7     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER07,MODEL=XP,
               MFILM=YES,ATTACH=SIM,PEND=YES,IMAGE=YES,CODELINE=US
*
```

```
*********************************************************************
*        READER SORTER 8
*
RDR8     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER08,MFILM=YES,MODEL=A,  -
               ENDORSE=YES,INF=YES,ATTACH=SIM,CODELINE=US
*
*********************************************************************
*        READER SORTER 9
*
RDR9     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER09,MFILM=YES,MODEL=A,   -
               ENDORSE=YES,INF=YES,ATTACH=SIM,CODELINE=US
```

DKNSPDEF Options

The CPCSI.V01R01.SDKNSAM2 member SPTYP002 provides the sort definition for this sample problem. The O-record contains all the options for subset processing.

Position	Option
6	Subsets
	ƀ = No subsets
	S = Subset processing
13	Tracers in kill pockets
	Y = Tracers placed before each subset in a kill pocket
	N = No tracers in kill pockets

15 Subset Pause Option

Y = MICR will pause between subsets

N = MICR will not pause between subsets

Note: If you use SPTYP002, you will not find tracers in kill pockets, and there is no pause between subsets. If you want to test either option, you must change the sort pattern definition.

```
**********************************************************************
* This definition of SPTYP002 is for Sample Problem 1.               *
*                                                                    *
* The "P" (Pass Description) statement defines the following:        *
* Column 2 - 1000000 - Pass pocket history                           *
* Column 9 - SAMP001Z - Stacker select module name                   *
* Column 25 - 02 - Number of rehandle pockets created this pass      *
* Column 27 - 1 - Credits precede Debits                             *
* Column 39 - 2 - Item number position 2 on Prime Pass               *
* Column 40 - 0030 - Kill bundle minimum count                       *
* Column 56 - 2 - Endorsing Position 2                               *
*                                                                    *
* The "R" (Rehandle Description) statement specifies that three      *
* tracers will be put in each rehandle pocket 3 (Col. 6) and         *
* pocket 6 (Col. 12).                                                *
*                                                                    *
* The "O" (Run Option) statement defines the following:              *
* Column 6 - S - Subsets processing is selected                      *
* Column 13 - N - No tracers in kill pockets                         *
* Column 15 - N - No pause between subsets                           *
*                                                                    *
* The first two "K" statements define the kill pockets and their    *
* associated endpoints.  The first of these defines pocket 1 as      *
* a kill pocket with an endpoint of 88888888.                        *
* This endpoint designation is for an on-us kill pocket with no      *
* kill list.  Pockets 2, 4 &  5 are similarly defined.               *
* Pockets 9, 10, 11 & 12 have endpoint codes that are in the         *
* endpoint table.                                                    *
*                                                                    *
* The second "P" statement defines the subsequent pass for the       *
* pocket 3 data.  Note the pass/pocket history has 2 in the pass     *
* number and 03 for the pass 1 pocket.                               *
*                                                                    *
* The "K" statements for the subsequent pass perform the same        *
* functions as those in the prime pass.                              *
*                                                                    *
* The third "P" statement defines the subsequent pass for the        *
* pocket 6 data.  Note the pass/pocket history has 2 in the pass     *
* number and 06 for the pass 1 pocket.                               *
*                                                                    *
* The "K" statements for the subsequent pass perform the same        *
* functions as those in the prime pass.                              *
**********************************************************************
P1000000SAMP001Z        011           20008            2
R    03    03
O    S      N N
K018888888802888888880488888888058888888809000000000510000000010
K110000001511200000025
P2030000SAMP1SPZ          0222223      0030
K07010000050301000010040100001505010000200601000025
P2060000SAMP1SPZ          0222223      0030
K07010000300301000035040100004005010000450601000050
```

Pocket Selection for Sample Problem 1

Figure 5-1, Figure 5-2, and Figure 5-3 describe the contents of each of the pockets for this sample problem. In the column labeled *Routing/Transit*, the letter **n** can be replaced by any digit.

Figure 5-1. Sample Problem 1: Pocket Selection Criteria - Prime Pass

Pocket	Type	Routing/ Transit	Other Field	Description
RJ	Reject	Not used	Not used	System reject pocket
01	---	N/A	N/A	Not used
02	Kill	11nn-nnnn	PC=AAAA33	On-us credit
03	R/H	11nn-nnnn	PC=AAAAAA	Rehandle
04	Kill	11nn-nnnn	Return Item	On-us return item
05	---	N/A	N/A	Not used
06	R/H	N/A 07nn-nnnn 08nn-nnnn 09nn-nnnn 16nn-nnnn 17nn-nnnn 18nn-nnnn 19nn-nnnn 20nn-nnnn 21nn-nnnn 22nn-nnnn	N/A	Second deadline transits
07	---	N/A	N/A	Not used
08	---	N/A	N/A	Not used
09	Kill	12nn-nnnn	N/A	First deadline transit EP=09000000
10	Kill	13nn-nnnn	N/A	First deadline transit EP=10000000
11	Kill	14nn-nnnn	N/A	First deadline transit EP=11000000
12	Kill	15nn-nnnn	N/A	First deadline transit EP=11000000

Figure 5-2 describes the pocket selection criteria that you use when you run the subsequent pass for rehandle pocket 03.

Figure 5-2. Sample Problem 1: Pocket Selection Criteria - Rehandle Pocket 03

Pocket	Type	Routing/ Transit	Other Field	Description
RJ	Reject	N/A	N/A	System reject pocket
01	---	N/A	N/A	Not used
02	---	N/A	N/A	Not used
03	Kill	18nn-nnnn	N/A	Second deadline transit EP=02100002
04	Kill	19nn-nnnn	N/A	Second deadline transit EP=02130235
05	Kill	20nn-nnnn	N/A	Second deadline transit EP=02100030
06	Kill	21nn-nnnn	N/A	Second deadline transit EP=02100008
07	Kill	All Other R/T codes	N/A	Second deadline transit EP=02000000
08	---	N/A	N/A	Not used
09	---	N/A	N/A	Not used
10	---	N/A	N/A	Not used
11	---	N/A	N/A	Not used
12	---	N/A	N/A	Not used

Figure 5-3 describes the pocket selection criteria that you use when you run the subsequent pass for rehandle pocket 06.

Figure 5-3. Sample Problem 1: Pocket Selection Criteria - Rehandle Pocket 06

Pocket	Type	Routing/ Transit	Other Field	Description
RJ	Reject	N/A	N/A	System reject pocket
01	---	N/A	N/A	Not used
02	---	N/A	N/A	Not used
03	Kill	18nn-nnnn	N/A	Second deadline transit EP=02100002
04	Kill	19nn-nnnn	N/A	Second deadline transit EP=02130235
05	Kill	20nn-nnnn	N/A	Second deadline transit EP=02100030
06	Kill	21nn-nnnn	N/A	Second deadline transit EP=02100008
07	Kill	All other R/T codes	N/A	Second deadline transit EP=02000000
08	---	N/A	N/A	Not used
09	---	N/A	N/A	Not used
10	---	N/A	N/A	Not used
11	---	N/A	N/A	Not used
12	---	N/A	N/A	Not used

Data for Sample Problem 1

The data is supplied in member STEST02 of CPCSI.V01R01.SDKNSAM2. The data provides three sets of tracers with subset data to let you test subsets and associated functions. These tracers (0020, 0021, and 0022) create three subsets when run through MICR with the supplied sort definition (SPTYP002).

Note: If you use the simulator for this sample problem, you must include the fields represented by periods as shown in the records. If you use a physical document processor, do not encode the fields represented by the periods.

Operating Instructions for Sample Problem 1

If you use the simulated sorter, you must also change the run JCL to use the correct sort-pattern definitions and data. If you use a physical document processor, inscribe the data provided on the appropriate documents.

1. Start CPCS-I by submitting the job in member DKNJCPCS from the CPCSI.V01R01.SDKNSAM1 data set.
2. From the CPCS-I logo screen, log on to the relevant terminals with the command

   ```
   SGON xxx
   ```

   where *xxx* is the CPCS-I operator ID.
3. Designate one terminal as the system supervisor terminal with the command

   ```
   SUPV ON,SYST
   ```

   or by pressing **PF1**.
4. Activate cycle 8 by running the DKNCYCL program. The command is

   ```
   CYCL 8,A
   ```

   Let the cycle and endorse dates default to the current date. Follow the display instructions by pressing **ENTER** until the `READY` prompt appears. CPCS-I sends a message to the supervisor terminal indicating the change of a cycle's status.
5. Start MICR on any nonsupervisor terminal by entering the command

   ```
   MICR
   ```

   Open the selected sorter (1 through 5; sorter 2 if simulated) by responding with

   ```
   O n
   ```

   on the MICR Options screen, where **O** is for Open and *n* is the sorter number.
6. After opening the sorter, begin the MICR run by responding with a **B** on the Options screen. Specify cycle 8, sort pattern 002 (non-XF sort with subsets), and entry number 0020.

   Note: When using the simulator, the JCL must point to the Sample Problem 1 test data for its input file. This data is in member STEST02 in CPCSI.V01R01.SDKNSAM2.
7. If you have selected the Subset Pause Option in the DKNSPDEF O-statement, DKNMICR prompts you to enter **GO** to continue after each subset. Otherwise, the DKNMICR completes the entry without intervention.

8. End the prime-pass run when you see the `Intervention Required` message on the MICR Status display after the last subset is processed. Do this by entering **E** at the prompt. Enter another **E** to verify the end command.
9. After ending the MICR run, proceed to the MICR Options screen by pressing **ENTER**. Use the CLOSE command to end the MICR session. DKNDIST displays messages on the supervisor terminal, indicating the end of the distribution task for each subset.

   Note: Enhanced System Manager workflows must be generated to automatically start DKNKILL.
10. Run the OLRR task to correct the rejects from the prime pass. Start OLRR by specifying the entry number *nnnn*, where *nnnn* is the tracer group number for the selected subset and the operator ID is 01. For example, the command for subset 2 (which is tracer 0021) would be

    ```
    OLRR 0021,01
    ```

    For this sample problem, subsets are defined by the following tracer groups:

Subset	Tracer Group
001	0020
002	0021
003	0022

    Three documents in the first subset are encoded with digit errors. Correct the errors as follows:

    a. R/T error: `12*1-2222` should be `1221-2222`
    b. Amt error: `00000*0002` should be `0000010002`
    c. R/T error: `1110-006*` should be `1110-0062`.

    Two documents in the second subset are encoded with digit errors. Correct these errors as follows:

    a. R/T error: `111*-0062` should be `1110-0062`
    b. Amt error: `00000*0002` should be `0000000002`.

    Four documents in the third subset are encoded with digit errors. Correct the errors as follows:

    a. R/T error: `12*1-2222` should be `1221-2222`
    b. Amt error: `00111*0000` should be `0011110000`
    c. R/T error: `1110-006*` should be `1110-0062`
    d. Amt error: `00002222*0` should be `0000222220`.

    Note: After you correct the rejects for each subset, the OLRR task ends automatically. You must restart the OLRR task for each subset.
11. After you correct all rejects for all subsets, start the DKNMRGE task for the corrected entry with the command

    ```
    MRGE 1,0020
    ```

    DKNMRGE option 1 merges the subset strings for an entry and creates an M-string for each subset.

    If Enhanced System Manager is active, it starts the DKNMRGE task.
12. Run the final merge to combine the subset M-strings. Do this by entering

    ```
    MRGE 7,0020
    ```

This step combines the M-strings for each of the subsets into a final M-string and deletes the reject strings for the prime pass.

Note: DKNMRGE creates the M-strings by combining the I-string (from the MICR prime pass) with the R-string (produced by OLRR). DKNMRGE then automatically starts the DKNPLST task for the corrected entry.

DKNMRGE displays all messages on the terminal from which it was started. DKNPLST displays messages on the supervisor terminal.

13. The Exception Entry Master List report identifies one block as being out of balance. The block that is out of balance is the block slip with serial number 49. This out-of-balance condition is caused by the intentional entry of an incorrect amount during OLRR for this block.

14. At this point, you need to run the Cash Letter Summary (DKNCLSM) task to create the cash letters. Run the DKNCLSM task for the current cycle (cycle 8) with the command

    ```
    CLSM ALL,8
    ```

    Select the Endpoint Table option and use table EPT001 when prompted for the table name.

15. Start a MICR subsequent pass for those documents sorted to the rehandle pockets (pockets 03 and 06) on the prime pass for this entry. After starting a MICR session, open the sorter and proceed to the Begin screen to specify information for the subsequent pass. Enter

    ```
    ,,tttt-sss
    ```

    where *tttt* is the first 4 digits of the account number for the first tracer in the subset's rehandle pocket, and *sss* is the last 3 digits of the account number for this tracer document.

    You do not need to reenter the cycle number and sort pattern, because MICR uses the information you provide to get this data from the system. The two commas preceding the tracer number replace the cycle and sort pattern codes.

    Note: The operands are positional. You must either enter a value or replace the expected value by a comma.

    If you are running the sample problem on a physical document processor, use the entry and sequence numbers printed on the first tracer in the pocket. For simulator entries, the first tracer in pocket 03 (the first rehandle pocket) is sequence number 10 for tracer group 0020. The entry for performing the subsequent pass on this pocket is `0020-010`.

    To run the subsequent pass for pocket 06 (the second rehandle pocket), you must identify the first tracer put in that pocket. Because the sort pattern definition specified three tracers for each rehandle pocket (pockets 03 and 06), the first tracer in pocket 06 should be sequence number 13. The entry for the pocket 06 subsequent pass, therefore, is `0020-013` because sequence numbers 011 and 012 are in pocket 03.

    You run the subsequent pass for the other two subsets the same way you did for the first subset. The tracer group for the second subset is 0021. The first tracer sequence number (found in pocket 03) should be 010. Therefore, the entry for this pass should be `0021-010`. For pocket 06 of the second subset, it should be `0021-013`. The third subset entries follow the same pattern.

Note: As their entry number in the MDS, all subset I-string names for the entry retain the original entry number (0020 in this case).

The system responds by displaying the subsequent-pass data on the MICR Begin screen. Acknowledge the correct data by pressing **ENTER**.

Alternate Method: If you run the sample problem on a physical sorter, you can use another option — processing all the subsets in a rehandle pocket in one subsequent pass. This approach reduces the number of subsequent passes and results in fewer short kill bundles.

To combine your rehandle pocket subsets into one subsequent pass, enter the tracer and sequence number of the first tracer in the desired rehandle pocket and append a subset number of 000 to it. For example, enter `,,0020-010-000` to process all subsets in pocket 03 in one subsequent pass.

16. End the subsequent-pass run when you see the `Intervention Required` message on the MICR Status screen.
17. After you end the MICR run, proceed to the MICR Options screen by pressing **ENTER** and end the MICR session with the CLOSE command.
18. Enhanced System Manager or a computer operator starts DKNDIST automatically. Also, because this is a subsequent pass, DKNDIST starts the subsequent-pass Master List task (DKNSLST). This list identifies any out-of-balance condition between the prime-pass and subsequent-pass runs on an item-count and dollar-amount basis. There should be no exception conditions for this pass.
19. As in the prime pass, Enhanced System Manager or the computer operator automatically starts KILL/RMIT to produce the kill lists, if the workflows were generated.
20. Run the Cash Letter Summary (DKNCLSM) task for the current cycle (cycle 8) with the command

    ```
    CLSM ALL,8
    ```

    Select the Endpoint Table option and use table EPT002 when prompted for the table name.
21. Run the Master Create (DKNMCRE) task for the current cycle (cycle 8) with the command

    ```
    MCRE ALL,8
    ```

    This task transfers outgoing strings, which are no longer needed, to a master file and then deletes the killed and on-us D-strings, the subsequent-pass reject D-strings, and the R-strings.
22. Run the Input Create (DKNICRE) task for the current cycle (cycle 8) with the command

    ```
    ICRE ALL,8
    ```

    This task transfers all M-strings for the cycle to the input create file.
23. Run the Cycle task (DKNCYCL) to deactivate the current cycle (cycle 8) with the command

    ```
    CYCL 8,D
    ```
24. Run the List Directory (DKNLDIR) task for the current cycle (cycle 8), with the command

```
LDIR 8,D
```

This task lists all string entries for the particular cycle and deletes rehandle D-strings and subsequent-pass I-strings.

25. Run the End Cycle (DKNECYC) task with the command

```
ECYC 8
```

This task deletes the active M-strings, ensures that there are no other active strings present for this cycle, and removes all pass-to-pass control records for the cycle. DKNECYC also automatically starts the End Cycle 2 (DKNECY2) task, which transfers all kill-bundle records from the kill-bundle data set to a kill-bundle summary file.

26. End the CPCS-I task with the command

```
STOP
```

Chapter 6. Sample Problem 2: Enhanced Reject Processing

This sample problem shows the capabilities of alternate reject pockets and enhanced reject processing. The instructions provided tell you how to:

- Run a prime pass that automatically distributes to all pockets.
- Prepare kill bundles and cash letters for the first deadline.
- Process the rehandle pocket. DKNMICR, DKNKILL, and DKNCLSM process the data to produce the kill bundles and cash letters for the second deadline.
- Repair and process the transit rejects. This step also creates kill bundles and cash letters to satisfy the third deadline.
- Repair the bad data from the on-us reject pocket.
- Process the consolidated rejects. This step takes all the bad records from pockets that were originally good kill pockets and creates a new string for inclusion in the revised kill bundles and cash letters.
- Fix the errors from the system reject pocket. These repaired items are also included for the final revisions.
- Merge all the corrected data using DKNMRGE.
- Perform final processing. This includes backing up the data, generating reports, and deleting the data that is no longer useful.

Preparing to Run the Sample Problem

The test material contains:

- MDEF parameters for DKNMTASK generation
- CPCSRDR macro options for enhanced reject processing
- Sort pattern definition for enhanced reject processing options
- Test data for the problem
- SCI and OLRR changes for enhanced reject processing.

DKNMTASK Generation Options

The member SMICRUS in CPCSI.V01R01.SDKNSAM2 contains the options for DKNMTASK generation. An example of SAMPMDEF is shown on page 10-2. To run the sample problem as described, run GENMTASK with the options defined in SMDEFUS.

DKNMICR Generation Options

DKNMICR requires no special options in the CPCSRDR macro for this sample problem.

DKNSPDEF Options

The member SPTYP003 in CPCSI.V01R01.SDKNSAM2 provides the sort definition for this sample problem.

```
*********************************************************************
* This definition of SPTYP003 is for Sample Problem 2.
*
* The "P" (Pass Description) statement defines the following:
* Column 2 - 1000000 - Pass pocket history
* Column 9 - SAMP030Z - Stacker select module name
* Column 25 - 03 - Number of rehandle pockets created this pass
* Column 27 - 1 - Credits precede Debits
* Column 39 - 1 - Item number position 1 on Prime Pass
* Column 40 - 0030 - Kill bundle minimum count
* Column 44 - XF - Expanded Format Sort Definition
* Column 55 - M - Microfilm On
* Column 56 - 1 - Endorsing On
* Column 57 - 1 - Item Number Feature On
*
* The "RP" (Run Profile Description) statement is required for an XF
* sort.  The Run Profile name, SAMPXF01, is optional but the "XP" in
* columns 41-42 is required for an XP processor.
*
* The "BMSG" (BEGIN Message Definition) is optional and serves to
* provide the message displayed on the "BEGIN" screen during MICR
* processing.  The message for the prime pass is
* "ENHANCED REJECT PROCESSING - PASS 1"
*
* The "O" (Run Option) statement defines the following:
* Column 28 - Y - Enhanced reject processing
*
* The "J" (Reject Pocket Number) statement defines
* two pockets, 07 & 08, as alternate reject pockets.
* Four pockets, 09, 10, 11, & 12 are defined as eligible for
* bad data.
*
* The "I" (Image) statement defines the following:
* Column 5 - P - Prime only
* Columns 8 through 9 - BW - Black and white
* Columns 17 through 18 - BW - Black and white
*
* The "R" (Rehandle Description) statement directs that
* three tracers precede data in the rehandle pocket 06 (Col. 12)
* and alternate reject pockets 07 and 08 (Columns 14 & 16).
*
* The first two "K" statements define the kill pockets and their
* associated endpoints.  The first of these defines pocket 01, 02,
* 03, and 13 as kill pockets with an endpoint of 88888888.
* This endpoint designation is for an on-us kill pocket with
* no kill list.  Pockets 09, 10, 11 & 12 have endpoint codes that
* are found in the endpoint table EPT001.
*
*****************************************************************
*
* The second "P" statement defines the subsequent pass for the
* pocket 06 data.  Note the pass/pocket history has 2 in the pass
* number and 06 for the pass 1 pocket.
*
* The "RP" and "BMSG" statements fulfill the same purpose as for the
* prime pass, except that the "XP" in columns 41-42 has been
* replaced with "PE" for power encode.

* The "K" statements for this pass define pockets
* 04 through 08 and their corresponding endpoints.
*
*****************************************************************
```

```
*
* The third "P" statement defines the second subsequent pass.  This
* pass will re-sort the data that was originally in pocket 07, an
* alternate reject pocket.  The pass will be run after the data in
* the pocket has been repaired.
*
* For the "RP" and "BMSG" statements, refer to pass 2.
*
* The "H" statement defines the following:
* Column 10 - Y - MICR read system 1 ON.
* Column 11 - Y - MICR read system 2 ON.
* Column 14 - P - Power Encode option ON.
* Column 16 - Y - Power Encode path 1 ON.
* Column 17 - Y - Power Encode path 2 ON.
* Columns 24-30 - 1234567 - Encode all fields.
*
* The "O" (Run Option) statement defines the following:
* Column 16 - MIPIEX30 - DKNMIPI user-exit name.
* Column 24 - 09 - DKNMIPI work field number - See "FLD" following.
*
* The "FLD" statement defines a  user field, field 09, that is used
* by the simulator for the power endorser feature.
*
* The "K" statements associated with the second subsequent pass
* defines pockets 01, 02, 03, & 13 as on-us kill pockets and pockets
* 04 through 12 as transit kill pockets.
*
*****************************************************************
*
* The fourth "P" statement defines the third subsequent pass.  This
* pass uses the corrected data from pocket 08, another alternate
* reject pocket, as input.
*
* For the "RP" and "BMSG" statements, refer to pass 2.
* The "H" statement defines the following:
* Column 10 - Y - MICR read system 1 ON.
* Column 11 - Y - MICR read system 2 ON.
* Column 14 - P - Power Encode option ON.
* Column 16 - Y - Power Encode path 1 ON.
* Column 17 - Y - Power Encode path 2 ON.
* Columns 24-30 - 1234567 - Encode all fields.
*
* The "O" (Run Option) statement defines the following:
* Column 16 - MIPIEX30 - DKNMIPI user-exit name.
* Column 24 - 09 - DKNMIPI work field number - See "FLD" following.
*
* The "FLD" statement defines a  user field, field 09, that is used
* by the simulator for the power endorser feature.
*
* The "K" statements associated with the third subsequent pass
* defines pockets 01, 02, 03, & 13 as on-us kill pockets and pockets
* 04 through 12 as transit kill pockets.
*
*****************************************************************
```

```
P1000000SAMP030Z        031           10030XF         M11
RP      SAMPXF01                        XP
BMSG  ENHANCED REJECT PROCESSING - PASS 1
O                         Y
J         RREEEE
I   P  BW       BW
R          030303
K0188888888028888888803888888881388888888
K0909000000101000000011110000001212000000
**********************************************************************
P2060000SAMP230Z          1            0030XF
RP      SAMPXF01                        XP
BMSG  ENHANCED REJECT PROCESSING - PASS 2, PKT 06 - TRANSIT REHANDLE
K0402100002050213023506021000300702100008080200000O
**********************************************************************
P2070000SAMPR30Z          1            0030XF
RP      SAMPXF01                        PE
H        YY  P YY      1234567
O              MIPIEX3009
FLD09     045
BMSG  ENHANCED REJECT PROCESSING - PASS 2, PKT 07 - ONUS ALT REJECT
K0188888888028888888803888888881388888888
K0402100002050213023506021000300702100008080200000O
K0909000000101000000011110000001212000000
**********************************************************************
P2080000SAMPR30Z          1            0030XF
RP      SAMPXF01                        PE
H        YY  P YY      1234567
O              MIPIEX3009
FLD09     045
BMSG ENHANCED REJECT PROCESSING - PASS 2, PKT 08 - TRANSIT ALT REJECT
K0188888888028888888803888888881388888888
K0402100002050213023506021000300702100008080200000O
K0909000000101000000011110000001212000000
************************************************************
```

Figure 6-1. Sort Pattern Definition for Enhanced Reject Processing

Pocket Selection for Sample Problem 2

Figure 6-2, Figure 6-3, and Figure 6-4 on page 6-7 describe the contents of each of the pockets for this sample problem. In the column labeled *Routing/Transit*, the letter **n** can be replaced by any digit.

Figure 6-2. Sample Problem 2: Pocket Selection Criteria - Prime Pass

Pocket	Type	Routing/ Transit	Other Field	Description
RJ	Reject	Invalid	Auto-selects, control documents	System reject pocket
01	Kill	11nn-nnnn	Amt > $1M	On-us high dollar debits
02	Kill	11nn-nnnn	PC=AAAA33	On-us credits
03	Kill	11nn-nnnn	PC=AAAAAA	On-us debits
04	---	N/A	N/A	Not used
05	---	N/A	N/A	Not used
06	R/H	07nn-nnnn 08nn-nnnn 09nn-nnnn 16nn-nnnn 17nn-nnnn 18nn-nnnn 19nn-nnnn 20nn-nnnn 21nn-nnnn 22nn-nnnn	N/A	Second deadline transits
07	R/H	11nn-nnnn	Any bad field	On-us rejects
08	R/H	Anonymous	Bad amounts	Transit rejects
09	Kill	12nn-nnnn	N/A	First deadline transit EP=09000000
10	Kill	13nn-nnnn	N/A	First deadline transit EP=10000000
11	Kill	14nn-nnnn	N/A	First deadline transit EP=11000000
12	Kill	15nn-nnnn	N/A	First deadline transit EP=12000000
13	Kill	11nn-nnnn	EXT PC = AA & PC=AAAAAA	On-us debit return items

Figure 6-3. Sample Problem 2: Pocket Selection Criteria - Rehandle Pocket 06

Pocket	Type	Routing/ Transit	Other Field	Description
RJ	Reject	Invalid On-us	Autoselects, control documents	System reject pocket
01	---	N/A	N/A	Not used
02	---	N/A	N/A	Not used
03	---	N/A	N/A	Not used
04	Kill	18nn-nnnn	N/A	Second deadline transit EP=02100002
05	Kill	19nn-nnnn	N/A	Second deadline transit EP=02130235
06	Kill	20nn-nnnn	N/A	Second deadline transit EP=02100030
07	Kill	21nn-nnnn	N/A	Second deadline transit EP=02100008
08	Kill	07nn-nnnn 08nn-nnnn 09nn-nnnn 16nn-nnnn 17nn-nnnn 22nn-nnnn	N/A	Second deadline transit EP=02000000
09	---	N/A	N/A	Not used
10	---	N/A	N/A	Not used
11	---	N/A	N/A	Not used
12	---	N/A	N/A	Not used

Figure 6-4. Sample Problem 2: Pocket Selection Criteria - Subsequent-Pass Alternate Reject Pockets

Pocket	Type	Routing/ Transit	Other Field	Description
RJ	Reject	Invalid	Auto-selects, Control documents	System reject pocket
01	Kill	11nn-nnnn	Amt > $1M	On-us high dollar debits
02	Kill	11nn-nnnn	PC=AAAA33	On-us credits
03	Kill	11nn-nnnn	PC=AAAAAA	On-us debits
04	Kill	18nn-nnnn	N/A	Third deadline transit EP=02100002
05	Kill	19nn-nnnn	N/A	Third deadline transit EP=02130235
06	Kill	20nn-nnnn	N/A	Third deadline transit EP=02100030
07	Kill	21nn-nnnn	N/A	Third deadline transit EP=02000008
08	Kill	07nn-nnnn 08nn-nnnn 09nn-nnnn 16nn-nnnn 17nn-nnnn 22nn-nnnn		Third deadline transit EP=02000000
09	Kill	12nn-nnnn	N/A	Third deadline transit EP=09000000
10	Kill	13nn-nnnn	N/A	Third deadline transit EP=10000000
11	Kill	14nn-nnnn	N/A	Third deadline transit EP=11000000
12	Kill	15nn-nnnn	N/A	Third deadline transit EP=12000000
13	Kill	11nn-nnnn	EXT PC=AA & PC=AAAAAA	On-us debit return items

Data for Sample Problem 2

The data supplied for this problem is in member STEST03 of CPCSI.V01R01.SDKNSAM2. This sample data lets you test enhanced reject processing by providing data that includes error items for the alternate reject pockets, error items that will be sorted into the eligible kill pockets, and an error for the system reject pocket.

If you use the simulator for this sample problem, you must include the fields represented by periods, as shown in the records. If you use a physical document processor, do not encode the fields represented by the periods.

For a description of the commands used within the data, see the information about the 3890/XP Test Aid in the *3890/XP MVS Support and 3890/XP VSE Support Program Reference*.

Reports for Sample Problem 2

The reports that CPCS-I produces when you run this sample problem are in CPCSI.V01R01.RPTLIB. Print out the members of this library and compare your results with these reports for accuracy.

Operating Instructions for Sample Problem 2

You must also change the run JCL to use the correct sort pattern definitions and data, if you use the simulated sorter. If you use a physical document processor, inscribe the data provided on the appropriate documents.

1. Start CPCS-I by submitting the job in member DKNJCPCS from the CPCSI.V01R01.CTRL data set.
2. From the CPCS-I logo screen, log on to the relevant terminals with the command

   ```
   SGON xxx
   ```

   where xxx is the CPCS-I operator ID.
3. Designate one terminal as the system supervisor terminal with the command

   ```
   SUPV ON,SYST
   ```

   or by pressing **PF1** and then **ENTER**.
4. Activate cycle 8 by running the DKNCYCL program. The command is

   ```
   CYCL 8,A
   ```

   Let the cycle and endorse dates default to the current date. Follow the display instructions by pressing **ENTER** until the `READY` prompt appears. CPCS-I sends a message to the supervisor terminal indicating the change of a cycle's status.
5. Start MICR on any nonsupervisor terminal by entering the command

   ```
   MICR
   ```

   Open the selected simulated sorter by responding with

   ```
   O 3
   ```

   on the MICR Options screen.

   Note: You must use sorter number 3 for this sample problem.

6. After opening the sorter, begin the MICR run by selecting **B** on the Options screen. Specify cycle 8, sort pattern 003 (non-XF sort with subsets), and entry number 0030.

   Note: If you use the simulator, the JCL must use the test data supplied in STEST03 from CPCSI.V01R01.SDKNSAM2 as the input file.

7. End the prime-pass run when you see the `Intervention Required` message on the MICR Status screen. Do this by entering **E** at the prompt. Enter another **E** to verify the end command.

8. After ending the MICR run, go to the MICR Options screen by pressing **ENTER**. From the MICR Options screen, press **ENTER** to accept the default distribution options. Use the CLOSE command to end the MICR session.

   Note: Because no specific option was specified for distribution, the default causes DKNDIST to perform a full distribution when MICR ends.

9. Run DKNKILL to produce kill bundles for the prime-pass kill pockets. Do this by entering

   ```
   KILL 8
   ```

   (where the 8 represents the active cycle) and pressing **ENTER**. Select option **T** for Endpoint Table. When DKNKILL prompts you to respond to whether this is a reprint, press **ENTER** to select **NO**. When prompted for the endpoint table name, enter

   ```
   EPT001
   ```

   which is the table for the prime pass endpoints. Select **C** on the Options screen. This causes DKNKILL to automatically start DKNCLSM, which produces cash letters for the kill bundles.

Next, you will process the rehandle pocket (pocket 06).

10. Run a subsequent pass: Start MICR and enter

    ```
    O 3
    ```

    to OPEN sorter 3.

    Note: Use sorter number 3 for all subsequent passes.

    After entering **B** to begin, enter

    ```
    ,,0030-010
    ```

    on the Begin screen, where 0030 is the entry number and 010 is the sequence number of the first tracer in the rehandle pocket. DKNDIST runs automatically to distribute the data for the second deadline.

11. Run DKNKILL to generate kill lists. DKNKILL also calls DKNCLSM, which produces corresponding cash letters. Enter

    ```
    KILL 8
    ```

    (where 8 is the cycle number) and press **ENTER**. Select option **T** for Endpoint Table. Press **ENTER** to select **NO** when DKNKILL asks whether this is a reprint. When prompted for the endpoint table name, enter

    ```
    EPT002
    ```

    for the endpoint table name. Select option **C** for the cash letter option.

At this point, you have produced the kill bundles and cash letters for the second deadline.

Next, you repair the data in the kill alternate reject pocket. The repaired string runs through a subsequent pass and is redistributed to provide the third deadline kill bundles and cash letters.

12. To repair the incorrect records, enter

    ```
    OLRR
    ```

    and press **ENTER**. DKNOLRR displays a screen that includes a default string name. Because the alternate reject pocket is Z8, the string name you enter is

    ```
    0030-1-Z8-00-00-00-D-000
    ```

    The operator ID is

    ```
    01
    ```

    Leave BYPASS and RESTART set at **N** and press **ENTER**.

    Note: The incorrect records appear with asterisks (*) in place of the unreadable digits. Replace all the asterisks with 1s for this problem.

13. Run the string concatenation task (DKNSCAT) to rename the newly created R-string for DKNMRGE. Do this by entering

    ```
    SCAT 0030-1-08-00-00-01-R-000
    ```

    This creates a new string 0030-1-08-00-00-00-R-000 for use by DKNMRGE.

    Note: OLRR has already changed the name in the first pass-pocket history from Z8 to 08.

14. Run the subsequent pass using MICR. To do this, start MICR, open sorter 4 (a 3892 document processor with the power encode feature) and enter **B** to display the Begin screen. On the Begin screen, enter

    ```
    ,,0030-016
    ```

    to identify the first tracer selected to pocket 08. Ignore the following message:

    ```
    WARNING=MICMSPFL  1002 FLD09 BYTE LEN 045 IS GREATER THEN MDX LEN 000
    ```

    DKNMICR again starts DKNDIST to produce the D-strings for the data. End MICR and close the sorter by entering **C**.

15. Run DKNKILL to create the kill bundles and cash letters for the repaired data. To do this, enter

    ```
    KILL 8
    ```

    and press **ENTER**. You receive a prompt for:

Type of data to run	Response is **T**
Rerun – yes or no	Press **ENTER** for no
Endpoint table ID	Enter EPT002 for subsequent pass
Cash letter option	Enter **C**.

    This step completes the processing for the third deadline.

16. Run DKNOLRR to repair the data in the alternate reject pocket (pocket 07) selected for on-us errors. To do this, enter

    ```
    OLRR
    ```

    and press **ENTER**. For the string name, enter

    ```
    0030-1-Z7-00-00-00-D-000
    ```

    and press **ENTER**. Replace the error digits with 1s for this data.

17. Run DKNSCAT to rename the R-string created by OLRR. Enter

    ```
    SCAT 0030-1-07-00-00-01-R-000
    ```

    and press **ENTER**.

18. Run DKNMICR for the repaired data. To do this, start MICR and OPEN sorter 4 (a 3892 document processor with the power encode feature). On the Begin screen, enter

    ```
    ,,0030-013
    ```

    to select the first tracer in pocket 07. Ignore the following message:

    ```
    WARNING=MICMSPFL  1002 FLD09 BYTE LEN 045 IS GREATER THAN MDX LEN 000
    ```

    After you end the MICR run and close the sorter, DKNDIST starts and distributes the on-us data.

19. Run DKNDIST to distribute the consolidated reject pocket. To do this, enter the command

    ```
    DIST
    ```

    and press **ENTER** to display the menu for DKNDIST. In response to the first menu, enter the string name for the I-string

    ```
    0030-1-00-00-00-00-I-000
    ```

    The second screen from DIST asks for the type of distribution. Enter

    ```
    RC
    ```

    for the option and press **ENTER** to create the consolidated reject string. The resulting RC-string contains a copy of records, from the eligible pockets, that have one or more invalid fields.

20. Run the OLRR task to correct the data in the consolidated reject string. Enter

    ```
    OLRR
    ```

    and press **ENTER**. When the menu appears, enter

    ```
    0030-1-RC-00-00-00-D-000
    ```

    and

    ```
    01
    ```

    for the operator ID. Leave BYPASS and RESTART set at **N** and press **ENTER**.

    Correct the data in the consolidated reject pocket by replacing the asterisks with 1s.

21. Run DKNSCAT. This task creates a new string name to make it acceptable to DKNMRGE. To run DKNSCAT, enter the command

    ```
    SCAT 0030-1-RC-00-00-01-R-000
    ```

    where the RC in the pass 1 pocket field is the pocket number for the consolidated reject pocket and the 01 in the pass 4 pocket field shows the first (or only) R-string created for a repaired entry. DKNSCAT changes 01 to 00 and creates a new string.

22. Run the OLRR task to correct the rejects from the prime-pass system reject pocket. Start OLRR by specifying the entry number *nnnn* and operator ID 01. For the supplied data, enter

    ```
    OLRR 0030,01
    ```

    One document in the SYSTEM REJECT data is encoded with a digit error. Correct the error as follows:

    R/T error: `12*1-2222` should be `1221-2222`

    Note: After you correct the reject, the OLRR task ends automatically.

In the next few steps, DKNMRGE combines all the repaired data from the system reject pocket, the alternate reject pockets, and the consolidated reject string.

23. First you merge the repaired system reject data with the original I-string. To do this, enter

    ```
    MRGE 2,0030,000
    ```

    DKNMRGE option 1 forms an M-string by merging the original I-string with the R-string created by OLRR from the system reject data. DKNMRGE then deletes the system reject D-string for the prime pass and automatically starts the DKNPLST task, which deletes the prime-pass I-string and reports on the corrected entry.

    DKNMRGE displays all messages on the terminal from which it was started. DKNPLST displays its messages on the supervisor terminal.

24. The Exception Entry Master List report identifies one block as being out of balance. The out-of-balance block is the block slip with serial number 49. This condition is a result of the intentional entry of an incorrect amount during OLRR for this block.

25. Merge the corrected, consolidated reject D-string with the M-string created in the previous MRGE run. Enter

    ```
    MRGE 4,0030,000
    ```

    Note: There are no strings deleted in this step.

26. Now, merge the repaired data for the two alternate reject pockets. To perform this function, enter

    ```
    MRGE 6,0030,000,07
    ```

    to merge the pocket 07 data, followed by

    ```
    MRGE 6,0030,000,08
    ```

    to merge the corrected data from pocket 08.

    Note: There are no strings deleted in this step.

27. You must now do a final merge. Start the DKNMRGE task to create the final M-string from the existing interim M-string, with the command

    ```
    MRGE 7,0030
    ```

    DKNMRGE automatically starts the DKNPLST task for the corrected entry.

28. Run DKNMDIS to redistribute the corrected data in the M-string. To do this, enter

    ```
    MDIS
    ```

    and press **ENTER**. When the menu appears, enter

    ```
    0030-1-00-00-00-00-M-000
    Select option 4 for forced redistribution.
    ```

    The DKNMDIS task creates two new strings for the corrected data from alternate reject pockets 07 and 08.

    ```
    0030-1-07-00-00-00-D-000
    0030-1-08-00-00-00-D-000
    ```

    DKNKILL uses the new strings.

29. Run DKNKILL for the repaired items. DKNKILL produces the revised kill bundles and associated revised cash letters. To run the function, enter

    ```
    KILL 8
    ```

    (where 8 is the current cycle number) and press **ENTER**. Select option **T** for the Endpoint Table option. Enter **Y** to select **YES** when DKNKILL asks whether this is a reprint. When prompted for the table name, enter

    ```
    EPT003
    ```

    This table contains all the endpoints for both the prime pass and the subsequent-pass kill pockets. Select option **C** for the cash letter option.

30. Run Master Create (DKNMCRE) for the current cycle (cycle 8) with the command

    ```
    MCRE ALL,8
    ```

    This task transfers outgoing strings that are no longer needed to a master file. It then deletes the killed and on-us D-strings, the subsequent-pass reject D-strings, and the R-strings.

31. Run Input Create (DKNICRE) for the current cycle (cycle 8) with the command

    ```
    ICRE ALL,8
    ```

    This task transfers all M-strings for the cycle to the input-create file.

32. Run the Cycle task (DKNCYCL) to deactivate the current cycle (cycle 8) with the command

    ```
    CYCL 8,D
    ```

33. Run the List Directory task (DKNLDIR) for the current cycle (cycle 8) with the command

    ```
    LDIR 8,D
    ```

    This task lists all string entries for the particular cycle and deletes rehandle D-strings and subsequent-pass I-strings.

    Note: At this point in an actual check processing run, all strings for the cycle except the M-string have been deleted. This might not occur when running the sample problem. To delete any extraneous strings from the MDS, use the following steps:

    a. Run DKNSDIR. This job lists all strings in the MDS database. If any string other than the final M-string shows on the screen, it must be deleted.

To display the strings in the MDS, enter

```
SDIR
```

and press **ENTER**.

b. Enter

```
DELE xxxx-x-xx-xx-xx-xx-x-xxx
```

for each string to be deleted, where *xxxx-x-xx-xx-xx-xx-x-xxx* is the string name, and press **ENTER**.

34. Start the Microfilm Report task (DKNFILM). This task produces a report of the records in the microfilm data set and marks them for deletion by DKNECYC. Start the task with the command:

    ```
    FILM
    ```

    Enter **1** as the response to message 01 and press **ENTER**.

35. Run the End Cycle task (DKNECYC). This is the last task for this sample problem. Use the command

    ```
    ECYC 8
    ```

    This task deletes the active M-strings and ensures that there are no other active strings present for this cycle. It removes all pass-to-pass control records for the cycle. DKNECYC also automatically starts the End Cycle 2 (DKNECY2) task, which transfers all kill-bundle records from the kill-bundle data set to a kill-bundle summary file.

36. Start the Compress task (DKNCOMP). This task deletes marked records from the kill-bundle data set and compresses the data set to free space for the next cycle. To do this, enter

    ```
    COMP
    ```

    and press **ENTER**.

37. End the CPCS-I task with the command

    ```
    STOP
    ```

Chapter 7. Sample Problem 3: Unqualified Data

This sample problem simulates the use of a power encoder to inscribe unqualified fields. The prime pass contains four transit kill pockets, one transit rehandle pocket, and two on-us kill pockets (one is a credit kill pocket and the other is a debit kill pocket).

The sample problem sorts all documents that require power encoding to pockets defined as *unencoded*. You can process these documents through a key entry application to enter the amounts. You then enter the rehandle pocket on a subsequent pass through a 3892/XP Document Processor with the power encode feature to transfer the electronic image to the paper document. The subsequent pass sorts these documents to kill pockets.

To capture the amounts, you must also use a key entry application to process the documents sorted to the kill pockets on prime pass. However, because they are sorted to kill pockets, you do not need to power encode the amounts. After you repair these items, you can kill them in the normal way.

Preparing to Run the Sample Problem

The test material contains:

- Sort-pattern definition for an XF unqualified entry
- Data, with amount fields only, for control documents processing.

DKNSPDEF Options

The member SPTYP004 in CPCSI.V01R01.SDKNSAM2 contains the sort-pattern definition for this sample problem. This sort pattern includes a special definition for an unqualified prime-pass entry and a power encode subsequent-pass entry. The J-record identifies one unqualified pocket and six alternate reject pockets. The H-record in the subsequent pass specifies the power encode parameters.

```
P1000000SAMP040Z        011           10030XF          11 2
BMSG  UNQUALIFIED ENTRY PROCESSING - PASS 1
RP      SAMPXF01                        XP
*   0
O           Y
J    RR  U  RRRR
R          03
K028888888803888888881388888888
K0909000000101000000011110000001212000000
*********************************************************************
P2060000SAMP240Z          1            0030XF
BMSG  UNQUALIFIED ENTRY PROCESSING - PASS 2, PKT 06 - TRANSITS
RP      SAMPXF01                        XP
H        Y   P YY      1
FLD01     005  10             N
K040210000205021302350602100030070210000808020000000
```

Pocket Selection for Sample Problem 3

Figure 7-1 and Figure 7-2 describe the contents of each of the pockets for this sample problem. In the column labeled *Routing/Transit*, the letter **n** can be replaced by any digit.

Figure 7-1. Sample Problem 3: Pocket Selection Criteria - Prime Pass

Pocket	Type	Routing/ Transit	Other Field	Description
RJ	Reject	Invalid	Auto-selects, control documents	System reject pocket
01	---	N/A	N/A	Not used
02	Kill	11nn-nnnn	PC=AAAA33	On-us credit
03	R/H	11nn-nnnn	PC=AAAAAA	On-us debits
04	---	N/A	N/A	Not used
05	---	N/A	N/A	Not used
06	R/H	07nn-nnnn 08nn-nnnn 09nn-nnnn 16nn-nnnn 17nn-nnnn 18nn-nnnn 19nn-nnnn 20nn-nnnn 21nn-nnnn 22nn-nnnn	N/A	Second deadline transits
07	---	N/A	N/A	Not used
08	---	N/A	N/A	Not used
09	Kill	12nn-nnnn	N/A	First deadline transit EP=09000000. Unqualified items sent here.
10	Kill	13nn-nnnn	N/A	First deadline transit EP=10000000. Unqualified items sent here.
11	Kill	14nn-nnnn	N/A	First deadline transit EP=11000000. Unqualified items sent here.
12	Kill	15nn-nnnn	N/A	First deadline transit EP=12000000. Unqualified items sent here
13	Kill	11nn-nnnn	Ret I=AA and PC=AAAAAA	On-us debit return items. Items are prequalified.

Figure 7-2 (Page 1 of 2). Sample Problem 3: Pocket Selection Criteria - Rehandle Pocket 06

Pocket	Type	Routing/ Transit	Other Field	Description
RJ	Reject	Invalid	Auto-selects, control documents	System Reject Pocket
01	---	N/A	N/A	Not used
02	---	N/A	N/A	Not used
03	---	N/A	N/A	Not used
04	Kill	18nn-nnnn	N/A	Second deadline transit EP=02100002
05	Kill	19nn-nnnn	N/A	Second deadline transit EP=02130235

Figure 7-2 (Page 2 of 2). Sample Problem 3: Pocket Selection Criteria - Rehandle Pocket 06

Pocket	Type	Routing/ Transit	Other Field	Description
06	Kill	20nn-nnnn	N/A	Second deadline transit EP=02100030
07	Kill	21nn-nnnn	N/A	Second deadline transit EP=02100008
08	Kill	07nn-nnnn 08nn-nnnn 09nn-nnnn 16nn-nnnn 17nn-nnnn 22nn-nnnn	N/A	Second deadline transit EP=02000000

Data for Sample Problem 3

The input data is from member STEST04 in CPCSI.V01R01.SDKNSAM2 and contains unencoded items for both the kill and the rehandle pockets.

Note: If you use the simulator for this sample problem, you must include the fields represented by periods as shown in the records. If you use a physical document processor, do not encode the fields represented by the periods.

Reports for Sample Problem 3

The reports that CPCS-I produces when you run this sample problem are in CPCSI.V01R01.RPTLIB. Print out the members of this library and compare your results with these reports for accuracy.

Operating Instructions for Sample Problem 3

The installation tape contains all JCL necessary to run this sample problem. You can change this as needed. If you are using the simulated sorter, you must also change the run JCL to use the correct sort-pattern definitions and data. If you use a physical document processor, you must inscribe the data provided on the appropriate documents.

1. Start CPCS-I by submitting the job in member DKNJCPCS from the CPCSI.V01R01.CTRL data set.
2. From the CPCS-I logo screen, log on to the relevant terminals with the command

   ```
   SGON xxx
   ```

   where *xxx* is the CPCS-I operator ID.
3. Designate one terminal as the system supervisor terminal with the command

   ```
   SUPV ON,SYST
   ```

   or by pressing **PF1** and then **ENTER**.
4. Activate cycle 8 by running the DKNCYCL program. Let the cycle and endorse dates default to the current date. The command is

   ```
   CYCL 8,A
   ```

   Follow the display instructions by pressing **ENTER** until the `READY` prompt appears. CPCS-I sends a message to the supervisor terminal indicating the change of a cycle's status.
5. Start the MICR task on any nonsupervisor terminal by entering the command

   ```
   MICR
   ```

   From the MICR Options screen, open the sorter by entering

   ```
   O 4
   ```

   where **O** is for Open and **4** is the sorter number (points to entry 0040).
6. After opening the sorter, begin the MICR run by entering **B** on the Options screen. Specify cycle 8, sort pattern 004, and entry number 0040. Press **ENTER** to start the run.
7. End the prime-pass run when you see the `Intervention Required` message on the MICR status display. Do this by entering **E** at the prompt. Enter another **E** to verify the end command.
8. After ending the MICR run, go to the MICR Options screen by entering **O** and pressing **ENTER**. End the MICR session with the CLOSE command. DKNDIST displays a message on the supervisor terminal, indicating the end of the distribution task.

9. Use DKNOLRR on the following alternate reject kill pocket D-strings

   ```
   0040-1-Z9-00-00-00-D-000
   0040-1-A0-00-00-00-D-000
   0040-1-A1-00-00-00-D-000
   0040-1-A2-00-00-00-D-000
   ```

   To do this, enter **OLRR** and press **ENTER**. This starts the task and provides a menu display. Enter the string name and press **ENTER**. Repair each amount by using the value in the account number field as the amount. This simulates the use of an amount key entry application and creates the following R-strings

   ```
   0040-1-09-00-00-01-R-000
   0040-1-10-00-00-01-R-000
   0040-1-11-00-00-01-R-000
   0040-1-12-00-00-01-R-000
   ```

10. Run the String Concatenation task (DKNSCAT) on each of the R-strings produced by DKNOLRR. To do this, enter the following command

    ```
    SCAT 0040-1-pp-00-00-01-R-000
    ```

    where *pp* is the pocket number.

    This creates the following R-strings

    ```
    0040-1-09-00-00-00-R-000
    0040-1-10-00-00-00-R-000
    0040-1-11-00-00-00-R-000
    0040-1-12-00-00-00-R-000
    ```

11. Run the OLRR task to process the system reject pocket from prime pass. Specify the entry number and operator number 01 using the following command

    ```
    OLRR 0040,01
    ```

    There should be only one rejected document in the reject pocket. Correct the errors as follows:

    - R/T error: `13*1-2222` should be `1321-2222`
    - Copy the account number to the amount field.

12. Run the DKNMRGE option 1 for the prime-pass entry to create the initial M-string. To do this, enter the command

    ```
    MRGE 1,0040
    ```

    where 0040 is the entry number.

13. Run the pocket merge (Merge option 5) for each of the strings created by DKNSCAT. To do this, enter the command

    ```
    MRGE 5,0040,pp
    ```

    where 0040 is the entry number and *pp* is the pocket number.

14. Start the Kill List (DKNKILL) task for the current cycle (cycle 8) with the command

    ```
    KILL 8
    ```

    Select option **T** for Endpoint Table. When DKNKILL prompts you to respond to whether this is a reprint, press **ENTER** to select **NO**. When prompted for the endpoint table name, enter

    ```
    EPT001
    ```

This table contains the endpoints for each prime-pass kill pocket. Select option **D** on the Cash Letter Override screen.

15. Start the Cash Letter Summary task (DKNCLSM) for the current cycle (cycle 8) with the command

    ```
    CLSM ALL,8
    ```

    Select option **T** for Endpoint Table. When prompted for the endpoint table name, enter

    ```
    EPT001
    ```

    When DKNCLSM prompts you to respond to whether this is a duplicate letter, press **ENTER** to select **NO**.

16. Use DKNOLRR on the transit rehandle pocket, 0040-1-06-00-00-00-D-000. Enter

    ```
    OLRR
    ```

    and press **ENTER**. This starts the OLRR task and provides a menu display.

    Enter the string name and press **ENTER**. Repair each amount by using the value in the account number field as the amount. This simulates the use of an amount key-entry application and creates the following R-string:

    ```
    0040-1-06-00-00-01-R-000
    ```

17. Run the String Concatenation (SCAT) task on the R-string produced by DKNOLRR in the previous step. This creates the following R-string:

    ```
    0040-1-06-00-00-00-R-000
    ```

18. Run a MICR subsequent pass for the documents sorted to the rehandle pocket (pocket 06) after the prime pass for this entry. After starting a MICR session, open sorter 4 and go to the Begin screen.

    Note: The document processor for this run must be defined as a 3892/XP with the power encoder feature. Specify the same information as before (cycle 8, sort pattern 004), except for the entry number. The entry number is the first tracer sorted to that pocket:

    ```
    0040-yyy
    ```

    where 0040 is the entry number and *yyy* is the tracer number. For the sample data, *yyy* is 010.

    The system responds with subsequent-pass data displayed on the MICR Begin screen. Acknowledge the correct data by pressing **ENTER**.

19. As with the prime-pass entry, end the subsequent-pass run when you see the `Intervention Required` message on the MICR Status screen.

20. After ending the MICR run, enter **O** and press **ENTER** and end the MICR session with the CLOSE command.

21. System Manager workflows automatically start DKNDIST when you end the entry. Because this is a subsequent pass, System Manager automatically starts DKNSLST (Subsequent-Pass Master List).

22. Start the Kill List task for the subsequent-pass items with the command

    ```
    KILL 8
    ```

    where 8 is the current cycle number.

    Select option **T** for Endpoint Table. When DKNKILL prompts you to respond to whether this is a reprint, press **ENTER** to select **NO**. When prompted for the endpoint table name, enter

    ```
    EPT002
    ```

    for pass 2. This table contains the endpoints for each subsequent-pass kill pocket. Select option **D** on the Cash Letter Override screen.

23. Start the Cash Letter Summary (DKNCLSM) task for the subsequent-pass items with the same command as before

    ```
    CLSM ALL,8
    ```

    where 8 is the current cycle number.

    Select option **T** for Endpoint Table. When prompted for the endpoint table name, enter

    ```
    EPT002
    ```

    This table contains the endpoints for each subsequent-pass kill pocket. When DKNCLSM prompts you to respond to whether this is a duplicate letter, press **ENTER** to select **NO**.

24. Use DKNOLRR to repair the following on-us kill D-strings from the prime pass:

    ```
    0040-1-Z2-00-00-00-D-000
    0040-1-Z3-00-00-00-D-000
    ```

    Enter **OLRR** and press **ENTER**. This starts the OLRR task and provides a menu screen.

    Enter the string name and press **ENTER**. Repair each amount by using the value in the account number field as the amount. This simulates the use of an amount key-entry application and creates the following R-strings:

    ```
    0040-1-02-00-00-01-R-000
    0040-1-03-00-00-01-R-000
    ```

25. Run the String Concatenation (SCAT) task on each of the R-strings created in the previous step. To do this, enter the following command

    ```
    SCAT 0040-1-pp-00-00-01-R-000
    ```

    where *pp* is the pocket number. This creates the following R-strings:

    ```
    0040-1-02-00-00-00-R-000
    0040-1-03-00-00-00-R-000
    ```

26. Run the pocket merge (Merge option 5) for each of the following strings:

    ```
    0040-1-02-00-00-00-R-000
    0040-1-03-00-00-00-R-000
    0040-1-06-00-00-00-R-000
    ```

    To do this, enter the command

    ```
    MRGE 5,0040,pp
    ```

    where 0040 is the entry number and *pp* is the pocket number.

27. Run final merge (Merge option 7) for the entry to create the final M-string. To do this, enter the command

    ```
    MRGE 7,0040
    ```

28. Run the Master Create (DKNMCRE) task for the current cycle (cycle 8) with the command

    ```
    MCRE ALL,8
    ```

    This task transfers outgoing items, which do not require further processing, to a master file and deletes those items from the system.

29. Run the Input Create (DKNICRE) task for the current cycle (cycle 8) with the command

    ```
    ICRE ALL,8
    ```

    This task transfers all M-strings for the particular cycle to the input-create file.

30. Run the Cycle task (DKNCYCL) to deactivate the current cycle (cycle 8) with the command

    ```
    CYCL 8,D
    ```

31. Run the List Directory (DKNLDIR) task for the current cycle (cycle 8) with the command

    ```
    LDIR 8,D
    ```

    This task lists all string entries for the particular cycle and deletes all strings that do not need further processing by the system. After you run this task, the only string left in the mass-data-set is the M-string for the prime pass.

32. Run the End Cycle (DKNECYC) task for the current cycle (cycle 8) with the command

    ```
    ECYC 8
    ```

    This task ensures that there are no active strings present for this cycle, other than the prime-pass M-strings. It deletes these M-strings and removes all pass-to-pass control records for the cycle. It also automatically starts the End Cycle 2 (DKNECY2) task, which transfers all kill-bundle records from the kill-bundle data set to a kill-bundle summary file.

33. End the CPCS-I task with the command

    ```
    STOP
    ```

Chapter 8. Sample Problem 4: Divider Resynchronization

This sample problem shows the use of divider documents to resychronize data during a subsequent pass.

Preparing to Run the Sample Problem

The options and input data for this sample problem are the same as for Sample Problem 6, except for the O-record of the sort-pattern definition.

For this problem, you must change the sort-pattern definition by adding (or changing) the O-record for the prime-pass data. The O-record must contain a D in column 5. You must also enter the divider merge count in columns 9 through 12. The divider merge count value must be small enough to have at least three divider documents sorted to the rehandle pocket.

```
P1000000SAMP001Z        011           20030           2
R          03
O   D   0020
K018888888802888888880488888888058888888809090000001010000000
K11110000001212000000
P2060000SAMP1SPZ          0222223      0030
K0702000000030210000204021302350502100030060210000 8
```

Operating Instructions for Sample Problem 4

This sample problem uses the same procedures as in Sample Problem 6.

Note: This problem cannot be run using the simulator. You must use a physical document processor and encode the data on actual test documents. You must use sorter 5, which is a 3890/XP channel-attached document processor.

1. Set up and run Sample Problem 6 using SPTYP005 on a physical document processor. (For instructions, see “Operating Instructions for Sample Problem 6” on page 10-7.)
2. The data supplied for this problem is in member STEST05 of CPCSI.V01R01.SDKNSAM2. Encode the data on actual documents. Do not encode the fields represented by a period (.). Use the entry number 0050.
3. Before loading the rehandle pocket documents into the document processor’s feed unit, remove all documents beginning with the first divider slip and continuing up to, but not including, the second divider slip.

   Place these items at the end of the subsequent-pass entry.
4. Continue running the sample problem.

   There should be no missing or free documents after the run ends. The output should be the same as the output from Sample Problem 6.

Sample Problem 4: Divider Resynchronization

Chapter 9. Sample Problem 5: Base Functions with Expanded MDS

This sample problem shows the basic functions of CPCS-I with an expanded mass data set (MDS). It guides you through a sample run that shows the following:

- Data capture
- Distribution
- Online reject reentry (OLRR)
- Merging of the repaired data with the input data
- Producing kill lists and cash letters
- Subsequent-pass processing
- End-of-cycle processing.

Preparing to Run the Sample Problem

The test material contains:

- MDEF macro parameters for DKNMTASK generation
- CPCSRDR parameters for DKNMICR generation
- Sort-pattern definition for sorting
- Data containing tracer slips, block slips, and batch slips
- SCI and OLRR definitions for sorting.

Note: For more information about tailoring CPCS-I operation using expanded MDS records, see the *CPCS-I Customization Guide*.

This sample problem shows you how the system operates in an expanded environment. The data and reports are the same as for Sample Problem 6 with the following additional steps:

- Field 6 contains data so that you can show the return item code for on-us debits. There is no expansion of this field.
- Fields 9 and 12 should each contain 50 bytes to match the sample test data.
- The amount field should contain 8 bytes (16 digits).
- The SCI program sorts amounts over one million dollars to a special high-dollar pocket.

DKNMTASK Generation Options

The member SAMPMDEF in CPCSI.V01R01.SDKNSAM2 contains the options for generating DKNMTASK. The block size (BLKSIZE) parameter must be set as shown in the following example. This block size is a multiple of the expanded MDS record size of 161.

```
         TITLE 'CPCS/3890    SAMPLE MAIN TASK GENERATION'
         MDEF  SPOOL=12,
               MAXBUF=255,
               MAXTASK=30,
               BLKSIZE=3220,
               BLKSEG=10,
               SEGDEV=250,
               MAXDA=1,
               MAXOPEN=60,
               BFRAT=0,
               TPBFNM=0,
               DUPLEX=0,
               SCRLINC=10,
               KILLAUT=0,
               MDIREND=1,
               IDIREND=3,
               RDIREND=4,
               DDIREND=10,
               NDIRBLK=21,
               NUMPTR=3,
               DYNAMIC=0,
               TIMECK=0,
               SMACT=NO,
               SCRTY=NONE,
               LOG=NO
          END
```

DKNMICR Generation Options

These are the same as the MICR options for Sample Problem 6.

DKNSPDEF Options

The member SPTYP006 in CPCSI.V01R01.SDKNSAM2 contains the sort-pattern definition for this sample problem.

```
P1000000SAMP060Z        011           20030XF      U11 2
RP      SAMPXF01                        XP
BMSG  XF SAMPLE PROBLEM USING PROLOG2
R          03
K01777700010288888888038888888804888888880588888888090900000010100000000
K11110000001212000000
P2060000SAMP062Z           2222223     0030XF
RP      SAMPXF02                        XP
BMSG  XF SAMPLE PROBLEM USING PROLOG2 - PASS 2
K07020000000302100002040213023505021000300602100008
```

Pocket Selection for Sample Problem 5

Figure 9-1 and Figure 9-2 describe the selection criteria of each of the pockets for this sample problem. In the column labeled *Routing/Transit*, the letter **n** can be replaced by any digit.

Figure 9-1. Sample Problem 5: Pocket Selection Criteria - Prime Pass

Pocket	Type	Routing/ Transit	Other Requirements for Selection	Description
RJ	Reject	N/A	N/A	System reject pocket
01	Kill	11nn-nnnn 88nn-nnnn	N/A	High dollar on-us kill
02	Kill	11nn-nnnn	PC=AAAA33	On-us credit
03	Kill	11nn-nnnn	PC=AAAAAA	On-us debit
04	Kill	11nn-nnnn	Return item	On-us return item
05	---	N/A	N/A	Not used
06	R/H	07nn-nnnn 08nn-nnnn 09nn-nnnn 16nn-nnnn 17nn-nnnn 18nn-nnnn 19nn-nnnn 20nn-nnnn 21nn-nnnn 22nn-nnnn	N/A	Second deadline transits
07	---	N/A	N/A	Not used
08	---	N/A	N/A	Not used
09	Kill	12nn-nnnn	N/A	First deadline transit EP=09000000
10	Kill	13nn-nnnn	N/A	First deadline transit EP=10000000
11	Kill	14nn-nnnn	N/A	First deadline transit EP=11000000
12	Kill	15nn-nnnn	N/A	First deadline transit EP=11000000

Figure 9-2 (Page 1 of 2). Sample Problem 5: Pocket Selection Criteria - Subsequent Pass

Pocket	Type	Routing/ Transit	Other Requirements for Selection	Description
RJ	Reject	N/A	N/A	System reject pocket
01	---	N/A	N/A	Not used
02	---	N/A	N/A	Not used
03	Kill	18nn-nnnn	N/A	Second deadline transit EP=02100002
04	Kill	19nn-nnnn	N/A	Second deadline transit EP=02130235

Figure 9-2 (Page 2 of 2). Sample Problem 5: Pocket Selection Criteria - Subsequent Pass

Pocket	Type	Routing/ Transit	Other Requirements for Selection	Description
05	Kill	20nn-nnnn	N/A	Second deadline transit EP=02100030
06	Kill	21nn-nnnn	N/A	Second deadline transit EP=02100008
07	Kill	All other	N/A	All other transit checks EP=02000000
08	---	N/A	N/A	Not used
09	---	N/A	N/A	Not used
10	---	N/A	N/A	Not used
11	---	N/A	N/A	Not used
12	---	N/A	N/A	Not used

Data for Sample Problem 5

The data for this test is in member STEST06 of CPCSI.V01R01.SDKNSAM2 and consists of one set of tracers with associated data. It also contains block and batch slips. This data contains deliberate errors to show the balancing features of the system.

If you use the simulator for this sample problem, you must include the fields indicated by periods and the data for fields 9 and 12, as shown in the records. If you use a physical document processor, do not encode the fields represented by the periods. The document processor cannot interpret fields 9 and 12.

Reports for Sample Problem 5

The reports that CPCS-I produces when you run this sample problem are in CPCSI.V01R01.RPTLIB. Print out the members of this library and compare your results with these reports for accuracy.

Notice that all amount fields are expanded. The hexadecimal-string list report is included to show fields 6, 9, and 12 in an expanded format report.

Operating Instructions for Sample Problem 5

Run this sample problem the same way you run Sample Problem 6 (see “Operating Instructions for Sample Problem 6” on page 10-7) with the following additional steps:

1. Generate an expanded MDS. (For information about setting up an expanded environment, see the *CPCS-I Customization Guide*.)
   - Fields 9 and 12 should be 50 bytes each to match the sample data.
   - The amount field should be 8 bytes.
2. You can run an expanded format (XF) sort only on an XP model document processor. To run a simulated XF POD, you must open an XP simulated sorter. We use sorter 6 for Sample Problem 5.
3. Run MICR with a sort-pattern definition for expanded format. Use sort pattern 006 for this sample problem.
4. Use entry number 0060.

Chapter 10. Sample Problem 6: Basic Functions

This sample problem shows the basic functions of CPCS-I. It guides you through sample runs that show the following:

- Data capture
- System Manager processing
- Distribution
- Online reject re-entry (OLRR)
- Merging of the repaired data with the input data
- Producing kill lists and docket control voucher/cash letter summaries (DCV/CLSM)
- Subsequent-pass processing
- End-of-cycle processing

Preparing to Run the Sample Problems

The test material contains:

- MDEF macro parameters for DKNMTASK generation
- CPCSRDR parameters for DKNMICR generation
- DKNSPDEF sort-pattern definition
- Sample data containing tracer slips and block slips
- SCI and OLRR definitions for sorting
- Sample electronic data for DEFT processing.

DKNMTASK Generation Options

The MDEF macro requires the following parameters for the sample problems. The member SAMPMDEF in CPCSI.V01R01.SDKNSAM2 contains the options for the DKNMTASK definition.

```
         TITLE 'CPCS/3890    SAMPLE MAIN TASK GENERATION'
         MDEF  SPOOL=12,                                        -
               MAXBUF=255,                                      -
               MAXTASK=50,                                      -
               BLKSIZE=1450,                                    -
               BLKSEG=16,                                       -
               SEGDEV=200,                                      -
               MAXDA=1,                                         -
               MAXOPEN=100,                                     -
               MAXSORT=3,                                       -
               DUPLEX=0,                                        -
               KBLRECL=200,                                     -
               KBBLKSZ=4800,                                    -
               LOG=NO,                                          -
               MDIREND=1,                                       -
               IDIREND=3,                                       -
               RDIREND=4,                                       -
               DDIREND=10,                                      -
               NDIRBLK=21,                                      -
               NUMPTR=14,                                       -
               DYNAMIC=1,                                       -
               TIMECK=0,                                        -
               SCRTY=NONE,                                      -
               BFRAT=0,                                         -
               TPBFNM=0,                                        -
               RCVY=NO,                                         -
               LNTNAME=,                                        -
               LANG=US
          END
```

DKNMICR Generation Options

The member SAMPMICR in CPCSI.V01R01.SDKNSAM2 contains the options that are necessary for the MICR task generation. DKNMICR requires the following parameters for the sample problems in the CPCSRDR macro:

ATTACH=(CHANNEL | LU62 | SIM)

The value must be SIM if you use the simulator; it must be CHANNEL or LU62 if you use a physical sorter. The default value is CHANNEL.

DDRDRIN=*ddname*

If you are using a physical document processor, you must also change this parameter. You can find a description of DDRDRIN in the *CPCS-I Customization Guide*.

```
         TITLE 'CPCS - MICR TASK GENERATION'
*********************************************************************
*        READER SORTER 1
*
RDR1     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER01,MODEL=A,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,                    -
               CODELINE=US
*
*********************************************************************
*        READER SORTER 2
*
RDR2     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER02,MODEL=B,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,                    -
               CODELINE=US
*
*********************************************************************
*        READER SORTER 3
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,          -
               MFILM=YES,ATTACH=SIM,PEND=YES,                    -
               CODELINE=US
*
*********************************************************************
*        READER SORTER 4
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,          -
               MFILM=YES,ATTACH=SIM,PEND=YES,                    -
               CODELINE=US
*
*********************************************************************
*        READER SORTER 5
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,          -
               MFILM=YES,ATTACH=SIM,PEND=YES,                    -
               CODELINE=US
*
```

```
*********************************************************************
*        READER SORTER 6
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,                     -
               CODELINE=US
*
*********************************************************************
*        READER SORTER 7
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,                     -
               CODELINE=US
*
*********************************************************************
*        READER SORTER 8
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,                     -
               CODELINE=US
*
*********************************************************************
*        READER SORTER 9
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,                     -
               CODELINE=US
*
```

DKNMICR Generation Options

The member SAMPMICR in CPCSI.V01R01.SDKNSAM2 contains the options that are necessary for the MICR task generation. DKNMICR requires the following parameters for this sample problem in the CPCSRDR macro:

ATTACH=(CHANNEL | LU62 | SIM)

If you use the simulator, the value must be SIM; if you use a physical sorter, the value must be CHANNEL or LU62. The default value is CHANNEL.

DDRDRIN=*ddname*

If you are using a physical document processor, you must also change this parameter. For a description of DDRDRIN, see the *CPCS-I Customization Guide*.

```
         TITLE 'CPCS - MICR TASK GENERATION'
*********************************************************************
*        READER SORTER 1
*
RDR1     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER01,MFILM=YES,MODEL=A,  -
               ENDORSE=YES,INF=YES,ATTACH=SIM,CODELINE=US
*
*********************************************************************
*        READER SORTER 2
*
RDR2     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER02,MODEL=B,            -
               MFILM=YES,ATTACH=SIM,CODELINE=US
*
*********************************************************************
*        READER SORTER 3
*
RDR3     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER03,MODEL=XP,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,IMAGE=YES,CODELINE=US
*
*********************************************************************
*        READER SORTER 4
*
RDR4     CPCSRDR TYPE=3892-6,DDRDRIN=SORTER04,MODEL=XP,           -
               MFILM=YES,ATTACH=SIM,POWER=YES,PEND=YES,CODELINE=US
*
*********************************************************************
*        READER SORTER 5
*
RDR5     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER05,MODEL=XP,           -
               MFILM=YES,ATTACH=CHANNEL,PEND=YES,CODELINE=US
*
*********************************************************************
*        READER SORTER 6
*
RDR6     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER06,MODEL=XP,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,CODELINE=US
*
*********************************************************************
*        READER SORTER 7
*
RDR7     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER07,MODEL=XP,           -
               MFILM=YES,ATTACH=SIM,PEND=YES,IMAGE=YES,CODELINE=US
*
```

```
*********************************************************************
*        READER SORTER 8
*
RDR8     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER08,MFILM=YES,MODEL=A,  -
               ENDORSE=YES,INF=YES,ATTACH=SIM,CODELINE=US
*
*********************************************************************
*        READER SORTER 9
*
RDR9     CPCSRDR TYPE=3890-6,DDRDRIN=SORTER09,MFILM=YES,MODEL=A,  -
               ENDORSE=YES,INF=YES,ATTACH=SIM,CODELINE=US
*
*********************************************************************
         END
```

Pocket Selection for Sample Problem 6

Figure 10-1 and Figure 10-2 describe the contents of each of the pockets for this sample problem. In the column labeled *Routing/Transit*, the letter **n** can be replaced by any digit.

Figure 10-1. Sample Problem 6: Pocket Selection Criteria - Prime Pass

Pocket	Type	Routing/ Transit	Other Requirements for Selection	Description
RJ	Reject	N/A	N/A	System reject pocket
01	---	N/A	N/A	Not used
02	Kill	11nn-nnnn	PC=AAAA33	On-us credit
03	Kill	11nn-nnnn	PC=AAAAAA	On-us debit
04	Kill	11nn-nnnn	Return item	On-us return item
05	---	N/A	N/A	Not used
06	R/H	07nn-nnnn 08nn-nnnn 09nn-nnnn 16nn-nnnn 17nn-nnnn 18nn-nnnn 19nn-nnnn 20nn-nnnn 21nn-nnnn 22nn-nnnn	N/A	Second deadline transits
07	---	N/A	N/A	Not used
08	---	N/A	N/A	Not used
09	Kill	12nn-nnnn	N/A	First deadline transit EP=09000000
10	Kill	13nn-nnnn	N/A	First deadline transit EP=10000000
11	Kill	14nn-nnnn	N/A	First deadline transit EP=11000000
12	Kill	15nn-nnnn	N/A	First deadline transit EP=11000000

Figure 10-2. Sample Problem 6: Pocket Selection Criteria - Subsequent Pass

Pocket	Type	Routing/ Transit	Other Requirements for Selection	Description
RJ	Reject	N/A	N/A	System reject pocket
01	---	N/A	N/A	Not used
02	---	N/A	N/A	Not used
03	Kill	18nn-nnnn	N/A	Second deadline transit EP=02100002
04	Kill	19nn-nnnn	N/A	Second deadline transit EP=02130235
05	Kill	20nn-nnnn	N/A	Second deadline transit EP=02100030
06	Kill	21nn-nnnn	N/A	Second deadline transit EP=02100008
07	Kill	All other	N/A	All other transit checks EP=02000000
08	---	N/A	N/A	Not used
09	---	N/A	N/A	Not used
10	---	N/A	N/A	Not used
11	---	N/A	N/A	Not used
12	---	N/A	N/A	Not used

Data for Sample Problem 6

The data for this sample problem consists of one set of tracers with associated data, including block, batch, and subbatch slips. This data also contains deliberate errors to show the balancing features of the system.

The member STEST01 in CPCSI.V01R01.SDKNSAM2 contains this input. If you use the simulator for this sample problem, you must include the fields represented by periods as shown in the records. If you use a physical document processor, do not encode the fields represented by the periods.

Reports for Sample Problem 6

The reports that CPCS-I produces when you run this sample problem are in CPCSI.V01R01.RPTLIB. Print out the members of this library and compare your results with these reports for accuracy.

Operating Instructions for Sample Problem 6

1. Start CPCS-I by submitting the job in member DKNJCPCS from the CPCSI.V01R01.CTRL data set.
2. From the CPCS-I logo screen, log on to the relevant terminals with the command

   ```
   SGON xxx
   ```

   where *xxx* is the CPCS-I operator ID.

3. Designate one terminal as the system supervisor terminal with the command

   ```
   SUPV ON,SYST
   ```

   or by pressing **PF1** and then **ENTER**.

4. Activate cycle 8 by running the DKNCYCL program. The command is

   ```
   CYCL 8,A
   ```

   Let the cycle and endorse dates default to the current date. Follow the display instructions by pressing **ENTER** until the `READY` prompt appears. CPCS-I sends a message to the supervisor terminal indicating the change of a cycle's status.

5. Start the MICR task (DKNMICR) on any nonsupervisor terminal by entering the command

   ```
   MICR
   ```

   Open sorter 1 by entering

   ```
   O 1
   ```

   on the MICR Options screen.

6. After opening the sorter, begin the MICR run. Specify cycle 8, sort pattern 001 (non-XF sort), and entry number 0001.

7. End the prime-pass run when you see the `Intervention Required` message on the MICR status screen. Do this by entering **E** at the prompt. Enter another **E** to verify the end command.

8. After ending the MICR run, go to the MICR Options screen by entering **O** and pressing **ENTER**. End the MICR session with the CLOSE command. The distribution task (DKNDIST) presents a message on the supervisor terminal indicating the end of the distribution task.

9. Run the OLRR task (DKNOLRR) to correct the rejects from the prime pass. Specify the entry number 00*x*0, where *x* is the sorter number, followed by operator ID 01. For example, an entry run on sorter number 1 requires the command

   ```
   OLRR 0001,01
   ```

   Three documents in the example contain digit errors. Correct the errors as follows:

   - R/T error: `12*1-2222` should be `1221-2222`
   - Amt error: `00000*0002` should be `0000010002`
   - R/T error: `1110-006*` should be `1110-0062`

10. After you correct the rejects, the OLRR task ends automatically. When it finishes, start the string merge task (DKNMRGE) for the corrected entry with the command

    ```
    MRGE 1,0010
    ```

    DKNMRGE option 1 merges all data into an M-string by combining the I-string from the prime pass with the R-string produced by OLRR processing.

11. DKNMRGE then automatically starts the entry master list (DKNPLST) task for the corrected entry. DKNMRGE displays messages on the terminal that started the task. DKNPLST displays messages on the supervisor terminal.

12. The Exception Entry Master List report, produced by DKNPLST, identifies one subbatch as being `OUT OF BALANCE`. The subbatch that is out of balance is the

subbatch slip with the serial number of 51. Because this subbatch is out of balance, the batch and block are also out of balance.

This out-of-balance condition is caused by the intentional entry of an incorrect amount ($100.02) during OLRR for this subbatch. ($.02 is the correct amount.)

13. Start the Kill List task (DKNKILL) for the current cycle (cycle 8) with the command

    ```
    KILL 8
    ```

    Select the Endpoint Table option, accept the default value for reprint (press **ENTER**), and enter **EPT001** when prompted for the table name. This table contains the endpoints for each prime-pass kill pocket.

14. Start the Cash Letter Summary task (DKNCLSM) for the current cycle (cycle 8) with the command

    ```
    CLSM ALL,8
    ```

    Select the Endpoint Table option and use table EPT001 when prompted for the table name. When DKNCLSM prompts you to respond to whether this is a duplicate letter, press **ENTER** to select **NO**.

15. Start the MICR subsequent pass for the documents sorted to the rehandle pocket (pocket 06) on prime pass for this entry.

    a. Open the sorter and proceed to the Begin screen.

    b. Specify the same information as before (cycle 8, sort pattern 001), except for the entry number.

    c. The entry number is the first tracer sorted to the rehandle pocket. Enter it in the format

       ```
       00x0-yyy
       ```

       where *x* is the sorter number and *yyy* is the tracer number. For a simulated sorter subsequent pass, the entry number is always 00x0-010.

       Subsequent-pass data appears on the MICR Begin screen.

    d. Acknowledge the correct data by pressing **ENTER**.

16. End the subsequent-pass run by entering **E** when you see the `Intervention Required` message on the MICR Status screen.

17. After ending the MICR run, enter **O**, press **ENTER**, and end the MICR session with the CLOSE command.

18. The distribution task starts automatically when you end the entry. Because this is a subsequent pass, System Manager automatically starts DKNSLST to produce a Subsequent Pass Master List.

    The Subsequent Pass Master List identifies, on an item-count and dollar-amount basis, any out-of-balance condition between the prime-pass and subsequent-pass runs. For this entry, there are no exception conditions.

19. Start the Kill List task for the subsequent-pass items with the command

    ```
    KILL 8
    ```

    where 8 is the current cycle number.

    Select the Endpoint Table option, take the default for rerun, and use table EPT002 (for subsequent-pass items) when prompted for the table name. This table contains the endpoints for each subsequent-pass kill pocket.

20. Start the Cash Letter Summary (DKNCLSM) task for the subsequent-pass items with the command

    ```
    CLSM ALL,8
    ```

    where 8 is the current cycle number.

    Select the Endpoint Table option and use table EPT002 when prompted for the table name. This table was created with the endpoints for each subsequent-pass kill pocket. When DKNCLSM prompts you to respond to whether this is a duplicate letter, press **ENTER** to select **NO**.

21. Run the Master Create task (DKNMCRE) for the current cycle (cycle 8) with the command

    ```
    MCRE ALL,8
    ```

    This task transfers outgoing items, which do not require further processing, to a master file and deletes these items (D-strings) from the system.

22. Run the Input Create task (DKNICRE) for the current cycle (cycle 8) with the command

    ```
    ICRE ALL,8
    ```

    This task transfers all M-strings for the particular cycle to the input create file.

23. Run the Cycle task to deactivate the current cycle (cycle 8) with the command

    ```
    CYCL 8,D
    ```

24. Run the List Directory task (DKNLDIR) for the current cycle (cycle 8) with the command

    ```
    LDIR 8,D
    ```

    This task lists all string entries for the particular cycle and deletes all strings that are fully processed by the system. After you run this task, the only string left on the mass data set is the M-string for the prime pass.

25. Run the End Cycle task (DKNECYC) for the current cycle (cycle 8) with the command

    ```
    ECYC 8
    ```

    This task ensures that there are no active strings present for this cycle, other than the prime-pass M-strings.

    It deletes these M-strings and removes all pass-to-pass control records for the cycle. It also automatically starts the End Cycle Two task (DKNECY2), which transfers all kill-bundle records from the kill-bundle data set to a kill-bundle summary file.

26. End the CPCS-I task with the command

    ```
    STOP
    ```

Operating Instructions for the Sample Problem with HSRR

To run the sample problem with HSRR, you can either use a physical sorter, or use the simulated sorters provided with CPCS-I.

If you use a physical sorter:

- Follow the instructions in this section and make the changes required to use the tracer group numbers of the actual tracer group slips that you are using. You will also need to use the sorter numbers of the sorters that you are using.
- For the prime pass, turn several documents upside down so that they are rejected on the prime pass.
- For the HSRR pass, turn the documents that you want HSRR to read accurately to their correct orientation.

If you use the simulated sorters and data, use sorter 8 for the prime pass and sorter 9 for the HSRR passes.

1. Start CPCS-I by submitting the job DKNJCPCS from the CPCSI.V01R01.CTRL data set.
2. From the CPCS-I logo screen, log on to the CPCS-I terminals by entering

   ```
   SGON xxx
   ```

   where *xxx* is the CPCS-I operator ID.
3. Designate one terminal as the system supervisor terminal with the command

   ```
   SUPV ON,SYST
   ```

   or by pressing **PF1** and then **ENTER**.
4. Activate cycle 8 by entering

   ```
   CYCL 8,A
   ```

   Let the cycle and endorse dates default to the current date. Follow the instructions by pressing **ENTER** until the `READY` prompt appears. CPCS-I sends a message to the supervisor terminal, indicating the change of a cycle's status.
5. Run the prime-pass entry.
   a. Start the MICR task (DKNMICR) on any nonsupervisor terminal by entering the command

      ```
      MICR
      ```

      Open sorter number 8 by entering

      ```
      O 8
      ```

      On the MICR Options screen, where **O** is the Open command and 8 represents the sorter number.

   b. After opening the sorter, begin the MICR run. Specify cycle 8, sort pattern 001 (non-XF sort), and entry number 0010. If you are using a physical sorter, use the number of your first tracer slip as the entry number.

      Note: Remember this entry number. It is used later in these instructions. Future references to this number will refer to the *prime-pass entry number*.

   c. End the prime pass when you see the `Intervention Required` message on the MICR Status screen. End the MICR run by entering **E** twice.

   d. After ending the MICR run, go to the MICR Options screen by entering **O**. End the MICR session with the CLOSE command. DKNDIST displays a message on the supervisor terminal indicating the end of the distribution task.
6. High Speed Reject Reentry (HSRR)

a. If you use a physical sorter, manually recondition the rejects from the prime-pass system reject pocket to ensure that all documents that are to be corrected on the HSRR pass are positioned correctly, and remain in the same sequence as the prime pass.

   Place a new set of tracer slips in front of the documents from the reject pocket (including the prime-pass tracer slips).

   Note: Remember the new tracer slip number. It is referred to later in these instructions as the *HSRR entry number*.

b. Start MICR on a nonsupervisor terminal by entering the command

   ```
   MICR
   ```

   If you are using the simulated sorters, open sorter 9 with the command

   ```
   O 9
   ```

   on the MICR options screen. If you are using a physical sorter, use the number of that sorter.

c. Begin the MICR run by entering Begin. At the Begin screen, enter

   ```
   8,1,0100,0010
   ```

   Where 8 is the cycle number, 1 is the sort pattern, 0100 is the HSRR entry number, and 0010 is the prime-pass entry number. If you are using a physical sorter, you must use the correct entry numbers for your tracer slips. The first four digits of the entry numbers are all that is required.

7. End the HSRR run when all documents have passed through the sorter or when you see `Intervention Required` on the MICR Status screen. End the HSRR run by entering **E** twice.

8. Return to the MICR Options screen by entering **O** and pressing **ENTER**. End the MICR session with the CLOSE command.

9. The three items encoded with digit errors should remain in the reject pocket, while all other noncontrol documents are sorted to their correct pockets. You must use OLRR to correct the remaining rejects.

10. Run the OLRR task to correct the rejects from the HSRR pass. Specify the HSRR entry number as you did previously in step 9 on page 10-8. For example, if the HSRR entry number was 0100, use the command

    ```
    OLRR 0100,01
    ```

    Three documents in the example contain digit errors. Correct the errors as follows:

    - R/T error: `12*1-2222` should be `1221-2222`
    - Amt error: `00000*0002` should be `0000010002`
    - R/T error: `1110-006*` should be `1110-0062`.

11. After you correct the rejects, the OLRR task ends automatically. When it finishes, start the string merge task (DKNMRGE) for the corrected HSRR entry with the command

    ```
    MRGE 1,nnnn
    ```

    where *nnnn* is the HSRR entry number. DKNMRGE option 1 merges all data into an M-string by combining the I-string from the prime pass with the HSRR I-string (produced by the HSRR MICR run) and the R-string (produced by OLRR). The M-string is for the prime-pass entry number.

12. DKNMRGE then automatically starts the entry master list (DKNPLST) task for the corrected entry. DKNMRGE displays messages on the terminal that started the task. DKNPLST displays messages on the supervisor terminal.
13. The Exception Entry Master List report identifies one subbatch as being `OUT OF BALANCE`. The subbatch that is out of balance is the subbatch slip with the serial number of 51. Because this subbatch is out of balance, the batch and block are also out of balance.

    This condition is the result of the intentional entry of an incorrect amount ($100.02) during OLRR for this subbatch. ($.02 is the correct amount.)
14. Start the Kill List task (DKNKILL) for the current cycle (cycle 8) with the command

    ```
    KILL 8
    ```

    Select the Endpoint Table option, accept the default value for reprint (press **ENTER**), and use table EPT001 when prompted for the table name. This table contains the endpoints for each prime-pass kill pocket.
15. Start the Cash Letter Summary task (DKNCLSM) for the current cycle (cycle 8) with the command

    ```
    CLSM ALL,8
    ```

    Select the Endpoint Table option and use table EPT001 when prompted for the table name.
16. You can run the subsequent pass in one of two ways. You can either run a separate subsequent pass for the prime pass and HSRR rehandle pockets, or you can run both of the rehandle pockets in one subsequent pass.

    To run each rehandle pocket as a separate subsequent pass, do the following:

    a. Start the MICR subsequent pass for the documents sorted to the rehandle pocket (pocket 06) on prime pass for this entry. After starting a MICR session, open sorter 8 and proceed to the Begin screen. Specify the same information as before (cycle 8 and sort pattern 001), except for the entry number. The entry number is the first tracer sorted to the rehandle pocket. Enter it in the format

       ```
       0010-yyy
       ```

       where *0010* is the prime-pass entry number and *yyy* is the tracer number. For a simulated sorter subsequent pass, the entry number is always 0080-010.

       The system responds with subsequent-pass data displayed on the MICR Begin screen. Acknowledge the correct data by pressing **ENTER**.

    b. End the subsequent-pass run when you see the `Intervention Required` message on the MICR Status screen. End the subsequent-pass run by entering **E** twice.

    c. After ending the MICR run, enter **O** and press **ENTER** to go to the MICR Options screen. End the MICR session with the CLOSE command.

    d. System Manager workflows automatically start the distribution task (DKNDIST) when you end the entry. Because this is a subsequent pass, System Manager automatically starts the subsequent-pass Master List task (DKNSLST).

The Subsequent Pass Master List identifies, on an item-count and dollar-amount basis, any out-of-balance condition between the prime-pass and subsequent-pass runs. For this entry, there are no exception conditions.

e. Start the MICR subsequent pass for the documents sorted to the rehandle pocket (pocket 06) on the HSRR run for this entry. After starting a MICR session, open sorter 9 and proceed to the Begin screen. Specify the same information as for the last subsequent pass (cycle 8 and sort pattern 001), except for the entry number. The entry number is the first tracer sorted to the rehandle pocket on the HSRR run. Enter it in the format

```
nnnn-yyy
```

where *nnnn* is the HSRR entry number and *yyy* is the tracer number. For a simulated sorter subsequent pass where the HSRR entry number is 0100, the entry number is 0100-010.

The system responds with subsequent-pass data shown on the MICR Begin screen. Acknowledge the correct data by pressing **ENTER**.

f. End the subsequent-pass run when you see the `Intervention Required` message on the MICR Status screen. End the subsequent-pass run by entering **E** twice.

g. After ending the MICR run, enter **O** and press **ENTER** to go to the MICR Options screen. End the MICR session with the CLOSE command.

h. System Manager workflows automatically starts DKNDIST when you end the entry. Because this is a subsequent pass, System Manager automatically starts DKNSLST (Subsequent Pass Master List).

17. Start the Kill List task for the subsequent-pass items with the command

```
KILL 8
```

where 8 is the current cycle number.

Select the Endpoint Table option, use the default for the reprint option, and use table EPT002 (for subsequent-pass items) when prompted for the table name. This table contains the endpoints for each subsequent-pass kill pocket.

18. Start the Cash Letter Summary task for the subsequent-pass items with the command

```
CLSM ALL,8
```

where 8 is the current cycle number.

Select the Endpoint Table option and use table EPT002 when prompted for the table name. This table was created with the endpoints for each subsequent-pass kill pocket.

19. Run the Master Create (DKNMCRE) task for the current cycle (cycle 8) with the command

```
MCRE ALL,8
```

This task transfers outgoing items, which do not require further processing, to a master file and deletes these items (D-strings) from the system.

20. Run the Input Create (DKNICRE) task for the current cycle (cycle 8) with the command

```
ICRE ALL,8
```

This task transfers all M-strings for the specified cycle to the input-create file.

21. Run the Cycle task to deactivate the current cycle (cycle 8), with the command

    ```
    CYCL 8,D
    ```

22. Run the List Directory (DKNLDIR) task for the current cycle (cycle 8) with the command

    ```
    LDIR 8,D
    ```

    This task lists all string entries for the cycle and deletes all strings that do not need further processing by the system. After you run this task, the only string left on the mass data set is the M-string for the prime pass.

23. Run the End Cycle (DKNECYC) task for the current cycle (cycle 8) with the command

    ```
    ECYC 8
    ```

    This task ensures that there are no active strings present for this cycle other than the prime-pass M-string. It deletes this M-string and removes all pass-to-pass control records for the cycle. It also automatically starts the DKNECY2 task, which transfers all kill-bundle records from the kill-bundle data set to a kill-bundle summary file.

24. End the CPCS task with the command

    ```
    STOP
    ```

Appendix A. Recommended Enhanced System Manager Workflows for DCVR

This section describes the Enhanced System Manager workflows for DCV Reconciliation (DCVR).

See the *CPCS Enhanced System Manager User's Guide* for additional information.

Task Profiles

Define the task profiles for:

- Sub-pass Balancing Application Task XXXXXX

  XXXXXX indicates *your* Sub-pass Balancing Application Task
- DKNIDCR
- DKNODCR
- DKNDFTO (optional)

Use these values to define the task profiles:

Task	UOW-Grouping	Grouping Level	User Level
XXXXXX	1	1	0
DKNIDCR	1	1	0
DKNODCR	1	1	0
DKNDFTO (optional)	1	1	0

Important!

For your Sub-pass Balancing Application, you must code *PREVIOUS* under the *DISPATCH* heading.

If the sending bank utilizes electronically transferred DCVs as input to its Outwork DCV Reconciliation, the DKNDFTO task profile is *required*.

DCV Inwork Workflow Detail

This table describes the recommended workflow detail for DCV Inwork Credits and Debits. XXXXXX indicates *your* Sub-pass Balancing Application.

Figure A-1 (Page 1 of 2). DCV Inwork Credits and Debits Workflow Detail

Workflow	Category Desc	Task	Priority	Auto
Credits Only 78-M String	DCV 78-M	DKNHCDM XXXXXX	7 7	1 0
Debits Only 78-M String	DCV 78-M	DKNHCDM XXXXXX	7 7	1 0
Credits Only 79-M String	DCV 79-M	DKNIDCR DKNDFTO DKNICRE	7 7 7	1 1 1

Figure A-1 (Page 2 of 2). DCV Inwork Credits and Debits Workflow Detail

Workflow	Category Desc	Task	Priority	Auto
Debits Only 79-M String	DCV 79-M	DKNIDCR	7	1
		DKNDFTO	7	1
		DKNICRE	7	1

Note: The DKNDFTO task is required if the sending bank is using electronically transferred DCVs as input to its OutWork DCV Reconciliation to avoid the paper capture of the returned DCVs.

DCV Outwork Workflow Detail

The following table describes the recommended workflow detail for DCV Outwork Credits and Debits. `XXXXXX` denotes *your* Sub-pass Balancing Application.

Figure A-2. DCV Outwork Credits and Debits Workflow Detail

Workflow	Category Description	Task	Priority	Auto
Credits Only 78-M String	DCV 78-M	XXXXX	7	0
Credits Only Corrected 78-M String	CORR 78-M	DKNHCDM	7	1
		XXXXXX	7	0
Debits Only 78-M String	DCV 78-M	XXXXXX	7	0
Debits Only Corrected 78-M String	CORR 78-M	DKNHCDM	7	1
		XXXXXX	7	0
Credits Only 79-M String	DCV 79-M	DKNODCR	7	1
		DKNICRE	7	1
Outwork Debits Only 79-M String	DCV 79-M	DKNODCR	7	1
		DKNICRE	7	1

Glossary

This glossary defines important terms and abbreviations used in this manual. If you do not find the term you are looking for, refer to the Index or to the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

A

ABA. American Bankers Association.

ABA number. (1) A numbering system devised by the ABA to provide exact identification of financial institutions. The code structure also identifies the Federal Reserve Bank and branch. (2) The MICR-inscribed field on a US document, containing the financial institution identification number.

account number field. An encoded field, on a check or a deposit slip, that indicates the account held by the drawer of the debit or the recipient of the credit.

adjustment. A change to a credit or debit document that adjusts the balance status of a deposit group (or transaction group).

advice. A letter that is sent to a financial institution or customer from whom checks have been received, advising that errors have been detected in the checks or in the listing that accompanied the checks.

ALS. Application Library Services.

American Bankers Association (ABA). Among the functions of this group is the specification of banking industry standards for US check-handling documents and procedures.

amount due field. This field is on some UK credit documents, typically utility payments, indicating the amount that is due for payment. It might or might not be the same as the actual amount field which will be encoded by the presenting bank when the credit is paid in.

amount field. An encoded field on an item that represents the amount of that item.

Application Library Services (ALS). See *ImagePlus HPTS Application Library Services.*

application tasks. Those application tasks that are delivered as part of the base CPCS-I program product or product feature.

application program task control block (APTCB). A CPCS-I area created by the applications task (DKNATASK) for every active subtask in the system. This area contains operating system control blocks that are related to the subtask; it also contains addresses and constants used by the CPCS-I executive programs.

APTCB. Application program task control block.

assist document (AST). A document that accompanies incoming work and that supplies information about the work. A remittance/kill list is an example of an assist document.

AST. Assist document.

automatic restart. The process of restarting (continuing) an interrupted entry without having to find and rebatch any item.

B

balanced M-string. The M-string that has been balanced by a balancing product. The balanced M-string is denoted by the string name *eeee-p-p1-p2-p3*-99-*t-sss*.

balancing. The act of bringing two sets of related figures into agreement (for example, reconciling accumulated-detail totals and input-control totals).

bank control file (BCF). A CPCS-I data set that contains control information for multiple bank processing.

Bank Giro Credit (BGC). A UK credit document that may be paid in only through a clearing bank. It may be encoded in MICR or in a mixture of MICR and OCR, but the format of the codeline is broadly similar to a check.

base CPCS-I application tasks. See *application tasks.*

basic direct access method (BDAM). An access method used to directly retrieve or update particular blocks of a data set on a direct access device.

batch. The lowest required level that has monetary control established by a control document. See also *Docket Control Voucher.*

batch number. The number that uniquely identifies a specific batch of documents.

batch slip. A level of control for balancing items. See also *batch* and *Docket Control Voucher.*

BCF. Bank Control File.

BDAM. Basic direct access method.

BGC. Bank Giro Credit.

block. (1) A prime-pass control level consisting of one or more batches. In CPCS-I, this control level is used to total multiple batches. A block can also represent work from a specific source. (2) A data-processing term used to refer to a series of logical records stored contiguously on external storage devices. (3) To insert control documents in preparation for a prime-pass sorter run. See also *data preparation.*

block slip. A level of control for balancing batches. See also *block.*

branch separators. A UK term for user control documents used to separate work for different branches in on-us output pockets.

buffer. A main storage area used as a data-transfer area for physical records being read or written.

bundle. A bundle is a set of documents grouped together for processing and prefixed, for control purposes, by slips (for example, batch).

C

capture. (1) To read the codeline that is inscribed on a document. (2) To make a digitized image of a document. In the HPTS system, full-item images can be captured by the Image Capture System attached to the document processor or by a low-speed scanner attached to a workstation.

cash letter summary. In the US, a listing that summarizes kill lists by giving monetary totals and item controls for each kill list. In the UK, this is referred to as a DCV Summary.

CDM. Codeline Data Matching.

CDMP. Codeline Data Matching Prime.

CDMR. Codeline Data Matching Rejects.

check. (UK = cheque) A draft drawn on a financial institution and payable on demand on or after the date indicated.

check number. See *serial field* or *reference.*

Check Image Management System Data Base (CIMS Data Base). A program in ImagePlus HPTS Application Library Services that stores, gets, and manages document images.

cheque. UK spelling of “check.”

CIMS. Check Image Management System. See *Check Image Management System Data Base.*

clearing house. An organization, established by financial institutions in the same locality, through which checks and other instruments are exchanged and net balances settled.

codeline data matching (CDM). A method by which a computer system controls items on a detail level by comparing the internal data records from a previous pass with data that it reads on the current pass.

codeline data matching prime (CDMP). The process of performing codeline data matching during a CPCS-I prime pass. Document codeline data is matched against DEFT data transmitted from another bank or a branch of the processing bank. See also *document-based electronic funds transfer.*

codeline data matching rejects (CDMR). The process of performing codeline data matching on CPCS-I prime-pass rejects. Document codeline data is matched against Prime/HSRR codeline data that has been repaired (for example, in OLRR or HPTS key entry).

codeline data record. See *data record.*

cold start. An initiation of the CPCS-I region that causes the deletion of the previous contents of the mass data set and the control data sets.

complete task status. This indicates that this task processed successfully for this UOW. See also *task status.*

complete UOW status. This indicates that all tasks in the task list processed successfully or had a bypass status.

component. A set of modules that performs a major function within a system; for example, a compiler or a master scheduler.

component internal data. All data accessible to any modules within a particular component, but not accessible to any part of the system outside this component.

concurrent kill. Producing remittance/kill lists for kill pockets in an entry before the entire entry is processed. The concurrent kill feature is available only with subset processing.

concurrent processing. A system where the processing of prime capture work through subsequent processes (such as reject handling, rehandle sorting, or remittance printing) begins before completing capture for the whole entry.

control block. A storage area that a computer program uses to hold control information.

control document. An encoded document that contains control information, such as the total of the checks that the document controls, the source of the checks, and a code that describes the level of control.

control slip. See *control document.*

control total. The total value or item count for a group of documents.

copy library. A library that contains statements to be modified by the user, accessed by the assembler instruction copy, and inserted into some of the CPCS-I programs.

correspondent financial institution. A financial institution that carries a deposit balance for, or engages in an exchange of services with, another financial institution.

CPCS-I. Check Processing Control System International MVS/ESA.

credit. The opposite of a debit. Common examples are deposit slips and utility payments.

cross record. See *XREC.*

cutoff. (1) The financial institution's designated point for balancing or releasing work before processing continues. (2) The designated time after which the financial institution cannot accept work for processing.

cycle. (1) A group of work or an identification of a group of work processed completely as a single entity. (2) A convenient grouping of work. A cycle normally contains a variable number of entries.

D

DASD. Direct access storage device.

data preparation. The preparation of documents for processing by a high-speed check-processing system.

data record. The electronic representation of the codeline captured from a check, deposit, debit, credit, or control document. The electronic representation can include additional data to help identify the record.

data space. An area of virtual storage that a program can ask the system to create. The area's size can range from 4K bytes to 2 gigabytes, according to the program's request. Unlike an address space, a data space contains only data. Program code cannot run in a data space. Unlike data in a Hiperspace, data in a data space is directly addressable.

DCV. Docket Control Voucher.

DCV summary. A listing that summarizes all of the kill bundles in a DCV summary report by giving monetary and item controls for each remittance list. See also *cash letter summary.*

DCV summary report. Report listing the group of items to be delivered to an endpoint. Grouping of the items is usually by kill bundle.

debit. A transaction that increases an asset or decreases a liability. In normal check-collection terminology, a check is considered a debit.

deferred printing. The method by which data is processed, transferred to a storage device, and later printed (as opposed to printing during the processing of data).

DEFT. Document-based Electronic Funds Transfer.

DEFT input. Electronically captured data that supports processing of paper documents in a codeline data-matching prime pass.

deleted UOW status. This indicates that the string associated with this UOW is deleted. No more processing can be done for this UOW.

deposit slip. A document that details a deposit. The total of the deposit is encoded on the deposit slip. A deposit is considered a credit.

DFD. Data Flow Diagram.

direct access storage device (DASD). A device in which access time is independent of the location of the data.

distributed string (D-string). The distribution task reads I-strings that the MICR task created and produces D-strings. Each D-string contains the records that correspond to all of the documents in a given pocket of the document processor.

divider slip. A control document that is used to separate kill bundles during machine sorting. It can also be used to support the resynchronization of codeline data matching during subsequent-pass processing.

Docket Control Voucher (DCV). A UK document used to prefix a batch of documents for exchange between clearing operations. A DCV is considered a Batch Slip by CPCS-I. See also *batch.*

document-based electronic funds transfer (DEFT). The transmission, reception, and processing of codeline data sent or received electronically from another

location together with the documents. The data is used in codeline data matching and reconciliation to reduce rejects and balance work.

document processor. A device that can read encoded characters from documents and sort the documents into multiple pockets.

document processor station. A work station consisting of a document processor and a terminal for operator communication.

drawer. The person on whose account a check is being drawn.

D-string. Distributed string.

E

ECDM. Extended codeline data matching.

enclosed and not listed. A condition that exists when an item is in a batch of checks but is not listed on the incoming kill/remittance list or inscriber tape.

encode. To imprint a MICR field on a document. The CPCS-I database contains the information that is encoded. Synonymous with *inscribe*.

encoder. A machine that encodes or inscribes. Synonymous with *inscriber*.

endorsement. (1) The signature of the endorser; (2) the stamp of a financial institution or company.

endorser. (1) A person or financial institution, other than the maker, who presents a check for payment. (2) A device that stamps an endorsement.

endpoint. The destination of an item (debit or credit).

enhanced reject processing. The pockets used in this processing are alternate reject pockets, eligible to receive a reject item and/or an unencoded reject item. These pockets are defined in the J sort pattern definition record with values of J, E, and U respectively.

entry. A variable number of documents that are processed as a single group of work. Normally consists of a number of blocks and batches.

entry number. The number of the first tracer group within an entry.

EPC. Extended process control field.

ERP. Enhanced reject processing.

error description. The detailed description of an error created, detected, and corrected by the processing financial institution.

exception printing. The printing of only the data that requires action external to a computer.

extended codeline data matching (ECDM). A feature available on the 389x/XP Series document processors. It allows the matching criteria to be changed on a per-document basis (based on the perfectly read fields or on the number of digit errors in a field) and increases the chance of a successful match.

extended process control field (EPC). An optional encoded field that indicates special handling (such as return or truncation).

F

fine-sort. (1) The sorting of items, for example, into account number order for filing. (2) The sorting of items for a single account into serial-number order as a customer service.

fine sort group (FSG). A group of documents that have been block-sorted under CPCS-I for fine sorting. Each FSG has a unique CPCS-I endpoint and does not enter fine sorting until all work for that FSG has been processed through all preceding passes.

flip-flop. An event that occurs when the volume to which you are writing a file becomes full. The writing continues on a new volume and the full volume is backed up.

float. The portion of a financial institution's total deposits, or of a depositor's account, that represents items (for example, checks) in the process of collection.

flow code. A 3-digit number (mnemonic) that represents an ordered list of tasks.

flow control. The pairing of a CPCS-I string with a task list through the specification of sort type, pass-pocket history, string type, and flow code.

FSG. Fine sort group.

full-page printing. A method of page formatting in which items are listed in as many columns as can be contained on the page (for example, the first 50 items in column 1, the second 50 in column 2, and so on).

functional unit of work. This unit of work corresponds to a CPCS-I string or subset string.

funds availability. The portion of the financial institution's total deposits or of a depositor's account that represents items (for example, checks) that have been collected and are now available. This includes cash deposited and checks drawn on the depositor's financial institution.

G

generated total. The total value or item count of checks that are processed by the computer.

H

held task status. This indicates that this task should be the next task to process, but a condition external to CPCS-I must complete first. See also *task status.*

High Performance Transaction System (HPTS). See *ImagePlus High Performance Transaction System.*

high-speed reject re-entry. The re-entering into the document processor of reconditioned documents that have previously been sorted to the system reject pocket (pocket 1-1).

Hiperspace. A range of up to two gigabytes of contiguous virtual storage addresses that a program can use as a buffer. Like a data space, a Hiperspace holds only data, not common areas or system data; code does not execute in a Hiperspace. Unlike data in a data space, data in a Hiperspace is not directly addressable.

holdover. (1) Items that were not processed in time to meet their deadline. (2) Items that are held for the next processing cycle.

HPTS. High Performance Transaction System. See *ImagePlus High Performance Transaction System.*

HSRR. High-speed reject re-entry.

I

image. The captured facsimile (picture) of an item represented in digital form suitable for computer processing and storage, and visual display to an operator.

ImagePlus High Performance Transaction System (HPTS). An IBM system that adds image processing capabilities to document processing.

ImagePlus HPTS Application Library Services. An IBM licensed program that supplies the HPTS system with services such as communication, data-storage management, recognition facilities, data compression, data reconstruction, and device support. The program consists of Image Host Application services, Image Processor Recognition Services, and Image Workstation Application Services.

import/export. The sending of information (export) from one system or application and the acceptance of information (import) by another system or application.

inclearings/inwork. A UK term describing checks and credits drawn on your financial institution. Similar to the term “on-us.”

incoming sequence number. A number that defines the incoming sequence of an item within the input stream. This unique number is associated with the item throughout the whole cycle of computer processing.

input string (I-string). A string of documents created by the MICR task. On each document processor run, an I-string is created. The string includes every document read by the document processor, including control documents and rejected documents. Related information, such as the pocket selected, is also stored in each record. The string also includes internally generated control records.

inscribe. Synonym for *encode.*

inscriber. A machine that encodes and inscribes in a particular format. Synonym for *encoder.*

interbank settlement sheet. A UK interbank report, produced by Inwork DCV Reconciliation, summarizing the Inwork DCV totals and the settlement figure.

Inwork. A UK term for incoming on-us work from other banks or institutions.

Inwork DCV Detail Report. A UK term for a report produced by Inwork DCV Reconciliation for each responding bank listing the DCVs and WDs that are being returned.

Inwork DCV Recapture File. A UK term for a file created by Inwork DCV Reconciliation by recapturing the Inwork DCVs and WDs after balancing. This file is matched against the Inwork DCV Summary File to produce the Inwork DCV Reconciliation File.

Inwork DCV Reconciliation File. A UK term for a file created by Inwork DCV Reconciliation by matching the Inwork DCV Recapture File against the Inwork DCV Summary File.

Inwork DCV Reconciliation Report. A UK term for a report produced by Inwork DCV Reconciliation that lists the free and missing Inwork DCVs detected.

Inwork DCV Summary File. A UK term for a file created by DKNIDCS after the completion of Prime Balancing. It contains details of all DCVs and WDs captured in the Inwork cycle and is input to Inwork DCV Reconciliation.

interface. A named and shared boundary between two functional units, (for example, component interface, subcomponent interface) defined by functional characteristics, or other characteristics, as appropriate.

invocation. Any method of starting a function within a component, subcomponent, or module, such as a direct call with parameters, use of a queue, or event control blocks (ECBs).

inwork. Checks and credits that are drawn on the financial institution that is processing them. Also termed “on-us.”

I-string. Input string.

item. A check, deposit slip, or other machine-readable document.

item-sequence number. A number that defines the sequence of an item within the input stream. This unique number is associated with the item throughout the entire cycle of computer processing.

J

jam. A condition that exists when items form a blockage anywhere in the transport mechanism of a document processor.

JGC. Joint Giro Credit.

job control language (JCL). A control language used to identify a job to an operating system and to describe the job's requirements.

JCL. Job Control Language.

JES. Job entry subsystem.

job entry subsystem (JES). A system facility for spooling, job queuing, and managing input and output.

joggler/jogger. A device that straightens and aligns items before high-speed sorting, principally to line up the lower edge and right side of a group of documents. This device is an integral component of some document processors.

Joint Giro Credit (JGC). A UK credit that may be paid in either through a clearing bank or through a post office. The two JGC types are (1) long joint giro, and (2) short joint giro. The only difference between the two types is that the long version has an Amount Due field and the short JGC does not.

K

kill. To process items to a point where no further distribution is required. See also *remit* .

kill bundle. A group of items in a kill pocket, delineated by divider slips, that forms a batch or remittance to another bank. With concurrent kill, this group can span strings. See also *remittance list.*

kill list. A document that accompanies a kill bundle, listing detail and controls for the items.

kill pass. A pass on which items are distributed to their endpoint pockets.

kill pocket. A document-processor pocket assigned to items that are sent and remitted to another bank or destination without further sorting.

L

legal tender. Any money that must, by law, be accepted in payment of debts. A personal check is not legal tender.

link-edit. To use a linkage editor to create a loadable computer program.

listed and not enclosed. A condition that exists when an item is listed on an incoming remittance/kill list or inscriber tape but is not enclosed in the kill bundle.

logical unit (LU). A port through which a user accesses SNA-network functions to communicate with another user on the network.

low-speed transit. The manual sorting and processing of checks.

LU. Logical unit.

LU 6.2. Logical unit 6.2 protocol.

LU 6.2 protocol. An SNA service that receives requests from users and from the system services control point. This service provides session management and other services for sessions between two logical units.

M

magnetic ink character recognition (MICR). The reading of magnetically encoded data on the 5/8" clear band that runs along the bottom of a document. The MICR system uses ten specially coded digits and four special symbols.

Management Information System (MIS). A DB2 system that maintains data on overall check processing. This is a subcomponent of ImagePlus HPTS Application Library Services (IALS).

manual restart. The process of physically finding and rebatching, before resuming an interrupted entry, the items to be recaptured.

mass data set (MDS). A file that contains records of all active document strings. This file consists of two direct access data sets: a directory index and a data record set.

master list. A list of all items that are read during a computer pass.

MDS. Mass data set.

merged string (M-string). The M-string, produced by DKNMRGE, represents the merging of images from the prime-pass I-string with corrected reject data. Reports that result from the M-string let you reconcile and balance input to ensure that all items were captured.

MICR. Magnetic ink character recognition.

microfilm number. The assigned item number that is also captured on microfilm.

MIS. Management Information System.

misread. A condition that occurs when a document processor interprets a character as a good character other than that which actually appears on the document codeline. Synonymous with *substitution.*

missort. An item that is found in a pocket other than the pocket to which it was sorted. This might be the result of a misread.

M-string. Merged string.

Multiple Virtual Storage (MVS). An operating system that consists of MVS/System Product (MVS/SP)* , MVS/ESA*, and the MVS Data Facility Product operating on a System/370 processor.

O

OCR. Optical character recognition.

OLMS. Online manual split.

OLRR. Online reject re-entry.

online fine sort. A computer-controlled sorting of documents (for example, checks) by either or both the account number and the serial number sequence for filing. This process commonly uses codeline data match techniques.

online manual split (OLMS). The process that sorts reject data from the MDS to produce remittance/kill lists and branch reports in the same sequence as manually sorted rejects.

online reject re-entry (OLRR). Manual entry or correction of MICR data through a display terminal.

on-us. Documents belonging to a bank that are sent to its clearing center from other banks or financial institutions. See also *inwork.*

Optical character recognition (OCR). Character recognition that uses optical means to identify graphic characters.

optional field 1. An optional, encoded field used by some US financial institutions for check truncation. It can also be used for other internal purposes.

out-clearing. A UK term meaning the sorting of documents to external destinations. The US term is *transit*. See also *outwork.*

outgoing sequence number. A sequence number or unique identification assigned to each item, identifying the kill bundle in which the item left the financial institution.

outwork. Documents that when processed leave the bank for collection from other institutions. See also *out-clearing.*

Outwork DCV Detail Report. A UK term for a report produced by Outwork DCV Reconciliation for each responding bank. It is essentially a listing of the Outwork DCV Reconciliation File.

Outwork DCV File. A UK term for a file produced by Remittance (Kill) processing. It is essentially an electronic version of the Outwork DCV Report and is used to power encode DCVs.

Outwork DCV Interbank Settlement Sheet. A UK term for a report produced by Outwork DCV Reconciliation for each responding bank, summarizing the agreed DCV totals and the figure for settlement.

Outwork DCV Recapture File. A UK term for a file created by Outwork DCV Reconciliation by recapturing the DCVs returned by other banks. This file is then

* Trademark of IBM

matched against the Outwork DCV Summary File created on the previous day.

Outwork DCV Reconciliation File. A UK term for a file created by Outwork DCV Reconciliation by matching the Outwork DCV Recapture File against the Outwork DCV Summary File.

Outwork DCV Reconciliation Report. A UK term for a report produced by Outwork DCV Reconciliation for each responding bank listing the missing and free DCVs detected.

Outwork DCV Report. A UK term for a report produced by Remittance (Kill) processing. It is similar to a CPCS-I cash letter and summarizes a number of kill bundles. It is not sent with the documents but is used to manually encode DCVs.

Outwork DCV Summary File. A UK term for a file produced by Remittance (Kill) processing. It contains a record for every Remittance (Kill) bundle processed and is grouped by endpoint within a cycle. It is used as input to Outwork DCV Reconciliation when the DCVs are returned by the responding bank on the following day.

P

pass. A single reading and sorting of a group of checks and control documents on a document processor.

pass-to-pass control. A process that maintains the total amount and item control of a group of documents on subsequent passes, when control has been established on the previous pass.

path. The path of a functional unit of work is the ordered list of tasks processed for the associated CPCS-I string. See also *flow code* and *flow control.*

pending status queue. A first-in-first-out System Manager queue through which CPCS-I applications interface to the System Manager, in sequence, to perform UOW creations, deletions, inquiries, and updates.

piggyback item. An item that was missing from its assigned pocket in a sorter and sorted “free” to an unidentified pocket, as when one document attaches itself to or overlaps another during processing.

pocket 1-1. See *system reject pocket.*

PRAD. Propagation of Adjustments.

presenting bank. A UK term for the bank sending documents and DCVs and requesting funds for the DCVs.

prime pass. The first pass of an entry on a document processor.

printing after the fact. See *deferred printing.*

process control field. Used in the US by the payor bank to know which process applies to each item. In the UK this field is called *transaction code* and is used to identify document types.

proof. Receives checks that come from tellers, mail and night depository, and internal departments of the financial institution. Proof balances transactions and inscribes or encodes the monetary amount in MICR.

proof of deposit. The act of totalling items at the deposit level and ensuring that the total of the credits equals the total of the debits.

propagation of adjustments. The process of ensuring that adjustments made in Balancing and elsewhere are carried forward to kill/remittance and other system output processes.

R

RACF. Resource Access Control Facility.

RBA. Relative block address.

reconcile. To find and correct the cause of a difference between two sets of totals.

reconciliation. See *balancing.*

reconditioning. The process of straightening folded items, inverting upside-down items, flipping reversed items, and removing any residual staples or rubber bands.

reference. A UK term for a field encoded on credit documents, corresponding to the 6-digit Serial field on debits. The Reference field may be up to 18 digits in length and (if printed in OCR) may contain alphanumeric characters.

rehandle pocket. A document processor pocket that receives items for multiple endpoints. Items directed to rehandle pockets are processed again on a later pass.

reject. A document that cannot be read in its entirety by a document processor or that fails certain editing checks. This document is normally directed to a special pocket called a reject pocket.

reject string (R-string). Strings that are created by the online reject re-entry task. Each R-string represents checks that have been re-entered online. R-strings are input to the DKNMRGE task.

relationship. Shows the parent/child hierarchy of units of work.

relative block address (RBA). In CPCS-I, the calculated location of a specific record.

remit. A UK term; to send items to another financial institution.

remittance file. A UK term for an MVS data set that is created by Remittance (Kill) processing. It is essentially an electronic version of the remittance list and may be used to support DEFT input processing at the receiving institution.

remittance list. A UK term for a CPCS-I Kill List that is produced to support negotiation and settlement of a batch of documents prefixed by a DCV. It is used for conventional interchange between clearing operations.

repass. See *rehandle pocket.*

rerun. A group of items that are sorted into a pocket on one pass and later brought into a document processor for more sorting.

Resource Access Control Facility (RACF). An MVS security subsystem that determines the validity of each operator's ID password and that controls operator access to application tasks and transactions.

responding bank. A UK term for the bank making payment on documents/DCVs received from the presenting bank.

restart. An initiation of the CPCS-I system after a system failure. A restart is generally used to start the system (after an abnormal end of a task) to cause the executive routines to re-establish the system to the status that existed before the failures.

restart buffer. An area where records are stored in an IBM 389x/XP Series document processor during online operations until they are sent to the host. The buffer is accessed during automatic restart.

resynch document. A control document used in DEFT processing to match DEFT data to the documents currently being processed on Prime and also used to separate and identify kill bundles on output.

return item. A check that is not honored by the maker's financial institution and that is returned to the depositor's financial institution.

routing/transit number field. An encoded check field that represents the financial institution on which the check is drawn. In the UK, this is referred to as the *Sort Code*.

R-string. Reject string.

S

SCI. Stacker Control Instruction.

scroll. The ability to use the DKNSCRL application to page through or look at the scroll data set. This data set includes supervisor terminal messages and DKNATASK log messages.

SDE. String directory entry.

separator. See *divider slip.*

sequence number. A number, assigned to a document, that uniquely identifies its position in a group of incoming or outgoing work.

serial field. A UK term for the 6-digit field, (equivalent to the check number in the US), which is normally the serial number of a check. On credits, the same field is called a Reference and may be up to 18 digits in length.

settlement. The act of bringing sets of related figures from two financial institutions into agreement. Adjustments are made to offset the differences.

simulated sorter. A CPCS-I facility that allows a user to run MICR, using an input file without a physical sorter.

slip. A slip is a control document used to prefix bundles for control purposes.

SMOF. System Manager Online Functions.

SNA. Systems Network Architecture.

sort code. A UK term for the field (equivalent to the routing transit field in the US) which identifies the bank and branch to which a debit or credit item belongs. It is in the format *BB-bbbb*, where *BB* identifies the bank, and *bbbb* identifies the branch within that bank. It may be printed in MICR (on checks and some credits) or in OCR (on some credits). If printed in MICR, the two parts of the field are separated by a dash (SS4).

sorter station (also document-processor station). A work station consisting of a document processor and a terminal for operator communications. Synonym for document-processor station.

sort pattern. A table used by the sort routine to determine the pocket to which a check is to be directed.

sort-pattern definition file. A collection of records that contains control information that MICR in CPCS-I uses to set up and control document sorting; it also contains data about endpoints.

sort routine. A time-dependent routine that does all processing required to direct a document to a specific document processor pocket.

sort program. A routine that performs all processing required to select a document to a pocket.

spool data set. A data set used to store printed output lines. Each spool (Simultaneous Peripheral Operations On-Line) data set is written by a CPCS-I application task and is read by the CPCS-I output writer as it is being printed.

SSB. String status block.

SSM. String segment map.

Stacker Control Instruction (SCI). SCI is the name of a language used to write programs to control the sorting of documents on a 389x document processor.

statistics. The processing of unit-of-work (UOW) data through a statistical program such as the ImagePlus Application Library Services (MIS) system. This term can also refer to the processing of unit-of-work data through a user-written statistical program.

string. The data records representing a group of items, for example, an I-string, a D-string, or an M-string. See related definitions for details.

string segment map (SSM). One of three types of segment maps in CPCS-I. Each string in the system is associated with a string segment map. Each bit in a map represents a segment of direct access storage.

string status block (SSB). This CPCS-I control block is maintained by the MDS programs for every open string.

STV. Subtotal voucher.

subcomponent. Functional subset of a component where subsetting is appropriate based on data use, logic flow, or other factors relating to modules.

subcomponent internal data. All data accessible to any modules within this particular subcomponent, but not accessible to any part of the system outside this subcomponent.

subsequent pass. A pass on which previously sorted items are resorted for further distribution.

subset. A defined portion of an entry, indicated by one or more tracer groups.

subset processing. Processing a portion of an entry beyond the document-entry step before the whole entry is run through the document processor.

subset string. A predefined group of data records that represents a portion of the physical items in an entry. A subset string can contain multiple tracer groups.

substitution. See *misread.*

subtotal voucher (STV). An optional UK document that can be inserted into a batch of documents to mark the point at which a cumulative subtotal is printed on the accompanying remittance list.

supervisor. (1) An MVS term used to refer to the system nucleus in internal storage. (2) A person responsible for operation of a financial institution area.

supervisory terminal. A special terminal or operating mode used in CPCS-I.

System Manager. A subsystem of CPCS-I that directs and controls the operations.

System Manager Online Functions (SMOF). A set of application-level tasks that monitor and modify the queues and databases of System Manager.

system reject pocket. The first physical pocket on the document processor. It is used by CPCS-I to hold machine and user-selected rejects.

System Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration of, networks.

T

tab key. A keyboard function key. The tab key causes the cursor to position to the next colon on the screen or to the top of the screen.

task. A CPCS-I application or function. User-written tasks must be in the CPCS-I BLDL list.

task list. The ordered list of tasks to be performed for a unit of work. It is determined by selecting the flow code for a given flow control record.

task status. A representation of what will happen, what is happening, or what happened during processing of this unit of work. Can be pending, ready, or complete. See related definitions for details.

total system. A system in which the computer is used for all phases of an operation.

tracer. A check-processing document used to provide pass-to-pass control.

tracer group. A grouping of documents between sets of tracers for control purposes. If subset processing is

in operation, this tracer group normally becomes a unit of work that can be processed independently of other units of work within that entry.

tracer ID. The tracer group and slip numbers corresponding to a tracer slip.

transaction code. A UK term for the 2-digit field that identifies debit, credit and control document types (similar to the Process Control Field in the US). A blank transaction code is a valid identifier for a check.

transit. The sorting of checks to external destinations. See also *out-clearing* and *outwork.*

U

unit of work (UOW). A logical entity that the System Manager uses to track a piece of work through CPCS-I. It can be informational or functional. See also *functional unit of work.*

UOW. Unit of work.

UOW status. This status represents the state of a unit of work and its associated string. Can be pending, ready, or complete.

V

Virtual Storage Access Method (VSAM). An access method for indexed or sequential processing of fixed or variable-length records on direct access storage devices.

Virtual Telecommunications Access Method (VTAM). A set of programs that control the communication between terminals and application programs.

VSAM. See *Virtual Storage Access Method.*

VTAM. See *Virtual Telecommunications Access Method.*

W

warm start. An initiation of the CPCS-I system, causing the contents of the MDS and the control data sets to be retained. A warm start is generally used for restarting CPCS-I after a normal ending.

WD (wrongly delivered). A UK term for items (debits or credits, not DCVs) that have been dispatched to the wrong bank. They are returned rather than redirected.

XREC. The dynamic control block that maps the string data at various points in the system. It cross-records or maps the string as it is in the data base, or as it is in the data space.

work. Any document or group of documents that CPCS-I processes.

work flow. An ordered list of tasks for a specific CPCS-I string. Each CPCS-I string must have a work flow.

Z

zero-balancing. The procedure that ensures that generated totals for a group of items plus any documented errors minus the control total equals zero.

Numerics

3890/XP Document Processor. A document processor in the 3890/XP Series of document processors that can read and sort documents at a rate of up to 2400 documents per minute.

3890/XP Series document processors. A series of high-speed document processors that can read and sort up to 1000, 1700, or 2400 documents per minute. These document processors include the IBM 3890/XP Document Processor, the IBM 3891/XP Document Processor, and the IBM 3892/XP Document Processor.

3891/XP Document Processor. A document processor in the 3890/XP Series of document processors that can read and sort documents at a rate of up to 1700 documents per minute.

3892/XP Document Processor. A document processor in the 3890/XP Series of document processors that can read and sort documents at a rate of up to 1000 documents per minute.

3892/XP Power Encoder Feature. An optional device that can be attached to the 3892/XP Document Processor to encode the MICR codeline field on a document.

99 M-string. See *balanced M-string.*

Bibliography

The publications in this bibliography contain information related to CPCS-I.

ACF/VTAM Publications

The following publications are related to the ACF/VTAM Version 3 Release 4 product:

IBM ACF/VTAM Programming, SC31-6436

IBM Planning and Reference for NetView, NCP, and VTAM, SC31-6124

IBM VTAM Programming for LU 6.2, SC31-6437.

Document Processor Support Publications

The following publications are related to document processor support:

IBM 3890/XP Series Document Processor General Information, GA34-2012

IBM 3890/XP Series Programming Guide, GC31-2662

IBM 3890/XP Series SPXServ Reference, GC31-2704

IBM 3890/XP MVS Support and 3890/XP VSE Support Program Reference, SC31-2654

High Performance Transaction System Publications (Version 1)

The following publications are related to the IBM ImagePlus High Performance Transaction System (Version 1):

IBM ImagePlus High Performance Transaction System General Information Manual, GC31-2706

IBM ImagePlus High Performance Transaction System Application Library Services Programming Reference, SC31-2794

IBM ImagePlus High Performance Transaction System Installation Guide, SC31-3943

High Performance Transaction System Publications (Version 2)

The following publications are related to the IBM ImagePlus High Performance Transaction System (Version 2):

IBM ImagePlus High Performance Transaction System Planning Guide, GC31-4005

IBM ImagePlus High Performance Transaction System Installation Guide, GC31-4006

IBM ImagePlus High Performance Transaction System Application Library Services Operations Guide, SC31-4010

IBM ImagePlus High Performance Transaction System Application Library Services Programming Guide, SC31-4011

IBM ImagePlus High Performance Transaction System Application Library Services Programming Reference, SC31-4012

MVS Publications (Version 5)

The following publications are related to MVS:

IBM MVS/ESA Programming: Extended Addressability, GC28-1468

IBM MVS/ESA Installation Exits, SC28-1459

IBM MVS/ESA Initialization and Tuning Reference, SC28-1452

IBM MVS/ESA JCL User's Guide, GC28-1473

IBM MVS/ESA System Messages, Volume 1 (ABA-ASA), GC28-1480

IBM MVS/ESA System Messages, Volume 2 (ASB-EWX), GC28-1481

IBM MVS/ESA System Messages, Volume 3 (GDE-IEB), GC28-1482

IBM MVS/ESA System Codes, GC28-1486

IBM MVS/DFP Version 3 Release 3: Utilities, SC26-4559

To help you find other MVS library references for various release levels, check:

MVS/ESA Library Guide for MVS/ESA System Product Version 4, GC28-1601

MVS/ESA System Product Library Guide with JES2, GC28-1423

RACF Publications

The following publications are related to RACF:

IBM Resource Access Control Facility Command Language Reference, SC28-0773

IBM Resource Access Control Facility Security Administrator's Guide, SC28-1340

IBM System Programming Library: Resource Access Control Facility, SC28-1343.

IBM Resource Access Control Facility Master Index, GC28-1035

CPCS Enhanced System Manager Publication

The following publication is related to Enhanced System Manager:

IBM Check Processing Control System: Enhanced System Manager User's Guide, SC31-4002

Index

Numerics

A

B

C

D

E

F

G

S

T

U

V

W

Communicating Your Comments to IBM

Check Processing Control System
International MVS/ESA
Installation Guide
Release 1

Publication No. GC31-2942-01

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

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

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

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

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments electronically, use this network ID:

  Internet: www.ibm.com/Products/CPCS

Make sure to include the following in your note:

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

Readers' Comments — We'd Like to Hear from You

Check Processing Control System
International MVS/ESA
Installation Guide
Release 1

Publication No. GC31-2942-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

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

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
GC31-2942-01

Cut or Fold
Along Line

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

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

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

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Payment Solutions
Department 58G, MG96/204
8501 IBM Drive
Charlotte NC 28262-8563

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

Cut or Fold
Along Line

Program Number: 5799-FKT

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

GC31-2942-01