Check Processing Control System

Customization Guide

Release 11

SC31-2853-06

Check Processing Control System

Customization Guide

Release 11

SC31-2853-06

Note!

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

| **Seventh Edition (November 1999)**

| This edition is a revision of SC31-2853-05. This edition applies to Version 1 Release 11 of the Check Processing Control System
| licensed program (Program No. 5734-F11). This publication is current as of PTF numbers UQ34786 and UQ34787. Each change is indicated by a vertical line (revision bar) in the left margin.

For automatic distribution of updates to the documentation for this product, the user must be enrolled in the System Library Subscription Service (SLSS). See your marketing representative for enrollment procedures.

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 given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader’s comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Payment Solutions, Department 58G, MG96/204, 8501 IBM Drive, Charlotte, NC 28262, U.S.A. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

Figures

Notices

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact: IBM Corporation, 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 Commercial Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

Programming Interface Information

This guide is intended to help the customer install, customize and maintain the IBM Check Processing Control System (CPCS).

This guide also documents General-Use Programming Interface and Associated Guidance Information.

General-Use programming interfaces allow the customer to write programs that obtain the services of the Check Processing Control System.

General-Use Programming Interface and Associated Guidance Information is identified where it occurs, by an introductory statement to the chapter or section.

Year 2000 Compliance

IBM announces that the Check Processing Control System, Version 1 Release 11, at PTF numbers UN99696 and UN99801, supports Year 2000. This IBM product, when used in accordance with its associated documentation, is designed to be capable of correctly processing, providing, and receiving date data within and between the twentieth and twenty-first centuries. This has been done by allowing the user to set the date format as a default throughout the system.

In the complex global computing environment that we have today, this IBM product's support for Year 2000 is, of course, dependent on the capabilities of all

the other products that are working together (for example, hardware, software, and firmware) to properly exchange accurate date data.

Trademarks

The following are trademarks of International Business Machines Corporation in the United States and/or other countries:

IBM®, ACF/VTAM®, COBOL/370™, ES/9000®, MVS/DFP™, MVS/ESA™, MVS/SP™, RACF™, System/370™, VTAM®, 3090™.

Other company, product, and service names may be trademarks or service marks of others.

About This Book

This book supplies customization information for personnel who install and maintain the IBM Check Processing Control System (CPCS).

This book provides information about:

- CPCS operating requirements in an MVS environment
- Installation and generation procedures for tailoring CPCS to your requirements
- Loading and editing profiles for controlling the behavior of CPCS.
- Creating user exits and other user programming requirements.

Who Should Read This Book?

System programmers, application programmers, and operational personnel use this book, the *CPCS Installation Guide*, and the *CPCS Terminal Operations Guide*, in order to install, maintain, and run CPCS.

Programmers and operational personnel responsible for the ImagePlus High Performance Transaction System might also need to refer to this book.

How Is This Book Organized?

This book contains the following chapters:

Chapter 1, “System Programming and Operations,” describes CPCS configurations, software requirements, data set allocation, and operational considerations.

Chapter 2, “Installation and Generation Procedures,” presents information on Master task generation, MICR task generation, activating Logging, creating an expanded MDS environment, Virtual Telecommunication Access Method (VTAM) installation configuration, and other installation tasks.

Chapter 3, “System and Application Profiles,” describes how to edit and load the profile members that control the behavior of CPCS and of its various application tasks.

Chapter 4, “User Programming Requirements and Exits,” describes MICR and Online Reject Reentry (OLRR) stacker-select routines, MICR user exits, extract programming, and Resource Access Control Facility (RACF) or user-defined security options.

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

Related Publications

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

- *CPCS General Information*, GH20-1008

  This manual gives a general introduction to the Check Processing Control System (CPCS). It describes various characteristics and advantages of CPCS and the hardware and software requirements for operating CPCS. It also discusses CPCS support of the IBM 3890 Document Processor and the 3890/XP Series document processors, along with some of the features of these processors.

- *CPCS Installation Guide*, GA34-2178

  This guide describes the steps necessary for using the IBM System Modification Program Extended (SMP/E) procedures to install CPCS software. It also provides installation procedures for generating CPCS modules and creating operational data sets. It provides a set of sample problems to test and verify operations after CPCS installation.

- *CPCS Online Adjustments Guide*, GC31-2723

  This guide provides the program descriptions and terminal operation instructions for online adjustments. It includes information about customization, system and user requirements, the user adjustment-code data set, the adjustment-record formats, and sample reports.

- *CPCS Programming and Diagnostic Guide*, SC31-2854

  This guide contains guidelines for CPCS programmers, including descriptive information about application-program processing, problem analysis and documentation procedures, and CPCS module descriptions.

- *CPCS Terminal Operations Guide*, SH20-1229

  This guide describes the CPCS tasks and task initiation formats, and explains how to perform these tasks for the CPCS operators. It also explains application-task commands and supervisor commands. CPCS messages, which used to appear in this book, are now in *CPCS Messages and Codes*.

- *CPCS Messages and Codes*, SC31-4004

  This manual contains terminal and supervisor messages, their responses, and return code information for CPCS application tasks.

- *CPCS Enhanced System Manager User’s Guide*, SC31-4002

  This manual contains the guidelines for the CPCS personnel who use the Enhanced System Manager subsystem. It explains the Enhanced System Manager’s interface functions, online functions, and processing. This publication is shipped with the Enhanced System Manager (ESM) product.

- *CPCS Master Index*, SC31-2857

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

Summary of Changes for SC31-2853-06

- **PTF Number:** This publication is current as of PTF numbers UQ34786 and UQ34787.
- **CPCS Password Protection:** CPCS provides password protection to ensure it can be executed only for customers who are registered with IBM. IBM does provide a preset grace period for CPCS customers; during this grace period, CPCS can be started without a password. However, once this grace period expires, CPCS cannot be started without a valid password. Customers must use this grace period to contact IBM and to obtain a valid CPCS password.
- **System Manager:** Only Enhanced System Manager (not System Manager) is supported by CPCS Version 1 Release 11.
- **Subset Processing:** All subset processing has been disabled from CPCS; however, all subset processing features have not been removed from CPCS, for example, in screens, settings, etc. All these features will be removed in the near future.
- **CPCS Transaction Charging:** The CPCS License Agreement requires that any transaction that a customer introduces into CPCS must now use the transaction charging output modes when writing the transactions to the CPCS mass data set. The output modes and the transaction charging guidelines are detailed in the *CPCS Programming and Diagnostic Guide*, Chapter 2, “Accessing the Mass Data Set and Its Index.”
- **Sample Problems:** All the sample problems in the *CPCS Installation Guide* have been rewritten so the sort patterns fit on a six-pocket sorter.
- **Exit Point for MTASK:** An exit point has been added for MTASK (DKNMTASK_MICR_ABEND_EXIT0100).
- **Electronic Cash Letter Support:** A new module (DKNETCSH) creates electronic cash letters. Three new user exits support this function.
- **Electronic Transaction Data Matching:** This feature allows an institution to match the codelines of items from strings within a CPCS environment. The task matches strings created through a CPCS MICR task against strings that were either created as part of the initial MICR capture or strings that were imported to CPCS using the electronic transaction import process.
- **System Profile DKNPRCVY:** This profile is the Logging system configuration that contains all the logging parameters.
- **Changes to System Profile DKNPEXIT:** More user exits were added.

Summary of Changes for SC31-2853-05

Extended String Support: CPCS supports strings with records having one or more associated user areas.

Electronic Transaction Input/Output: The term ETIO refers to the importing and exporting of electronic transactions. It includes the following new CPCS features: electronic transaction output (ETOT), electronic transaction input (ETIN), and electronic transaction input/output display (ETIO).

Electronic Transaction Output: The ETOT task allows a financial institution to export strings from CPCS to a sequential data set (known as transmission data

sets). Strings can be extracted either individually, by endpoint, or can be grouped by using Enhanced System Manager.

Electronic Transaction Input: The ETIN task allows a financial institution to create strings within CPCS from a transmission data set. The electronic records can be in any format; consequently, the import process can load transmission data sets from various sources.

Electronic Transaction Input/Output Display: The electronic transaction input/out (ETIO) task allows a financial institution to display online the status of either electronic control file records or electronic string file records. The institution may then display and update or delete records on either of these files.

Electronic Transaction Utility: The electronic transaction utility (ETUT) task allows the loading and unloading of both standard strings and extended strings for testing and maintenance purposes.

Summary of Changes for SC31-2853-04

The main changes for this revision are:

Merge Before Main Support: Dividers can optionally precede kill bundle items in a capture. With a Sort Pattern Definition option, a merge feed document can be sprayed to each kill pocket before any main hopper item is processed. Also, when this feature is active, CPCS post-capture tasks associate kill bundles to their preceding dividers. Merge-Before-Main is only supported for expanded-sort types using stacker-select routines that include PROLOGX.

User Exit Facility: The CPCS user exit facility (UEF) is designed to isolate CPCS programs from user exit processing. To use this facility, a CPCS program simply calls the user exit manager (UEM) at a defined exit point, and UEM does the rest.

User Area Manager: The CPCS User Area Manager (UAM) isolates user data from CPCS control blocks and other data. User data may be added to applicable control blocks and processed through UAM.

System Manager Module Name Changes: All system manager module names changed from “DKNSM*” to “DKNZM*”.

New MDS Exit Point: The mass data set FREE exit point (MDS_FREE_EXIT) allows multiple exits to be invoked during MDS string FREE processing. You may use these exits to deny a FREE SPACE request.

DKNPEXIT Profile Member: The DKNPEXIT profile member is a system profile data set that contains records that are used to specify user exits to be run at the specific exit point with which they are associated. More than one exit may be specified for a single exit point, and the exits are called in the order specified in this profile member.

Feature Disable/Disengage Option: If selected features have been initialized in the IREC and are subsequently disabled, you can have the sorter issue a disengage.

Summary of Changes for SC31-2853-03

The main changes for this revision are:

Application Profiles: The Application Profile (DKNAPPL) data set's members are profiles that each control the behavior of a different CPCS application task.

Summary of Changes for SC31-2853-02

The main changes for this revision are:

Year 2000 Changes:

- **Date Customization:** CPCS now provides the ability to specify a date format at the system level (as a default). This format is propagated throughout CPCS in reports, screens, and in data sets.

  For more information regarding CPCS code compliance with the Year 2000, see "Year 2000 Compliance" on page vii. For a summary of Year 2000 changes, refer to the special appendix on this subject in the *Programming and Diagnostic Guide*.

Enhanced Prime: CPCS supports the capture of multiple entries on a single prime pass without the use of subsets.

Task Groups: This enhancement allows multiple BLDL tasks to be grouped for performance tuning.

System and Application Profile Data Sets: These data sets are used to pass information to programs now and in the future. These profiles contain configuration and run-time option information for the CPCS system and application programs.

MICR JAM Enhancement: You can configure the MICR task to have a refreshed enhanced jam screen displayed at the end of a runout on the 3890/XP.

Sequence Number Assignment: When the **P** record in the sort pattern specifies an **XF** sort, the item sequence number is returned to CPCS from the 3890/XP for each item processed.

PTF Numbers: This publication is current as of PTF Numbers UN99696 and UN99801.

New Web Site: Visit us at our new web site:
www.ibm.com/Products/CPCS

Enhanced System Manager Support: CPCS supports the IBM Enhanced System Manager feature, which provides workflow management functions including task starting (based on workflow, time of day, after CPCS End Cycle, after Cold Start, after Warm Start, etc.), task tracking (auto-started and manual, Task Suppression (deadline management), Unit of Work (UOW) functions, and automatic generation of workflows.

Summary of Changes for SC31-2853-01

The main change for this revision is:

Enhanced System Manager Support: CPCS supports the IBM Enhanced System Manager feature, which provides work scheduler functions including task starting (based on workflow, time of day, after CPCS End Cycle, after Cold Start, after Warm Start, etc.), task tracking (auto-started and manual), Unit of Work (UOW) functions, and automatic generation of workflows.

Summary of Changes for CPCS Release 11

The new and enhanced functions in this release are in the following areas:

Automatic Restart: When a host or link failure occurs, CPCS restarts interrupted item-capture operations without the physical intervention of operators. The 3890/XP control program maintains a restart buffer where it stores a copy of the records that it sends to the host. When it is recovering from a host or link failure, CPCS retrieves the records contained in the restart buffer and replaces the intermediate buffer records that were lost when the failure occurred.

Blocked BDAM: To use disk storage space more efficiently, you can change the basic direct access method (BDAM) files that are used in CPCS to a blocked format. This capability can improve processing time, especially during startup, when initialization and compression of files occur.

Data-Set Duplexing: CPCS maintains the integrity of all data sets through the data-set duplexing (DUPLEX) function. CPCS duplexes all data sets that are critical to its operation. If a disk failure occurs, you can re-initialize or re-create duplexed data sets through predefined procedures.

Enhanced Logging: CPCS logging has some significantly enhanced features for the automatic recovery of mass-data-set strings. These features include:

- Tracer data-set recovery.
- Automated tracking (history) of mass-data-set strings and logging data sets.
- Elimination of dedicated tape drives.
- Automated full mass-data-set recovery. (Recovery automatically determines which strings should be recovered.)
- Recovery of strings with different mass-data-set definitions. (Recovery performs data conversions to match the existing mass-data-set definition.)
- Recovery of multiple strings with one pass of the logging data set.
- Selective recovery of strings by a generic search capability.

Enhanced Reject Processing: CPCS provides for two types of reject pockets: system reject pockets and user-defined reject pockets.

Expanded Document Processor Read Record: The 3890/XP Series document processors give your sort program access to a larger read record. You can now expand the read record to contain as many as 12 header bytes and 244 data bytes. CPCS supports 15 logical fields, 14 of which are available to you. The document processors write data from the code line into the first seven fields. CPCS uses

field 8 during code-line data-match processing. Your sort program can set the remaining seven fields. For example, you can use the additional fields for:

- The depositor's account number
- An alternate account-numbering system
- A pass-pocket history of the pocket used on each pass
- Optical character recognition (OCR) data capture.

Full Document Image Capture: When you add the IBM 3897 Image Capture System to your 3890/XP Document Processor, CPCS can perform full document image capture. This system, when supported by the IBM ImagePlus High Performance Transaction System (HPTS) Application Library Services licensed program, lets you capture an image (both front and back) of each item in an entry. These images can assist in reject repair. You can also use them for amount capture, eliminating the need to encode amounts before entry. You can store the images for later use in back-end systems, such as exception-item processing, cycle sorting, and statement assembly.

Merged String Distribution: The merged string (M-string) distribution module (DKNMDIS) enables CPCS to distribute M-strings.

Online Adjustments: The Online Adjustments function lets you correct mass-data-set errors found during the CPCS reconciliation process. This function processes updated M-string information to provide a balanced and accurate file for analysis and account posting.

The Online Adjustments function includes the following modules:

- Adjustment entry
- Adjustment listing
- Trial balancing.

Power Encoding: CPCS supports the IBM 3892/XP Power Encoder feature. With the Power Encoder feature, the 3892/XP can encode up to 500 documents a minute. It encodes the amount data on your checks and the entire MICR code line for MICR rejects. To ensure accuracy, the Power Encoder feature compares the code-line data from the prime pass with the data on the check that it is encoding. The Power Encoder verifies the data and stops processing when it reaches a set number of consecutive discrepancies. You determine this number during installation.

Enhanced Prime Capture: CPCS supports the capture of multiple entries on a single prime pass.

Chapter 1. System Programming and Operations

Overview

This chapter contains information about establishing and using the IBM Check Processing Control System (CPCS) operating environment and includes information about:

- System programming requirements, including system configuration
- CPCS startup and shutdown
- Operational considerations
- External storage requirements
- Data-set recovery procedures
- CPCS-supplied command procedures.

Important!

The job control language (JCL) referenced in this chapter represents systems and procedures that are tested at the time this book was published. You must customize most, if not all, JCL that is supplied with CPCS to meet user and installation requirements. For the most current information, refer to the files on the installation tape.

System Programming Requirements

CPCS supports document processors in the following modes:

3890s Channel-attached

3890/XPs Channel-attached or attached on a token-ring network, using the logical unit (LU) 6.2 protocol

3891/XPs and 3892/XPs
Attached, using the LU 6.2 protocol, through a synchronous data link control (SDLC) or a token-ring network.

CPCS operates only on Multiple Virtual Storage (MVS) systems with extended architecture.

For extensive coverage on both the hardware and software requirements for CPCS, refer to the *CPCS Program Directory*.

Minimum System Configuration

CPCS requires, as a minimum hardware configuration, a System/370 processor operating in extended architecture mode with at least 4 megabytes (8MB recommended) of main storage. In addition to the extended architecture requirement, the following features and input/output (I/O) devices are necessary:

- Two direct access storage devices, with sufficient space for the CPCS data sets.
- One 3270 series display terminal or an equivalent device.
- One 3890, or one document processor from the 3890/XP Series, with the item numbering and endorsing feature. The document processor should also include the microfilming feature.

  Note: The standard CPCS mass data set configuration stores the routing-transit (R/T) number as 8 digits. If a 9 digit, all numeric R/T number is

used, only the first 8 digits are stored. RPQ S00391 converts 9-digit R/T numbers to 8 digits and a dash for use by CPCS. This RPQ is for the 3890 Document Processor, it is not required for 3890/XP Series document processors. You can also code an SCI program to accomplish this change or expand the R/T field in the mass data set to accommodate 9 digits.

Sample Configuration

Correct system configuration depends upon each user's volume of work, processing techniques, and performance requirements. The following configuration information should serve only as a guide. Your IBM marketing representative can help you find further assistance with detailed configuration planning.

Quantity	Hardware
One	4381 Processor, Model Group 2 (8 megabytes)
One	3811 Printer Control Unit
Two	3203 Printer Model 5s
Two	Document processors with item numbering, endorsing, and microfilming features
Two	3274 Display Control Unit Model 31Ds
One	3287 Printer Model 2
Ten	3278 Display Terminal Model 2s
Three	Direct Access Storage Devices
One	3880 Storage Controller Model 1
One	3803 Tape Controller Model 2
Four	3420 Magnetic Tape Unit Model 3s

CPCS Startup and Shutdown

This section describes how to start and stop CPCS. It highlights the specific types of startup procedures. It also includes instructions on restarting the system, and descriptions of the job control language (JCL) parameters and statements supplied in the sample JCL on the installation tape.

CPCS Password Protection

CPCS provides password protection to ensure it can be executed only for customers who are registered with IBM. IBM does provide a preset grace period for CPCS customers; during this trace period, CPCS can be started without a password. However, once this grace period expires, CPCS cannot be started without a valid password. Customers must use this grace period to contact IBM and obtain a valid CPCS password.

Starting CPCS

CPCS enters MVS as a standard job with JCL and start procedures. Once started and initialized, CPCS runs primarily in response to input commands from the CPCS display terminals at the item-processing site. System operation requirements consist only of responding to tape handling requests and CPCS error conditions. After CPCS completes initialization, the CPCS VTAM logo appears on terminals that are defined with the operand CPCS=YES coded in the VNODE macro. CPCS is now ready to process any start requests that are entered through the display terminals.

Figure 1-1 on page 1-6 summarizes the effect of various start parameters on CPCS data sets. “CPCS Startup JCL Definition” on page 1-7 provides a detailed description of the CPCS startup JCL.

Figure 1-1. Effect of CPCS Start Parameters on Critical Data Sets

Start Parameters			Critical Data Sets							
STYPE	**FTYPE**	**CTYPE**	**Mass Data Set**	**Index Data Set**	**Tracer Data Set**	**Kill Bundle Data Set**	**Microfilm Data Set**	**Scroll Data Set**	**Spool Checkpoint Record**	**Cycle Data Set**
Omitted (defaults to WARM)	Omitted (defaults to NOFMT)	Omitted	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Compresses the data set	Compresses the data set	Does not clear the data set	Does not clear the checkpoint record	Does not clear the data set
COLD	FMT[1]	Omitted (assumes CKPT)	Formats the data set and deletes all SDE data	Clears the data set	Clears the data set	Clears and reformats the data set	Clears and reformats the data set	Clears the data set	Clears the checkpoint record	Clears the data set
COLD	NOFMT	Omitted (assumes CKPT)	Does not format the data set	Clears the data set	Clears the data set	Clears and reformats the data set	Clears and reformats the data set	Does not clear the data set	Clears the checkpoint record	Clears the data set
COLD	SCFMT	Omitted (assumes CKPT)	Does not format the data set	Clears the data set	Clears the data set	Clears and reformats the data set	Clears and reformats the data set	Clears the data set	Clears the checkpoint record	Clears the data set
WARM	FMT	Omitted	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Compresses the data set	Compresses the data set	Does not clear the data set	Does not clear the checkpoint record	Clears the data set
WARM	FMT	CKPT	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Compresses the data set	Compresses the data set	Does not clear the data set	Clears the checkpoint record	Clears the data set
WARM	NOFMT[1]	Omitted	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Compresses the data set	Compresses the data set	Does not clear the data set	Does not clear the checkpoint record	Clears the data set
WARM	NOFMT[1]	CKPT	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Compresses the data set	Compresses the data set	Does not clear the data set	Clears the checkpoint record	Clears the data set
WARM	SCFMT	Omitted	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Compresses the data set	Compresses the data set	Clears the data set	Does not clear the checkpoint record	Clears the data set
WARM	SCFMT	CKPT	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Compresses the data set	Compresses the data set	Clears the data set	Clears the checkpoint record	Clears the data set
REST	FMT	Omitted	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Does not compress the data set	Does not compress the data set	Does not clear the data set	Does not clear the checkpoint record	Clears the data set
REST	FMT	CKPT	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Does not compress the data set	Does not compress the data set	Does not clear the data set	Clears the checkpoint record	Does not clear the data set
REST	NOFMT[1]	Omitted	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Does not compress the data set	Does not compress the data set	Does not clear the data set	Does not clear the checkpoint record	Does not clear the data set
REST	NOFMT[1]	CKPT	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Does not compress the data set	Does not compress the data set	Does not clear the data set	Clears the checkpoint record	Does not clear the data set
REST	SCFMT	Omitted	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Does not compress the data set	Does not compress the data set	Clears the data set	Does not clear the checkpoint record	Does not clear the data set
REST	SCFMT	CKPT	Sets open I-strings and R-strings to restart mode	Does not clear the data set	Does not clear the data set	Does not compress the data set	Does not compress the data set	Clears the data set	Clears the checkpoint record	Does not clear the data set

1 If you omit the FTYPE parameter, the system defaults to this setting for this STYPE parameter. *If you specify the FTYPE parameter, you must specify the STYPE parameter*.

CPCS Startup JCL Definition

The JCL for starting CPCS (member DKNJRUN in CPCS.V1R11.CTRL) is a the basis for the descriptions of the CPCS start parameters, device specification, file allocation and initialization, and recovery procedures in this section.

You must preallocate data sets that are assigned a disposition (DISP) of share (SHR) or old (OLD). The space allocations are enough for test purposes only. Each CPCS installation must perform the detailed calculations for file allocation based on the expected volume of work and retention periods.

Execute Statement

```
//CPCS EXEC PGM=DKNMTASK,PARM='STYPE,FTYPE,CTYPE,ESM',
//     TIME=1439
```

The CPCS EXEC statement must contain the following positional parameters:

STYPE Start type defines how CPCS is started.

FTYPE Format type defines how the mass data set (MDS) is formatted and whether the scroll data set is cleared.

CTYPE Checkpoint type defines whether the checkpoint record is cleared.

ESM Start type for Enhanced System Manager

Figure 1-1 on page 1-6 contains a description of the effect of the different parameter combinations on some of the critical data sets.

These parameters determine the type of CPCS process, as shown on the following pages. Be sure you understand the implications of each option and establish your installation procedures accordingly. Since CPCS is often a continuously running task, select a value for the TIME operand that prevents premature ending of the job.

Note: If you omit the PARM operand on the EXEC statement, the default options are WARM, NOFMT. If you specify an FTYPE parameter, you must also specify an STYPE parameter.

STYPE Parameter

COLD

- Rebuilds the CPCS system parameter log file (MCMPL).
- Deletes, by rewriting the string directory index, all string information that exists in the MDS.
- Clears all information from the pass-to-pass control tracer data set and the cycle data set.
- Calls DKNCOMP to clear and reformat the kill bundle data set and the microfilm data set.
- The FTYPE setting determines whether to clear the MDS and the scroll data set. (See Figure 1-1 on page 1-6.)
- Assumes that CTYPE=CKPT.

A cold start can be time consuming, but it is a requirement if:

- MDS parameters change

- MAXTG parameters change
- MAXRP parameters change
- Certain kinds of errors occur.

Note: If you want to initialize the IGEN data set, run JCL first to delete and then to re-create the IGEN data set (DKNISEQ).

WARM

- Scans the CPCS system parameter log file (MCMPL) for changes from the last run of CPCS. The CPCS system editor determines if any parameters have changed and if those parameters require a COLD start. If the CPCS system editor determines that a COLD start is required, the system shuts down and the system log file is *not* updated. If a COLD start is *not* required, the CPCS system parameter log file (MCMPL) is rebuilt.
- Scans the MDS for I-strings or R-strings that were open when CPCS was last stopped and sets them to restart mode.
- Does not change the pass-to-pass control data set or the cycle data set.
- Compresses fully processed records from the kill bundle data set and the microfilm data set.
- Clears all information from the scroll data set unless NOFMT is the option in the FTYPE parameter.
- Clears all information from the writer checkpoint record when CKPT is the option in the CTYPE parameter.

REST

- Scans the CPCS system parameter log file (MCMPL) for changes from the last run of CPCS. The CPCS system editor determines if any parameters have changed and if those parameters require a COLD start. If the CPCS system editor determines that a COLD start is required, the system shuts down and the system log file is *not* updated. If a COLD start is *not* required, the CPCS system parameter log file (MCMPL) is rebuilt.
- Scans the MDS for I-strings that were open when CPCS was last stopped and sets them to restart mode.
- Does not change the pass-to-pass control data set or the cycle data set.
- Does not compress the kill bundle data set or the microfilm data set. If REST is used consistently, the COMP task should be scheduled after end-cycle processing.
- Does not change in the scroll data set.
- Gives the quickest restart of CPCS without loss or change of data.

RECV

- Recovery of the MDS and index should be complete before CPCS can operate. See the description of the FTYPE parameter for recovery options.
- Scans the MDS for I-strings or R-strings that were open when CPCS was last brought down and sets them to restart mode.

- Calls DKNCOMP to scan the kill bundle data set and the microfilm data set to determine the next available key. These files are not compressed.
- Permits use of previous information in the scroll data set.

For a description of the effect of this option with the various FTYPE parameters for data set recovery, see "Data-Set Recovery Procedures" on page 1-33.

FTYPE Parameter

FMT

- FMT is the default if COLD is the option for the STYPE parameter.
- If you change the size of the MDS record or any of the MDS parameters in the CPCS system profile member DKNPCPCS, you must also use FMT.
- Formats the MDS when STYPE is COLD.
- Clears the scroll data set when STYPE is COLD.

NOFMT

- NOFMT is the default when WARM or REST is the option for the STYPE parameter.
- Does not format the MDS.

SCFMT Clears the information from the scroll data set on a restart.

Note: You can use the following FTYPE parameters only if STYPE is specified as RECV. For more information about selecting these parameters, see "Recovery Procedures for the MDS and the Index Data Set" on page 1-34.

MDS CPCS recovers the MDS from user-specified log tapes.

INDEX CPCS recovers the MDS index from the MDS.

BOTH CPCS recovers the MDS and the MDS index. CPCS recovers the MDS from user-specified log tapes, then assembles the index from the MDS. This parameter is valid for logging.

SEL You specify this option (selective recovery) if an MDS or BOTH recovery resulted in incomplete strings and the tapes necessary for completeness originated before the tapes used in the preceding recovery. SEL specifies that only the incomplete strings are recovered from these earlier tapes. This parameter is valid for logging.

RMDS You specify this option (restart MDS recovery) if the recovery tapes were in the sequence 1, 2, 3, 5, 4 in the restart JCL (DKNRT) and only 4 and 5 are necessary for RMDS.

RBTH You specify this option if you want to restart BOTH recovery under the same conditions as for RMDS.

NBTH You specify this option when you want to recover the mass data set and index but do not want to alter the tracer data set. The recover process is the same as if you specify PARM='RECV,BOTH', except the tracer data set is not initialized.

CTYPE Parameter

CKPT Indicates that CPCS must clear the writer-checkpoint record whenever either the printer or spool configurations change.

Note: All spooled output is lost.

The checkpoint record contains information about the spool data sets and the printers assigned to CPCS. It must be the third positional parameter, if specified. When the STYPE parameter is specified as COLD, CKPT is the default value for the CTYPE parameter. In this case, you can omit CTYPE and substitute a positional null.

ESM Parameter

XXXX Specifies the type of startup that the Enhanced System Manager should perform. See the *Enhanced System Manager User's Guide* for the possible values for this parameter.

Library Statements

STEPLIB Data Set Statements: This set of concatenated data definition (DD) statements should reference the CPCS load library and any user-load libraries required by the installation. COBOL/370 programs might require the correct system library if the COBOL/370 and sort libraries are not in the MVS link list.

SORTLIB Data Set Statements: CPCS requires this library when running DKNFILM, DKNCREF, and any user-written COBOL programs that use the SORT verb.

Document Processor Statements: These DDs are a requirement if you use channel-attached document processors. They refer to the DDRDRIN parameters of the CPCSRDR macros. With dynamic allocation, you must specify these DDs only if allocation is necessary at startup. These DDs, and the 3890s contained in the DKNDSAT table for MVS dynamic allocation, associate the DDNAME parameters of the CPCSRDR macros with their physical document processors. DDs for 3890 simulated document processors are also included.

Printer Device Statements: DKNITASK uses these DDs and the printers in the DKNDSAT table for dynamic allocation to complete the printer table at initialization of CPCS. If you use a unit reference in the form of a channel and unit address, the device must be a physical printer. If the ddname is TAPEOUT, DKNITASK includes a tape-output printer in the printer table.

Temporary and Permanent CPCS Data Sets: Permanent CPCS data sets should be cataloged in the system and allocated in accordance with the volume requirements of the installation. For information on sizing calculations, see "External Storage Requirements" on page 1-19.

CPCS temporary data sets should be sized to meet the needs of the installation and, for performance reasons, should be specified as single buffered (BUFNO=1) data sets.

Spool Data Set Statements: You can enter up to the maximum number of spool data sets supported by CPCS, based on the number specified by the SPOOL parameter in the CPCS system profile member DKNPCPCS. The first 5 characters

of the ddname must be SPOOL. One to three digits follow the ddname SPOOL (for example, SPOOLxxx, where *xxx* can be 1 to 3 characters).

You preallocate spool data sets because they are used by application programs as intermediate data sets. The CPCS output writer then reads from these data sets and directs them to a printer. As a guide, the working minimum should equal the number of application tasks that require a printer and that you expect to have running concurrently.

Note: Generating only a small number of spool data sets increases the chances that none will be available when there is a request to start an application task that needs one. In this case, the task waits in a queue for a spool data set to become free. An excessive number of spool data sets wastes direct access space. You can allocate a minimum of two and a maximum of 255 spool data sets.

High-Volume Spool Data Set Statements: CPCS supports tasks that generate a large amount of spooled output (for example, DKNKILL). This option minimizes the amount of space necessary for all spool data sets by sending spooled output to specific data sets. The user must conform to the specifications as explained in the *CPCS Programming and Diagnostic Guide* and must code the letter L as the eighth character of the ddname in the DD statement (for example, SPOOL01L).

DKNATASK allocates these spool data sets to selected application tasks, provided that this requirement is specified in the DKNBLDL list by the HIVOL=1 operand. You must determine, and specify in the DD statement, the amount of disk space to allocate to the larger spool data sets.

Application tasks can write directly to job entry subsystem (JES) spool files. The maximum number of output spools is 255. DKNBLDL saves additional information for assigning JES spools. This information includes:

- The class to which reports are sent for this task
- Routing information, if applicable.

See “JES Considerations with Dynamic Allocation” on page 1-16 for more information.

Tape Data Set Statements: If you elect to make them tape data sets, the input create file (DKNIN), the master create file (DKNMD), and the end cycle file (DKNSK) can be allocated to a single tape unit. These tapes are dynamically allocated using the definitions in DKNDSAT.

The recovery tape (DKNRT) is used for both selective and full MDS recoveries. If you install logging, and elect to log directly to tape, then the MDS log tape (DKNLT) and its option duplex (DKNLTD) should each have its own drive. If possible, secondary units should be available to minimize delays that result from end-of-reel conditions.

Logging Data Set Statements: These DDs are a requirement if the master task generation defines LOG=YES. With dynamic allocation, the DKNLD, DKNLDD, and DKNDTD statements can be dynamically allocated if the DD definitions are in DKNDSAT. DKNRINIT should preallocate and initialize all other DDs.

System Print Data Set Statements: Various system tasks and user tasks (for example, the SORT program called by COBOL tasks) generate printed output messages during the running of CPCS. For this reason, you should include a SYSOUT DD in the JCL.

In the event of program abends and possible system errors, you should code a SYSUDUMP or a SYSABEND DD to obtain the documentation necessary for problem determination. DKNATASK deallocates and reallocates SYSUDUMP each time a CPCS application program returns with a nonzero system or user return code.

Normal Shutdown of CPCS

The MVS system operator can remove CPCS from the system through the normal operating system facilities by canceling the job. However, the operator should not cancel the CPCS job unless directed to do so by a CPCS supervisor. The supervisor can shut down CPCS at any time with the STOP command. This type of shutdown is a ***scheduled shutdown***. When it occurs, a console message informs the system operator that CPCS has been shut down because of a supervisor request. For more information on the STOP command, see the executive task commands section of the *CPCS Terminal Operations Guide*.

Operational Considerations

This section describes system operation recommendations for CPCS. These recommendations are intended to help you run CPCS more efficiently on your host system. However, depending on the host system configuration, some of these recommendations might not be appropriate or necessary. The topics include job priority, terminal and sorter attachment, automatic restart, printing, job entry subsystem (JES), and system management facilities (SMF).

MVS Operations

You should run CPCS with the highest available job priority on your system. You must run CPCS as non-swappable. Include an entry for the CPCS master task (DKNMTASK) in member SCHED00 of SYS1.PARMLIB. Place the entry in the following columns:

```
----+----1----+----2----+----3----+----
PPT     PGMNAME(DKNMTASK)
        NOSWAP
```

This marks the CPCS region as non-swappable, but does not prevent it from being paged. For details on running a non-swappable task, see *MVS/ESA Initialization and Tuning*.

A secondary method for making CPCS non-swappable is to have the CPCS master task invoke a non-swappable SVC. For further information, see “Nonswap SVC Generation Task” on page 2-31.

Data-Space Considerations

The merged string (M-string) distribution (DKNMDIS) module uses IBM licensed program Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA*) data spaces to temporarily store D-strings that DKNMDIS builds. The following formula determines the size of the data space: `data-space size = ((maximum number of D-strings x length of string directory entry) + (number of codelines x (length of one codeline + 8)) + 4095) ÷ 4096`.

For example, given a 50,000 item entry, the size of the data space would be `2046 = ((99 x 736) + (50,000 x (158 + 8)) + 4095 ÷ 4096`.

The result is 2046 pages of memory, which is approximately 9MB (MB equals 1,048,576 bytes) of memory. The sample BLDL allows three copies of MDIS to be active at the same time, which is 3 x 9MB or 27MB.

DKNMDIS maintains a forward and backward pointer to the codelines, and it requires approximately 8 bytes per codeline. MDIS also uses approximately one page of the data space for control variables. When DKNMDIS attaches, it cannot determine the exact size of the data space. Therefore, DKNMDIS checks the size of the data space each time it adds an item to the data space. If the size of the data space is too small, MDIS increases the size up to the maximum data-space size or up to the system installation limit. Each time a copy of DKNMDIS attaches, a new data space is created. The data space is deleted when DKNMDIS ends.

Note: If DKNMDIS cannot extend the data space to fulfill the requirements of the entry, DKNMDIS ends and no permanent data sets are updated. The maximum number of codelines in a string that DKNMDIS can distribute is approximately 10,000,000 lines. The standard mass data set provides a data space of 2 gigabytes. For more information about data spaces, see *MVS/ESA Application Development Guide: Extended Addressability*.

Region Size

The CPCS configuration needs a region of at least 4M bytes. However, for efficient operation, the region size may need to be much larger; at least 8M bytes is recommended.

Spool Checkpoint

The writer interface module DKNWTRIF performs the checkpointing of the spool and printer allocation table. The checkpoint record is rewritten each time the status of an entry in the table changes. In the event of a system failure, restarting CPCS prompts DKNWTRIF to recover the existing checkpoint record. This record is used to restore the existing spool and printer allocation table entries to the status that existed at the time of the last checkpoint. DKNWTRIF scans the printer table, flagging all printers available except those designated offline; the output class remains the same.

The spool table is scanned to determine whether any status change is required. The status of a spool changes under the following conditions:

- A spool that was printing at the time of the failure is flagged as queued.
- A spool in-use (assigned to an application task for data creation) is flagged as available, because any data on the spool is considered unreliable and the application task is normally rerun.
- Spools waiting to print are flagged as queued.

After checking and changing each table entry, DKNWTRIF performs its normal interface functions by checkpointing the newly modified table and scheduling the correct spools for print processing.

Display Terminal Considerations

MAXTERM=nn (where nn=1 to 99) must be specified in the CPCS system profile member DKNPCPCS. Specifying a number greater than the actual number of physical terminals creates extra terminal entries in the terminal table. CPCS uses VTAM functions for allocation of the terminals.

Document Processor Considerations

CPCS supports the following three host attachments for document processors:

- Channel attachment
- LU 6.2 attachment
- Host-simulated (no physical attachment).

You define how the document processor attaches to the host as part of the MICR task generation. For more information about the MICR task generation, see “MICR Task Generation” on page 2-33.

Channel Attachment

Channel-attached document processors use QSAM for I/O operations. If you specify the 3890/XP Document Processor during the MICR task generation, CPCS requires the 3890/XP MVS Support product, which provides enhanced QSAM support. You must use the MODEL=XP option and the TYPE=3890-nn option on the CPCSRDR macro to define the 3890/XP Document Processor.

You can define the physical document processor channel-unit addresses through data definition (DD) statements in your JCL. If you use dynamic allocation, an operator defines the physical document processor channel-unit address using the DKNALLO program, and the DKNDSAT data set must contain a corresponding DSAT macro entry. For more information about using the DSAT macro, see "Dynamically Allocating Data Sets" on page 2-23.

LU 6.2 Attachment

An LU 6.2-attached document processor requires 3890/XP MVS Support Version 1 Release 2 or higher and ACF/VTAM Version 3 Release 2 or higher. You must define LU 6.2-attached document processors to the VTAM products. For more information about defining an LU 6.2-attached document processor, see the *3890/XP MVS Support and 3890/XP VSE Support Program Reference*.

You must supply an LU 6.2 node table module to define the physical LU 6.2-attached document processors. The LNTNAME parameter in the CPCS system profile member DKNPCPCS specifies the name of the LU 6.2 node table module. For information about creating an LU 6.2 node table module, see the *3890/XP MVS Support and 3890/XP VSE Support Program Reference*.

The DKNALLO program begins and ends communication with an LU 6.2-attached document processor and associates a CPCS logical document processor to a physical LU 6.2-attached document processor. During MICR processing, LU 6.2-attached document processors operate the same way as do channel-attached document processors. LU 6.2-attached document processors and channel-attached document processors can replace one another as part of MICR OPEN processing.

Host-Simulated Attachment

The host-simulated document processor requires the 3890/XP MVS Support program. You must define each simulated document processor in your JCL with additional DD statements. For an example of the DD statements, see the sample definition statements in DKNJRUN in the CPCS.V1R11.CTRL data set. For more information about defining DD statements for document processors, see the *3890/XP MVS Support and 3890/XP VSE Support Program Reference*.

Automatic Restart Considerations

Different CPCS jobs that access the same group of document processors must have different ARST parameter values. Automatic restart is performed only for 3890/XP Series document processors.

The restart method is always manual for simulated document processors. MICR-user changes must not exclude any paper and automatic restart exception control records from being written to the MDS. Otherwise, automatic restart might fail and manual restart would be forced.

Note: Automatic exception control records are written to the MDS whenever CPCS processes exception or SCI error headers. These control records, which are

identified by X'80' in field DIFLAG2 and X'E3' in field DITYPEI, are necessary for the automatic-restart matching process.

If the Run Profile Drives/Paths do not specify the standard directory (INIT), the Restart Run Profile (RESTART.RPR) must be copied into the nonstandard directories. Otherwise, automatic restart will fail with an `AUTOMATIC RESTART NOT IMPLEMENTED` status message, and manual restart would be forced.

Printer Considerations

If you do not use dynamic printer allocation, you should define a dedicated device for print output. In most cases, this would be a printer. However, an option exists to substitute a tape drive or SYSOUT=A, using the CPCS TAPEOUT DD statement.

To substitute a tape drive, change the JCL according to the following specifications:

- The DD statement for the tape unit must contain the ddname TAPEOUT.
- The disposition parameter must be DISP=(MOD,PASS).
- The DCB parameter must specify BLKSIZE, BUFL, and BUFNO.
- Do not code the volume serial number (VOL=SER=nnnnnn).

If you use dynamic printer allocation, you do not need to specify printers in the JCL. You must include these printers in the DKNDSAT table. With this approach, the printers can be allocated to other jobs when CPCS is started. They can be allocated to CPCS, when needed, by using the DKNALLO task.

CPCS can also use JES output facilities, see “JES Considerations with Dynamic Allocation.”

If you specify a printer with the universal character set (UCS) feature, the character image should be a default layout. Specify both the forms control buffer (FCB) and UCS parameters for use with a forms control buffer.

For efficiency in printing, the number of copies (MAX) for the DKNWTR task specified in DKNBLDL should be equal to the number of printers specified in the CPCS system profile member DKNPCPCS. For more information on the DKNWTR task, see the *CPCS Programming and Diagnostic Guide*.

Do not direct output to a particular printer because, for example, if a printer is experiencing hardware errors, all output to that device is lost. In this case, change the printer to an offline status or change its class to one that is not used. You can use DKNFORM to change the class. For more information about DKNFORM, see the *CPCS Terminal Operations Guide*.

JES Considerations with Dynamic Allocation

When using dynamically allocated SYSOUT data sets and JES writers, perform the following steps to generate CPCS printed output:

- Include JESPRT*nc* entries in DKNDSAT, using the DSAT macro to provide a pool of SYSOUT data sets for use by the CPCS writer task. For more information on SYSOUT data sets, see the *CPCS Programming and Diagnostic Guide*.

If you need to print large reports without tying up spool data sets, use DKNADCB3. This subroutine changes the calling program’s data control block (DCB).

DKNADCB3 dynamically allocates a JES SYSOUT spool and sets the calling program's DCB to the JES ddname.

For dynamic allocation of data sets, see "Dynamically Allocating Data Sets" on page 2-23 for a description of using the DSAT macro. For a description of DKNADCB3, see the *CPCS Programming and Diagnostic Guide*.

System Management Facilities Considerations

CPCS customers who use System Management Facilities (SMF) must specify a job-wait-time parameter for SMF generation. This parameter governs the length of time a job can be in a WAIT state before automatic system termination. Because check processing operations can be lengthy, CPCS might have to remain in a wait state for a long period of time. If it is not feasible to increase the time parameter value (for example, TIME=1439) to accommodate a long wait, unscheduled CPCS shutdowns can occur.

Enhanced Prime Capture Considerations

Enhanced Prime captures have the following characteristics:

- The MICR capture begin information is entered only once, before starting the run, with the PRIME field containing the tracer group number corresponding to the first entry to be captured. All stacked entries run under the same sort type, on the same cycle and under the same MICR user parameters and exit routines.
- The MICR capture end commands are entered only once, after having captured the last stacked entry. The capture end screens show counts and totals corresponding to the last entry captured.
- The number of tracer groups that identify an entry is determined by the value that is specified in the Sort Pattern Definition O, card columns 41-43. The default value is 001.
- Output I-strings follow the same naming conventions as all other normal I-strings.
- MICR status message 118 may optionally be displayed, before each new entry is captured, to allow the MICR operator to pull the pocketed items before the next entry is captured. The capture may also be suspended or ended at this point, instead of continuing capturing more entries.
- If the pocketed items are not pulled after capturing each entry, kill items belonging to different entries may be separated by activating the "tracers in kill pockets" option.
- All normal prime pass features and options are available.
- Kill bundles do not span through entries. A "Sorter Initialization at Entry Breaks" option is available to avoid small bundles at the beginning of the entries.
- The function and its options are activated by using the sort pattern definition options (O) record.

Data Facility Storage Management Subsystem Considerations

CPCS data sets can reside on Data Facility Storage Management Subsystem (SMS) controlled volumes, except the mass data set. You must adequately define the CPCS data set requirements. You should use the following guidelines:

1. SMS cannot control the CPCS mass data set. The mass data set is allocated as follows:

   ```
   DD DSN=CPCS.V1R11.MASSDS,
   DCB=(RECFM=F,LRECL=1012,BLKSIZE=1012,DSORG=DAU),VOL=SER=XXXXXX
   ```

   You should allocate the mass data set as contiguous and unmovable, especially with multi-pack allocation.
2. The data sets cannot have space released.
3. The duplex data sets cannot be on the same volume serial as primary data sets. The data sets can be on the same volume serial only when you use split storage groups or guaranteed space storage classes.
4. Data sets should not be migrated.
5. Blocking factors cannot be automatically generated by SMS.
6. No DCB or VSAM allocation parameter can be altered by SMS.
7. You should write the SMS access control system (ACS) routine to use all dynamic allocation parameters. You must decide if SMS uses specific volume serial requests.
8. CPCS BDAM data sets are not allocated with extents.
9. Fast I/O response time for certain data sets is required. Caching volumes may be required for high I/O data sets, (such as the Divider data set).
10. CPCS logging data sets are not allocated with *free space*. When a D37 system completion code occurs the data set should not be increased. A data class should not be selected that assigns a volume count or secondary space to the data set.
11. Data sets required for user written, third party vendor, and business partner programs that run in the CPCS environment must be analyzed and considered for any unique restrictions that may apply to those programs.
12. Sorter allocation is not controlled by SMS.

External Storage Requirements

This section describes the external storage requirements for CPCS. It includes information about direct access storage, magnetic tape storage, and data set preparation.

Direct Access Storage Device Requirements

Within each of the following categories, the data sets are listed alphabetically by ***ddname***. (Depending on the degree to which you have customized your own CPCS run JCL, your ddnames may not be the order in which they occur the supplied sample CPCS run JCL.)

Important!

Before you run CPCS, you must allocate and pre-format the VSAM data sets.

Permanently Allocated Data Sets

The following ddnames must be coded in your run JCL when you start CPCS.

APTR (ATASK Printer)

The applications task, DKNATASK, writes a log of events as they occur. Whenever a task is attached, detached, or queued, DKNATASK writes a message to the log. It also writes these messages to the scroll data set. You should allocate APTR to a SYSOUT data set. If you do not want this function, allocate the DD statement to DD DUMMY.

CKPTDS (Writer Checkpoint Record)

This data set contains status information about the spools and printers assigned to CPCS. DKNWTRIF rewrites this single record each time a status change occurs. It is a sequential data set that consists of one record that contains both the spool and the printer tables. This data set must be pre-allocated. It requires no special formatting; DKNWTRIF formats it, based on the PARM operand values in the JCL EXEC statement.

```
CKPTDS size = (4 + (48 x SPOOL)) + (4 + (20 x NUMPTR)) bytes
```

CYCLDS1 (Cycle Table Data Set)

This data set contains records for status, date, endorse date, name, and cycle quantity. You must pre-allocate this data set with LRECL=664 and BLKSIZE=664.

CYCLDS2 (Cycle Table Duplex Data Set)

This data set is a duplicate of CYCLDS1 when DUPLEX=1. The storage requirements are the same. Allocate it on a different DASD volume than the one used for CYCLDS1.

DKNAB (Endpoint Name and Address Data Set)

The CPCS installation creates the endpoint name and address data set for reference by DKNKILL and DKNCLSM. One record exists for each routing number entered in the file. DKNLOAD creates this file each time it runs (see “Data-Set Preparation” on page 1-31). The data set is unblocked and the

logical record length is 600. You should allow 20% additional space for expansion.

DKNADJCD
This data set defines the adjustment codes used by the DKNADJ online adjustments program. Each record is 30 bytes in length, and contains the user's two-digit adjustment code, the equivalent internal DKNADJ flag, and a comment.

DKNAPPL
DKNAPPL is a partitioned data set with a logical record length of 80. Its members are profiles that control the behavior of individual CPCS tasks. The profile for a specific task is reloaded each time that task is run.

DKNBCF (Bank Control File Data Set)
The CPCS installation creates a set of 80-byte records that describe each processing bank. DKNLOAD uses this input data set, created outside of CPCS, to create the records in the DKNBCF data set. DKNMICR and various report modules use this data set as input.

DKNCMPRS (Work Data Set for DKNCOMP)
This is a work data set used by DKNCOMP. The space allocation for this data set should be large enough to hold a complete copy of the kill bundle data set or the microfilm data set, whichever is larger. It must be permanently allocated, because it contains the only complete copy of the kill bundle or microfilm data set during compress operations.

DKNDIV (Divider Slip Data Set)
This optional data set is a VSAM-keyed, sequential data set (KSDS) that provides the location of a divider slip in the MDS. DKNDIST, DKNMDIS, and DKNTDUP (a subroutine of DKNSCAT) all update this data set. Its purpose is to resynchronize sorter codeline data match buffers during a subsequent-pass capture.

DKNEP (Endpoint Tables Data Set)
This is a partitioned data set whose members are endpoint tables (see "Data-Set Preparation" on page 1-31). The user creates and maintains this data set. The number of endpoints that you have defined determines its total size.

DKNFREE (Work Data Set for DKNIMGS)
This data set area is required for shops that run CIMS and HSRR capture, and use DKNIMGS to provide CIMS with a cross-reference between the prime pass and HSRR pass sequence numbers. It is a sequential data set, fixed blocked, with a logical record length of 29 and a BLKSIZE of 290. A record is written to DKNFREE for each HSRR I-string codeline that could not be matched back to its prime-pass system reject D-string.

DKNHELP (System Help Data Set)
This is a VSAM-keyed, sequential data set (KSDS) used to hold help screens. It is allocated and loaded during CPCS installation. Thereafter, any task can display any help screen by passing the appropriate task name and screen number keys to the Help Function Manager module, DKNHELP.

DKNISEQ (Item Sequence Number Data Set)
This data set is a VSAM relative-record data set that DKNIGEN uses to maintain unique sequence numbers for up to 96 document processors.

Provision is made for other electronic data. DKNMICR and DKNOLRR update this data set. The CPCS installation process includes a stand-alone job that initializes this data set.

DKNKB (Kill Bundle Data Set)

DKNKILL writes a record to the kill bundle data set for each printed kill bundle. These records are deleted by cycle when DKNECYC runs. DKNCOMP pre-formats the kill bundle data set with a specific number of records, based on the expected maximum number of kill bundles.

Apply the following formula when sizing the kill bundle data set:

```
K = (N ÷ A) x 1.1
```

where:

K = Number of kill bundle records required
N = Number of items killed
A = Average number of items for each killed bundle

Base the calculation on peak-day volume.

Notes:

1. The value 1.1 is an allowance for kill bundles for high-speed reject reentry (HSRR) runs and for the partial bundles at the end of each prime and subsequent pass for kill pockets.
2. You can use a blocked BDAM format for this data set. For information about formatting this data set using blocked BDAM, see “Using Blocked BDAM Data Sets” on page 1-28.

DKNKD (Duplex Kill Bundle Data Set)

This data set is a duplicate of DKNKB, and its storage requirements are the same. Allocate it on a different DASD volume from the volume used for DKNKB.

If you specify DUPLEX=1 in your master task generation, you must include a DD statement for this data set in your JCL. You cannot have a DD DUMMY statement for this data set.

DKNMC (Master Create Work Data Set)

DKNMCRE uses this temporary work data set. It contains one record for each kill bundle (DKNKB records) transferred to the master tape and one record for each R-string, on-us D-string, user-to-process D-string, or subsequent-pass reject D-string.

This data set must be large enough to contain both the records for the maximum number of kill bundles that one run of master creation can process and records for the maximum number of R-strings, on-us D-strings, user-to-process D-strings, or subsequent-pass reject D-strings in the run.

DKNMF (Microfilm Data Set)

DKNFILM produces microfilm reports from this data set. It is a required data set even if microfilming is not present or used. DKNDIST and DKNMDIS write a record for every batch and block slip they process. DKNFILM reads records for its reports, and marks them for removal; DKNCOMP deletes the marked records. DKNCOMP pre-formats the microfilm data set with a specific number of records, based on the expected maximum number of batch and block slips.

Consider how often you run DKNFILM and how often you compress the data set by running using DKNCOMP (including CPCS startup) when calculating the size of the data-set. An allocation of one track is enough if microfilming is not present or not used because only three records are written to this file by CPCS. You can allocate the microfilm data set as a blocked BDAM data set. For information about formatting this data set using blocked BDAM, see “Using Blocked BDAM Data Sets” on page 1-28.

DKNMFD (Duplex Microfilm Data Set)

This data set is a duplicate of DKNMF. Its storage requirements are the same. Allocate it on a DASD volume that is different from the volume used for DKNMF.

If you specify DUPLEX=1 in your master task generation, you must include a DD statement for this data set in your JCL. You cannot have a DD DUMMY statement for this data set.

DKNMISS (Work Data Set for DKNIMGS)

This data set area is required for shops that run CIMS and HSRR capture, and use DKNIMGS to provide CIMS with a cross-reference between the prime pass and HSRR pass sequence numbers. It is a both sequential data set, fixed blocked, with a logical record length of 29 and a BLKSIZE of 290. A record is written to DKNMISS for each prime-pass system reject D-string codeline that could not be matched to its HSRR I-string.

DKNSMSGD (System Message Data Set)

This is a VSAM keyed sequential data set (KSDS) used to hold error messages. It is allocated and loaded during CPCS installation. Thereafter, any task can fetch any message by passing the appropriate subsystem, task name, and message number keys to the System Message Handler module, DKNSMSG.

DKNSPDEF (Sort Pattern Definition Library)

The user creates and maintains this data set. This is a partitioned data set. You must define it as unblocked.

DKNSUBMT (Default JCL Submit Data Set for DKNSUB)

This is an optional partitioned data set with a logical record length of 80. If it is present in the CPCS run JCL, DKNSUB searches it for the JCL members that it is to submit. If DKNSUBMT is not present in the CPCS run JCL, DKNSUB assumes its start parameter names a data set as well as a member, and it attempts to allocate and submit from that data set.

DKNTG (Pass-to-Pass Control Data Set)

This is the tracer data set used in pass-to-pass control. DKNPCTL formats this data set on a cold start. The calculations for space requirements are as follows:

`SPACE = N blocks` where:

N = `Q x (R + X)`

- **Q =** MAXTG + 1 (defined under “CPCSOPTN Macro Parameters” on page 2-43)
- **R =** MAXRP + 9 (defined under “CPCSOPTN Macro Parameters” on page 2-43)

X = Number of prefix index records that are equal to the nearest integer greater than `(4 x R) ÷ BLKSIZE` (from Figure 1-2 on page 1-23)

DKNTG is a "keyed" file and must be allocated with the DCB parameter, where *XXX* is the block size without the key length added (see Figure 1-2 below).

DCB parameter needed:

```
DCB=(KEYLEN=12,RECFM=F,DSORG=DA,BLKSIZE=XXX)
```

Figure 1-2. Allocation of DKNTG with the DCB parameter

Number of Pockets (largest sorter in MICR gen)	BLKSIZE (w/o Key)	LRECL (with Key)
6	318	330
12	318	330
18	414	426
24	510	522
30	606	618
36	702	714

Note: DKNPCTL calculates the required BLKSIZE when a CPCS cold start is executed. This BLKSIZE may differ from the above chart depending on your CPCS maintenance level. If the BLKSIZE differs from the above chart, you should use the DKNPCTL calculated BLKSIZE to determine the proper number of prefix index records (X) in the calculation for space requirements.

DKNTGD (Pass-to-Pass Control Duplex Data Set)

This data set is a duplicate of DKNTG, and the storage requirements are the same. Allocate it on a different DASD volume than the one used for DKNTG.

If you specify DUPLEX=1 in your master task generation, you must include a DD statement for this data set in your JCL. You cannot have a DD DUMMY statement for this data set.

IMGSSEQ (Work Data Set for DKNIMGS)

This data set is required for shops that run CIMS and HSRR capture and use DKNIMGS to provide CIMS with a cross-reference between the prime pass and HSRR-pass sequence numbers. It is a sequential data set, fixed blocked, with a logical record length of 47 and a BLKSIZE of 470. One record is written for each codeline that matches, or fails to match, between the HSRR I-string and its corresponding prime-pass system reject D-string.

INDEX (MDS directory index)

The MDS directory index is a random access data set. To maximize processing efficiency, we recommend that the data set physically reside on a device separate from the MDS. The MDS directory index consists of string directory index (SDI) records for all active strings.

Specify the number of physical records allocated to the MDS directory index (NDIRBLK), based on the number of anticipated active strings. To aid in searching the index for a named string, SDI records are grouped by string type within the MDS directory index. You specify the number of physical records allocated to each string type by specifying the last relative record allocated to a

specific string type (EDIREND, MDIREND, IDIREND, RDIREND, and DDIREND).

A randomizing routine distributes active SDI records across the records allocated for a specific string type. If a record is completely filled, **one** overflow record is chained to the primary record. If both the primary and overflow records contain no free SDI records, a request to open an output string to that record results in an error and a return to the calling program.

Figure 1-3 shows the relationship between the types of strings and the number of physical records:

Figure 1-3. MDS directory index relationships

String Type	Number of Physical Records
NDIRBLK	42
EDIREND	1
MDIREND	2
IDIREND	5
RDIREND	6
DDIREND	21

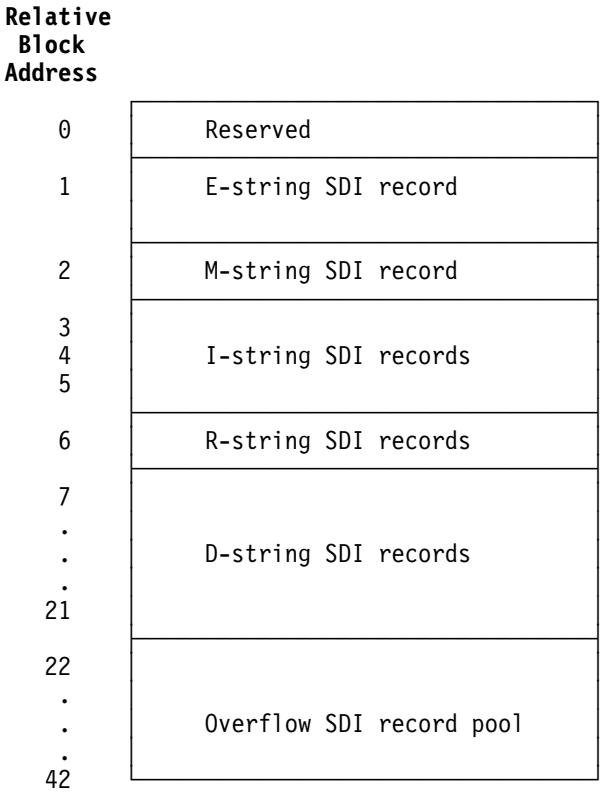

Figure 1-4. MDS Index Block Allocation

Note: It is possible for each primary directory block to be chained to an overflow directory block. Therefore, the maximum size of the overflow pool equals the total number of primary blocks.

MAIL01 (Electronic Mail Data Set)

This is a VSAM-keyed, sequential data set (KSDS) used by DKNMAIL to route messages from one CPCS task to another. Each call to DKNMAIL results in one message being added, read, or removed from the data set. For more

information on DKNMAIL, refer to the *CPCS Programming and Diagnostic Guide*.

MASSDS (Mass Data Set)

The mass data set (MDS) is a random access data set that resides on one or more direct access volumes. You should not allocate the MDS to a device that contains other frequently accessed data sets.

The MDS logically consists of segments that are defined as a specified number of physical records. A segment is the basic unit of space allocation. The first record in the first segment allocated to a string contains the string directory entry (SDE) and the string segment map (SSM). The relative block address of the SDE and the string itself are in the MDS directory index. Multiple segments allocated to the same string are chained together. The segments allocated to a string are reused when the string is deleted and space is freed.

Use the following variables to help you to determine the size of the MDS:

- Direct access storage device type
- Length of time that the strings for a particular entry are active before the space is reusable
- Maximum number and size of concurrently active strings
- Block size as specified in the CPCS system profile member DKNPCPCS.

MDS services (DKNMDSVC) directs a message to the system operator and the supervisor terminal when the MDS is 80% filled and for each 5% increment thereafter until the data set is completely filled. When the MDS becomes completely filled, all tasks enter a wait state until space becomes available. You can run DKNICRE, DKNMCRE, DKNDELE DKNSTGD, DKNFREE, or DKNLDIR with the delete option, to free MDS data set space. For information about using these, see the *CPCS Programming and Diagnostic Guide* and the *CPCS Terminal Operations Guide*.

MCMPL (System Log File)

The application task, MTASK, writes a system log file with the latest system parameters. This log is used to identify what parameters have been modified from the previous run and what type of start CPCS requires. You must pre-allocate this data set with LRECL=200 and BLKSIZE=200.

MPTR (CPCS System Parameter Edit Report)

This is a report DD that displays the CPCS system parameters generated and any errors associated with the edits. This DD is defined to the CPCS run JCL (DKNJRUN) and the Batch System batch editor JCL (DKNJEOTP). This JCL is located in the CPCS.V1R11.CTRL data set.

SCROLL (Online Activity Log)

The communication task (DKNVTASK) writes supervisor terminal messages and ATASK log messages to this data set. The operator can use the DKNSCRL application task to browse this data set.

Warning: The data set must be previously allocated as a single contiguous extent.

The communication task formats this data set during a cold start of CPCS. It can optionally format the data set during a warm start of CPCS.

The average number of supervisor terminal messages and ATASK log messages that a particular installation receives determines the number of tracks allocated to the data set. This is not a history data set. Writes to this data set wrap around when it is full. Allocate this data set in cylinders.

SORTWK01, SORTWK02, SORTWK03 (Sort Work Data Sets)
These work data sets are required for the sort-merge programs. If you use concurrent sorting, do not code them in the startup JCL because CPCS modules acquire the work data sets dynamically.

SYSTPROF (System Profile Data Set)
This is a partitioned data set with a logical record length of 80. Its members are profiles that control the behavior of CPCS. They are loaded once CPCS is started and are used thereafter for the remainder of CPCS job execution.

Temporary Data Sets

The following data sets are all temporary work files. You should code entries in your DKNDSAT table, using a DYNDSN of 35 characters or less, a DYNSTAT of NEW, a DYNNORM of DELETE, and a DYNCOND of DELETE. For more information, see “Dynamically Allocating Data Sets” on page 2-23. The standard DKNDSAT shipped with CPCS already contains entries for all these data sets.

Important!

If you expand your mass data set, the DCB parameters, and possibly the space allocation, for the DKNMRGE and DKNSCA2 data sets (MRGEIN, MRGEOUT, RSTGFIL, MATCHFIL, HSRRFIL, and SCATIN) may have to change. The updates would take place either in DKNDSAT, if you are configured for dynamic allocation, or in the CPCS run JCL, if you are not. For more information, see “Expanding the Mass Data Set” on page 2-48.

ALSTSORT
This is a work data set used by the DKNALST report program of the Online Adjustments system. Each record is 18 bytes in length. As many records should be allocated as there are adjustment M-strings you might conceivably have to list. (A separate adjustment M-string is created for each individual tracer group within an adjusted entry.) You should size ALSTSORT accordingly.

CLDSI (Temporary Work Data Set for DKNCLSM)
DKNCLSM uses this data set to store kill bundle records. When you request cash letter summaries, DKNCLSM writes the kill bundle records that meet the request criteria of this data set.

CLDSO (Temporary Work Data Set for DKNCLSM)
DKNCLSM sorts the data on the CLDSI data set, using the following keys:

- Endpoint
- Sending financial institution
- Cycle code.

It writes the output of the sort to this data set.

CREFIN, CREFOUT (Temporary Work Data Sets for DKNCREF)
These data sets contain one active record for each kill bundle that meets the criteria for the current run of DKNCREF.

DKNIW (Input Creation Work Data Set)
This data set serves as an intermediate data set during the input creation task. It saves information about M-strings being transferred during this run of the task. Therefore, the data set should contain one record for each M-string processed during an input creation run.

KLDSO (Temporary Work Data Set for DKNKILL)
This data set contains one record for each string that is kill-listed during this run of DKNKILL. This data set is input and output to an internal sort of the string names that DKNKILL uses to generate the kill lists.

MFSORTED (Microfilm Work File)
This data set contains the sorted output from the two-sort run in DKNFILM. Each record contains an entire microfilm record of 50 bytes and a 4-byte sort key, for a total of 54 bytes for each record. Enough space should be allocated so that the work file can contain a complete data set of microfilm records.

MRGEIN, MRGEOUT, RSTGFIL, MATCHFIL, HSRRFIL (Temporary Work Data Sets for DKNMRGE)
These work files are used for various purposes by DKNMRGE. At most, two records are written to each file for every codeline that DKNMRGE processes. Therefore, each file should have enough primary space allocation to hold twice the size of your largest M-string.

SCATIN (Temporary Work Data Set for DKNSCA2)
As directed by DKNSCAT, DKNSCA2 reads each R-string, one after the other, into this data set to construct the final concatenated R-string. This data set should be sized to hold the largest R-string that can exist on your system.

Logging and Recovery Data Sets

If you plan to install the CPCS Logging Subsystem, the following data sets must be added to your CPCS run JCL. See “Logging CPCS Data” on page 2-7 for more information.

Required

DKNLD1
DKNLD2
DKNRCVCK
DKNRCVSR
DKNRCVTD
DKNRCVUT
DKNRCVY

Required If Duplexing “Flip-flop” Files

DKNLD1D
DKNLD2D

Required If Duplexing the Control Files

DKNRCVCD
DKNRCVSD
DKNRCVT2
DKNRCVYD

Enhanced System Manager Data Sets

If you plan to install the Enhanced System Manager feature, the following data sets must be added to your CPCS run JCL. For more information, refer to the *CPCS Enhanced System Manager User's Guide*.

ESMCNTL
ESMLOG
EVEDATA
HSADATA
MEMDATA
MWFLDD
SMSPDEF
TLRDATA
TPDD
TPDDO
UOWDATA
UOWNRDD
WFLDD
WFLDDO
WRBDATA

Using Blocked BDAM Data Sets

You can use a blocked BDAM format for the DKNKB and DKNMF data sets. Follow these steps if you plan to use blocked BDAM for these data sets.

1. Determine the blocking factor you need for the DKNKB and DKNMF data sets. The maximum blocking factor is 255.

   For DKNKB, this value must be a multiple of the logical record length of 400. The maximum value is 32400.

   For DKNMF, this value must be a multiple of the logical record length of 50. The maximum value is 12750.

   Note: You might need to increase the amount of working storage requested for the distribution (DIST) task in DKNBLDL. For more information about DKNBLDL, see "Describing an Application Task's Environment" on page 2-15.

   The BLKSIZE and LRECL parameters must be equal for each file that uses blocked BDAM.

2. Allocate the DKNKB and DKNMF data sets with the new logical record length and block size.
3. Change the KBBLKSZ and MFBLKSZ parameters in the CPCS system profile member DKNPCPCS to the new block size values.

Important!

- If you are using duplexed data sets, repeat steps 1 and 2 for the DKNKD and DKNMFD data sets.
- Be sure that the DUPLEX parameter in the CPCS system profile member DKNPCPCS is set to 1. When you use duplexed data sets, you must allocate the duplex versions for all four of the data sets that are duplexed. For more information about duplexed data-sets, see “Using Data-Set Duplexing” on page 2-6.
- Only the DKNKB and DKNMF data sets (and their duplexed versions) can be allocated as blocked BDAM data sets. You can choose to allocate only one of the data sets (and its duplex version) using blocked BDAM.

Magnetic Tape Storage Requirements

The input creation (DKNICRE), master creation (DKNMCRE), and end cycle (DKNECY2) tasks have data sets that optionally can be written to tape.

If more than one data set is allocated to a drive, all data sets except the first must have `UNIT=AFF=FILE1` in the JCL of the data description (DD) for this data set. To ensure that two data sets do not use the same tape, the second parameter of the DISP operand must be KEEP. This parameter causes all tapes for this data set to unload when the data set is closed, which means that the operator must mount an additional tape the next time the task is run.

If you do not have enough tape drives, you should dynamically allocate tape data sets wherever possible. Dynamic allocation lets you free up tape drives when an application module does not require them. If you use dynamic allocation, you do not need to allocate multiple data sets to a single tape drive. For information about setting up tape DD statements for dynamically allocated tape data sets, see the *MVS/ESA JCL User's Guide*. For information about CPCS dynamic allocation requirements, see “Dynamically Allocating Data Sets” on page 2-23 and the *CPCS Programming and Diagnostic Guide*.

If the LOG=YES and LOGDEV=TAPE functions are active, the logging task requires that you mount one tape throughout CPCS operation. You should mount two tapes for this task to save time mounting and dismounting the tapes. When a log tape is not mounted, any CPCS task that issues a write to the MDS goes into a wait state and this has a serious effect on all CPCS operations. For instructions on assigning two tape drives to one data set, see the *MVS/ESA JCL User's Guide*.

When CPCS opens a tape data set, it sends a message to the system operator indicating that it is creating a data set. CPCS identifies the external label for the tape. When CPCS closes the tape data set, it sends another message to the system operator stating that it created the data set and repeats the label information.

The following instructions required to implement tape handling for DKNICRE and DKNCIRET also apply to the data sets for the master create and end cycle tasks. Program changes are the same except for the naming convention.

DKNICRE

Calls the subroutine DKNICRET to open, write, or close the DKNIN data set. DKNICRET passes back to DKNICRE the unit-serial number that is used while

opening the file. This file is used for each run of DKNICRE. This information goes to the supervisor terminal and the scroll data set, if they are used.

DKNICRET

Reads the JFCB to establish addressability to the Task Input/Output Table (TIOT). The TIOT contains the address of the unit control block (UCB). The UCB contains the volume serial number. The JFCB contains the 44-byte data-set name. This information is extracted at open and EOV time. If the data-set name is a generation data group (GDG), it is used as it exists the first time DKNIN is opened. The data-set name is incremented by 1 in the generation level, and the serial number field is cleared to its original status. At close time, the serial numbers that are used are cataloged if index levels exist. Suitable index levels must be created by running IEHPROGM or IDCAMS.

If the data set is on tape and is not a GDG, DKNICRET appends .T*dddhhmm* to the end of the DSN, where *ddd* is the Julian day of the year and *hhmm* is the hour and minute the data set was opened. DISP=(NEW,KEEP) is required in the JCL before these changes to the data-set name occur. If DISP=OLD or DISP=MOD, no data-set name changes occur.

DKNICRE creates one data set each time it runs. The JCL for DKNIN should specify VOL=PRIVATE and LABEL=EXPDT or RETPD to prevent the operator from accidentally overwriting the output tapes.

Optional Tape Data Sets

The following tape data sets are listed by ddname and must appear with that same ddname in the JCL that starts CPCS.

Note: Dynamic allocation applies to the DKNRT, DKNSK, DKNIN, and DKNMD data sets.

DKNIN (Input Data Set)

This data set contains the data from the M-strings for a cycle and bank. It can contain multiple volumes. The record length is the MDS record length + 1 byte, and you can specify whether it is blocked. This data set may be either tape or DASD.

DKNLT, DKNLTD (Log Data Set)

Thess data sets are required for logging if LOGDEV=TAPE is selected in the options module. DKNLT and DKNLTD replace the DKNLD and DKNLDD statements.

DKNMD (Master Data Set)

This data set contains the data for a cycle and bank from the killed D-strings, subsequent-pass reject D-strings, R-strings, and on-us D-strings. It can contain multiple volumes. The record length is the MDS record length + 9, and you can specify whether it is blocked. This data set may be either tape or DASD.

DKNRT (Recovery Tape Data Set)

CPCS uses this data set to restore the MDS. CPCS creates these tapes as a real-time log of MDS activity. When RECV is specified, the data set is opened, read, and closed as part of CPCS initialization. The DCB parameters are obtained from the header label of the tape. You supply the volume serial numbers for the data set.

DKNSK (Summary Kill Bundle Data Set)
This data set contains kill bundle records for a cycle. The record length is 31 bytes, and you can specify whether it is blocked. This data set may be either tape or DASD.

TAPEOUT (Printer Tape Output)
When specified as a tape in the DD statement, this tape substitutes for a dedicated printer. It contains CPCS output writer information from data written to spool data sets. For more information on tape-drive substitution, see "Printer Considerations" on page 1-16.

Data-Set Preparation

These data sets are listed by ddname and you must format them with data before running CPCS. You determine the space, volume-serial-number, and number of records for each data set.

DKNAB (Endpoint Name and Address Data Set)
DKNLOAD creates the endpoint name and address data set, a basic direct access method (BDAM) data set, each time it runs. The data set contains:

- The routing number
- The receiving financial institution's name and address
- Flags that identify the type and format of the kill bundles
- Any other instructions to or from the financial institution
- Twelve bytes in each record to store codes that might be meaningful to the local operation.

There is one record for each endpoint number entered into the data set. DKNKILL and DKNCLSM use the information as a reference during the processing cycle.

DKNBCF (Bank Control File Data Set)
DKNBCF, a BDAM data set, contains the name and address of each financial institution that has work being processed. It also contains special processing requirements and any CPCSOPTN overrides to the site defaults. For more information about this data set, refer to "Bank-Control-File Record Formats" on page 4-5.

Note: There are special BDAM data-set considerations. You must allocate enough space for BDAM data sets and include an additional 20% space for future growth.

You must use the ***extended search*** feature, available as an option for BDAM data sets. Call this option by including the following parameters in the data control block (DCB) for the DKNAB and DKNBCF data sets:

```
OPTCD=E
LIMCT=n
```

The first parameter activates the feature. The second parameter specifies the number of tracks (*n*) that BDAM uses during an extended search. As shipped, CPCS has LIMCT set to 13 for the DKNAB and DKNBCF data sets. If this is changed from the shipped value, the new value must be put in member DKNCRTRK in data set CPCS.V1R11.SLIB. For more information, see "COBOL

Copy Definitions" on page 2-5. Then all modules using the member must be either recompiled or assembled, then link edited.

You must rerun DKNLOAD for loading the endpoint name, address, and BCF data set. Use the JCL in the member DKNGBCAB of CPCS.V1R11.CTRL.

If the DKNLOAD program ends abnormally, you must rerun the program.

DKNEP (Endpoint Tables Data Set)

This is a partitioned data set. Each member contains a list of endpoints. DKNCLSM and DKNKILL accept member names as input. CPCS uses the endpoints in the table you select to determine what to include in the kill list or to summarize in a cash letter.

You create the data set using the IEBCOPY utility. Each member can contain from 1 to 100 endpoints. Member names must be EPT, followed by three numbers. Each input record contains as many as nine endpoints, each separated by a space. The last endpoint in the last input record for a member must be an endpoint of eight @s. See Figure 1-5.

You should use the JCL in member DKNGEPT of CPCS.V1R11.CTRL to create the endpoint tables data set.

Figure 1-5. Example of Endpoint Data-Set Member Contents

EPT001	EPT002
09000000	02100002
10000000	02130235
11000000	02100030
12000000	02100008
@@@@@@@@	02000000
	@@@@@@@@

You can run the IEBGENER utility to update the endpoint table. For information about using IEBGENER, see *MVS/ESA Data Administration: Utilities, Version 3.1*.

DKNISEQ (Item Sequence Number Data Set)

You must initialize this VSAM relative-record data set after you define it. Use the JCL in member DKNGISN of CPCS.V1R11.CTRL for this initialization.

DKNSPDEF (Sort Pattern Definition Library)

Sort patterns are defined to CPCS through a partitioned data-set member in the DKNSPDEF data set. The MICR task retrieves the information pertaining to a particular sort pattern when a document processor entry is made. The member of the sort pattern definition data set is accessed by the type-of-work parameter. All member names must be of the format SPTYP*xxx*, where *xxx* is the type of work. The *xxx* is numeric, and can have a value from 1–255.

Use the JCL in member DKNGSPDF of CPCS.V1R11.CTRL to set up the sort pattern definitions used during CPCS operation.

Standard 3890 SPDEF Example

```
P1000000S001S1          021     0     20100            20
RP      RP001
R            0101
M              01
K01125000020212500057031250001004125000110512500013060000012509888888888
K10888888881188888888
P2070000S001S2          001222220     10100             0
RP      RP001
K01125000280212500182031250018804125007130512500755061250075607125007 80
K08125007850932507 1171012510007
P2080000S001S3          001222220     10100             0
RP      RP001
K01000000010200000002030000000304000000040500000005060000000607000000 07
K08000000080900000009100000001011000000 11
```

Expanded Format 3890/XP Series SPDEF Example

```
P1000000S002S1          011     0     10100XF          10
RP      RP002                           XP
R                   01
M                   01
K01888888880288888888038888888804009999990512500002061250005707125000 10
K08125000110912500013100000012 5
P2110000S002S2          001222220     10100             0
RP      RP002
K01000000010200000002030000000304000000040500000005060000000607000000 07
K08000000080900000009100000001011000000 11
```

For a detailed description of the SPDEF format, see “Sort-Pattern-Definition Record Formats” on page 4-17.

Sample Profiles

Your CPCS.V1R11.SAMPLIB distribution library contains a number of sample profile members. CPCS.V1R11.CTRL jobs DKNGSYST and DKNAPPL load these members into, respectively, the SYSTPROF System Profile and DKNAPPL Application Profile data sets. The recommended procedure for customizing profiles is to copy them from SAMPLIB, edit them, then use jobs DKNGSYST and DKNAPPL to transfer the edited copies into your CPCS.

Changes made to the DKNAPPL application profile data set take effect immediately, while CPCS is still up.

For changes made to the SYSTPROF System Profile data set, you must stop and restart CPCS before they have any effect.

Data-Set Recovery Procedures

This section describes the types of data-set recovery procedures provided by CPCS.

Recovery Procedures for the MDS and the Index Data Set

Recovery for the MDS and the index data set is separate from recovery for the duplexed data sets. The type of recovery necessary depends on which files you want to recover. When you have determined the recovery type, you must specify the proper parameters for the PARM operand of the CPCS JCL. For the suggested recovery parameters, see Figure 1-6 on page 1-35.

DKNMDSV1 performs MDS and MDS index recoveries. When the recovery is complete, DKNMTASK initializes CPCS in a restart mode. Logging does not take place during recovery. Logging starts with the completion of CPCS initialization.

Whenever you cannot access data on the MDS or the index, run an MDS or index recovery to restore the data. CPCS sends messages that indicate I/O errors to the system operator and supervisor. I/O errors on the index result in the console messages DKNMDSVC02 or DKNMDSV104, which state the requirements for an index recovery. CPCS handles write errors on the MDS. Read errors on the MDS are more serious, and the task requesting the read handles them individually. You must determine how important the inaccessible data is and must decide whether some form of recovery is necessary.

Note: I/O errors can indicate that data cannot be processed. Recovery does not solve hardware failure problems unless you allocate a new data set to a different device.

Figure 1-6 on page 1-35 contains the suggested recovery procedures for the CPCS mass data set recovery. Before you use this table, you must first know the status of the following files involved in the recovery:

- RCVY support files
- Mass data set
- Mass data set index
- Tracer data set.

You must determine whether these files are available and contain valid data. In Figure 1-6 on page 1-35, YES means that the files are available and contain valid data. NO means that the files were lost or are not available and have been reallocated empty.

Note: If any one of the following files has been lost, first check to see if its duplex can be used in its place. If the duplex has also been lost, or if you do not have your recovery system configured for control file duplexing, then the answer for the RCVY Support Files is *NO*.

DKNRCVY DD name for the logging recovery index.
DKNRCVSR DD name for the logging VOL-SER file.
DKNRCVTD DD name for the logging status file.

Important!

Initialize the tracer data set only when necessary. Any time the tracer data set is reinitialized, all I-strings that were not closed through the MICR task **cannot** be restarted. The restart information is lost when the tracer data set is reinitialized.

Figure 1-6. Suggested Recovery Procedures

RCVY Support Files	Mass data set	Mass data set index	Tracer Data Set	Recovery Procedure Steps
NO	NO	NO	NO	1, 2, 3, 8, 10, 11, 13
NO	NO	NO	YES	1, 2, 4, 8, 10, 11, 13
NO	NO	YES	NO	1, 2, 5, 8, 9, 10, 11, 13
NO	NO	YES	YES	1, 2, 5, 8, 10, 11, 13
NO	YES	NO	NO	1, 2, 6, 9, 10, 11, 13
NO	YES	NO	YES	1, 2, 6, 10, 11, 13
YES	NO	NO	NO	2, 3, 10, 11, 14, 15
YES	NO	NO	YES	2, 4, 10, 11, 14, 16
YES	NO	YES	NO	2, 3, 10, 11, 14, 15
YES	NO	YES	YES	2, 4, 10, 11, 14, 16
YES	YES	NO	NO	2, 3, 10, 11, 13, 14, 15
YES	YES	NO	YES	2, 6, 10, 13
YES	YES	YES	NO	2, 3, 10, 11, 14, 15

Recovery Procedure Steps

Listed below are suggested steps for performing the recovery procedures shown in Figure 1-6.

1. Reallocate the logging files.

   Update the DKNGLALL JCL and run this job. This job allocates the logging files.

   Notes:

   a. All files are allocated as empty.

   b. If the logging recovery index is still available, update the DKNJLSRP JCL and run this job. This job creates a report of all the strings within the logging facility.

   c. If you have parameter AUTO_INIT=NO in the logging profile (DKNPRCVY), you should update DKNGLINT JCL and run this job. The DKNRINIT program does *not* preserve any data.

2. Reallocate lost files, preferably to a different device.

3. Change the PARM statement to `PARM='RECV,BOTH'`.

   For more details about these parameters, see “BOTH Recovery (PARM='RECV,BOTH')” on page 1-39.

4. To destroy the existing MDS data, change the PARM statement to `PARM='RECV,NBTH'`. If the existing MDS data is to be kept, change the PARM statement to `PARM='RECV,RBTH'`.

   For more details about these parameters, see “BOTH Recovery (PARM='RECV,NBTH')” and “Restart BOTH MDS Recovery (PARM='RECV,RBTH')” on page 1-40.

5. Change the PARM statement to `PARM='RECV,MDS'`.

For more details about these parameters, see “Mass Data Set Recovery (PARM='RECV,MDS')” on page 1-37.

6. Change the PARM statement to `PARM='RECV,INDEX'`.

   For more details about these parameters, see “Mass Data Set Index Recovery (PARM='RECV,INDEX')” on page 1-37.

7. Change the PARM statement to `PARM='RECV,SEL'`.

   For more details about these parameters, see “Selective Recovery (PARM='RECV,SEL')” on page 1-40.

8. Change the CPCS startup JCL by specifying recovery tapes under the DKNRT DD statement.

   The recovery tape serial numbers should be specified for the recovery data set (DKNRT). You should specify, in the order of their creation, all tapes that contain required strings.

9. Copy the duplex tracer group file over to the newly allocated tracer group file (DKNTG).
10. Start CPCS.
11. Sign on to CPCS as a supervisor.

    If you specify a recovery start parameter, the RCVY task starts automatically on the supervisor terminal.

12. Enter RCVY to see the Recovery Start option menu.
13. Select option 3.

    This synchronizes the CPCS MDS recovery files with the mass data set.

14. Select the Full Mass Data Set Recovery (option 1) of RCVY.

    RCVY automatically determines which strings and which tapes are necessary. For more details, see the description of DKNRCVY in the *CPCS Programming and Diagnostic Guide*.

15. Enter YES in the RCVY Tracer Group File Update option.

    Before beginning recovery, RCVY presents the Recovery Cycle Specification screen. Enter YES in the Tracer Group File Update option. This option updates the tracer file for the strings being recovered.

16. Enter NO in the RCVY Tracer Group File Update option.

    Before beginning recovery, RCVY presents the Recovery Cycle Specification screen. Enter NO in the Tracer Group File Update option. This option does not update the tracer file for the strings being recovered.

Recovery Parameter Descriptions

Listed below are the detailed descriptions of the CPCS recovery parameters. These parameters are valid only when LOG=YES is specified in the CPCS system profile member DKNPCPCS.

Mass Data Set Index Recovery (PARM='RECV,INDEX')

Delete the old index data set and allocate a new data set, preferably on a different device. Tape input is not necessary for index recovery. The index is rebuilt from the existing data on the mass data set.

Processing Description: The index data set is formatted and the MDS is unchanged. You can recover the index data set by processing all the active SDEs on the MDS and constructing index records (SDIs) for those strings. If the strings are closed, CPCS updates the master and existing segment maps. This recovery option does not update the tracer data set.

Note: OPEN and RESTART strings are added to the corresponding map during the automatic CPCS restart that follows recovery.

MDS read errors can result in a loss of data. If the error segment contained an active SDE and data, then part or all of the string can be lost. Other READ errors do not affect the index recovery. Run the program DKNLDIR and check the output to determine whether all strings are present. For MDS READ errors, the recovery routine writes console message DKNMDSV120 and continues processing.

SDI READ and WRITE errors on the index data set cause messages DKNMDSV107 and DKNMDSV108 to be sent to the system operator. Recovery ends with a user ABEND (U0073). Reallocate the data set and run recovery again.

After recovery is complete, the RCVY screen is automatically presented to the system supervisor when the supervisor signs on. Normally, no additional recovery is necessary. It might be necessary, in some cases, to use option 3 on this screen to synchronize the MDS with the RCVY recovery index. You must determine whether any additional recovery steps are necessary, depending on the specific circumstances.

Mass Data Set Recovery (PARM='RECV,MDS')

Note: Use this recovery option only when the recovery support files have been lost. If the recovery support files are intact, the recovery process RCVY automatically determines which strings and tapes are active and recovers accordingly. For the recovery procedures relating to specific recovery situations, see Figure 1-6 on page 1-35.

The option does not update the tracer data set. Delete the old MDS and allocate a new MDS, preferably to a different device. Change the CPCS startup JCL by specifying the recovery tape serial numbers for the recovery data set (DKNRT).

Each time CPCS is shut down, DKNLOGX closes the data set. Therefore, more than one log data set can exist. Regardless of the end-of-volume (EOV) or end-of-file (EOF) condition, if you specify OPTCD=B in the DCB parameter of the recovery data set, DKNMDSV1 processes all tapes in the sequence indicated in the JCL. You must specify tape JCL serial numbers in the sequence in which they were created.

Processing Description: The MDS is formatted and the index is left as-is. DKNMDSV1 reads the header record from the tapes and writes it to the MDS. Each tape record is sequence-checked as to date and time, which appear in the header record. An out-of-sequence condition means that the tapes have been specified in the wrong sequence. DKNMDSV1 sends message DKNMDSV111 to the system operator. The operator has the option of ignoring the error or canceling

the recovery. The data already restored is not lost if the recovery is cancelled. The recovery can be restarted (see “Restart MDS Recovery (PARM='RECV,RMDS')” on page 1-40). The data is checked for completeness after all the tapes have been read and processed. DKNMDSV1 considers the string incomplete if an index entry exists and one or more of the following conditions exist:

- No SDE exists.
- The SDE does not contain the correct IDs.
- The first data record of the string does not contain the correct ID.

DKNMDSV1 sends message DKNMDSV110 to the system operator, specifying the names of the incomplete strings; and the recovery process ends with a user ABEND (U0074) after all strings have been checked. Incomplete strings mean that you have not supplied enough tapes to recover all active data. (CPCS does not process any incomplete strings.) Three possibilities exist:

- The missing tape or tapes should have been specified sequentially before all the other tapes (see “Case 1”). You can recover the missing data by performing a selective recovery. See “Selective Recovery (PARM='RECV,SEL')” on page 1-40.
- The missing tape or tapes should have been specified after all the other tapes (see “Case 2”). You can recover the missing data by performing a restart MDS recovery, using only those tapes that were omitted. See “Restart MDS Recovery (PARM='RECV,RMDS')” on page 1-40.
- The missing tape or tapes should have been specified somewhere in the middle (see “Case 3”). You can recover the missing data by performing a restart MDS recovery, starting with the missing tapes and including all subsequent tapes. See “Restart MDS Recovery (PARM='RECV,RMDS')” on page 1-40.

Examples:

Case 1: There are five tapes in the recovery data set. Tapes 3, 4, and 5 are used in an MDS recovery, resulting in incomplete strings. Therefore, use tapes 1 and 2 in a selective recovery to complete the strings.

Case 2: There are five tapes in the recovery data set. Tapes 1, 2, 3, and 4 are used in an MDS recovery, resulting in incomplete strings. Therefore, use tape 5 in a restart MDS recovery to complete the strings.

Case 3: There are five tapes in the recovery data set. Tapes 1, 2, 4, and 5 are used in an MDS recovery, resulting in incomplete strings. Therefore, use tapes 3, 4, and 5 in a restart MDS recovery to complete the strings.

Recovery Tape Read Errors: Each time a read error occurs on the recovery tape, DKNMDSV1 sends message DKNMDSV114 to the system operator. DKNMDSV1 skips the tape data record and reads the next record. DKNMDSV1 loses the MDS information in the skipped tape record. Should 25 unsuccessful READs occur in succession, DKNMDSV1 sends message DKNMDSV113 to the system operator and ends the recovery process with a user ABEND (U0073). For additional security, back up the recovery tapes using an MVS copy utility.

MDS Write Errors: These errors indicate that CPCS cannot process the newly allocated MDS. DKNMDSV1 sends message DKNMDSV117 to the system operator and ends the recovery process with a user ABEND (U0073). You must rerun MDS recovery, using a reallocated MDS.

MDS Read Errors: These errors indicate that CPCS cannot process the newly allocated MDS. DKNMDSV1 sends message DKNMDSV106 to the system operator and ends the recovery process with a user ABEND (U0073). You must rerun MDS recovery, using a reallocated MDS.

SDI Read Errors: These errors indicate that the index data set also needs to be recovered. DKNMDSV1 sends message DKNMDSV107 to the system operator and ends the recovery process with a user ABEND (U0073). To recover both the MDS and the index, you must run a BOTH recovery.

BOTH Recovery (PARM='RECV,BOTH')

Important!

If you use this option, the tracer data set is initialized. Once it is initialized, any existing I-strings in restart status cannot be restarted because the MICR task saves the Restart Control Block (RSCB) in the tracer data set. Without the RSCB information for an entry, MICR cannot restart the entry.

Although it may not be necessary to delete the mass data set and its index, it is recommended that they be deleted and reallocated. At this point, you must determine whether the recovery support files are available and contain valid information. If these files have been lost, you must change the CPCS startup JCL by specifying the recovery tape serial numbers for the recovery data set (DKNRT). If the recovery support files have not been lost, you do not need to specify the recovery tape serial numbers in the CPCS startup JCL. The RCVY process determines which strings were active, which tapes are necessary, and in which sequence to process the tapes.

Processing Description

- Recovery Support Files (or their duplexes) do exist

  DKNRCVY DD name for the logging recovery index
  DKNRCVSR DD name for the logging VOLSER file.
  DKNRCVTD DD name for the logging status file.

  If the recovery support files exist, (see previous list), using the parameter PARM=(‘RECV,BOTH’) causes the initialization of the MDS, the MDS Index, and the tracer data set. This is followed by CPCS startup. Do not specify any recovery tape serial numbers for the recovery data set (DKNRT). They are not necessary if the recovery support files exist. No recovery takes place until the system supervisor uses the RCVY process to start recovery.

  RCVY is started automatically on the supervisor terminal when the supervisor signs on. The CPCS Recovery Start Options Menu screen is displayed on the system supervisor terminal. The Full Mass Data Set Recovery option is on this screen. The supervisor should select that option to begin full MDS recovery. You can find additional information on the RCVY process in the *CPCS Terminal Operations Guide* and the *CPCS Programming and Diagnostic Guide*.

- Recovery Support Files do not exist

  If the recovery support files do not exist, using the parameter PARM=('RECV,BOTH') causes the initialization of the MDS, the MDS Index, and the tracer data set. The processing is the same as described for MDS recovery (see "Mass Data Set Recovery (PARM='RECV,MDS')" on page 1-37 and "Mass Data Set Index Recovery (PARM='RECV,INDEX')" on page 1-37. If you use this type of recovery, the tracer data set will only be initialized, but not updated.

Restart MDS Recovery (PARM='RECV,RMDS')

This option does not cause the initialization of the tracer data set. Run this type of recovery only when a previous recovery ('RECV,MDS') did not recover the MDS as expected. The data recovered must be more recent than the data already residing on the MDS. MDS recovery and restart MDS recovery processes are the same, except that the MDS is not formatted in the restart MDS recovery process.

Restart BOTH MDS Recovery (PARM='RECV,RBTH')

This option does not cause the initialization of the tracer data set. This type of recovery is only necessary when you are recovering by specifying tape volume serial numbers under the DKNRT DD in the CPCS startup JCL.

Use this type of recovery when you need to recover more data to the MDS without destroying the data that already exists. The index needs reassembling after the MDS recovery. The data recovered must be more recent than the data that already resides on the MDS. The BOTH recovery process and the Restart BOTH recovery process are the same, except that the MDS is not formatted in the Restart BOTH recovery process.

BOTH Recovery (PARM='RECV,NBTH')

This recovery is the same as with PARM='RECV,BOTH', except that the tracer data set is not initialized. Also, the tracer update option on the RCVY screen is set to NO. You should leave this option set to NO if the tracer data set is intact.

Selective Recovery (PARM='RECV,SEL')

This option does not cause the initialization of the tracer data set. Selective recovery assumes that the MDS contains incomplete strings (for more information, see "Mass Data Set Recovery (PARM='RECV,MDS')" on page 1-37). You cannot specify any additional string names for recovery. When selective recovery starts, it scans the MDS to locate any incomplete strings. If there are no incomplete strings, the selective recovery ends.

Therefore, you should run selective recovery only when a previous MDS recovery or a BOTH recovery results in incomplete strings. If the recovery tapes were in the middle or at the end of the sequence of tapes used in the previous recovery, consider a restart MDS recovery or a restart BOTH recovery. However, the DKNRT *ddname* must be specified in the JCL as DD_DUMMY, as the task DKNMDSV1 attempts to open and close this file. For examples of recovery options, see page 1-38.

Processing Description: The index contains an indication of every incomplete string. This information is used to assemble a list of incomplete strings. As it reads each recovery tape, the program compares the data records against the list. If it finds a match, the program writes the record and checks the next record

against the MDS. If there is not a match, the program checks the next record. After all the tapes are processed, the MDS is scanned for any incomplete strings. You can run CPCS with incomplete strings; however, the system does not process these strings.

Note: As the list of incomplete strings is assembled, the complete string indications are removed from the index. Therefore, if selective recovery abnormally ends, the index indicates that there are no incomplete strings.

Before you can rerun a selective recovery, you must run a restart MDS recovery, specifying DD DUMMY for the recovery data set (DKNRT). A restart MDS recovery scans the MDS for incomplete strings. It updates the index to indicate those incomplete strings.

Suggestion

- After a recovery, you should run DKNLDIR and check the list to verify that all active strings were recovered. DKNLDIR provides a list of all active strings on the MDS.

Recovering Duplexed Data Sets

A basic recovery procedure exists for all duplexed data sets in CPCS. All data sets that are duplexed can be recovered by:

1. Ending CPCS as quickly as possible to start the recovery procedure
2. Determining the causes of failure (for example, a disk head crash)
3. Formatting (or allocating) the original data set and copying the duplexed data set to the newly allocated data set
4. Starting CPCS.

The following CPCS data sets are duplexed:

CYCLDS1	the cycle-table data set
DKNTG	the pass-to-pass control data set
DKNKB	the kill-bundle data set
DKNMF	the microfilm data set.

If you have installed logging, the following data sets may also be duplexed:

DKNLD1	disk log 1 (used if you elect to log to disk)
DKNLD2	disk log 2 (used if you elect to log to disk)
DKNRCVCK	the Recovery System checkpoint file
DKNRCVSR	the Recovery System volume serial file
DKNRCVTD	the Recovery System logging status file
DKNRCVY	the Recovery System string name file

You can use IEBGENER to restore lost data for all data sets except the pass-to-pass control data set. The IBM Data Facilities/Data Set Services (DFDSS) program can be used to restore the pass-to-pass control data set because it is a keyed direct-access data set.

For a description of how to use IEBGENER, see *MVS/ESA Data Administration: Utilities Version 3.1*.

For more information about how to use DFDSS, see *Data Facilities/Data Set Services User's Guide and Reference*.

CPCS-Supplied Command Procedures

The following supplied command procedures are a series of job steps, each containing EXEC statements for assembly, compile, and link-edit. The load library is named CPCS.V1R11.LLIB. The assembly and link-edit parameters for the assembler programs are coded in accordance with the table of link-edit parameters. The data sets CPCS.V1R11.SLIB and CPCS.V1R11.LLIB are assumed to be cataloged. If the data sets are not cataloged, the volume identification must be coded.

Use the member DKNJCOB in CPCS.V1R11.CTRL to compile and link-edit all COBOL source modules necessary for CPCS. You can change this member to meet your installation standards.

These PROCs are used by the ASSEMBLY JCL that follows in this section.

Use the member DKNJASM1 in CPCS.V1R11.CTRL to assemble and link-edit those source modules that are called by other source modules.

Use the member DKNJASM2 in CPCS.V1R11.CTRL to assemble and link-edit all other assembler source necessary for CPCS, except the MICR modules. This JCL places load modules in CPCS.V1R11.LLIB. You can change this member to meet your installation standards.

Use the member DKNJMICR in CPCS.V1R11.CTRL to assemble and link-edit all MICR assembler source modules necessary for CPCS. This JCL places load modules in CPCS.V1R11.LLIB. You can change this member to meet your installation standards.

Note: Before using *your* JCL to assemble or compile any modules, be sure that you complete the changes described in the following sections. For additional information on assembling and compiling your CPCS source modules, see the *CPCS Installation Guide*.

Assembly Considerations

CPCS assembler modules contain code that indicates the addressing mode (AMODE) and the residency mode (RMODE) of the CSECT. If a CPCS assembler module contains the following lines, you should **not** override either AMODE or RMODE at link-edit time:

```
DKNcccc CSECT
DKNcccc AMODE 31
DKNcccc RMODE ANY
```

or

```
DKNcccc CSECT
DKNcccc AMODE 31
DKNcccc RMODE 24
```

- Copy or link-edit DKNMPSR into SYS1.IMAGELIB. You can skip this step if you have installed 3890/XP MVS Support Version 1 Release 2 on your system.

COBOL for MVS/VM Compiler Considerations

You must install COBOL for MVS/VM Release 1.0, or higher, to compile CPCS COBOL modules that can run above the 16MB line. The COBOL for MVS/VM modules that IBM ships with the CPCS system are all designed to run above the 16MB line (AMODE 31, RMODE ANY).

If you write COBOL for MVS/VM modules that you want to run above the 16MB line, each module must contain a parameter override record as the first input record in the source deck, as follows:

```
CBL RENT,RES,DYNAM,DATA(31),TRUNC(BIN)
```

- The LKED.SYSLIN should include the COBOL run-time options module, CPCS.V1R11.LLIB (CEEUOPT), appended after the object deck. See “COBOL for MVS/VM Compile Options” on page 1-45 for instructions on assembling CEEUOPT.

COBOL for MVS/VM Compile Options

CPCS requires the following compile options for COBOL for MVS/VM programs.

```
  ADV
  APOST
NOAWO
  BUFSIZE(4096)
NOCMPR2

NOCOMPILE(S)
  DATA(31)
NODBCS
NODECK
NODUMP
  DYNAM

NOEVENTS
NOEXIT
NOFASTSRT
  FLAG(I)
NOFLAGMIG
NOFLAGSAA

NOFLAGSTD
  LANGUAGE(EN)
  LIB
  LINECOUNT(60)
NOLIST
NOMAP
NONAME

NONUMBER
  NUMPROC(NOPFD)
  OBJECT
NOOFFSET

NOOPTIMIZE
  OUTDD(SYSOUT)
  RENT
  SEQUENCE
  SIZE(MAX)
  SOURCE
  SPACE(1)

NOSSRANGE
NOTERM
NOTEST
  TRUNC(STD)
NOVBREF
NOWORD
  XREF(FULL)
  ZWB
```

These options may be supplied either from your installation's defaults, from a COBOL statement at the top of your program source, or from the compile JCL parameter field. The compiler merges from all three sources to locate the final option set.

If you want your programs to run above the line, use DATA(31).

COBOL for MVS/VM User Options

For proper operation, CPCS programs that execute in a COBOL for MVS/VM environment must be compiled with CPCS-compatible runtime options. IBM supplies sample runtime options in it member CEEUOPT of the source library CPCS.V1R11.SLIB. It is strongly recommended that you compile with these options, as described below.

Any modifications to the COBOL for MVS/VM runtime options must be made at the CPCS level. Include the value OVR (override option) on an installation-level statement that corresponds to the statement to be changed at the CPCS level. For example, on the HEAP parameter, you might want to change the installation default values. Make the changes as follows:

- At the installation level, include the value OVR on the HEAP parameter as shown. Do not change other values at the installation level.

```
CEEXOPT
      HEAP=((64K 64k,ANYWHERE,KEEP,16K,16K,OVR),
```

- At the CPCS level, change the values you require. In this example, the 64K values changed to 12K and 50K; the 16K value changed to 12K.

```
CEEXOPT CSECT
  HEAP=((12K 50k,ANYWHERE,KEEP,12K,12K,),
```

The COBOL for MVS/VM user options module (CEEUOPT) and the JCL to assemble the module are in the COBOL installation library.

The sample COBOL370 member CEEUOPT parameters for CPCS can be found in the CPCS.V1R11.SLIB data set. Prior to compiling any COBOL modules, submit member DKNJASM1 from the CPCS.V1R11.CTRL data set. This member contains the CEEUOPT assemble JCL.

Link-Edit Considerations

- Allocate your load library using the IEFBR14 utility.
- The LKED.SYSLMOD in the link-edit step should specify the name of the user load library to use.
- A SYSLIB data set in the link-edit step should include a reference to the CPCS load library.
- Modules using the NCAL option can receive a link-edit warning message. The unresolved external reference resolves when the main module that uses the subordinate modules is link-edited.
- DKNMOLRI *must* be link-edited with RENT attributes.
- OLRR user-edit routines *must* be reentrant and be link-edited with RENT attributes. DKNMOLRI *must* be included with DKNOLRR.
- SCI and table modules *must* be link-edited as RENT, but need not be reentrant.

Changing CPCS Control Blocks

Certain CPCS processing modules maintain the integrity of the CPCS Object Code Only (OCO) programs. These token-table modules and their corresponding control blocks are as follows:

Token Table	Control Block
DKNAPCGT	DKNAPCB
DKNAPTGT	DKNAPTCB
DKNBCFGT	DKNBCF1
DKNPRMGT	DKNPARM
DKNRSCGT	RSCB

The modules dynamically calculate displacements for their respective control block fields. This function allows you to dynamically add fields affecting the CPCS OCO delivered modules. You should use the following procedures to add a new field to a control block:

1. Add the field to the appropriate control block.
2. For each field addition to a control block, add a TOKENDEF macro invocation to its corresponding token table.

   Note: You should use the same field name in both the control block and the token table.

   Use the TOKENDEF macro to add the new field name to the token table. You must specify the field name on the FNAME parameter. For example, the following macro invocation adds the new field xxxxxx to DKNAPTGT:

   ```
   TOKENDEF DNAME=APTCB,FNAME=xxxxxx
   ```

   The TOKENDEF parameters are specified as follows:

   ```
   FNAME = {xxxxxx}
   ```

   where *xxxxxx* is the field name added to the control block. The FNAME should be the same as the field name added to the token table.

   ```
   DNAME = {yyyyyyyy}
   ```

   where *yyyyyyyy* is the DSECT containing the field specified by FNAME, or it defaults to PARMLST.

3. Assemble and link-edit the token tables based on the changes made in step 2. Use the member DKNJASM2 in CPCS.V1R11.CTRL to assemble and link-edit. This job creates the token tables for the control blocks.

   Notes:

   1) You cannot change the existing field names or field types in the control blocks.

   2) You must re-assemble the remainder of CPCS, except the CPCS OCO modules, when additions are made to these control blocks.

Remote Site Support

Using the VNODE macro, you may assign sites, both “owning” and “processing” to a particular terminal. When VNODE is run, and a string is captured through MICR, the site information associated with the MICR terminal is then propagated into MICR’s APTCB for that MICR session, and ultimately the unit of work corresponding to the mass data set string. From this point, DKNATASK continuously propagates this site identification from the work request block of the Enhanced System Manager task that was automatically started, to the APTCB of those tasks, and on to the communication buffers.

For more information on the VNODE macro, see “VNODE Macro Description” on page 2-57.

Chapter 2. Installation and Generation Procedures

Overview

To simplify installation and maintenance, CPCS is packaged using the IBM System Modification Program Extended (SMP/E) product. For complete installation instructions, see the *CPCS Installation Guide* and the *CPCS Program Directory* which accompanies the installation tape for this release. The files on the CPCS tape are:

- SMP/E control file
- CPCS source, copybooks, and macros
- Sample data
- Installation documentation and sample JCL
- SMP/E JCL
- Sample report listings.

To make the best use of SMP/E packaging, complete the following steps during CPCS installation:

1. Use SMP/E to unload the distribution tape to your system, and run the installation job streams as described in the *CPCS Installation Guide*.
2. Run the sample problems to verify that CPCS has been correctly installed. See the *CPCS Installation Guide* for instructions on how to run the sample problems.
3. Customize the CPCS generation to the requirements of your system by performing a DKNMICR generation, which describes the document processors and control documents that you are using. (For information on generating DKNMICR, see “MICR Task Generation” on page 2-33.) Link the MICR modules together in a workable DKNMICR module.
4. Allocate and initialize all CPCS data sets in accordance with the requirements of your system.
5. Edit and load your system and application profiles. Chapter 3, “System and Application Profiles” contains instructions for customizing profiles.
6. Design, code, and test user document-processing routines. Chapter 4, “User Programming Requirements and Exits,” describes the requirements for creating user document-processing routines.
7. Generate data security tables and routines as required for your system. For controlling CPCS resource access, “Security Options” on page 4-151 describes the various methods.
8. Code user-exit routines as required for your system.
9. Customize CPCS reports by changing the endpoint name-and-address data set (DKNAB) and the bank-control-file data set (DKNBCF).

Note: CPCS runs in an MVS extended architecture environment. If you want to capture images, you must run CPCS in an MVS/ESA environment.

Macro Descriptions

The CPCS macro parameters described in this chapter are presented in the following format:

Name	Operation	Parameters
Symbolic name	Macro name	Required and optional parameters

The following list describes the symbols for coding macros:

Capital letters Capital letters represent values that you code directly, without change.

Lowercase letters Lowercase letters represent parameters for which you must supply a value or name.

Brackets [ ] Brackets enclose parameters or symbols that are either optional or conditional. Conversely, the lack of brackets indicates that you must code an item or group of items.

Vertical “or” bar (|)
A vertical bar between operands indicates that you must code one parameter from among the values separated by the “or” bar.

Parentheses, equal signs, and commas
Parentheses, equal signs, and commas must be coded as shown.

Underlined values An underlined value represents the default value that CPCS uses if you omit the parameter.

Braces { } Braces indicate mutually exclusive parameters.

Never code brackets, the “or” bar, underlines, braces, or subscripts.

COBOL Copy Definitions

The COPY definitions DKNCRPKT, DKNCRTRK, DKNCRTG, and DKNCRTGU are supplied to you with default values. If the default values do not meet your requirements, you can change these values and incorporate the changes by using the IEBUPDTE utility.

You can obtain listings of the definitions in CPCS.V1R11.SLIB by using the IEBPTPCH utility to list them.

Each of the following is a one-statement member of the above library. You must set the values in these statements to reflect the configuration.

DKNCRPKT — This value should be 1 less than the number of pockets on the largest sorter: 5, 11, 17, 23, 29, or 35.

DKNCRTRK — The DKNLOAD program uses this parameter when loading the endpoint name and address file (DKNAB) and the bank control file (DKNBCF). DKNKILL and DKNCLSM also use this parameter when reading these files. DKNCRTRK issues as a level 77 statement and should be set equal to the nearest prime number that is less than the number of tracks allocated to the file. Then all modules using the member must be either recompiled or assembled, then link edited.

DKNCRTG — You must change this data description to contain the number of pockets of the largest sorter (including simulated sorters), minus 1. Change the OCCURS statement for the number of pockets.

DKNCRTGU — You must change this data description to contain the number of pockets of the largest sorter (including simulated sorters), minus 1. Change the OCCURS statement for the number of pockets.

Using Data-Set Duplexing

Because a check processing system is a continuous cycle of throughput, loss of access to certain data sets can result in a serious loss of work-in-progress. CPCS relies on the data-set duplexing functions to re-establish the database and to continue processing existing work under dollar control. You can select the duplex option by specifying DUPLEX=1 in the master task generation options.

Data sets that represent dollar control are **duplexed** for recovery. Duplexing creates copies of these files that should reside on different disk volumes.

The files that are duplexed in CPCS are:

DKNTG	Pass-to-pass control data set. The duplex data set name is DKNTGD.
DKNKB	Kill bundle data set. The duplex data set name is DKNKD.
DKNMF	Microfilm data set. The duplex data set name is DKNMFD.
CYCLDS1	Cycle table data set. The duplex data set name is CYCLDS2.

The DKNKB and DKNMF data sets can be allocated as blocked BDAM data sets. (For information about allocating these data sets using blocked BDAM, see “Using Blocked BDAM Data Sets” on page 1-28.) You can recover the duplexed data sets by using an MVS utility to copy from the backup file to the newly created file.

Capture of Duplexed Data Sets

Each time CPCS writes to one of the duplexed data sets, it performs a second write to the corresponding data-set copy. You should allocate the data-set copies on a DASD unit other than the primary data set.

Recovering Duplexed Data Sets

To recover a CPCS data set that is duplexed, you must stop CPCS. Use standard MVS utilities to copy from the duplicate data set to a newly allocated, primary data set. For more information about recovering duplex data sets, see page 1-41.

Logging CPCS Data

The CPCS Logging subsystem maintains the integrity of the CPCS mass data set (MDS) and associated index. This subsystem records or logs to tape or disk all MDS activity. This lets you reinitialize or re-create these data sets and restore logged data if a disk failure occurs during CPCS processing.

The CPCS system is shipped with the Logging function inactive (LOG=NO in the CPCS system profile member DKNPCPCS). You can use the sample problems described in the *CPCS Installation Guide* without activating Logging.

Note: IBM recommends using the dynamic allocation option for your tape and disk data sets, document processors, and printers. Dynamic allocation lets you release or regain control of these devices without stopping CPCS processing, changing JCL statements, and restarting CPCS.

CPCS Logging supports a number of options that enhance the CPCS data-recovery process. These options include logging to disk, duplexed log tapes, tracking of log tapes, tracer data set recovery, and streamlined recovery functions.

The Logging subsystem maintains an index of unexpired log tapes with another index of all string names that are on each tape.

During both selective and full mass-data-set (MDS) recovery, Logging automatically selects the tapes that are required to recover the requested strings.

When you use disk logging, Logging uses two disk log files (with optional duplex files) and alternates between them. As each one fills up, a backup job starts to back up the disk log file to tape. Optionally, you can duplex this tape backup.

When you use tape logging, duplexing is available as an option.

Installing the Logging Subsystem

Installation of the Logging subsystem requires the following steps:

1. Change the CPCS system profile member DKNPCPCS parameters to activate Logging.
2. Change the Logging profile member DKNPRCVY.
3. Verify the RCVUNTBL entries.
4. Verify the DKNLOGM routing destination.
5. Change the DKNDSAT parameters.
6. Assemble the Logging program modules.
7. Assemble the DKNLOGU user exit.
8. Allocate the Logging data sets.
9. Initialize the Logging data sets.
10. Create the generation data group (GDG) for the log data-set backups.
11. Update the CPCS startup JCL.
12. Restart CPCS with PARM='COLD' in the EXEC statement of the JCL.
13. Back up the Logging files.
14. Change the DKNLJCRH JCL.

These procedures assume that you have already completed the installation procedures described in the *CPCS Installation Guide*.

Step 1: Change the DKNPCPCS parameters to activate logging

To activate the logging functions of CPCS, you must change the CPCS system profile member DKNPCPCS parameters. Change the following CPCS system profile DKNPCPCS parameters:

- Set the buffer ratio (BFRAT) to a non-zero value.
- Set the number of tape buffers (TPBFNM) to a nonzero value.
- Set the LOG parameter to YES.

  Note: If you only want to recover existing logged data and do not want to log new data, code LOG=NO and RCVY=YES in the CPCS system profile member DKNPCPCS.
- As a part of the CPCS system edit routine (see page 1-25) the block size for the DKNLD data set has been calculated using the formula below. The block size is written to the CPCS system edit report. The report ddname is MPTR.

  ```
  (BFRATx(BLKSIZE + 12)) + ((BFRAT x 4) + 16)
  ```

For details about the CPCS system parameters, see “DKNPCPCS Profile Member” on page 3-4.

To deactivate logging, set the LOG parameter in the CPCS system profile member DKNPCPCS to NO.

Step 2: Change DKNPRCVY to Specify the Logging Options

The Logging options module controls the environment for Logging. A sample DKNPRCVY profile member is in CPCS.V1R11.SAMPLIB. See “DKNPRCVY Profile Member” on page 3-12, for more details.

Step 3: Verify the RCVUNTBL Copybook Entries

RCVUNTBL is a table that contains the dynamic allocation device names and types that are available to the DKNLOGX and DKNBKUP programs. However, the names contained in this copybook might not be valid for your system.

You can change RCVUNTBL to show the correct device names and types for your system. Type over the names (IBM unit numbers) in the unit-name column of the table with the names assigned to your devices. For example, instead of 3480 your system might use TAPE as a device name. You would replace `3480` with `TAPE` to specify this as your unit type (see Figure 2-1 on page 2-9). If you modify this table after installing CPCS, activate the changes by generating a new version of the DKNRUTBL module.

```
         DC    CL6'2311  ',XL2'2001'
         DC    CL6'2301  ',XL2'2002'
         DC    CL6'2303  ',XL2'2003'
         DC    CL6'2302  ',XL2'2004'
         DC    CL6'2321  ',XL2'2005'
         DC    CL6'2305  ',XL2'2006'
         DC    CL6'2305  ',XL2'2007'
         DC    CL6'2314  ',XL2'2008'
         DC    CL6'3330  ',XL2'2009'
         DC    CL6'3340  ',XL2'200A'
         DC    CL6'3350  ',XL2'200B'
         DC    CL6'3375  ',XL2'200C'
         DC    CL6'3330  ',XL2'200D'
         DC    CL6'3380  ',XL2'200E'
         DC    CL6'3390  ',XL2'200F'
         DC    CL6'2400  ',XL2'8001'
         DC    CL6'3400  ',XL2'8003'
         DC    CL6'3480  ',XL2'8080'
         DC    CL6'3490  ',XL2'8081'
         DC    CL6'      '             END OF TABLE MARKER
```

Figure 2-1. RCVUNTBL Copybook Example

Step 4: Verify the DKNLOGM Routing Destination

This step outlines the preparations required to change the routing destination of the WTO messages sent out by DKNLOGX. Because DKNLOGX calls DKNLOGM to send WTO messages, modify DKNLOGM to send the messages to either the programmer or to the MVS console *and* the programmer.

This example changes the message LOGLOGX 0006 from the default setting of “send to the MVS console” to “send to the programmer only.”

```
 T_NL106I DS     0CL20
     DC     CL8'DKNL106I'                .* T_MSG_NO..MESSAGE NUMBER
     DC     AL4 (C_PROGRAMMER_OPERATOR)  .* T_MSG_DT..WTO TO USE
     DC     AL4(C_106I)                  .* T_MSG_TX..MESSAGE TEXT
     DC     0XL4                         .* T_MSG_SM..ZEROS = NOT USED
     DC     AL1(T-MSG_WTO)               .* T_MSG_SM1..SEND VIA WTO
     DC     XL1'00'                      .* T_MSG_SM2..SEND VIA GB2
     DC     XL1'00'                      .* T_MSG_SM3..SCROLL OR SUPER
     DC     XL1'00'                      .* T_MSG_SM4..RESERVED
```

Figure 2-2. The code showing the default setting (send to MVS console)

```
 T_NL106I DS     0CL20
     DC     CL8'DKNL106I'                .* T_MSG_NO..MESSAGE NUMBER
@MOD001 DC  AL4 (C_PROGRAMMER_ONLY)      .* T_MSG_DT..WTO TO USE
     DC     AL4(C_106I)                  .* T_MSG_TX..MESSAGE TEXT
     DC     0XL4                         .* T_MSG_SM..ZEROS = NOT USED
     DC     AL1(T-MSG_WTO)               .* T_MSG_SM1..SEND VIA WTO
     DC     XL1'00'                      .* T_MSG_SM2..SEND VIA GB2
     DC     XL1'00'                      .* T_MSG_SM3..SCROLL OR SUPER
     DC     XL1'00'                      .* T_MSG_SM4..RESERVED
```

Figure 2-3. The code changing from default to send to programmer only

Step 5: Modify DKNDSAT

The DKNDSAT data set describes dynamically allocated data sets. Verify the file names on the logging entries for DKNLD and DKNLDD.

For information about using the DSAT macro to create DKNDSAT entries, see "Dynamically Allocating Data Sets" on page 2-23.

Step 6: Assemble the Logging Program Modules

Assemble and link-edit the Logging assembler source modules, using member DKNJASML in CPCS.V1R11.CTRL.

Step 7: Assemble the DKNLOGU User Exit

Assemble and link-edit the DKNLOGU sample user exit, using DKNJSAMP in CPCS.V1R11.CTRL.

Step 8: Allocate the Logging Data Sets

Allocate the logging data sets by running the DKNGLALL job from CPCS.V1R11.CTRL.

The following data sets (with high-level qualifier CPCS.V1R11) are allocated:

DKNRCVY
DKNRCVSR
DKNRCVCK
DKNRCVTD.

You need the following data sets only if DPXCNTL=YES:

DKNRCVYD
DKNRCVSD
DKNRCVCD
DKNRCVT2.

You need the following data sets only if LOGDEV=DISK:

DKNLD1
DKNLD2.

You need the following data sets only if LOGDEV=DISK and DPXDISK=YES:

DKNLD1D
DKNLD2D.

Step 9: Initialize the Logging Data Sets

If the AUTO_INIT=YES is specified in the Logging profile member DKNPRCVY, the Logging system automatically performs the initialization of the Logging data sets. Each time the initialization routine runs, the logging system creates a strings names report of the strings within the logging system. If the AUTO_INIT=NO is specified in the Logging profile member DKNPRCVY, you must run the batch editor (DKNJEDTP) and update member DKNGLINT and run the job. This job initializes the Logging data sets that were previously allocated in step "Step 8: Allocate the Logging Data Sets." For more details on the AUTO_INIT parameter, see "DKNPRCVY Profile Member" on page 3-12.

Step 10: Create the GDG for Logging Data Set Backups

The JCL for defining the generation data groups for the backup of the logging and duplex data sets is in member DKNLJGDG of CPCS.V1R11.CTRL. Set the LIM(NNN) parameter in the GDG statements to the number of catalog entries you want to maintain for each data set.

Step 11: Update the CPCS Startup JCL

Update the CPCS startup JCL (DKNJRUN) with the DD statements shown in Figure 2-4.

```
//*****************************************************************
//* LOGGING DATASETS
//*****************************************************************
//DKNRCVY  DD DSN=CPCS.V1R11.DKNRCVY,DISP=SHR
//DKNRCVYD DD DSN=CPCS.V1R11.DKNRCVYD,DISP=SHR
//DKNRCVSR DD DSN=CPCS.V1R11.DKNRCVSR,DISP=SHR
//DKNRCVSD DD DSN=CPCS.V1R11.DKNRCVSD,DISP=SHR
//DKNRCVCK DD DSN=CPCS.V1R11.DKNRCVCK,DISP=SHR
//DKNRCVCD DD DSN=CPCS.V1R11.DKNRCVCD,DISP=SHR
//DKNRCVTD DD DSN=CPCS.V1R11.DKNRCVTD,DISP=SHR
//DKNRCVT2 DD DSN=CPCS.V1R11.DKNRCVT2,DISP=SHR
//DKNLD1   DD DSN=CPCS.V1R11.DKNLD1,DISP=SHR
//DKNLD1D  DD DSN=CPCS.V1R11.DKNLD1D,DISP=SHR
//DKNLD2   DD DSN=CPCS.V1R11.DKNLD2,DISP=SHR
//DKNLD2D  DD DSN=CPCS.V1R11.DKNLD2D,DISP=SHR
//DKNLDD   DD DYNAM
//DKNLD    DD DYNAM
//DKNLT    DD DYNAM
//DKNLTD   DD DYNAM
//DKNSUBMT DD DSN=CPCS.V1R11.CTRL,DISP=SHR
//STGSF    DD DSN=&STGSF,DISP=(NEW,DELETE,DELETE),
               SPACE=(CYL,(10,2)),UNIT=SYSDA,
               DCB=(RECFM=F,LRECL=530,BLKSIZE=530)
//JESPRT      SYSOUT=*
```

Figure 2-4. CPCS Startup JCL Changes for Logging Data Sets

Step 12: Restart CPCS

To initialize the Logging subsystem in the CPCS environment, perform a startup of CPCS after specifying PARM='COLD' in the EXEC statement of the DKNJRUN job stream.

For a description of EXEC statement operands and their effects on CPCS startup, see “CPCS Startup JCL Definition” on page 1-7.

Step 13: Batch Backup of Logging Files

The JCL in member DKNJLBKP provides the ability to back up the disk logging files to tape in batch mode. Submit this batch job when one of these messages appears on the system console:

```
LOGLOGX 1003: BACKUP OF DISK LOG ONE HAS NOT COMPLETED.
PLEASE NOTIFY CPCS SUPERVISOR
```

or

```
LOGLOGX 1004: BACKUP OF DISK LOG TWO HAS NOT COMPLETED.
PLEASE NOTIFY CPCS SUPERVISOR
```

Step 14: Change the DKNJLCRH JCL

You must modify the JCL member DKNJLCRH to match your operating environment. To ensure that no logging data is lost, you must change the JCL after any abend of the MVS system. For more information about the DKNJLCRH functions, see the *CPCS Programming and Diagnostic Guide*.

Capture of Logged Data Sets

Logging maintains a real-time log of all write operations to the MDS. DKNMDCTL calls the program DKNLOGX after every write operation to the MDS. DKNLOGX buffers the disk record according to the value that you specified for the CPCS system profile member DKNPCPCS parameter BFRAT. You also specify the number of available buffers for DKNLOGX (CPCS system profile parameter TPBFNM). While a buffer remains free, there is no waiting on I/O operations.

Recovering Logged Data Sets

You can use the following methods to recover logged data sets:

Mass Data Set Recovery: Recover the mass data set (MDS) by starting CPCS with the correct start parameters. The program DKNMDSV1 performs the recovery by reading the log tapes and writing the data to a newly allocated MDS. CPCS checks the data to ensure the completeness of the recovered data. At the end of MDS recovery, a normal restart occurs.

MDS Directory Index Recovery: The program DKNMDSV1 performs the index recovery by reading the MDS and rebuilding the index that points to the strings found on the MDS. At the end of index recovery, a normal restart occurs.

BOTH Recovery: The CPCS start parameters support concurrent recovery of the MDS and index. An index recovery follows an MDS recovery. At the end of the index recovery, a normal restart occurs.

Selective and Restart Recovery: Tapes can be mistakenly left out of an MDS recovery or a BOTH recovery, resulting in strings that are not complete. Under these conditions, it is not necessary to rerun the complete recovery process. Use either a selective or a restart recovery to reload the data on the omitted tapes.

For additional information about recovering data sets, see the recovery procedures described in “Data-Set Recovery Procedures” on page 1-33.

Log-Tape Synchronization: Whenever CPCS or MVS ends abnormally, it is possible that the MDS contains more data than the log tape. This can happen because the tape-record block size is a multiple of the disk record size and there might be data in a buffer at the time of the error. During the restart, CPCS logs data from the MDS to ensure that the tape-log data set reflects all active MDS data.

Overriding the Log Data-Set Definition

The Logging subsystem writes the current log data-set definition in the first or second record of each log data set. RCVY normally uses this system-generated definition record to define the log data set being recovered.

The DKNRDX50 module defines the fields used when overriding the log data-set definition. DKNRDX50 is shipped with field values that match the sample CPCS system profile parameters and MDX record defaults. If a change is required, you can alter the field definitions, and assemble and link-edit a new load module. The log data-set definition can be overridden by changing the default setting of &RDX in the DKNROPTS macro to the new module name and assembling DKNRCVY, or by keying in the new module name on the Recovery Cycle Specification screen of RCVY.

Important!

Creating a new load module is not normally required. Use this option only when the log data-set definition does not exist as the first record in the data set.

Using an incorrect definition can cause errors during RCVY operation. Ensure that the definition used to recover a log data set is the correct definition for that data set.

If you choose to override the log data-set definition, you must create a new definition override module any time there are changes to the CPCS system profile member DKNPCPCS parameters (BFRAT, TPBFNM, or BLKSZ) or changes to the MDX record.

Tracer group update is supported when recovering strings from CPCS Release 10, or higher, which contains MICR group edit disengage records.

Use the following steps to create a new definition to override the module from the DKNRDX50 log data-set definition:

1. Change the RBFRAT field to match the CPCS system profile member DKNPCPCS parameter BFRAT value that was in effect when the data set was created.
2. Change the RTPBFN field to match the value of the CPCS system profile member DKNPCPCS parameter TPBFNM that was in effect when the data set was created.
3. Change the RBLKSZ field to match the value of the CPCS system profile member DKNPCPCS parameter BLKSZ that was in effect when the data set was created.
4. If the log data set was created under CPCS Release 10 or greater, change the RDXLGSDE field to Y; otherwise set the value to N.
5. Change the RDESC field to a 40-character description that describes this log data-set definition. This description appears on the RCVY Recover Status screen.
6. The remaining fields in the record correspond to the same fields that were in the MDX record when the log data set was created.

   Note: Ensure that all fields are updated to match exactly.

7. Assemble and link-edit the new module for your load library.

   Provide a unique name for each definition override module that you create. For example, if you have a log data set with a mass data set expanded by 3 bytes, you could name the new module DKNRDX53. However, any meaningful name can be used. This name can then be entered on the Recovery Cycle Specification screen.

Data-Set Utilities

There is one data-set duplexing utility. DKNCOPY copies the log tape to a second tape.

DKNCOPY: If MVS ends abnormally, and you are using LOG=YES and LOGDEV=TAPE, the CPCS log tape might not have an end-of-file marker. You can use the improperly closed log tape as input to the program DKNCOPY. DKNCOPY is a batch job that runs under the control of MVS, not CPCS. DKNCOPY copies the log tape to a second tape and closes the second tape. This tape takes the place of the copied log tape.

Customizing Item Sequence Numbers

At the generation time for the Item Sequence Number data set, you can select one of three different formats for the last 8 digits of the item sequence numbers. The formats are *ssssssss*, *rsssssss*, and *rrsssssss*, where *sssssssss/sssssss/ssssss* is a sequence number and *r/rr* is a region number. The CPCS item sequence number format and the item sequence number ranges are selected with the Item Sequence Number Data Set Generation Profile (see “DKNPISNG Profile Member” on page 3-18). CPCS must be cold-started after regenerating the Item Sequence Number Data Set (see “Step 18: Initialize the Item-Sequence-Number Data Set” in the *CPCS Installation Guide*. See also the “Sequence Number” section of the *CPCS Programming and Diagnostic Guide* for detailed documentation of the entire 12-digit item sequence number field. If region numbers are used:

- The user decides what they identify.
- Items captured under a non-XF sort type do not have the region number stamped as part of their sequence numbers.

Describing an Application Task's Environment

You can change the APCB macro keywords for performance, security, or control reasons. The APCB macros must be in alphabetical sequence by program module and can be found in DKNBLDL. Descriptions of the APCB keywords (operands) follow this table:

Name	Operation	Operands
SYMBOL	APCB	[NAME=*task name*] [AUTH={**1** | **2** | **3** | **4** | **5**}] [AUTO={**0** | **1**}] [CIMS={**0** | **1**}] [CLASS=*output class*] [COMP={**0** | **1**}] [DPCOD={**0** | *dispatching priority modifier*}] [ECYEND={**0** | **1**}] [ECYRTN={**0** | **1**}] [EXSEQ={**1 - 255**}] [EXTSK={**Y** | **N**}] [HIVOL={**0** | **1**}] [HLOG={**0** | **1**}] [ICLSS={**I** | *application class value*}] [INSTRG={**0** | *number of MDS input strings open concurrently*}] [ISPOOL={**000** | *number of document print files*}] [ISUPV={**0** | **1**}] [MAIL={**N** | **Y**}] [MAX={**1** | *number of concurrent runs permitted*}] [MAXM={**1** | *pages of above-the-line storage*}] [MSUPV={**0** | **1**}] [OUTSTRG={**0** | *number of MDS output strings open concurrently*}] [PRINT={**0** | *number of spool data sets required*}] [SMMULT={**0** | **1**}] [SMSTART={**0** | **1**}] [SMTRACK={**0** | **1**}] [SORT={**0** | **1**}] [SPEC={**0** | **1**}] [SSUPV={**0** | **1**}] [TASKGRP={**0** | *1-digit group number 0-9* }] [TIMECK={**0** | **1**}] [USREXIT=*optional user exit name*] [USRPARM={**X**}] [WORK=*amount of work storage required*] [WRKPOOL={**0** | **1** | **2**}]

NAME=*application task name*

Specifies the name of the application task. The first 3 characters of the name are user-variable; the last 4 characters are the transaction ID and must be unique. The names in the BLDL table must be in alphabetic sequence on the full length.

Note: You no longer need to use DKN as the first 3 characters of the name.

AUTH={1 | 2 | 3 | 4 | 5}

Specifies the authority class (0, 1, 2, 3, 4, or 5) of the task. This operand has meaning only if you selected the data security (DASC) option in the master task

generation. See the *CPCS Programming and Diagnostic Guide* for more information on the security option. If you do not specify `AUTH`, the task defaults to `AUTH=1`.

Note: For executive tasks, the value for AUTH always defaults to `AUTH=1`.

AUTO={0 | 1}
Specifies whether another task or an operator at a CPCS terminal can start this task.

AUTO=0 Either an operator (manually) or another task (automatically) can start this task (default value).

AUTO=1 Another task must start this task (automatically).

CIMS={0 | 1}
Specifies whether the task that is associated with this entry uses the services of the Check Image Management System (CIMS), an IBM licensed program that stores, retrieves, and manages document images.

CIMS=0 This task does not use CIMS services (default value).

CIMS=1 This task uses CIMS services. Specify `CIMS=1` only if you are running the High Performance Transaction System and the task will access the CIMS database.

CLASS=*output class*
Specifies the CPCS output class (A through Z or 0 through 9) of any printed output produced by the task. The CLASS parameter of the CPCS system profile member DKNPCPCS parameter determines the default value. For more information about the DKNPCPCS profile member, see "DKNPCPCS Profile Member" on page 3-4. Using this operand and the printer control capabilities of the DKNFORM task lets you direct output to specific printers. For example, you can direct the kill and cash-letter summary lists to one printer and the balancing lists to another printer. With job-entry subsystem (JES) print support, the last character of the JESPRT DD name is assigned as the CPCS printer class.

Note: If you make this last digit the same as the SYSOUT class of the CPCS JES printer, you can locate the print buffer data set more easily. See "Dynamically Allocating Data Sets" on page 2-23 for details.

COMP={0 | 1}
Specifies whether the task can run while the DKNCOMP task is running.

COMP=0 This task does not use DKNKB and DKNMF, so it *can* run while the DKNCOMP task is running (default value).

COMP=1 This task uses DKNKB or DKNMF, so it *cannot* run while the DKNCOMP task is running. Specify `COMP=1` for any task that requires the use of the kill bundle or microfilm data sets (except DKNCOMP).

DPCOD={0 | *dispatching priority modifier***}**
Specifies the task-dispatching priority modifier, which DKNATASK passes to MVS when it attaches the task.

DPCOD=0 This task has the highest dispatching priority (default value).

DPCOD=*dispatching priority modifier*

This value should be a negative number (-1, -2,...), with -1 having a higher dispatching priority than -2, followed by -3, -4, and so forth.

You can use this operand to optimize the performance of CPCS tasks. For example, when you specify `DPCOD=0` for DKNOLRR and `DPCOD=-1` for DKNMRG2, DKNPLST, and DKNDIST, you give a higher dispatching priority to DKNOLRR than to higher processing-unit usage programs. This priority is important for programs such as DKNOLRR that have a low processing-unit usage and a critical response requirement.

ECYEND={0 | 1}

Specifies whether this task is the last task in the end cycle series of tasks.

ECYEND=0 This is not the last task in end cycle processing (default value).
ECYEND=1 This is the last task in end cycle processing.

ECYRTN={0 | 1}

Specifies whether this task is part of the end cycle series of tasks.

ECYRTN=0 This is not an end cycle processing task (default value).
ECYRTN=1 This is an end cycle processing task.

EXSEQ={1–255}

Specifies the sequence in which DKNATASK will activate user executive tasks (EXTSK=Y). This parameter is valid only for auto-started tasks (AUTO=1). The default value is EXSEQ=50.

EXTSK={Y | N}

Specifies whether the associated task is a user executive task. User executive tasks can be manually or automatically started.

EXTSK=Y This is a user executive task.
EXTSK=N This is not a user executive task (default value).

HLOG={0 | 1}

Specifies whether DKNLOADR posts the host logon process.

HLOG=0 DKNLOADR does not post the host logon process. This is standard CPCS treatment (default value).

HLOG=1 DKNLOADR posts the host logon process when a task is successfully attached.

HIVOL={0 | 1}

Specifies whether the task requires a large spool data set for printed output.

HIVOL=0 This task does not require a large spool data set. The small printer spool data sets should be large enough to contain the largest printed output of this task (default value).

HIVOL=1 This task requires a large spool data set.

Notes:

1. If the task requires more than one spool data set, only one can be a large spool data set. When an application program calls DKNADCB to obtain spool data sets, the large spool data set should appear first in the parameter list.
2. This parameter has no effect on task output that prints through CPCS spools with the DKNADCB and DKNADCB2 macros. Use the CLASS parameter instead.

ICLSS={I | *application class value*}
Specifies the application class for printing directly to the JES spool. Application class values are 0 through 9 or A through Z. The default value is ICLSS=I.

INSTRG={0 | *number of MDS input strings open concurrently*}
Specifies the maximum number of MDS input strings that can be open concurrently for the task. If the number of concurrently open input strings is greater than this specification, the task will abend. This specification must not be too large, or the task might exceed the 64K maximum work area. The default value is INSTRG=0.

ISPOOL={000 | *number of document print files*}
Specifies the maximum number of print files, which can be a 3-digit number between 000 and 255. DKNATASK must assign as many ddnames for print files as this field indicates. The default value is ISPOOL=000.

Note: This parameter has no effect on task output that prints through CPCS spools via the DKNADCB and DKNADCB2 macros. Use the CLASS parameter instead.

ISUPV={0 | 1}
Specifies whether the INSC supervisor terminal must start the task.

ISUPV=0	This task can start on any terminal (default value).
ISUPV=1	This task can be started only on the INSC supervisor terminal.

MAIL={N | Y}
Specifies whether a module can receive electronic mail.

MAIL=N	This module cannot receive electronic mail (default value).
MAIL=Y	This module can receive electronic mail.

MAX={1 | *number of concurrent runs permitted*}
Specifies the maximum number of concurrent runs of this task that CPCS permits. The default value is MAX=1. You can use this operand to increase or decrease the maximum number for more flexibility and performance tuning.

Note: DKNCLSM is an example of a CPCS task that requires MAX=1 in its APCB macro in DKNBLDL, because DKNCLSM uses a single work data set.

MAXM={1 | *pages of above-the-line storage*}
Specifies the maximum number of pages of above-the-line storage that an application can request from DKNMEMI. The input value is between 1 and 500 000. Each page represents 4096 bytes of storage. The default value is MAXM=1. For more information about the storage manager, see the *CPCS Programming and Diagnostic Guide*.

MSUPV={0 | 1}
Specifies whether the MICR supervisor terminal must start this task.

MSUPV=0 This task can start on any terminal (default value).
MSUPV=1 This task can be started only on the MICR supervisor terminal.

OUTSTRG={0 | *number of MDS output strings open concurrently***}**
Specifies the number of MDS output strings concurrently open for normal performance of the program that is associated with this APCB macro. Unlike INSTRG, if more output strings are concurrently open than are specified here, the task does not abend; but some storage fragmentation might occur. If fewer than the specified output strings are concurrently open, the extra storage that DKNATASK obtains is unused. This specification must not be too large, since the task can exceed the 64K maximum. The default value is OUTSTRG=0.

PRINT={0 | *number of spool data sets required***}**
Specifies the number of spool data sets required to run this task. The maximum number that you can specify is 255. The default value is PRINT=0.

Note: This parameter has no effect on task output that prints directly to JES instead of with the DKNADCB3 macro. Use the ISPOOL parameter instead.

SMMULT={0 | 1}
Specifies whether one execution of this task may complete System Manager work request blocks.

SMMULT=0 When one copy of this task is executed, only one Enhanced System Manager WRB is marked "active" and subsequently completed (shown with a 'C'). The majority of CPCS tasks fall into this category.

SMMULT=1 When one copy of this task is executed, several Enhanced System Manager WRBs may be marked "active" and subsequently completed (shown with a 'C').

Specifying SMMULT=1 enables Enhanced System Manager to anticipate this scenario and process this type of work correctly.

Note: See the *CPCS Enhanced System Manager User's Guide* for more information.

SMSTART={0 | 1}
Specifies whether the task may be automatically started by the System Manager.

SMSTART=0 This task *cannot* be automatically started by System Manager.

SMSTART=1 This task *can* be automatically started by System Manager.

Note: See the *CPCS Enhanced System Manager User's Guide* for more information.

SMTRACK={0 | 1}
Specifies whether the task may be tracked but may *not* be started by the System Manager.

SMTRACK=0 This task *cannot* be automatically started or tracked by System Manager.

SMTRACK=1 This task *can* be tracked but *cannot* be started by System Manager.

Note: See the *CPCS Enhanced System Manager User's Guide* for more information.

SORT={0 | 1}
Specifies whether the task uses an internal sort.

SORT=0 This task does not use an internal sort (default value).
SORT=1 This task uses an internal sort.

SPEC={0 | 1}
Specifies whether the task requires special testing before DKNATASK can attach the task.

SPEC=0 This task requires no special testing (default value).

SPEC=1 This task requires special testing (for example, DKNCOMP and DKNECYC require special testing).

SSUPV={0 | 1}
Specifies whether the SYST system supervisor terminal must start the task.

SSUPV=0 This task can start on any terminal (default value).

SSUPV=1 This task can be started only on the SYST system supervisor terminal.

TASKGRP={0 | *1-digit group number 0–9***}**
Specifies the task group number for the task. Similar tasks may be grouped with the same task group number to restrict how many task of that group may be active at one time.

This is related to the System Profile data set member DKNPATSK, which is described in Chapter 3, "System and Application Profiles." Group 0 will cause the task to be started immediately if MAXTASK is not exceeded. Group 0 is the default if this parameter is omitted.

TIMECK={0 | 1}
Specifies whether DKNASGF automatically logs off the task.

TIMECK=0 This task is available for automatic logoff if it is inactive for a period of time that exceeds the period specified in the CPCS system profile member DKNPCPCS.

TIMECK=1 DKNASGF must *not* automatically log off this task *even* if the task is inactive for a period of time that exceeds the period specified in the CPCS system profile member DKNPCPCS.

USREXIT=*optional user exit name*
Specifies the name of a user-written exit routine. This is not a required parameter. When you use this parameter, the name of the exit must be no more than 8 characters in length; the first character must be alphabetic and the rest must be alphanumeric. Generally, user exits named in DKNBLDL are used for default bank 001 only. The bank control file and/or the sort pattern definition may name other user exits that override the user exit named here.

USRPARM={X}
Used by DKNICRE and DKNMRGE.

DKNICRE Specifies whether DKNICRE should extract an uncompressed data set. When you code X for this parameter, DKNICRE extracts data sets using the copybook DKNCRINA, an uncompressed format based on the extract field lengths in the MDX macro. If you do not code this parameter, DKNICRE uses the default copybook DKNCRIN, which has a compressed format.

| Change the record length of the ICRE file in the DKNGALLO JCL and in the DSAT to match the MDX (GENCLIB JCL) output, matching the ICRA note. Do not change the ddname.

DKNMRGE Specifies the threshold of uncorrected errors allowed when MRGE is run. If this threshold is met, MRGE 24 is issued and MRGE terminates. This may or may not indicate an error. MRGE inspects all flag bits including the flag bits for fields 9–15. If it can be determined that the suspect OLRR user edit processing the items correctly *but* flagging items incorrectly, then the error message can be eliminated by coding USPARM=9999. This does not fix the error, but eliminates the message, as a threshold of 9999 will probably not be exceeded.

WORK=*amount of work storage required*

Specifies the amount of user work storage that the task requires (applies to assembler programs only, usually reentrant).

Note:

WORK acquires 24-bit, below-the-line storage.

For expanded MDS records you must increase the work area for DKNOLRR by 50 bytes for every byte of MDS expansion.

Note: Before DKNATASK attaches a task, it must acquire a work area equal to the sum of the WORK parameter and the size of the APTCB and MDS control blocks and buffers required for INSTRG and OUTSTRG. The maximum size of this work area is 64K. You should specify the correct number of strings for INSTRG and OUTSTRG. If you are using blocked BDAM, you might need to increase the WORK= value for DKNDIST. Failing to do this results in an SOC4 abend in DKNDIST. The default value for this is 25,000. For maximum blocking factors, a value of 40,000 is sufficient.

WRKPOOL={0 | 1 | 2}

Specifies whether the task uses an internal work area (for example, sort work areas).

WRKPOOL=0 This task does not use an internal work area (default value).
WRKPOOL=1 This task uses an internal work area assigned outside subpool 0.
WRKPOOL=2 This task uses an internal work area assigned to subpool 0.

Examples of Adding a DKNBLDL Entry for a New Task: This following example shows you how to add a system supervisor task that the transaction identification WXYZ can start. The task reads one MDS input string and writes a report. Only one of these tasks can be run at a time.

```
APCB    NAME=DKNWXYZ,PRINT=1,INSTRG=1,SSUPV=1
```

The following example shows an application task that another task starts automatically:

```
APCB  NAME=DKNDEMO,AUTO=1,PRINT=1,ECYRTN=1,COMP=1,OKONDUMP=1, X
      MAIL=Y,ICLSS=I,ISPOOL=001,MAXM=2
```

The task, DEMO, is part of the end cycle process. It is authorized to receive messages in the electronic mailbox from other applications. In addition to a regular report, the DEMO task includes routines that use class I to process and send reports to a JES printer. The task uses memory above the line to store the records. This task cannot be called if DKNCOMP is running, and DKNCOMP cannot start if this task is running.

Dynamically Allocating Data Sets

This section explains how to use the DSAT macro to describe a dynamic allocation data set in DKNDSAT. For each dynamically allocated data set, there must be a DSAT macro.

Name	Operation	Operands
[symbol]	DSAT	DYNDEV={**TAPE** | **DISK** | **3890** | **PRTR**}
		,DYNDDN=*ddname*
		,DYNDSN=*dsname*
		,DYNSTAT={**OLD** | **MOD** | **NEW** | **SHR**}
		,DYNNORM={**UNCATLG** | **CATLG** | **DELETE** | **KEEP**}
		,DYNCOND={**UNCATLG** | **CATLG** | **DELETE** | **KEEP**}
		,DYNUNIT=*unit group name*
		[,DYNIO={**INPUT** | **OUTPUT**}]
		[,DYNPVI=**PRIVATE**]
		[,DYNVLSR=*volume serial number*]
		[,DYNEXPR=*Julian expiration date*]
		[,DYNRTEN=*data-set retention period*]
		[,DYNLRCL=*logical record length*]
		[,DYNBSIZ=*block size*]
		[,DYNLABL={**NL** | **SL** | **NSL** | **SUL** | **BLP** | **LTM** | **AL** | **AUL**]
		[,DYNSEQ=*tape sequence number*]
		[,DYNDEN={**0** | **1** | **2** | **3** | **4**}]
		[,DYNSTYP={**TRK** | **CYL** | **BLK**}]
		[,DYNBLKS=*average block size*]
		[,DYNPSPA=*amount of primary DASD space allocated*]
		[,DYNSSPA=*amount of secondary DASD space allocated*]
		[,DYNFCB=*forms control buffer identification*]
		[,DYNUCS=*universal character set name*]
		[,DYNDCB=*dsname*]
		[,DYNOPT={**B** | **H**}]
		[,DYNVLCT=*volume count*]
		[,DYNRECF=*record format*]
		[,DYNRLSE={**0** | **RLSE**}]
		[,DYNCLASS=*alphanumeric*]
		[,DYNOUT={*name1* | (*name1,name2,name3*)}]
		[,DYNCLOSE={**N** | Y:}]
		[,DYNBUFNO={0 | NUMBER FROM 1 TO 255}]
		[,DYNSMSSC={SMS Storage Class}]
		[,DYNSMSMC={SMS Management Class}]
		[,DYNSMSDC={SMS Data Class}]
		[,DYNSMSLI={SMS *model dsname*}]
		[,DYNSMSAV={U | K | M}]

The following list describes the DSAT parameters.

DYNDEV={TAPE | DISK | 3890 | PRTR}
Specifies the type of data set to allocate (tape, disk, 3890 channel-attached document processor, or printer).

DYNDDN=*ddname*
Specifies the data-definition name (ddname) that the calling program uses to reference the dynamically allocated file. Specify 1 to 8 characters, according to the JCL conventions for data definition names.

DYNDSN=*dsname*
Specifies the data-set name that CPCS uses for the dynamically allocated file. Use 1 to 44 characters, following the JCL conventions for data-set names.

DYNSTAT={OLD | MOD | NEW | SHR}
Specifies the data-set status of old, modified, new, or shared. If DYNDEV=DISK, and the status is not NEW, CPCS ignores the DYNBLKS, DYNPSPA, and DYNSSPA parameters.

DYNNORM={UNCATLG | CATLG | DELETE | KEEP}
Specifies the normal disposition of the allocated data set. The options are:

UNCATLG	Uncatalog the data set when the program ends.
CATLG	Catalog the data set when the program ends.
DELETE	Delete the data set when the program ends.
KEEP	Keep the data set when the program ends.

DYNCOND={UNCATLG | CATLG | DELETE | KEEP}
Specifies the conditional disposition of the data set. The options are:

UNCATLG	Uncatalog the data set when the program ends abnormally.
CATLG	Catalog the data set when the program ends abnormally.
DELETE	Delete the data set when the program ends abnormally.
KEEP	Keep the data set when the program ends abnormally.

DYNUNIT=*unit group name*
Specifies a unit group name as defined by the installation (for example, SYSDA, TAPE9). If you specify DYNDEV=3890, the character length is 6; otherwise, the character length is 8.

DYNIO={INPUT | OUTPUT}
Specifies whether CPCS allocates the data set as input-only or output-only. This is an optional parameter.

DYNPVI=PRIVATE
Specifies a data set as PRIVATE. This is an optional parameter.

DYNVLSR=*volume serial number*
Specifies the volume serial number. Use 6 characters following the JCL conventions for volume serial numbers.

DYNEXPR=*Julian expiration date*
Specifies the Julian expiration date for the file. The parameter may be 5 digits (*yyddd*) or 7 digits (*yyyyddd*) depending on the settings in copybook EXPDTV. This is an optional parameter. It is mutually exclusive with DYNRTEN.

DYNRTEN=*data-set retention period*
Specifies the data-set retention period. Use 1 to 4 numeric characters. This is an optional parameter. It is mutually exclusive with DYNEXPR.

DYNLRCL=*logical record length*
Specifies the actual or maximum length (in bytes) of a logical record. This is an optional parameter.

DYNBSIZ=*block size*
Specifies the length (in bytes) of a block. The maximum size depends on the device type. This is an optional parameter.

DYNLABL={NL | SL | NSL | SUL | BLP | LTM | AL | AUL}
Specifies the tape label processing that CPCS performs on the allocated data set. The options are:

NL	No label on the tape
SL	IBM standard label on the tape
NSL	Nonstandard label on the tape
SUL	IBM standard label and user label on the tape
BLP	Bypass label processing
LTM	Bypass leading tape mark on an unlabeled tape
AL	ANSI standard label on the tape
AUL	ANSI standard label and ANSI user label on the tape.

DYNSEQ=*tape sequence number*
Specifies the tape sequence number. This is an optional parameter.

DYNDEN={0 | 1 | 2 | 3 | 4}
Specifies the tape density setting. Options are 0 through 4, where:

0	200 bpi	(7-track tape only)
1	556 bpi	(7-track tape only)
2	800 bpi	
3	1600 bpi	(9-track tape only)
4	6250 bpi	(9-track tape only).

This is only a required parameter if you specified the TAPE option in DYNDEV; otherwise, this is an optional parameter.

DYNSTYP={TRK | CYL | BLK}
Defines the type of space allocation for new disk data sets. Valid options are tracks (TRK), cylinders (CYL), or blocks (BLK).

DYNBLKS=*average block size*
Specifies the average block size of the data set if DYNSTYP=BLK.

DYNPSPA=*amount of primary DASD space to allocate*
Specifies the amount of primary DASD space to allocate to the new disk data set.

DYNSSPA=*amount of secondary DASD space to allocate*
Specifies the amount of secondary DASD space to allocate to the new disk data set.

DYNFCB=*forms control buffer identification*
Specifies the forms control buffer record identification. Use 1 to 4 characters.

DYNUCS=*universal character set name*
Specifies the universal character set, such as PCAN or PCHAN. Use 1 to 4 characters.

DYNDCB=*dsname*
Specifies the referenced data-set name. This is an optional parameter.

DYNOPT={B | H}
Specifies the optional service code; only B and H are permitted. This is an optional parameter. This optional service code applies only to dynamically allocated tape data sets for the basic sequential access method (BSAM) or to the queued sequential access method (QSAM). For more information about this option, see the *MVS/ESA Data Administration: Macro Instruction Reference*.

DYNVLCT=*volume count*
Specifies the volume count.

DYNRECF=*record format*
Specifies the record format. The options are:

A Records contain ANSI control characters
B Records are blocked
D Data set contains variable length ISCII/ASCII tape records
F Record size is fixed
M Records contain machine-code control characters
S Data set contains spanned records
T Data set accommodates track overflow
U Record format is undefined
V Record size is variable.

For valid combinations of these options, see the *MVS/ESA Data Administration: Macro Instruction Reference*.

DYNRLSE={0 | RLSE}
Specifies the release (RLSE) of unused DASD space. This is an optional parameter.

DYNCLASS=*alphanumeric*
Specifies the SYSOUT class for JES printers.

DYNOUT={*name1* **| (***name1,name2,name3***)}**
Specifies one to three names that refer to an output record defined in the CPCS run JCL. Each name can be 1 through 8 characters in length.

DYNCLOSE={N | Y}
Specifies if the ddname is to be unallocated when the data set is closed.

DYNBUFNO={0 | NUMBER FROM 1 TO 255}
Specifies the number of QSAM buffers to be used for this data set. A value of 0 indicates this option is not to be used.

DYNSMSSC={SMS Storage Class}
Specifies the SMS Storage Class to be assigned to this data set. Use a 1- to 8-character, SMS-defined storage class name. DYNUNIT is optional when DYNSMSSC is specified.

DYNSMSMC={SMS Management Class}
Specifies the SMS Management Class to be assigned to this data set. Use a 1- to 8-character, SMS-defined management class name.

DYNSMSDC={SMS Data Class}
Specifies the SMS Data Class to be assigned to this data set. Use a 1- to 8-character, SMS-defined data class name.

DYNSMSLI={SMS Model Data Set Name}
Specifies the SMS Model Data Set Name to be used for this data set. Use a 1- to 44-character, SMS-defined model data set name.

DYNSMSAV={U | K | M}
Specifies the SMS average record scale modifier to be used for this data set.

Example of Coding the DSAT Parameter DYNOUT: The following examples show you how to code OUTPUT records for the CPCS run JCL and corresponding DSAT macro entries for DKNDSAT. For more information about using the OUTPUT operands, see the *MVS/ESA JCL User's Guide*.

Example 1: To route the same print job to multiple destinations with the same output class, add the following statements to the CPCS run JCL:

```
//CPCSP1  OUTPUT COPIES=1,DEST=PRT1   CPCS PRINT TO LOCATION 1
//CPCSP1  OUTPUT COPIES=1,DEST=PRT2   CPCS PRINT TO LOCATION 2
```

Add the following entry to the DKNDSAT data set:

```
JES  DSAT  DYNDEV=PRTR,              always PRTR for printer   X
           DYNDDN=JESPRT1C,          DDNAME                    X
           DYNCLASS=C,               SYSOUT class              X
           DYNOUT=(CPCSP1,CPCSP2)    OUTPUT record names
```

All printing is routed to class C, with destinations of PTR1 and PTR2.

Example 2: The following example shows how to code the JCL and the DSAT macro for different output classes and destinations. This example assumes that you have an online microfiche printer and are printing on a standard printer and the microfiche printer.

Add the following OUTPUT records to the CPCS run JCL:

```
//CPCSP   OUTPUT COPIES=1,CLASS=E,DEST=PRT1   REGULAR CPCS PRINT
//FICHE   OUTPUT COPIES=1,CLASS=F,DEST=FICHE  CPCS PRINT TO MICROFICHE
```

Add the following entry to the DKNDSAT data set:

```
JES  DSAT  DYNDEV=PRTR,              always PRTR for printer   X
           DYNDDN=JESPRT1A,          DDNAME                    X
           DYNOUT=(CPCSP,FICHE)      OUTPUT record names
```

Note: The DYNCLASS parameter is not necessary because the OUTPUT record in the JCL specifies the output class.

Example of Adding a DKNDSAT Entry for a New Tape Data Set: To dynamically allocate a tape data set, you should assemble the following DSAT macro in DKNDSAT. Set up the parameters according to your standards.

```
EXTR     DSAT DYNDEV=TAPE,          tape data set              X
              DYNDDN=USEREXTR,      user's DCB DDNAME          X
              DYNDSN=EXTRACT,       user's data set name       X
              DYNSTAT=NEW,          new allocation             X
              DYNNORM=KEEP,         KEEP normal disposition    X
              DYNCOND=DELETE,       DELETE error disposition   X
              DYNVLSR=123456,       VOLSER for allocation      X
              DYNEXPR=99365,        expiration date            X
              DYNUNIT=TAPE9,        allocate from 9 track pool X
              DYNLABL=SL,           standard label processing  X
              DYNSEQ=1,             data set sequence number   X
              DYNDEN=3              1600 bpi 9 track
```

Example of Adding a DKNDSAT Entry for a Disk Data Set: Use the following DSAT macro to add a dynamically-allocated data set for an extract system:

```
EXTR     DSAT DYNDEV=DISK,          disk data set              X
              DYNDDN=USREXT1,       user's DCB DDNAME          X
              DYNDSN=EXTR1.DSK,     user's data set name       X
              DYNSTAT=NEW,          status is NEW              X
              DYNNORM=KEEP,         KEEP normal disposition    X
              DYNCOND=DELETE,       DELETE error disposition   X
              DYNVLSR=CPCS01,       volume serial number       X
              DYNUNIT=SYSDA,        on-line disk               X
              DYNLRCL=63,           logical record length      X
              DYNBSIZ=630,          maximum block size         X
              DYNSTYP=CYL,          cylinder space allocation  X
              DYNPSPA=1,            primary space              X
              DYNSSPA=1             secondary space
```

Example of Adding a DKNDSAT Entry for a Document Processor: Use the following DSAT macro to dynamically allocate a channel-attached document processor, using DKNALLO, through terminal commands:

```
RS38901  DSAT DYNDEV=3890,          3890 is device           X
              DYNDDN=RS3890I1,      DDNAME                   X
              DYNUNIT=01B-01,       device address/logical #  X
              DYNLRCL=44,           logical record length    X
              DYNBSIZ=704           maximum block size
```

Note: The DYNUNIT parameter is in the format of *CUU-nn*, where *CUU* is the device address and *nn* is the logical device number. The logical device number is the same number that the MICR OPEN command uses.

If, along with the above 3890 DSAT entry, there is also a requirement for a logical sorter 2 that uses the same unit address, then you could specify another 3890 DSAT macro entry as follows:

```
RS38902      DSAT   DYNDEV=3890,          X
                    DYNDDN=RS3890I2,      X
                    DYNUNIT=01B-02,       X
                    DYNLRCL=44,           X
                    DYNBSIZ=704
```

At any one time, DKNALLO can allocate only one logical device for each physical unit.

Example of Coding a DKNDSAT Entry for a Printer: Use a DSAT macro to dynamically allocate a printer, using DKNALLO, through terminal commands.

```
PRNTER   DSAT DYNDEV=PRTR,          always PRTR for a printer X
              DYNDDN=PRINT1,        DDNAME                    X
              DYNUNIT=01E,          device address            X
              DYNLRCL=133,          logical record length     X
              DYNBSIZ=133,          maximum block size        X
              DYNFCB=xxxx,          forms control buffer      X
              DYNUCS=yyyy           universal character set
```

Substitute your system standard values for the values *xxxx* and *yyyy*.

To dynamically allocate this device, use the same procedure shown in “Example of Adding a DKNDSAT Entry for a Document Processor” on page 2-28. Use 01E as the device address in the DKNALLO commands. Increase the NUMPTR parameter in the CPCS system profile member DKNPCPCS, if required.

Example of Coding a DKNDSAT Entry for a JES Printer: Use the DSAT macro that follows to dynamically allocate a SYSOUT data set through DKNWTR for printing:

```
JES   DSAT  DYNDEV=PRTR,        always PRTR for printer      X
            DYNDDN=JESPRT1A     DDNAME                       X
            DYNCLASS=A,         SYSOUT CLASS (optional)      X
            DYNOUT=OUT1         OUTPUT CARD NAME (optional)
```

The DYNDDN statement has the following requirements:

- The first 6 characters of the ddname must be JESPRT.
- The seventh character can be any user-selected digit that makes the ddname unique.
- The eighth character is the CPCS internal class specification. This option lets application tasks that use spools have an assigned CPCS class (in DKNBLDL) that is the same as a JESPRT SYSOUT class. This removes the need to use FORM operands to direct output to the desired JES printer class. For information on DKNBLDL and the CLASS parameter, see “Describing an Application Task’s Environment” on page 2-15.

  The eighth digit can also be the same as the SYSOUT class.

Increase the NUMPTR parameter in the MDEF macro, if required.

Example of Adding a DKNDSAT Entry for a Unique Temporary Disk Data Set:
Use the following DSAT macro to add a temporary-work data set for an extract system:

```
EXTR   DSAT DYNDEV=DISK,      disk data set               X
            DYNDDN=USREXT1,   user's DCB DDNAME           X
            DYNDSN=EXTR1.DSK, user's data set name        X
            DYNSTAT=NEW,      status is NEW               X
            DYNNORM=DELETE,   DELETE normal disposition   X
            DYNCOND=DELETE,   DELETE error disposition    X
            DYNVLSR=CPCS01,   volume serial               X
            DYNUNIT=SYSDA,    on-line disk                X
            DYNLRCL=63,       logical record length       X
            DYNBSIZ=630,      maximum block size          X
            DYNSTYP=CYL,      cylinder space allocation   X
            DYNPSPA=1,        primary space               X
            DYNSSPA=1         secondary space
```

Note: The data-set name must be fewer than 35 characters in length. This accommodates the 9-character time stamp suffix that DKNTDYNA appends.

Nonswap SVC Generation Task

Sample input to the nonswap macro is shown as it is supplied in the SAMPSVC member of CPCS. The nonswap macro uses this input to create an SVC that marks the task and steplibs as “non-swappable” when the SVC is invoked from the CPCS master task. The keyword operands contained in this sample input should be changed to suit your installation’s needs. The input is in macro assembler format. The nonswap macro checks the operands for valid values. The keywords are:

SVC The desired SVC number. Must be numeric, between 200 and 255. If omitted, the number defaults to 203. Multiple operands are ignored.

PROG The authorized program name must be DKNMTASK.

LIBS The names of the data sets marked non-swappable. They must be 44 characters or less and must conform to MVS data set name rules. Multiple operands are allowed.

```
NONSWAP SVC=203,
      PROG=DKNMTASK,
      CPCS.V1R11.LLIB,
      CPCS.V1R11.TEST.LLIB
END
```

The nonswap macro creates the proper type 4 SVC CSECT name based on the SVC number specified. However, due to assembler and linkage editor limitations, a default member name for the SVC load module must be used. On successful completion, an MNOTE is issued during assembly, giving the proper SVC member name. You should rename the load module “TEMPSWAP” to the name issued in the MNOTE.

Concurrent Sort Generation

When you specify concurrent sorting (with MAXSORT) in the CPCS system profile member DKNPCPCS, you must create a SORT load module by assembling the CCSDEF macro.

Important!

If you code any SORTWK data sets in the CPCS run JCL, CPCS startup disables concurrent sorting.

This macro defines the sort-work data sets and the sort-message data set used by each application task that requires sorting functions. CCSDEF has no defaults. The format of the CCSDEF macro is as follows:

Name	Operation	Parameters
[symbol]	CCSDEF	NUMWORK=*n* ,{TRACKS=*nnn*} ,{CYLS=*nnn*} ,UNIT=*xxxxxxxx* ,VOLSER=*xxxxxx* ,SYSOUT=*x* ,SORTNAM=*xxxxxxxx* ,SORTMSG={**Y**, **N**}

The following list describes each CCSDEF parameter:

NUMWORK=*n*
Specifies the number of sort-work data sets to dynamically allocate for each task. A valid value is any number from 1 through 9.

TRACKS=*nnn*
Specifies the number of tracks allocated to each work data set. All sort-work data sets are the same size. This parameter and the CYLS parameter are mutually exclusive. Any numeric value is acceptable.

CYLS=*nnn*
Specifies the number of cylinders allocated to each sort-work data set. All sort-work data sets are the same size. This parameter and the TRACKS parameter are mutually exclusive. Any numeric value is acceptable.

UNIT=*xxxxxxxx*
Specifies the generic unit type to which the sort-work data sets are allocated. All sort-work data sets are allocated on this unit type. Specify any valid unit type up to 8 characters in length.

VOLSER=*xxxxxx*
Specifies a volume serial number to which the sort-work data sets are allocated. All sort-work data sets are allocated on this volume. This parameter is optional. A valid length is 1 to 6 characters.

SYSOUT=*x*

Specifies the SYSOUT class for the sort message data sets. Specify any valid SYSOUT class.

Note: If you do not want the SORT control statements and statistics output generated, specify your system's discard class value.

SORTNAM=*xxxxxxxx*

Specifies an alias name for your system's sort program. Ask your system programmer what names you can use for your program (other than SORT).

Note: Concurrent sort cannot be used if your installation does not use an alias for SORT.

SORTMSG={Y, N}

Specifies whether the sort module created from CCSDEF should allocate a sort output data set. To eliminate sort messages, you must pass appropriate parameters to your actual sort program. For example, using DFSORT, eliminate the sort output messages by having the DFSPARM data set specify:

```
'OPTION NOLIST, NOLISTX, MSGPRT=NONE'
```

MICR Task Generation

You perform the CPCS MICR task generation by assembling and link-editing a set of user-prepared, MICR task-generation macros.

The purpose of this assembly is to:

- Enable the user to designate specific installation options
- Construct the permanently resident MICR task control blocks and tables.

You can perform the assembly, using the procedures described in the *CPCS Installation Guide*. There are two MICR task-generation macros for which you select keyword parameters. These macros are CPCSRDR and CPCSOPTN.

You must specify at least one CPCSRDR macro for each 3890 or 3890/XP Series Document Processor that CPCS operates. The keyword parameters for the CPCSRDR macro are defined in "CPCSRDR Macro Parameters" on page 2-37.

Important!

- All CPCSRDR macros must precede the CPCSOPTN macro.
- If adding CPCSRDR macros or changing existing CPCSRDR macros changes the maximum number of stackers, you must perform a CPCS cold start.

  If the additions or alterations do not change the maximum number of stackers, you must either perform a CPCS cold start or use the HALTMICR and STRTMICR commands. For more information on CPCS cold start, see "STYPE Parameter" on page 1-7. For a description of the HALTMICR and STRTMICR commands, see the *CPCS Terminal Operations Guide*.

Use the JCL in member DKNGMICR of CPCS.V1R11.CTRL to generate DKNMICR for CPCS. Figure 2-5 on page 2-34 is an example of the member DKNMGEN, which is used as the SYSIN data set in the MGEN step of the DKNGMICR job.

Note: The CSBU task uses RDR99. If you are not using DKNCSBU, the RDR99 entry is not required.

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

Figure 2-5 (Part 1 of 3). Example of Macros for DKNMICR Generation

```
*        READER SORTER 6
*
RDR6     CPCSRDR TYPE=3890-2,ATTACH=SIM,DDRDRIN=SORTER06,   -
               MODEL=XP,MFILM=YES,PEND=YES
*
************************************************************
*        READER SORTER 8
*
RDR8     CPCSRDR TYPE=3890-1,ATTACH=SIM,DDRDRIN=SORTER08,   -
               LDPN=08,                                     -
               MODEL=A,MFILM=YES,ENDORSE=YES,INF=YES
*
************************************************************
*        READER SORTER 9
*
RDR9     CPCSRDR TYPE=3890-1,ATTACH=SIM,DDRDRIN=SORTER09,   -
               MODEL=A,MFILM=YES,ENDORSE=YES,INF=YES
*
************************************************************
*        READER SORTER 10
*
RDR10    CPCSRDR TYPE=3890-6,ATTACH=EMICR,DDRDRIN=SORTER10, -
               LDPN=10,                                     -
               MODEL=A,MFILM=NO
*
************************************************************
*
*----------------------------------------------------------*
*    ADD ALL NEW READER SORTERS HERE                       *
*----------------------------------------------------------*
*
************************************************************
*        READER SORTER 99
*
RDR99    CPCSRDR TYPE=3890-6,ATTACH=SIM,DDRDRIN=SORTER99,   -
               MODEL=XP,MFILM=YES,IMAGE=YES,LDPN=99
*
************************************************************
```

Figure 2-5 (Part 2 of 3). Example of Macros for DKNMICR Generation

```
        EJECT
*************************************************************
*        CPCS OPTIONS
*
OPTIONS  CPCSOPTN TRACER=(5555-5555,5),                     -
              BLOCK=(5533-3333,5),                         -
              BATCH=(5566-6666,5),                         -
              SBATCH=(5599-9999,5),                        -
              DIVIDER=(4666-6666,5),                       -
              CTLSEQ=PP,                                   -
              BEXIT=BEGNEXIT,                              -
              MAXTG=50,                                    -
              MAXRP=15,                                    -
              MAXSCI=256K
*
*             SSWORK=2040,    UNCOMMENT AND INCLUDE IN THE
*                             CPCSOPTN LIST IF YOU WISH TO
*                             RUN THE SAMPLE OLRR EDIT
*                             ROUTINES.
         END
```

Figure 2-5 (Part 3 of 3). Example of Macros for DKNMICR Generation

CPCSRDR Macro Parameters

Figure 2-6 lists the options available for the document processors. An NA for any option means that the option is *not applicable* for the sorter. Bold print and an underline indicate that the option is a default.

Figure 2-6. CPCSRDR Parameter Values

Option	3890	3890/XP	3891/XP	3892/XP
TYPE	3890-1–6	3890-1–6	3891-1–6	3892-1–6
DDRDRIN	ddname	ddname[1]	NA	NA
MFILM	YES | **NO**	YES | **NO**	YES | **NO**	YES | **NO**
INF	**YES** | NO	NA	NA	NA
ENDORSE	**YES** | NO	NA	NA	NA
MODEL	**A** | B | X |	XP	XP	XP
PEND	NA	**YES** | NO	**YES** | NO	**YES** | NO
POWER	NA	NA	NA	YES | **NO**
IMAGE	NA	YES | **NO**	YES | **NO**	YES | **NO**
ATTACH	**CHANNEL**|SIM|EMICR	**CHANNEL**|LU62|SIM|EMICR	**LU62**|SIM|EMICR	**LU62**|SIM|EMICR
LUNAME	NA	luname	luname	luname
EXT	NA	YES | **NO**	YES | **NO**	YES | **NO**
FLD1–7	(page 2-41)	(page 2-41)	(page 2-41)	(page 2-41)
FLD9–15	NA	FLDn=,	FLDn=,	FLDn=,
LDPN	(page 2-42)	(page 2-42)	(page 2-42)	(page 2-42)

[1] For LU 6.2-attached document processors, the ddname is NA.

The following is a description of all the possible parameters for the document processors.

Name	Operation	Parameters
[symbol]	CPCSRDR	TYPE={3890-*n* | 3891-*n* | 3892-*n*}
		[,ATTACH={**CHANNEL** | LU62 | SIM | EMICR}]
		[,DDRDRIN=*ddname*]
		[,LUNAME=*nnnnnnnn*]
		[,MFILM={**YES** | **NO**}]
		[,INF={**YES** | **NO**}]
		[,ENDORSE={**YES** | **NO**}]
		[,MODEL={**A** | **B** | **X** | **XP**}]
		[,PEND={**YES** | **NO**}]
		[,POWER={**YES** | **NO**}]
		[,IMAGE={**YES** | **NO**}]
		[,FLD*n*=(*d*,*b*,*o*,*i*)]
		[,EXT={YES | **NO**}]
		[,LDPN={*nn*}]

TYPE={3890-*n* | 3891-*n* | 3892-*n*}
Specifies the type of IBM document processor for this macro. You must select one of the model numbers and include the number of stacker modules. The number of stacker modules is *n*. Valid values are 1 to 6.

ATTACH={CHANNEL | LU62 | SIM | EMICR}
This optional operand defines how the document processor attaches to the host.

CHANNEL Specifies channel-attached document processors.

LU62 Specifies LU 6.2-attached document processors. This is the default for the 3891/XP and 3892/XP Document Processors.

SIM Specifies host-simulated document processors. The default value is CHANNEL, except for 3891/XP and 3892/XP processors, for which the default is LU62. If you specify either CHANNEL or SIM, the DDRDRIN= parameter is required and the LUNAME parameter is ignored. If you specify LU 6.2, the MODEL parameter defaults to XP, and you should not specify DDRDRIN.

ATTACH=SIM defines a host-simulated document processor, which lets you run MICR without a physical document processor. ATTACH=SIM requires the 3890/XP MVS Support licensed program. You can create a set of simulated items, including control documents, in a data set and have these captured into the MDS. The simulator feature supports codeline data matching in subsequent passes. The codeline data match record, which CPCS supplies to the simulated document processor, returns to CPCS as document read records. For more information on the simulator, see the *3890/XP MVS Support and 3890/XP VSE Support Program Reference*.

EMICR Specifies a host-simulated EMICR document processor for electronic string processing. As with SIM, this option allows you to run MICR without a physical document processor.

DDRDRIN=*ddname*
Specifies the name of the DD statement that identifies the document processor unit address. For a simulated document processor, the DD statement must reference the sequential data set containing codeline data. For more information about the unit address DD statement, see the *CPCS Programming and Diagnostic Guide*. For information on dynamically allocating document processors, see "Dynamically Allocating Data Sets" on page 2-23. You must add entries to the DKNDSAT table and then assemble and link DKNALLO. This parameter is not valid for ATTACH=LU62.

LUNAME=*nnnnnnnn*
Specifies the LU name for an LU 6.2 connection. This parameter is not valid for ATTACH=CHANNEL or ATTACH=SIM.

MFILM={YES | NO}
Specifies whether the microfilm feature is present on the document processor.

INF={YES | NO}
Specifies whether the item-numbering feature is present on the 3890 Document Processor. You must activate this feature on the prime pass. However, this option permits the use of a document processor without this feature on a subsequent pass. The 3890/XP Series document processors do not require this parameter. You must supply information for item-numbering in the sort pattern definition record for these document processors.

The item sequence number consists of three parts:

- Location of item on microfilm. This is true whether or not the document processor includes a microfilm unit.
- The document processor that you used to enter the work.
- The physical sequence of the items in the entry.

The physical sequence number is 6 digits for the 3890 and 8 digits for the 3890/XP Series document processors. The 3890 Document Processors print the last part of the item sequence number on each item.

CPCS stores the item sequence number in the old format (DISEQ) and the new format (DISEQ12). This provides compatibility with your own application tasks and with third-party vendor software. DISEQ contains the same digits as DISEQ12, except for the leftmost 4 digits of the sequential number. You should not consider DISEQ as the unique sequence number for CPCS. If you do not use DISEQ12 for your references, we cannot guarantee unique sequence numbers.

ENDORSE={YES | NO}
Specifies whether the endorsement feature is present on the 3890 Document Processor. The sort-pattern definition activates this feature for prime pass during the initialization record SETDEV process.

Note: This parameter applies only to non-XP document processors. Application of this parameter to a 3890/XP Series document processor results in an MNOTE 4 and in ENDORSE=NO being set automatically. To specify

whether the endorse feature is present on a 3890/XP Series document processor, set the PEND parameter.

MODEL={A | B | X | XP}
This optional parameter specifies the model of the document processor. The model code identifies the amount of storage available for your stacker control instruction (SCI) programs.

Model	Available Storage
A	10K
B	26K
X	42K
XP	Between 64K and 6M

Note: In all models, you have 3K additional storage available for prime-pass and high-speed entries. If you do not code this parameter for the 3890, the CPCSRDR macro uses **A** as the default. For the 3890/XP Series document processors, the macro uses **XP** as the default. If you use ATTACH=LU62, the default value is XP; using other MODEL options causes an error.

PEND={YES | NO}
Specifies whether the programmable endorse feature is present.

The sort-pattern definition controls the various endorse options (positioning, front, back, stamp) for the different sorts.

Note: This parameter applies only to the 3890/XP Series document processors.

POWER={YES | NO}
Specifies whether the power encode feature is present.

The sort pattern definition controls for the power encode options (path and font) for the different sorts.

Note: This parameter applies only to the 3892/XP Document Processors.

IMAGE={YES | NO}
Specifies whether the image feature is present.

Note: This parameter applies to the 3890/XP Series document processors.

FLD*n*=(*d*,*b*,*o*,*i*)
This optional parameter defines the settings for the 15 fields that can appear on a record (FLD1...FLD15). You can repeat this parameter for each field. The value of *n* can be 1 through 7 or 9 through 15. You cannot define field 8. The settings are positional so, if you omit one, you must insert a comma to represent it.

d Defines the total number of digits for this field. For variable-length fields, this is the maximum number of digits in the field. Do not use this value for fields 9 through 15. For extended fields, use the *b* parameter to specify the length.

b Defines the number of bytes necessary to store this field in the buffer record of the document processor. Each digit in the field requires a ***halfbyte*** (4 bits) of storage. This means that the value for this setting should be one-half the value for the first setting (*d*), rounded up if the first setting is an odd number.

o Defines the options associated with the field. If you use these options, they must be in the order shown below and you must not separate them by commas. The options are:

F Defines the field as a fixed-length field. If you omit this option, the field will be variable in length. If you define a field as fixed, the field must be on all documents.

Note: The shipped default for FLD1 and FLD5 is fixed.

D Specifies whether you want to transmit the dash from an encoded field to the record buffer of the document processor. If you omit this option, dash transmission does not occur and the dash does not occupy a halfbyte in the record.

Usage Notes:

1. You must specify this option for field 5. The position of the dash distinguishes Canadian checks from American checks.
2. You should be careful about using this option for other fields. For example, if you specify this option in field 3 (account number), and you inscribe a dash in field 3 of the tracer documents, unpredictable results occur when the tracers are read.

S Specifies whether you want symbol error correction for the field

I Specifies whether to use the field during codeline data matching on subsequent passes

i Specifies the digit error threshold of codeline data matching. See the *3890/XP Programming Guide* for a complete description of the codeline data matching function.

If you do not include a parameter for fields 1 through 3 or 5 through 7, the following default values are applied automatically:

[,FLD1=(**10**, | **5**, | **FSI**, | **10**)]

[,FLD2=(**6**, | **3**,| **SI**, | **6**)]

[,FLD3=(**14**, | **7**, | **SI**, | **14**)]

[,FLD5=(**9**, | **5** , | **FDSI**, | **9**)]

[,FLD6=(**2**, | **1** , | **SI**, | **2**)]

[,FLD7=(**10**, | **5**, | **SI**, | **10**)]

If you omit the parameter for field 4 and fields 9 through 15, the macro does not generate any data for the omitted parameter. There are no default values for these parameters.

Field 8 (not specified by the user) does have a default length of 6 bytes; this length should be used in process buffer-length calculations.

Important!

If you want to expand the MDS, you should use FLD*n*= parameters on the CPCSRDR macro. This enables your sorters to physically define the fields read by the sorters independently from the size of the MDS. Expansion of the MDS also automatically expands the size of the transmitted fields from the sorter if these FLDn= parameters are not specified.

A non-XP sorter is limited to 36 bytes of read data. A 3890/XP Series document processor is limited to 244 bytes of read data.

Violation of these limits generates a level-12 MNOTE (resulting in a bad assembly of DKNMICR) and the message:

```
EFA366 SUM OF BYTE LENGTH FOR FIELDS 1-15 IS nnn,
    CANNOT BE GREATER THAN xxx
```

where *nnn* is the sum of the read data lengths of all the defined fields, and *xxx* is either 36 for a 3890 Document Processor or 244 for a 3890/XP Series document processor.

EXT={YES | NO}

Specifies whether you are requesting the extended features of the 3890/XP Series document processors for this initialization record (IREC). This parameter allows for an expanded initialization record.

Note: You must specify EXT=YES if you defined fields 9 through 15.

LDPN={*nn*}

The logical document processor number is a unique 2-digit identifier for document processors. CPCS uses this number to find records in the item sequence-number data set and to determine whether the item sequence number is in use, needs to be updated, or is available.

- If you do not specify a value in the CPCSRDR macros for your document processors, the number defaults to 01 for the first document processor and increments by 1 for each subsequent macro without a value for LDPN.

  Note: Records 00 and 99 are reserved for CPCS use.

- If you specify a value, it must be greater than the LDPN value for the previous document processor.

The following table is an example of the effect of the LDPN parameter:

RDRn	Unit Number	LDPN Coded Value	LDPN Assigned Value	
RDR1	01	Omitted	01	Default
RDR2	02	Omitted	02	Default
RDR3	03	LDPN=11	11	Assigned
RDR4	04	Omitted	12	Default
RDR5	05	Omitted	13	Default

CPCSOPTN Macro Parameters

You must code one CPCSOPTN macro for the MICR task generation. The keyword parameters for the CPCSOPTN macro are as follows:

Name	Operation	Parameters
[symbol]	CPCSOPTN	TRACER=(*id*,*f*[,*f*])
		,DIVIDER=(*id*,*f*[,*f*])
		,BLOCK=(*id*,*f*[,*f*])
		,BATCH=(*id*,*f*[,*f*])
		,SBATCH=(*id*,*f* [,*f*])
		,ASTDOC=(*id*,*f* [,*f*])
		,CTLSEQ={**PP** | **PS** | **SP** | **SS**}
		,MAXSCI=(*nnnn*{K | M})
		[,MAXTG=**100** | *n*]
		[,MAXRP=**0** | *m*]
		[,TERMS=*x*]
		[,REXIT=*symbol*]
		[,BEXIT=*symbol*]
		[,MEXIT=*symbol*]
		[,MFOPT={**BOTH** | **FACE**}]
		[,SSWORK=*nnnn*]
		[,PCDASH={**YES** | **NO**}]
		[,ACTDASH={**YES** | **NO**}]
		[,SERDASH={**YES** | **NO**}]

The following operands describe the keyword parameters for the CPCSOPTN macro and also define the installation control documents for CPCS:

TRACER=(*id***,***f* **[,***f* **])**
DIVIDER=(*id***,***f* **[,***f* **])**
BLOCK=(*id***,***f* **[,***f* **])**
BATCH=(*id***,***f* **[,***f* **])**
SBATCH=(*id***,***f* **[,***f* **])**
ASTDOC=(*id***,***f* **[,***f* **])**

where:

id Specifies the control document identifier, which is a numeric character string up to 16 digits in length that identifies the control document. The fields in which you encode the identification determine the length of the control document identifier.

f Specifies which document fields contain the control document identifier.

For example, TRACER=(4777-7777,5,7) indicates that the CPCS PROLOG identifies a tracer document by the appearance of the character string 4777-7777 in document fields 5 or 7, or both.

Note: Field numbers used for this parameter coincide with MICR field numbers. See Figure 2-7 on page 2-44 for a description of the field numbers. Character strings cannot be larger than 16 digits or the maximum length of any field specified,

whichever is smaller. If you encode the field on the control document with a dash, you must include the dash in this parameter. You must encode the dash in **both** fields. While you can specify multiple control-document identifier fields, the recommended method is to use the routing number field solely for this purpose. See Figure 2-7 for field length limitations.

Figure 2-7. Control-Document Field Identifiers

Document Field	CPCS Identifier	Maximum Length
Amount	1	16 digits
Process control	2	16 digits
On-us (account number)	3	16 digits
Routing number	5	16 digits
Auxiliary on-us (serial number)	7	16 digits

The MICR task identifies a control document by comparing the contents of specified document fields with the character strings indicated. If you specify more than one field, then the task makes multiple comparisons to identify the document. A match on any field causes the MICR task to recognize the document as a control document. Specific fields of control documents are arbitrarily defined and cannot be used for control-document identifiers. See Figure 2-8 for control-document field definitions.

Figure 2-8. Control-Document Field Definitions

Document Type	Field ID				
	7	5	3	2	1
Tracer			TGID[1]		
Divider[2]	Kill bundle ID				
Block					Control amount
Batch					Control amount
Sub-batch					Control amount

Notes:

1. Tracer-group identification (TGID). You must pre-encode the TGID in field 3. It consists of a 7-digit number, XXXX-YYY, where XXXX is constant for all tracer slips within a group and YYY is greater than or equal to 1 and less than or equal to (MAXRP+9). YYY must be unique.

   If you code the parameter ACTDASH=YES, the TGID field on the tracer documents cannot contain a dash between the tracer group number and the slip number.

2. CPCS does not record data in the MDS record for fields 5, 2, and 1. CPCS control information occupies these fields. However, you can use these fields as document identifiers.

CTLSEQ={PP | PS | SP | SS}

Specifies the location of batch, sub-batch, and block control documents relative to the document stream, where:

P Specifies that the control document precedes the items it controls.
S Specifies that the control document follows the items it controls.

The first character refers to the position of block control documents and the second refers to the position of batch and sub-batch control documents. For example, CTLSEQ=PS indicates that block control documents precede the work and batch control documents follow the work.

MAXSCI=(*nnnn***[K | M])**
This operand limits the size of the temporary buffer used to load SCI programs. You can define space in 1024-byte (K) increments or in 1048576-byte (M) increments. If MAXSCI=0, no limit is set.

MAXTG=100 | *n*
Specifies the maximum number of tracer groups permitted in the system concurrently. When the number reaches this threshold, no more tracer groups can enter the system until there is an end-cycle operation. The default value for MAXTG is 100. Changing this parameter **requires a CPCS cold start** to initialize the pass-to-pass control file. For more information on CPCS cold start, see “STYPE Parameter” on page 1-7.

MAXRP=0 | *m*
Specifies the number of tracer slips required for the sort pattern with the most rehandle pockets. You can determine this number by totaling the number of slips specified in the R-records of each sort pattern and selecting the maximum. For example, if a specific sort pattern has five rehandle pockets, defined as two rehandle pockets on the prime pass and three rehandle pockets on pass 2, and each rehandle pocket receives two tracer slips, MAXRP is 10. If you specify the tracers in the Kill-Pockets Sort-Pattern Definition option, you should also include additional tracer slips to allow one tracer slip for each prime-pass kill pocket. The default value for MAXRP is zero.

For more information on sort patterns and sort pattern R records, see “Sort-Pattern-Definition Record Formats” on page 4-17.

Usage Notes:

1. For any given sort pattern, MAXRP plus 9 is the minimum number of tracer slips required in each tracer group for CPCS to effectively maintain pass-to-pass controls.
2. Changing this parameter requires a CPCS cold start to initialize the pass-to-pass control file. You should make this number larger than your current requirements to avoid cold starts.
3. MAXRP serves as a factor for calculating the physical record size for the tracer pass-to-pass data set. Records are established in the tracer data set to store the RSCB for each sorter. There is a direct size relationship between the tracer record calculated using the MAXRP and the tracer record required for MAXRP+9 sorters. The CPCSOPTN macro ensures that the number of sorters is no more than MAXRP+9 by issuing MNOTE 12 for violations of this rule.

TERMS=*x*
Specifies the number of CPCS terminals that can communicate concurrently with the MICR task. If you code this parameter, it must be large enough to include at least one terminal for every CPCSRDR macro specified. If you omit this parameter, it defaults to the number of CPCSRDR macros used.

REXIT=*symbol*

Specifies a user-written DKNMREAD exit routine that receives control each time a codeline data record is retrieved from the document processor. The exit routine permits user processing of items beyond those that the user stacker-selection routine has already processed. If specified, the routine loads during MICR task initialization and remains resident throughout CPCS runs. For correct operation, you must ensure that this routine is reusable or reentrant. The same copy of the program is used for all active document processors. For more information about the user exit for DKNMREAD, see "DKNMREAD Document-Processing User Exit" on page 4-113.

BEXIT=*symbol*

Specifies a user-written DKNMBEGN exit routine that receives control after the final verification by the operator of the DKNMBEGN display and before starting the entry. The exit routine lets the user check all data that the operator entered and decide whether to permit the entry to start or to return a message to the operator to correct the data. If you specify this routine, it loads during MICR task initialization and remains resident throughout CPCS runs. The user must ensure the re-usability or reentrance of this routine for correct operation. All active document processors use the same copy of the program. For more information about the user exit for DKNMBEGN, see "DKNMBEGN User Processing Exit" on page 4-107.

MEXIT=*symbol*

Specifies a user-written DKNMIPI exit routine that receives control each time DKNMIPI has prepared another record to be sent to the sorter for codeline data matching. The exit routine permits the user to suppress, edit, or insert records into the data-matching process. This DKNMIPI exit is a default only; other DKNMIPI exits, specified in bank control or sort pattern definition entries, can override it. Regardless of its source, you must ensure that the DKNMIPI routine is re-usable and reentrant. For more information about the DKNMIPI user exit, see "DKNMIPI User Processing Exit" on page 4-111.

MFOPT={BOTH | FACE}

Specifies how document processors with the microfilm feature initialize, where

BOTH Specifies duplex mode, which records both sides of the document

FACE Specifies duo mode operation, which records only the front of the document.

SSWORK=*nnnn*

Specifies the amount of additional 24-bit storage (in bytes) that you want for use as a work area during the running of:

- OLRR edit routine
- MICR user document processing exit (DKNMREAD).

The maximum size for SSWORK is 2040. For a description of the extended storage, see "MICR User-Parameter Area (MUPA)" on page 4-52.

PCDASH={YES | NO}
ACTDASH={YES | NO}
SERDASH={YES | NO}

Controls dash transmission from the document processor for the process control, account number, and serial number fields, respectively. (The default values shown above correspond to the original release of 3890 support in which these parameters were not available.)

If you use ACTDASH=YES, the TGID field on the tracer documents cannot contain a dash between the tracer group number and the slip number.

Expanding the Mass Data Set

The purpose of the MDS expansion is to enable you to define, within specified limits, all field sizes for records stored in the MDS. This lets you customize the MDS records according to your own needs. You can vary the size of fields 1 through 7, but not the sequence number field (field 8). Field 8 has internal requirements and dependencies.

MDX Macro Parameters

The MDX macro controls the MDS record expansion. IBM distributes CPCS with the MDX macro defined for the standard 50-byte record. You can customize the MDX parameter list to define a record according to your needs.

For example, if you want to include a field in your MDS record that contains remittance data, you can define one of the optional fields (fields 9 through 15) for this purpose. You might also want to define the amount field (field 1) on the MICR document with a length of 12 to take advantage of the larger field processing capabilities of the 3890/XP Series document processors.

Important!

1. If yours is a multi-bank installation, all of your financial institutions **must** use the same MDS record, even though they might not all need the expanded data. Each field must be as large as the largest required field size.
2. If you decide to expand the MDS, use FLDn= parameters on the CPCSRDR macro. This enables your sorters to physically define the fields read by the sorters independently from the size of the MDS. Expansion of the MDS automatically expands the size of the transmitted fields from the sorter if these FLDn= parameters are not specified.
3. A non-XP sorter is limited to 36 bytes of read data. A 3890/XP Series document processor is limited to 244 bytes of read data.

   Violation of these limits causes a level 12 MNOTE (resulting in a bad assembly of DKNMICR) and the message:

   ```
   EFA366 SUM OF BYTE LENGTH FOR FIELDS 1-15 IS nnn,
        CANNOT BE GREATER THAN xxx
   ```

   where *nnn* is the sum of the read data lengths of all the defined fields, and *xxx* is either 36 for a 3890 Document Processor or 244 for a 3890/XP Series document processor.

For more information about the CPCSRDR macro and the FLDn= parameter, see “CPCSRDR Macro Parameters” on page 2-37.

The following shows the syntax for the MDX macro:

Name	Operation	Parameters
[symbol]	MDX	**MDXF***yy* ,[*m*] ,['*descriptor*'] ,[*displen*] ,[*justification*] ,[*truncation*] ,[*fill character*] ,[*extrlen*]

MDXF*yy*

Specifies the field that you are defining. *yy* can be field numbers 01 through 15, except 08. (You must include the 0.)

An additional field may be defined to reserve space in the mass data set by coding **RP** in the *yy* location. If RP is used, only the internal field length is specified. Any other parameters are ignored. The reserved space is placed following field flag 3.

m Specifies the internal field length that you need. The field size must be less than or equal to the maximum size permitted. Field length refers to the number of bytes required to store the data in the MDS record. Data for fields 1 through 7 is stored at the rate of 2 digits per byte. Data for fields 9 through 15 is stored on a one-for-one ratio. Do not change field 8.

A size of zero indicates that the field is to be skipped and it will be commented out in the DSECT or copybooks. Zero is valid only for field 4 and fields 9 through 15. No other field can be set to zero.

'*descriptor***'**

Specifies the unique field descriptor that you want printed on reports. It can be up to 12 characters, including spaces and special characters. The descriptor must be in single quotation marks.

For example, you can define field 9 to be used for remittances. If you specify this parameter as `'REMITTANCE'`, your reports will have a REMITTANCE label on any information retrieved from field 9. For examples, see “MDX Macro Examples (Input Parameters)” on page 2-50.

displen

Specifies the number of displayable digits that are stored in the field. This must be more than zero, if you want the field to print on a report, and less than or equal to twice the defined field size (*m*, as described above). If you set this value for a field to zero, the field will not print on a report. If you do not code this parameter (in other words, if you leave it as a default by placing a comma in the corresponding position), the value defaults to twice the defined field size (*m*).

Note: This parameter is only valid for fields 1 through 7. For all other fields, omit the parameter but include the comma to delimit the positional parameter. The MDX macro sets this parameter to the defined size of the field.

justification
Specifies the justification of the field. The value is set to R or L.

truncation
Specifies the truncation of the field. The value is set to R or L.

fill character
The hexadecimal value for the fill character. The valid entries are 00, AA, or 40.

extrlen
Specifies the field size to be used for DKNICRE to create an extract file. If you set the BLDL parameter USRPARM, ICRE creates the alternate non-compressed file, using the field sizes specified in the *extrlen* parameter. This must be greater than 0 and less than or equal to twice the defined field size (m). The default value is twice the size of the defined length for fields 1 through 7, and the size of the defined length for fields 9 through 15.

Usage Notes:

1. The macro parameters are positional. If you do not use the optional parameters, you must mark the position with the comma. You can place several macros in succession. If you follow the parameter *displen* with a second MDXF*yy*, you must place a comma after *displen*. If *displen* is the final operand, a comma is not required.
2. For default (standard) and maximum field sizes, both internal and display, see the figure “MDS Record Field Lengths” in the *CPCS Programming and Diagnostic Guide*.
3. The MDX macro must follow macro coding rules for content, punctuation, and continuation. If you require more than one line on the statement, you must continue the statement with a comma (,) and a continuation character in column 72. The next line of continued parameters must start in position 16.

MDX Macro Examples (Input Parameters)

Some examples of the MDX macro are:

```
MDX MDXF01,6,,11,,,,,MDXF02,,'PROC. CTRL.',,,,
```

This example shows field 1, the amount field, defined with an internal field length of 6 bytes. The second comma after the 6 indicates that the default descriptor is being used. The display length is 11 characters.

A second set of parameters defines field 2. Only the field descriptor is changed. The other fields all retain the default settings.

```
MDX MDXF09,75,'REMITTANCE',,,,,
```

This example shows the addition of field 9. It is set at 75 bytes and has a unique descriptor. Because no other fields are changed with this macro, fields 1 through 3, 5 through 7, use the default settings.

```
MDXFRP,8
```

This example shows the addition of 8 bytes of reserved space.

Important!

You can assemble the MDX macro at any time; however, if you change the size of the MDS record, you must recompile or reassemble the CPCS modules and reinstall CPCS. This includes a cold start of CPCS after the install. For more information about a CPCS cold start, see “STYPE Parameter” on page 1-7.

“Expanded MDS Installation Procedure” describes the procedure for generating a new CPCS system with an expanded mass data set (MDS). You cannot recover data captured before the install.

Expanded MDS Installation Procedure

CPCS is delivered with a 50-byte MDS record length; however, you can modify the record length to meet the requirements of your installation. The following steps let you configure CPCS with an MDS record length that is larger than the default size.

Note:

These instructions assume that you have already installed a CPCS with a standard MDS record layout by following the instructions in *CPCS Installation Guide without* modification. If you have already expanded that Mass Data Set as part of your installation, you do not need to perform the following steps.

Modify the MDX macro in GENLIB by adding the input parameters shown in “MDX Macro Parameters” on page 2-48.

Step 1: Expand the Copybooks

If you change the MDS record in any way from the standard size, change the IBM-supplied member GENCLIB so that it points to the library that contains the CPCS source code. This member assembles the MDX macro to create copybooks reflecting the expanded fields. Change the MDX macro statements in GENCLIB to reflect your changes to the MDS.

Run the GENCLIB job stream to assemble the MDX macro and create copybooks that contain the expanded fields.

Step 2: Assemble and Compile the Programs Affected by Changes

This step assembles, or compiles, and link-edits Assembler and COBOL source modules that are affected by changes to the MDS. The JCL is in member DKNJBLD3 of CPCS.V1R11.CTRL.

If you have High Performance Transaction System Application Library Services installed, you must reassemble the DSS user exit. Refer to the *ALS Operations Guide* for information on how to do this.

Run job GENCLIB before you run this job. If necessary, change CPCS.V1R11 to a qualifier you have selected.

Step 3: Allocate Expanded CPCS Data Sets

This step allocates the expanded data sets. Make the following changes to the allocation job stream.

- Determine the MDS block-size parameter. The MDX macro reports the MDS record length; you need to multiply this by a blocking factor, taking into account the rules and limitations described where the BLKSIZE parameter is shown. Use the computed blocksize as-is on the BLKSIZE parameter of the CPCS system profile in "DKNPCPCS Profile Member" on page 3-4. Add 12 to the blocksize when allocating the Mass Data Set.
- Set the ICRE record length to the value reported to you by the MDX macro. If you've chosen to extract an uncompressed file by specifying USRPARM=X on DKNICRE's DKNBLDL entry, then allocate using the ICRA record length reported to you by the MDX macro. Adjust the block size accordingly.
- Set the following record lengths to the value reported to you by the MDX macro, and adjust the block size accordingly:

  MCRE record length
  MRGEIN record length
  MRGEOUT record length
  RSTGFIL record length
  HSRRFIL record length
  SCATIN record length

- Change the corresponding entries in DKNDSAT.

Important!

Increase the PRINT parameter value in the DKNBLDL macro for DKNSBAL from 2 to 4 and assemble a new version of DKNBLDL. If this value is not increased, DKNSBAL reports are not produced and the following message is issued to the supervisor terminal:

```
SBAL  MDXR RETURN CODE: 2088   FROM FUNCTION: 0000
```

- Reassemble DKNDSAT and DKNBLDL now that they have been modified.

Step 4: Generate the DKNMICR Module

Changing the MDS record length requires that you generate a new version of the DKNMICR module. The JCL is in member DKNGMICR of CPCS.V1R11.CTRL.

For information about generating DKNMICR, see "MICR Task Generation" on page 2-33.

Step 5: Generate the DKNMTASK Module

Changing the MDS record length requires a COLD start of CPCS. You can run the CPCS system parameter edit report with job DKNJEOTP found in CPCS.V1R11.CTRL. If you have LOG=YES in the CPCS system profile, you should take special note of the new DKNLD block size calculated. Set the MDEF BLKSIZE parameter to the new block size.

For information about the CPCS system profile parameters, see "DKNPCPCS Profile Member" on page 3-4.

VTAM Installation Requirements for CPCS

To run CPCS, you must perform the following Virtual Telecommunications Access Method (VTAM) tasks:

1. Assemble and link an installation node-name table (even if you do not specify any entries) into the CPCS load library and name it DKNVNODE.
2. Define the CPCS application and all CPCS terminals in the installation's VTAM configuration tables for System Network Architecture (SNA) local and non-SNA local terminals. You must define all SNA remote terminals within your network control program (NCP) generation.

To define CPCS as a VTAM application program, use the APPL statement in SYS1.VTAMLST. For user flexibility, the distributed DKNVTASK program takes the CPCS application program name from the job-step name of the EXEC statement in the CPCS job stream. Therefore, the CPCS application name on the APPL statement in SYS1.VTAMLST must correspond to the name you specify as the job-step name in the EXEC statement.

If you do not want to use this method to name the CPCS application, you can change two SETC variables within the DKNVTASK program (immediately following the DKNVTASK statement) to cause the EXEC statement to be ignored. You can change the &ACBLAB variable from `&ACBLAB SETC 0` to `&ACBLAB SETC VCVTASK`, which is the name field within the DKNVTASK module for the CPCS application name. This change causes the VTAM products to use the name specified in the DKNVTASK module as the application name.

The distributed name within the DKNVTASK module is CPCS (label VCVTASK). You can change this name by changing the 1- to 8-character alphanumeric value in the &ACBNAME variable. This variable specifies the CPCS application name and it must be the same as the name specified in the APPL statement within SYS1.VTAMLST.

DKNVTASK Installation

The DKNVTASK program includes a number of global variables that let you:

- Omit the abend processor (CSECT VPSTA000)
- Omit the issuing of some or all WTOs
- Omit the expansion of the CPCS, VTAM, or DKNVTASK control block DSECTs
- Omit the issuing of snap dumps for various error conditions
- Set the number of various threshold counters
- Assign an application name (APPLID) to the CPCS task or use the job-step name from the CPCS job stream
- Set session protocol definitions for the session BIND
- Assign program function keys to either CPCS commands or tasks or both.

Figure 2-9 on page 2-54 lists the DKNVTASK variables, their initial settings, and the functions they perform.

Figure 2-9 (Page 1 of 3). DKNVTASK Variables

Variable	Setting	Function
Control block expansions (0=do not expand, 1=expand):		
&PVTAMB	SETB 0	Expand VTAM control blocks.
&PVTASKB	SETB 0	Expand DKNVTASK control blocks (VDSECT).
&PVMSGS	SETB 0	Expand DKNVTASK message table.
&PCPCSB	SETB 0	Expand CPCS control blocks.
Include or omit processing routines (0=include, 1=omit):		
&SNAPS	SETB 0	**0=** Issue snap dumps.
		1= Do not issue a snap dump for control blocks when session error conditions occur (for example, I/O errors or data-stream errors).
&STOP	SETB 1	**0=** Stop DKNVTASK during initialization. Requires a reply on the console to continue (any reply OK).
		1= Do not stop DKNVTASK.
&STA	SETB 1	**0=** Include the DKNVTASK STAE routine.
		1= Do not include the DKNVTASK STAE routine. The DKNVTASK STAE routine intercepts DKNVTASK abends, takes a snap dump, issues abend messages, and attempts to recover the abending session.
&WTOALL	SETB 0	**0=** Include all WTOs.
		1= Omit all WTOs (routine VIWTO000 returns to the caller).
&WTOSES	SETB 0	**0=** Include logon/logoff WTOs.
		1= Omit logon/logoff WTOs.
&WTOIOR	SETB 0	**0=** Include I/O error WTOs.
		1= Omit I/O error WTOs.
&WTOABD	SETB 0	**0=** Include abend processor WTOs.
		1= Omit abend processor WTOs (except for the STA abend WTO).
Note: The DKNVTASK program is distributed with the above SETB settings.		
Threshold counters:		
&STACRSH	SETA 25	STAE routine-number of session stops that can occur before CPCS stops with an abend 119 (see &STAABRT).
&STAABRT	SETA 4	STAE routine-number of times that an abended session can recover before it stops. Each time a session stops, the crash counter increments by 1 (see &STACRSH).

Figure 2-9 (Page 2 of 3). DKNVTASK Variables

Variable	Setting	Function
Hardcopy scroll node name:		
&HCPYNDE	SETC 'HARDCOPY'	Node name identifies the VTAM-controlled printer for LU.T0 printers, such as the IBM 3284 printer. This value is ignored for SNA printers LU.T3. If any other name is used for an LU.T0 printer, use the DEVICE=PRT keyword to identify the node as a printer.
The following variables cause the CPCS APPLID to be set:		
&ACBNAME	SETC 'CPCS'	APPLID name if &ACBLAB is not set to 0.
&ACBLAB	SETC '0'	If this variable is set to 0, the CPCS APPLID is the job-step name from the CPCS EXEC statement. If this variable is **not** set to a 0, the CPCS APPLID comes from the variable &ACBNAME.
The following variables cause DKNVTASK to set protocol values before a VTAM session is activated:		
&SETPSNA	SETB 0	DKNVTASK sets the protocol definitions for all SNA sessions to: PRIPROT = X'81' SECPROT = X'90' COMPROT = X'3080'.
&SETPBSC	SETB 0	DKNVTASK sets the protocol definitions for all non-SNA sessions to: PRIPROT = X'71' SECPROT = X'40' COMPROT = X'2000'.
The following variables assign CPCS commands to program function keys. You can change the program function key assignments.		
&PF01	SETC	'SUPVON,SYST'
&PF02	SETC	'SUPVOFF,SYST'
&PF03	SETC	'SUPVON,INSC'
&PF04	SETC	'SUPVOFF,INSC'
&PF05	SETC	'SUPVON,MICR'
&PF06	SETC	'SUPVOFF,MICR'
&PF07	SETC	'STAT'
&PF08	SETC	'SDIR'
&PF09	SETC	'FORM'
&PF10	SETC	'SCRL'
&PF11	SETC	'MICR'
&PF12	SETC	'SGOF'
&PF13	SETC	'ANODE'
&PF14	SETC	'FNODE'
&PF15	SETC	'AHCPY'
&PF16	SETC	'DHCPY'

Figure 2-9 (Page 3 of 3). DKNVTASK Variables

Variable	Setting	Function
&PF17	SETC	'PDUMP'
&PF18	SETC	'PBUFF'
&PF19	SETC	'VWHO'
&PF20	SETC	Not defined
&PF21	SETC	Not defined
&PF22	SETC	'CANCEL'
&PF23	SETC	'STOP'
&PF24	SETC	'ALLO'

DKNVTASK Node-Name Table Description

The node-name table (DKNVNODE) is a non-executable CSECT that the DKNVTASK program loads at run time. It contains node names and passwords for the VTAM terminal devices. This table has two sections that are used to specify those terminals in the VTAM network that are *primary* and those terminals that log on CPCS as *auxiliary* terminals. If you specify a terminal as a primary CPCS terminal, it automatically logs on CPCS during initialization, and CPCS writes its logo to the screen. If you specify a terminal as an auxiliary CPCS terminal, it requires the standard VTAM command LOGON to log on CPCS. The LOGON command causes CPCS to write its logo to the screen and to identify the terminal as a standard CPCS terminal.

If you do not specify any primary CPCS terminals in the node-name table, all terminals that CPCS uses must log on the VTAM products as auxiliary terminals. In this case, terminals are not automatically assigned to CPCS during CPCS initialization. CPCS does not write logos to any terminal, and any terminal in the VTAM network can log on CPCS. If you do not specify any terminals as auxiliary CPCS terminals in the node-name table, any node in the VTAM network can log on CPCS and function as a CPCS terminal.

Along with node names, the node-name table can also contain logon passwords for each primary and auxiliary node specified. If the password is present, you must specify the password as data in the logon message. The password entered in the logon message is checked against the password in the node-name table for the terminal that is logged on. For auxiliary node names, the password is checked each time that the terminal is logged on. For primary node names, the password is only checked if the LOGOFF command logs the terminal off CPCS and a later attempt is made to log it back on CPCS with a VTAM LOGON request.

You create DKNVNODE by coding, assembling, and link-editing VNODE macro statements into your load library (see "VNODE Macro Description" on page 2-57). For each VNODE statement that you code, an entry is created as either a primary node-name entry or an auxiliary node-name entry in the node-name table.

On each VNODE macro statement, you must code a NODE=*node-name* parameter, but the PASSWD=*password* parameter is optional. Coding the CPCS parameter is optional for primary node names because the default is CPCS=YES. However, it is required for auxiliary CPCS terminals (CPCS=NO). You can code the DEVICE parameter to indicate whether the device is a display (CRT) or a scroll

printer (SCR). DEVICE=CRT is the default. The NEND=YES parameter is required for the last VNODE macro statement to end the assembly process. If the last VNODE macro statement does not contain the NEND=YES parameter, the VNODE macro does not generate the node-name table.

You must link-edit the output generated by the assembly of the VNODE macros into your load library as DKNVNODE.

VNODE Macro Description

The VNODE macro generates the node-name table CSECT as previously described in “DKNVTASK Node-Name Table Description” on page 2-56. Each valid issuance of the macro results in a node-name entry being placed in either the primary node name section or the auxiliary node-name section of the node-name table (DKNVNODE). The table generation process continues until it reaches a macro with the NEND=YES parameter or until there are no more macro requests in the input stream. If the last processed macro does not include a NEND=YES parameter, it does not create a CSECT END statement and does not generate assembly errors.

Name	Operation	Parameters
[SYMBOL]	VNODE	NODE=*xxxxxxxx* [,CPCS={**YES** | **NO**}] [,NEND={**NO** | **YES**}] [,PASSWD=*pswd*] [,DEVICE={**CRT** | **SCR** | **PRT**}] [,OSITE={**CPCSR11** | *xxxxxxxx*}] [,PSITE={**OSITE-value** | *xxxxxxxx*}]

NODE=*xxxxxxxx*

An alphanumeric, 1- to 8-character node name of the node to be placed in either the primary or auxiliary node-name table.

CPCS={YES | NO}

Specifies whether the node-name entry is in the primary or auxiliary node-name table, where

YES Specifies that the node-name entry is in the primary node-name table. The terminal is considered a primary CPCS terminal and is put into session with CPCS when CPCS starts.

NO Specifies that the node-name entry is in the auxiliary node-name table and must be put into session with CPCS by a VTAM LOGON request.

NEND={NO | YES}

Specifies either that this is the last node name in the node-name table and should cause a CSECT END statement, or that more macros follow. NEND=YES causes the macro processing to end.

PASSWD=*pswd*

An alphanumeric, 1- to 8-character password that you must enter with a VTAM LOGON request for a terminal to activate. If you include this parameter for a primary CPCS terminal, it is effective only if you take the terminal out of session by a VTAM LOGOFF request and put it back into session by a VTAM LOGON request.

DEVICE={CRT | SCR | PRT}
Specifies the type of device in use, where:

CRT Specifies that the device is a display terminal
SCR Specifies that the device is a scroll printer
PRT Specifies that the device is a scroll printer.

OSITE={CPCSR11 | *xxxxxxxx***}**
The alphanumeric, 1- to 8-character, site identifier of this terminal's "owning" site; for example, this site owns this terminal or this terminal belongs to this site.

PSITE={OSITE-value | *xxxxxxxx***}**
The alphanumeric, 1- to 8-character, name for the current processing site for this terminal; for example, the site from where this terminal's activities are now being processed.

Terminal Interface

The node-name table determines which terminals are CPCS terminals and which are not. You generate the node-name table by coding VNODE macro statements as described in "VNODE Macro Description" on page 2-57. To associate terminals with CPCS, you must specify (in CPCS VNODE macro statements) whether the node names on the LU and LOCAL statements are primary or auxiliary CPCS terminals, as follows:

Primary Terminals Primary terminals log on CPCS automatically during CPCS initialization. If a primary terminal is powered off, initialization continues. However, if that terminal becomes available, then it automatically logs on CPCS. When a primary terminal logs on CPCS, the CPCS logo appears, which indicates that entering the SGON command will start CPCS processing. For information about specifying a primary terminal, see "DKNVTASK Node-Name Table Description" on page 2-56.

Auxiliary Terminals: Unlike CPCS primary terminals, CPCS auxiliary terminals do not automatically log on CPCS. An auxiliary CPCS terminal requires the VTAM LOGON request to log on CPCS.

You must specify auxiliary terminals only if you want to limit which terminals within the VTAM network can or cannot log on the CPCS application.

Note: If you do not specify an auxiliary terminal, any terminal within the VTAM network can use the CPCS application. If you specify at least one terminal as a CPCS auxiliary terminal, only those terminals included in the node-name table as primary or auxiliary terminals, or both, can use the CPCS application.

Once an auxiliary CPCS terminal or a primary CPCS terminal (one that has been logged off CPCS) logs on CPCS, the CPCS logo appears and you can use the CPCS SGON command to log on CPCS. For information on specifying an auxiliary terminal, see "DKNVTASK Node-Name Table Description" on page 2-56.

Remote Site Support: When assigning site information to a terminal, use the VNODE macro. You may assign sites, both "owning" and "processing" to a particular terminal.

This information is propagated throughout the CPCS control blocks (communication buffer, APTCB, UOW, and others) and therefore throughout the system as well.

Application Interface

Application tasks communicate with a terminal through DKNVTASK. For additional information about using DKNVTASK and for an explanation of the operation codes, see the *CPCS Programming and Diagnostic Guide*.

VNODE Macro Error Messages

See the *CPCS Messages and Codes* manual for error messages that can occur during processing of VNODE macros to build the DKNVNODE table.

Terminal Definition

DKNVTASK supports the following devices unless otherwise noted:

- All 327x series devices, except the 3277 Display Station Model 1
- Any 3270 Control Unit in binary synchronous communication (BSC) mode (3271 Control Unit Models 1 and 2 or the 3274-1C Control Unit in BSC mode)
- Configuration Support A devices and all devices connected to a 3274-1D Control Unit as non-SNA Local (LU.T0) devices
- Configuration Support B devices as either SNA Local or SNA Remote (LU.T2 or LU.T3) devices
- 3290 and 3278 Model 5s as 80-column devices that use from 24 to 43 rows in both SNA and non-SNA modes.

This section briefly describes and gives some examples of the NCP macro statements and the configuration statements necessary to define CPCS as a VTAM product and to define CPCS terminals to VTAM. VTAM configuration statements must define all local terminals (non-SNA and SNA) and the CPCS application; SYS1.VTAMLST retains these definitions. The user's NCP generation must include the definitions (using the correct NCP generation macro) for all SNA remote devices.

Local Non-SNA Devices

For local non-SNA environments (LU.T0 devices), you must define the terminals available to CPCS as part of a major node. You do this by using the LBUILD statement to define the major node and a LOCAL statement to define each non-SNA terminal. In the following example, FINS3270 is a member of SYS1.VTAMLST and defines the major node, FINS3270, with the two minor nodes, EIGHT01 and HARDCOPY.

```
FINS3270 LBUILD
EIGHT01  LOCAL  CUADDR=BE0,TERM=3277,          X
                FEATUR2=(MODEL2,ANKEY,PFK),    X
                DLOGMOD=D4B32782
HARDCOPY LOCAL  CUADDR=BE7,TERM=3284,          X
                FEATUR2=MODEL2,DLOGMOD=S3270
```

Local SNA

For local SNA terminals (LU.T2 devices), you must use a VBUILD statement to define a major node along with a physical unit (PU) statement to define each physical unit (such as an SNA controller) and LU statements to define each logical unit. In the following example, LOCALSNA is a member of SYS1.VTAMLST, which defines a major node with one local cluster, LOC3274, and two local terminal nodes, LOC3278 and LOC3279.

```
LOCALSNA VBUILD  TYPE=LOCAL
LOC3274  PU      CUADDR=BE0,ISTATUS=ACTIVE,MAXBFRU=4
LOC3278  LU      LOCADDR=2,DLOGMOD=D4A32781
LOC3279  LU      LOCADDR=3,DLOGMOD=D4A32782
```

Remote SNA

For SNA remote terminals, you must define each CPCS terminal and its associated control unit within the user's NCP generation. You must define each 3274 with a PU statement and each terminal with an LU statement. The following example shows that portion of an NCP generation that describes one cluster for a 3274-1C (CLUSTI0) and two LUs for two display terminals (F327802 and F327804).

```
CLUSTI0  PU      ADDR=C0,
                 MAXDATE=265,
                 ISTATUS=ACTIVE,
                 SPAN=(SPANTR03)
F327802  LU      LOCADDR=2,
                 DLOGMOD=D4C32782
F327804  LU      LOCADDR=3,
                 DLOGMOD=D4C32784
```

Assigning VTAM Terminals to CPCS

Terminal or printer device node names that you specify as primary CPCS terminals in the node-name table automatically log on CPCS when CPCS is started. During initialization, there are no password checks and all terminals that you specified as primary CPCS terminals are reserved for the CPCS application.

If a terminal is powered off at the time of initialization, it is assigned to CPCS and becomes active when the device is powered on. If a primary CPCS terminal is not online to the VTAM products when CPCS is started, it requires a VTAM LOGON after the device has been activated to the VTAM products by a VARY ACTIVE command.

If a VTAM LOGOFF command releases a primary CPCS terminal, the terminal requires a VTAM LOGON command to log on CPCS again. If a CPCS DHCPY command releases a printer device to the VTAM products, the printer requires a CPCS AHCPY command to log on CPCS again.

Terminal or printer devices that you specify as auxiliary CPCS terminals in the node-name table require a manual operation to log on CPCS. For terminal displays, you can log on CPCS with a VTAM LOGON request from the display device. For terminal printers, you can log on CPCS with a CPCS AHCPY command.

If you specify any devices in the node-name table as CPCS auxiliary terminals, only those devices whose node names you specify in the node-name table as either primary or auxiliary CPCS terminals can log on CPCS. If you do not specify any devices as auxiliary CPCS terminals, any terminal in the complete VTAM network can log on CPCS.

The optional user-specified password parameter in the node-name table is a requirement only for a VTAM LOGON request. For primary CPCS terminals, it is a requirement only if the primary terminal logs off and then logs back on CPCS with a VTAM LOGON request. When a printer logs on CPCS with a CPCS AHCPY command, no password is necessary. Also, the optional password, if in the

node-name table, is a required entry *each* time an auxiliary CPCS terminal logs on CPCS.

Note: If a major node or a terminal node is activated with the logon option, the following conditions are enforced:

- If you specify any auxiliary node names, only those nodes in either the primary or auxiliary table, or both, can log on.
- CPCS rejects any node containing the optional password.

Releasing a CPCS Terminal

Terminals that CPCS uses are available to other VTAM products at any time. You must first use the CPCS SGOF command to log off CPCS. When the CPCS logo appears, use the LOGOFF command to release the terminal. After you enter the LOGOFF command, the logo disappears and the display terminal is available to other VTAM products.

You can enter the DHCPY command, from the supervisor terminal, to release a printer to the VTAM products. You can also use the FNODE command to release any terminal to the other VTAM products. For a detailed description of the CPCS commands, see the *CPCS Terminal Operations Guide*. Also, terminals can be released to a VTAM product if external error conditions exist. For example, the abend processor or an unrecoverable I/O error can release the terminal.

Terminal Screens and Key Usage

The CPCS system (non-application mode) uses a full-screen format for commands and messages. A terminal uses the number of rows defined in its logon mode table (DLOGMOD). CPCS writes commands and messages on a screen from the top to the bottom. The screen wraps to the first line and CPCS starts to write over the existing lines.

When in application mode, an application can also use the full screen. Again, the dimensions specified in the logon mode table (DLOGMOD) parameter limit the size of the screen.

DKNVTASK controls the screen format when it is not in application mode. The READY message determines where you can enter commands. All lines above the READY message are protected (you cannot use them for entering commands); all lines below the READY message are unprotected (you can use them for entering commands).

On color terminals, the protected area is blue, the unprotected area is green, the READY message appears in white, and the entry area appears in red.

On non-color terminals, the protected and unprotected areas appear in normal intensity. The READY message and entry areas appear in high intensity.

As with a non-supervisor terminal, CPCS writes all supervisor terminal messages in white for color terminals and in high-intensity for non-color terminals.

Program Function (PF) Keys: You can optionally assign PF keys to retrieve CPCS commands and CPCS tasks for performance (see the DKNVTASK variables on page 2-55). If you select this option and press a PF key assigned to a CPCS command or task, the assigned command or task appears to the right of the

READY message, with the cursor located to the right of the retrieved command or task name. The READY message is down one line, and the retrieved command or task returns to the input buffer as if you had previously entered it. If you press a program function key that is **not** assigned to a CPCS command or task, the **last** command or task that you entered appears.

Program Attention Keys (PA1, PA2, PA3): If you press a program attention key, the last command entered appears to the right of the READY message. The cursor appears to the right of the retrieved command and the READY message is on the next line (it can only retrieve the command that you entered earlier).

CLEAR Key: If you press **CLEAR**, the terminal screen clears and the READY message displays on the next line.

Screen Sizes

Under VTAM control, CPCS supports screen sizes based on the logical unit type. The logon-mode table (DLOGMOD) entry establishes the base default screen size and the alternate (large) screen size. For detailed information on defining screen sizes, see the following publications:

- *3270 Information Display System Customizing Guide Supplement for 3274 Control Unit*
- *ACF/VTAM Version 3 Programming*
- *ACF/VTAM Planning and Installation Reference*.

In general, the definitions in the VTAM logon-mode tables establish screen sizes. When specified in the DLOGMOD parameter, those tables bind the terminal session. Therefore, they determine the dimensions of a terminal screen.

You must specify at least 80 columns for CPCS. Only the characteristics of the device restrict the number of rows that you can specify.

- If you specify fewer than 80 columns, DKNVTASK resets this value to 80.
- If you specify fewer than 24 rows, DKNVTASK resets this value to 24.
- If you specify more than 43 rows, DKNVTASK resets the number of rows to 43.

Although you can code logon-mode tables, we recommend that you use those logon-mode tables distributed for the specific device type.

Accessing the VSAM Table

The CPCS VSAM manager modules access the VSAM table (DKNVSTBL) to determine whether an application is requesting access to a file in the table. The table specifies information on how the file is to be processed. Listing a file in the table, which lets the file be treated as a CPCS resource, provides minimum system overhead in file processing.

Note: There is no requirement that files used during CPCS processing be listed in DKNVSTBL.

The CPCS VSAM table has the following characteristics:

- The table entry has an 8-character *file identification* that is cross-referenced to the DDNAME of the file.

  This method of logical reference permits multiple table entries for a single VSAM file. This provides flexibility because each entry can specify different attributes for opening and accessing the file.

- The table entry specifies the level of data integrity protection for the file.

  When the table entry is created, you can specify either VSAM subtask-sharing protection or VSAM manager cross-region sharing protection.

- You can define file access for read, write, and update; or you can limit file access to read-only, based on values used to create the table entry.

- The table entry can specify that the file is to be emptied when a CPCS cold start occurs.

- For each VSAM table-file identification, the table entry can specify exit routines that run during CPCS startup, CPCS shutdown, or both.

DKNVSTBL contains control information used by DKNVSMGR and a series of entries generated by the DKNVSENT macro. To add a new VSAM file definition to the table, add a DKNVSENT macro to the existing DKNVSTBL source code, compile, and link. The definition of the CPCS message file, DKNSMSGD, should remain the first table entry.

The DKNVSTAB macro describes the VSAM table layout. The following coding generates a DSECT:

```
(label)   DKNVSTAB               .Generate a DSECT of the CPCS VSAM
*                                . table, DKNVSTBL
```

The DKNVSENT macro describes each entry in the table. The following coding generates a DSECT:

```
(label)   DKNVSENT   DSECT=Y     .Generate a DSECT of the DKNVSTBL
*                                . entries
```

CPCS VSAM Table

The following is a sample of CPCS VSAM table entries:

```
DIV      DKNVSENT DDN=DKNDIV,      - CPCS DIVIDER DATA SET DDNAME
                  FILEID=DKNDIV,   - FILE ID OF DIVIDER DATA SET
                  READONL=N,       - FILE IS READ/ WRITE
                  CROSSRG=N,       - USES VSAM SUBTASK SHARING
                  COLDCLR=Y        - CLEAR FILE AT CPCS COLD STARTS
```

The following is a sample of DKNVSENT macro coding:

```
SAMPLE   DKNVSENT DDN=DKNSAMP,     - A SAMPLE DATA SET DDNAME
                  FILEID=SAMPLE,   - FILE ID DIFFERENT FROM DDNAME
                  READONL=Y,       - FILE IS READ ONLY
                  CROSSRG=Y,       - SERIALIZE FILE ACCESS
                  COLDCLR=N,       - DON'T EMPTY FILE AT COLD STARTS
                  BEGEXIT=SAMPINIT,- CALL SAMPINIT AT CPCS START-UP
                  ENDEXIT=SAMPTERM - CALL SAMPTERM AT CPCS SHUTDOWN
```

DKNVSENT Keywords and Values

The following is a description of the DKNVSENT keywords and their values:

DDN=*ddname*

The DDNAME given to the file in the JCL that starts CPCS.

FILEID=*filename*

The file identification is up to 8 characters long. More than one FILEID can reference a single VSAM file.

READONL={Y | N}

Y The ACB that accesses the VSAM file, which this table entry defines, can be used only for reads. This is the default value.

N The ACB that accesses the VSAM file can be used for reads, writes, and updates.

CROSSRG={Y | N}

Y Provide cross-region data integrity. The file use is serialized by a systems-wide enqueue of the 44-character data-set name of the VSAM file. This is the default value.

N Only programs in a single address space can access the file. VSAM subtask sharing ensures data integrity.

COLDCLR={Y | N}

Y Empty the file when CPCS is cold started.

N Never empty the file. This is the default value.

BEGEXIT=*exit name*

The 8-character name for an optional initialization-exit module. This module is called during CPCS startup.

ENDEXIT=*exit name*

This 8-character name for an optional termination-exit module. This module is called during CPCS termination.

Processing Description

This section describes the CPCS VSAM Manager DKNVSMGR operations.

During CPCS Initialization

DKNMTASK loads the VSAM table and attaches DKNVSMGR. During this start-up, DKNVSMGR processes each VSAM table entry to determine the following:

- Whether to empty the file because of a cold start of CPCS (a flag on the entry indicates *empty* on CPCS cold starts)
- Whether to call an exit module to perform initialization for the VSAM file described by the entry.

When processing is complete for all the table entries, DKNVSMGR waits for additional requests to open, close, or end CPCS VSAM operations. Event control blocks that are incorporated in the VSAM table make requests and signal the completion of the requests.

During CPCS Operations

The VSAM manager can be activated to allocate and open or close ACBs for VSAM files. If an application program caller specifies a file identification (VSMFID) in the VSMVOFL control block, DKNVSMIO scans the VSAM table for a matching identifier. If no match is found, an error code returns. If a corresponding identifier is found and the table entry contains a zero ACB address, DKNVSMIO activates the VSAM manager to open the file.

The manager verifies that no ACB was previously allocated for the FILEID and then uses DKNVSOPN, the open file routine, to allocate an ACB and open the VSAM file. The file can remain open after the application program ends because the VSAM manager, instead of the application task, owns the ACB that accesses the file.

After a successful open, the VSAM manager copies the ACB address to the table entry for use by other CPCS applications and sets a flag in the requestor's VOFL control block to indicate that the VSAM manager opened the file. When an application closes a VSAM file listed in the CPCS VSAM table, DKNVSMIO uses this flag to determine whether to activate the VSAM manager to free temporary buffers that were allocated during open processing. The application does not close VSAM files that are CPCS system resources. When an application caller requests a close operation, the close does, however, free the temporary buffers that were allocated during file open processing.

During CPCS End Processing

DKNVSMGR scans the VSAM table for files that were opened during CPCS processing. If an entry defines a file that is open, the exit module that is listed in the entry is called to perform end processing. If an exit module is not specified, the file is closed.

CPCS Third-Party Vendor Definition

Use the DKNVENDR macro to define a third-party program vendor to CPCS. The definitions of the DKNVENDR keywords (operands) follow:

FUNCT=*requested macro expansion*

Specifies the code to be expanded for this macro invocation. The possible values are:

BEGIN Marks the beginning of the CPCS Third-Party Vendor Table

ELEMENT Generates a table element that defines a CPCS third-party vendor

END Generates the label that marks the end of the CPCS Third-Party Vendor Table

TYPE=*requested macro expansion type*

Specifies the type of macro expansion. The possible values are:

DATA Generates a Data Section

DSECT Generates a Data Section Map

PF=*macro label prefix*

Specifies a 1- to 3-character prefix on all labels

NAME=*vendor company name*

Specifies a 1- to 32-character vendor company name

ABBR=*vendor company abbreviation*

Specifies a 1- to 8-character vendor company abbreviation

PREFIX=*vendor company prefix*

Specifies a 3-character vendor company prefix that is used in user exit name comparisons to associate user exits to vendors. This enables CPCS to pass each user exit, the specific user area that belongs to the vendor who owns the exit.

MDSL=*Mass Data Set Vendor Area Length*

Specifies the length of the vendor's section of the overall mass data set user area. The first four bytes of this area are used to store the prefix and length of the area, so those four bytes should be included in the value for this parameter. For example, if 10 bytes of data are needed, a value of 14 bytes should be specified.

CPCS Mass Data Set Exit Points

CPCS provides several exit points for mass data set-related processing. These exit points are detailed in this section, as well as the user exit information relative to the exit points.

MDS_FREE_EXIT

The Mass Data Set FREE Exit point allows multiple exits to be invoked during MDS string FREE processing. You may use these exits to deny a FREE SPACE request.

Environment

This exit is given control within mass data set services TCB and can be detrimental to system performance.

Point of Processing

These exits are called prior to deleting the requested MDS string.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The MDS_FREE_EXIT=*keyword* is used to specify an exit for this exit point. Up to ten (10) exits may be specified for this exit point. Each exit is called, in the order they were specified in the DKNPEXIT profile, until there are no more exits or until one of the exits denies the request with a return code.

Figure 2-10. MDS_FREE_EXIT Point Parameter List

Offset	Description
+00	Caller's APTCB address
+04	Vendor-specific user area address, may be null if no MDS user area was specified for a vendor with a prefix that matches the first three characters of the exit name.
+08	MDS User Exit Control Block mapped by DKNMDS11

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return codes as described below:

- **00** Continue normally
- **04** Deny the caller's request

Other Programming Considerations

Notes:

1. You may alter your MDS user area data, if desired; however, your changes are not saved.
2. This exit point is located in a performance-critical area and, as a result, inefficient code affects overall CPCS performance.
3. IBM does not require this exit be re-entrant and recommends that extra STORAGE, GETMAIN, or FREEMAIN invocations be avoided to improve performance.

Example
No examples have been provided.

MDS_OPEN_EXIT

The Mass Data Set OPEN Exit point allows multiple exits to be invoked during MDS string OPEN processing. You may use these exits to initialize or modify the MDS user area specific to your exit. You may also request that CPCS not allow the string open request.

Environment
This exit is given control within mass data set services TCB and can be detrimental to system performance.

Point of Processing
These exits are called prior to opening the requested MDS string.

Activation
All exits for this exit point are specified in the CPCS system profile (DKNPCPCS), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The MDS_OPEN_EXIT=*keyword* is used to specify an exit for this exit point. Up to ten (10) exits may be specified for this exit point. Each exit is called, in the order in which it was specified in the DKNPEXIT profile until no more exits exist or until one of the exits denies the request with a return code.

Figure 2-11. MDS_OPEN_EXIT Point Parameter List

Offset	Description
+00	Caller's APTCB address
+04	Vendor-specific user area address, may be null if no MDS user area was specified for a vendor with a prefix that matches the first three characters of the exit name.
+08	MDS user Exit Control Block mapped by DKNMDS11

Register Contents When Control Is Passed to the Exit Routine

R0	Not applicable
R1	Parameter list address
R2-R12	Not applicable
R13	Caller's save area address
R14	Return address
R15	User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return codes as described below:

00 Continue normally.

04 Deny the caller's request.

Other Programming Considerations

Notes:

1. You may alter your MDS user area data, if desired.
2. This exit point is located in a performance-critical area and, as a result, inefficient code affects overall CPCS performance.
3. IBM does not require this exit be re-entrant and recommends that extra STORAGE, GETMAIN, or FREEMAIN invocations be avoided to improve performance.

Example

Member DKNMDSX1, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

MDS_RDIR_EXIT

The Mass Data Set Read Directory (RDIR) exit point allows multiple exits to be invoked during MDS string read directory processing. You may use these exits to interrogate the MDS user area specific to your exit. You may also request that CPCS not allow the string read directory request.

Environment

This exit is given control within mass data set services TCB, and can be detrimental to system performance.

Point of Processing

These exit(s) are called prior to returning the requested MDS string directory information to the caller.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The MDS_RDIR_EXIT=*keyword* is used to specify an exit for this exit point. Up to ten (10) exits may be specified for this exit point. Each exit is called, in the order in which it was specified in the DKNPEXIT profile, until no more exits exist or until one of the exits denies the request with a return code:

Figure 2-12. MDS_RDIR_EXIT Point Parameter List

Offset	Description
+00	Caller's APTCB address
+04	Vendor-specific user area address, may be null if no MDS user area was specified for a vendor with a prefix that matches the first three characters of the exit name.
+08	MDS User Exit Control Block mapped by DKNMDS11

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return code as described below:

- **00** Continue normally.
- **04** Deny the caller's request.

Other Programming Considerations

Notes:

1. This exit point is located in a performance critical area and, as a result, inefficient code affects overall CPCS performance.
2. IBM does not require this exit be re-entrant and recommends that extra STORAGE, GETMAIN, or FREEMAIN invocations be avoided to improve performance.

Example

Member DKNMDSX1, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

MDS_SRCH_EXIT

The Mass Data Set Search Directory (SRCH) exit point allows multiple exits to be invoked during MDS Search Directory processing. You may use these exits to interrogate the MDS user area specific to your exit. You may also request that CPCS not allow the string search directory request.

Environment

This exit is given control within mass data set services TCB and can be detrimental to system performance.

Point of Processing

These exit(s) are called prior to returning the requested MDS string directory information to the caller.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The MDS_SRCH_EXIT=*keyword* is used to specify an exit for this exit point. Up to ten (10) exits may be specified for this exit point. Each exit is called, in the order in which they were specified in the DKNPEXIT profile, until no more exits exist or until one of the exits denies the request with a return code:

Figure 2-13. MDS_RDIR_EXIT Point Parameter List

Offset	Description
+00	Caller's APTCB address
+04	Vendor-specific user area address, may be null if no MDS user area was specified for a vendor with a prefix that matches the first three characters of the exit name.
+08	MDS User Exit Control Block mapped by DKNMDS11

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return code as described below:

- **00** Continue normally.
- **04** Exclude current directory information from the search results.

Other Programming Considerations

Notes:

1. This exit point is located in a performance critical area and, as a result, inefficient code affects overall CPCS performance.
2. IBM does not require this exit be re-entrant and recommends that extra STORAGE, GETMAIN, or FREEMAIN invocations be avoided to improve performance.

Example

Member DKNMDSX1, in the CPCS.V1R11.SAMPLIB partitioned data set may be used as an example when coding assembler exit routines for this exit point.

MDS_COMPRESS_EXIT

This exit allows the user to control the way codelines are compressed, decompressed, expanded, and unexpanded prior to being stored or after being retrieved from the CPCS mass data set. See the “Definition of Terms” in the *CPCS Programming and Diagnostic Guide*. The format of the codeline returned to the program requesting CPCS mass data set services does not necessarily reflect the way the actual data is stored in the CPCS mass data set.

This exit allows the user to control this conversion process. The exit is optional in that it is not required for CPCS to operate. If the exit is installed, it is called for each codeline that is sent to Mass Data Set Services for compression, decompression, expansion, or non-expansion. Not all codelines are expanded or uncompressed. Mass Data Set services can be and is often called to retrieve the compressed version of the codeline. The user exit is *not* called for these codelines. Any manipulation of these codelines must be done in the user’s application program.

Environment

This exit is given control within mass data set services TCB, and can be detrimental to system performance.

Point of Processing

The user exit is called at CPCS initialization and each time thereafter that a codeline needs to be compressed, decompressed, expanded, or non-expanded. When called at initialization, the user exit obtains a re-entrant work area for use by the user exit. The address of this re-entrant work area is maintained by the calling program until CPCS ends. This re-entrant work area is not returned to the user exit when called to perform mass data set services, such as compressing or decompressing a codeline. Since there is no final call (in other words, the CPCS supervisor enters `STOP,STOP` at the ready prompt), users should *not* obtain this re-entrant work area from common storage.

When called by Mass Data Set services, the user must determine whether the re-entrant work area has already been obtained by testing a switch in the user exit control block passed by the calling program. If the re-entrant work area has not been obtained, the user exit must obtain working storage. Each time Mass Data Set Services is called and initialized, the user exit is called. Mass Data Set Services cannot determine if the requesting module needs the services of the Mass Data Set View modules. Therefore, it is possible the user exit may be called only twice, once to initialize and once to end Mass Data Set Services.

The user exit is controlled by function codes located in the user exit control block. Below is the list of function codes and a brief explanation of each code:

- Calls Made at CPCS Initialization

  INIT Initial call used to initialize variables

- Calls Made by CPCS Mass Data Set Services

  ZAIT An application module is requesting Mass Data Set Services; the user exit must obtain storage to save registers and program variables that are needed.

  ZAEN An application module is requesting Mass Data Set Services to free storage obtained for accessing the mass data set. The user exit should free any storage obtained during the ZAIT call.

  DIZD Convert the DIDSCT to the ZDDSCT view (decompress).

  DIDX Convert the DIDSCT to the DXDSCT view (expand).

  DIZX Convert the DIDSCT to the ZXDSCT view (decompress, expand)

  ZDDI Convert the ZDDSCT to the DIDSCT view (compress).

  DXDI Convert the DXDSCT to the DIDSCT view (unexpand).

  ZXDI Convert the ZXDSCT to the DIDSCT view (unexpand, decompress).

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The MDS_COMPRESS_EXIT=*keyword* is used to specify an exit for this exit point. Up to one (1) exit may be specified for this exit point. Each exit is called, in the order in which they were specified in the DKNPEXIT profile until no more exits exist or until one of the exits denies the request with a return code:

Figure 2-14. MDS_COMPRESS_EXIT Point Parameter List

Offset	Description
+00	DKNMDSX2 Control Block Address
+04	Caller's APTCB (DKNAPTCB) Address
+08	CPCS Parameter List (DKNPARM) Address
+12	ZA Control Block (ZADSCT) Address
+16	ZB Control Block (ZBDSCT) Address
+20	ZE Control Block (ZEDSCT) Address
+24	DI Control Block (DIDSCT) Address
+28	ZD Control Block (ZDDSCT) Address
+32	DX Control Block (DXDSCT) Address
+36	ZX Control Block (ZXDSCT) Address

Note: A fullword containing binary zeroes indicates the calling program did not provide this parameter.

Information is passed in the following interface control blocks:

Figure 2-15. MDS View Exit Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Call Parameters	None	DKNMDS2P
Processing modification requests are made with the reason-code field in this control block. The two requests are: **0000** Use default results **0004** The user exit has completed the requested function.		
CPCS APTCB	DKNCATCB	DKNAPTCB
CPCS Parameter List	DKNCPARM	DKNPARM
CPCS MDS Control Block	DKNCRZA	ZADSCT
MDS Compressed SDE	DKNCRZB	ZEDSCT
MDS Uncompressed SDE	DKNCRZE	ZBDSCT
MDS Compressed View of the Codeline	DKNCRZC	DIDSCT
MDS Uncompressed View of the Codeline	DKNCRDI	ZDDSCT
MDS Expanded Compressed View	DKNCRZX	DXDSCT
MDS Expanded Uncompressed View	DKNCRDX	ZXDSCT

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter List Address

R2-R12 Not applicable

R13 Caller's Save Area Address

R14 Return Address

R15 User Exit Entry Point Address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return Codes:

00 The user exit has been called successfully and Mass Data Set Services continues normally, in other words, the function code is DIZD, the user exit reason code is 00, and the user exit has chosen to let Mass Data Set Services uncompress this codeline.

04 The user exit has been called successfully and Mass Data Set Services continues normally, in other words, the function code is DIZD, the user exit reason code is 04, and

the user exit has chosen to uncompress the codeline. Mass Data Set Services does not uncompress the codeline again.

Other Programming Considerations

Notes:

1. This exit point is located in a performance critical area and, as a result, inefficient code affects overall CPCS performance.
2. **IBM requires this exit be re-entrant.**

Example

Member DKNMDSX2, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point. It also has code to change the default decompression for a batch statistic record. This is done for the first three batch statistic records found in the string. The batch statistic record is converted to a user control document that has no meaning in CPCS.

After DKNMRGE runs for sample problem 1, the M-string can be hex-listed using DKNHEXL. The three new user control records should show up in the hex listing. This proves that the user exit has been called successfully.

Chapter 3. System and Application Profiles

Profiles provide a simple and efficient means for customizing CPCS. They consist of members in a PDS (partitioned data set), each of which contains one or more keywords. The values assigned to these keywords control how CPCS behaves.

System profiles, stored in the SYSTPROF data set, affect the CPCS system as a whole. They are loaded once whenever CPCS is started and are used thereafter for the remainder of the CPCS run.

Application profiles, by contrast, affect individual CPCS tasks. As with system profiles, the application profiles for some modules set the values for some of the settings that dictate how CPCS runs. These values can be used to change the module's run-time characteristics, without having to recompile or reassemble the module or bounce CPCS. These application profiles reside in the DKNAPPL data set and are reloaded from there every time a fresh copy of the task they affect starts.

SYSTPROF and DKNAPPL are ddnames in your CPCS execution JCL. For more information, see "External Storage Requirements" on page 1-19.

The naming convention for the application profile members is DKNP*xxxx*, where *xxxx* is the name of the module. For example, DKNPMRGE would be the DKNAPPL member for DKNMRGE. DKNPMRGE could contain a record that would allow the user to turn on the TRACEIT statements within MRGE that are used to display the MRGE module's progression.

Rather than trying to document the record formats that each application profile member supports, see the prologue section of the appropriate application profile member for actual record formats.

Profile Keywords

Profile members consist of a number of keywords. All keywords share the same common characteristics:

- Only one keyword for each profile member line.
- Each keyword must start in column 1.
- A keyword can be of any length.
- A keyword must consist of either upper-case alphabetic characters, numeric digits, or underscores.
- A keyword may not contain imbedded spaces (use underscores instead).
- An equals sign (=) must immediately follow the last character of the keyword.
- The value to be assigned to the keyword must immediately follow the equals sign.
- A space character (" ") must immediately follow the last character of the value. Any comments to the right of this space are ignored.

  For example, the following is a portion of the DKNPMRGE Application Profile:

```
USE_SCATTED_SYSTEM_REJECT_R-STRING=YES
TRACEIT=NO
AUTOSTART_MNOP=NO
LAST_CARD * This card tells MRGE to stop reading the profile
*
*---+----1----+----2----+----3----+----4----+----5----+---
*
```

Note the use of two special keywords in this example, the asterisk (*), and LAST_CARD. These keywords are common to all profiles.

An asterisk (*) in column 1 indicates that the entire line consists of comments. Routines that read profiles always skip over lines that begin with an asterisk.

The LAST_CARD keyword is a directive to stop reading the profile at this point, ignoring all further lines. A common coding practice is to place all keywords at the top of a profile, followed by a LAST_CARD and copious comments. The LAST_CARD keeps programs from wasting time scanning through all the comments.

System Profile Data Set

The System Profile data set CPCS.V1R11.SYSTPROF is a data set that contains information used to define parameters used by CPCS. These parameters may be changed by editing the appropriate member within the data set.

DKNPCPCS Profile Member

The DKNPCPCS profile is the CPCS system configuration profile that contains all the CPCS system parameters. You must change the system profile to conform to your system requirements. Use the JCL in member DKNJEOTP of CPCS.V1R11.CTRL to edit the CPCS system parameters. CPCS also checks all system parameters on the initial startup of CPCS and updates the CPCS system parameter log file (MCMPL) when no errors are detected. The CPCS system parameter log file is used on all subsequent runs to verify all CPCS system parameters. If CPCS detects any parameter changes that require a COLD start, a message is issued and the system shuts down.

The following list describes the CPCS system parameters:

MAXTERM={12 | *n*}
: Specifies the maximum number of display terminals that CPCS supports. This number must also include the keyboard-printer terminal, if one is used to print supervisor terminal messages under CPCS control. This parameter determines the size of the terminal table.

MAXTASK={30 | *n*}
: Specifies the maximum number of tasks that can run concurrently under CPCS. MAXTASK should be at least twice the size of MAXTERM to account for automatically started tasks. The MAXTASK total also includes the number of all user executive tasks that have entries in the CPCS BLDL list. It is used to calculate the amount of storage needed for the ECB list.

MAXBUFF={255 | *n*}
: Specifies the maximum number of communication buffers. You can allocate a maximum of 2500 communication buffers. If you select a large quantity of buffers, you should increase the region size. If you allocate the maximum quantity of buffers, increase the region size by 256K.

Note: You should specify the minimum number of buffers as MAXTASK or MAXTERM. If a CPCS task is unable to obtain a communication buffer, it can continue processing, retry the request for a communication buffer, or end. The MAXBUF value should never be smaller than the values for either MAXTASK or MAXTERM.

NUMPTR={3 | *n*}
Specifies the maximum number of printers that CPCS supports. This value should include TAPEOUT, directly-allocated, and dynamically-allocated printers. Direct JES printers are not counted against this maximum limit.

SPOOL={12 | *n*}
Specifies the maximum number of CPCS spool data sets that CPCS supports. This parameter determines the maximum number of tasks requiring printed output that can run concurrently.

MSTRING={0 | 1}
Specifies whether the M-string is required for processing, where:

MSTRING=0 M-string not required.
MSTRING=1 M-string required.

If you specify MSTRING=1, set the USER-RELEASED flag before running DKNICRE. If you do not set this flag, DKNICRE does not transfer the M-string.

SCRDAYS={2 | *n*}
Specifies the number of days that the scroll multiple-dates table retains. The maximum is 33.

SCRTY={NONE | DASC | RACF | USER}
Specifies your various security options.

NONE Specifies that you do *not* use the data security option. If you omit the parameter, the default is none.

DASC Specifies the CPCS security through the DASC table.

RACF Specifies the MVS RACF security option. You must supply RACF definitions for this option.

USER Specifies a user-supplied security product.

See the *CPCS Programming and Diagnostic Guide* for details concerning the coding of security options and the minimum requirements for the DKNSECRX user exit.

Note: Specify the next three keywords (CHGPSWD, SCRCL, and SCRGP) only when SCRTY=RACF.

CHGPSWD={0 | 1}
Enables you to change passwords during signon, where:

CHGPSWD=0 Permits you to change passwords during signon.
CHGPSWD=1 Prohibits you from changing passwords during signon.

SCRCL=$CPCS
Specifies the RACF class to which you define the CPCS transactions.

SCRGP={0 | CPCS}
Specifies the RACF group to which you define the CPCS logon IDs.

SCRGP=0 TSO users (regardless of the RACF group) can log on CPCS. The RACF group or groups to which the user is defined determine the access to protected CPCS transactions.

SCRGP=CPCS (or any valid RACF group name)
Only TSO users who are defined to or connected to the RACF group can log on CPCS.

Note: The value CPCS is an example; this value can be any valid group name defined to RACF.

TIMECK={0 | *nn*}
Specifies the status of the auto sign-off function for inactive terminals, where:

TIMECK=0 Disables this function

TIMECK=nn Enables this function (where *nn* is the number of minutes, from 1 to 60, that a terminal can remain inactive before it is automatically logged off).

KILLAUT={0 | 1}
Specifies whether DKNDIST automatically starts DKNKILL.

KILLAUT=0 DKNKILL requires a manual start.

KILLAUT=1 An I-string name starts DKNKILL automatically. This is a required option when the concurrent kill option is set for one or more banks in the bank control file (BCF) data set.

Note: The next 10 parameters relate to the MDS. Before defining these parameters, you should perform MDS data-set space calculations to determine preliminary specifications for the MDS. If you change any of the MDS parameters, you must perform a CPCS cold start. For more information on CPCS cold start, see “STYPE Parameter” on page 1-7.

BLKSIZE={1000 | *nnnn*}
Specifies block size for the MDS.

- BLKSIZE (standard 50-byte record)

  `50 x blocking factor`, where the block size is greater than or equal to 50 and less than or equal to the track size of the device allocated for the MDS. The maximum value allowed is 32750. The block size specification must be a multiple of 50. This value plus 12 is the actual block size of the MDS. Select a BLKSIZE that makes effective use of the space allocated for the MDS. The MDS does not have keys. The 12 bytes are part of the block.

- BLKSIZE must be greater than:

  $$\left\lceil \frac{SEGDEV + 7}{8} \right\rceil \times MAXDA + 144$$

- BLKSIZE (nonstandard record)

  Specifies block size for the MDS when you change any of the standard field sizes. (`R x blocking factor`) is less than or equal to `32750`, where R indicates the MDS record length.

BLKSIZE must be a multiple of the record size. You must change the MDSLRECL constant to reflect the new MDS record size. This value plus 12 is the actual block size on the MDS.

For a description of the MDS record format, see the figure "MDS Record Field Lengths" in the *CPCS Programming and Diagnostic Guide*.

MAXDA={1 | *n*}
Specifies the number of direct access devices that the MDS uses. MAXDA x SEGDEV must be less than or equal to 65535.

Note: All DASD that the MDS uses must be the same type.

BLKSEG={10 | *nnnnn*}
You should specify BLKSEG so that a high percentage of any active string can be in one segment. BLKSEG must be greater than or equal to 4 and less than or equal to 32767.

SEGDEV={250 | *nnnnn*}
Specifies the number of logical segments for each device. SEGDEV must be less than 32767, and MAXDA x SEGDEV must be less than or equal to 65535.

NDIRBLK={42 | *nnnn*}
Specifies the number of physical record blocks in the MDS directory index. This should be twice the value of DDIREND so that each index block can have one overflow record if needed. NDIRBLK must be less than or equal to 65535.

EDIREND={1 | *n* }
Specifies the relative block address of the last physical record block of the index data set that contains E-strings.

MDIREND={2 | *n*}
Specifies the relative block address of the last physical record block of the index data set that contains M-strings.

IDIREND={5 | *n*}
Specifies the relative block address of the last physical record block of the index data set that contains I-strings.

RDIREND={6 | *n*}
Specifies the relative block address of the last physical record block of the index data set that contains R-strings.

DDIREND={21 | *n*}
Specifies the relative block address of the last physical record block of the index data set that contains D-strings.

MAXOPEN={60 | *nnn*}
Specifies the maximum number of different strings that can be open concurrently for input, output, or both. This number should be larger than MAXTASK because many tasks can have multiple open strings.

Note: The next three parameters (DUPLEX, BFRAT, and TPBFNM) control the duplexing of data sets and MDS logging options. If any of the parameters change, you must perform a CPCS cold start. For more information on CPCS cold start, see "STYPE Parameter" on page 1-7.

DUPLEX={0 | 1}
Specifies whether the duplex function is active.

DUPLEX=1 Duplex function is active
DUPLEX=0 Duplex function is not active.

For more information about the duplex function, see “Using Data-Set Duplexing” on page 2-6.

BFRAT={0 | *n*}
Specifies the ratio of MDS blocks to tape records. The greater the ratio, the less often tape I/O is performed.

TPBFNM={0–5}
Specifies the number of tape buffers that DKNLOGX uses. The maximum number of tape buffers is five. The tape buffers are written as they are filled. If another buffer is free, I/O operations occur without delay. The greater the number of buffers, the less likely that an I/O event delayed.

SUBBAL={SLST | SBAL | BOTH | NONE}
Specifies the subsequent-pass balancing list that DKNDIST automatically starts.

SLST Specifies that DKNDIST automatically starts DKNSLST.

SBAL Specifies that DKNDIST automatically starts DKNSBAL.

BOTH Specifies that DKNDIST automatically starts both DKNSLST and DKNSBAL.

NONE Specifies that DKNDIST does not automatically start DKNSLST or DKNSBAL.

MAXSORT={NO | *nn*}
Specifies whether concurrent sorting is active. If it is active, the value coded becomes the maximum number of concurrent tasks that can run a sort. You must generate a new module to complete activation of concurrent sorting for your CPCS system. For more information about concurrent sorting, see “Concurrent Sort Generation” on page 2-32.

CLASS={A | *n*}
Specifies the default CPCS output class (A through Z, 0 through 9). This class specification is assigned to application tasks that did not specify a class in DKNBLDL through the APCB macro. See “Describing an Application Task’s Environment” on page 2-15 for more information about the BLDL table.

DUMPCLAS={F | *n*}
Specifies the default CPCS Dump class (A through Z, 0 through 9). This class specification is assigned to any dumps produced in the system.

LNTNAME={*name*}
Specifies the name of an LU 6.2 node table module that loads during CPCS startup. The LU 6.2 node table module must reside in the CPCS load library. If you use LU 6.2-attached document processors, this parameter is required.

For information about creating this module, see the *3890/XP MVS Support and 3890/XP VSE Support Program Reference*.

LOG={NO | YES}
Specifies the Logging facilities used.

NO Does not log the mass data set.

YES Logs the mass data set to disk or tape. Logging is activated. You must generate a new module to complete activation of Logging in your CPCS system (see "Installing the Logging Subsystem" on page 2-7).

RCVY={NO | YES}
Specifies whether recovery is possible when LOG=NO. This parameter is ignored when LOG=YES. Set this parameter to YES only if you have completed the steps described in "Installing the Logging Subsystem" on page 2-7.

NO Does not perform the Logging initialization required for recovery during CPCS startup. Set this parameter to NO if you have installed CPCS without the Logging function.

YES This Logging initialization required for recovery of strings occurs during CPCS startup. When LOG=NO and RCVY=YES, new data added to the MDS is not logged, but any existing logged data sets can be recovered.

SMACT={NO | ESM}
Specifies whether the functions associated with the Enhanced System Manager are activated.

NO The Enhanced System Manager is not activated.

ESM The Enhanced System Manager is activated. For more information, see the *CPCS Enhanced System Manager User's Guide*.

KBBLKSZ={400 | *nnn*}
Specifies the kill-bundle block size. The kill-bundle data set must be allocated as unblocked, fixed-length records that are multiples of 400 (the default value is 400).

Note: An increase in the block size can effectively speed up processing but can also result in CLSM missing records from the kill-bundle data set. If KILL and CLSM are running simultaneously, CLSM can detect an EOF condition on the kill-bundle data set at the same time KILL is enqueued on the data set to add records.

MFBLKSZ={50 | *nnnn*}
Specifies the microfilm block size. The microfilm data set must be allocated as unblocked, fixed-length records that are multiples of 50 (the default value is 50).

MAXSUB={255 | *nnn*}
Specifies the maximum number of subsets that can be created for an entry. The valid range is 2 through 255, with a default of 255. This value is the default for all banks, but it may be overridden in the bank control file for banks other than 001. The value must be 3 digits, right-justified, and zero-filled.

SVC={NO | *nnn*}
Specifies the SVC number (*nnn*) assigned to the nonswap SVC for your site. This parameter must be coded only if you intend to use the nonswap SVC.

LANG={US | *nn*}
Specifies to CPCS the user's choice of environment. The valid values are US (for ENU environment), UK (for the United Kingdom environment) and OZ (for Australian environment). The environment selection causes CPCS to adapt specific processing to the selected environment. For example, selecting the Australian environment suppresses hyphen display in the routing field for

adjustment processing (see *CPCS Online Adjustments*) and for OLRR processing.

Important!

If you code LANG=OZ on this parameter, set the variable &DASHDSP to 0 in the MDX macro to fully disable the hyphen display in the routing field.

DATE={mm/dd/yy | *nnnnnnnnn*}
Specifies the CPCS default date format. Dates on all reports and screens will be displayed in this format. Valid date formats are:

mm/dd/yy	mm/dd/yyyy
dd/mm/yy	dd/mm/yyyy
yy/mm/dd	yyyy/mm/dd
yy.ddd	yyyy.ddd

IOMODE={31 | 24}
Specifies the I/O addressing mode. This value must be set to IOMODE=24 for customers who are running MVS/DFP.

Note: If this value is not set correctly, you may receive an assembly error for the APTR DCB because it specifies an invalid parameter of DCBE. If you receive this error, you must specify IOMODE=24.

SYSID={CPCS | *nnnn*}
Specifies a unique job identifier that CPCS can use to verify the existing installation level of operation or to identify test version operation. The Default value is CPCS

CIMSID={*nnnn*}
Specifies the four character identifier of the unique CIMS subsystem that processes CPCS document images. For information about defining the CMID, see the CPCS Programming and Diagnostic Guide.

AUTORST={Z | *n*}
Specifies a character that identifies the CPCS job that captured the the last items that passed through a document processor. You access this character during automatic restart processing.

Note:

1. Different CPCS jobs that access the same document processors must have different values specified for the AUTORST parameter.
2. If this parameter is not specified, automatic restart processing is deactivated.
3. If more than one character is specified, only the first one is used. The remaining characters are ignored.
4. The # character deactivates automatic restart

OVERRIDE={NO | YES}
Specifies whether to continue to WARM/REST start when the system noticed a change to one of the parameters which require a COLD start.

Important!

If you set the OVERRIDE parameter to YES during this condition, unpredictable results may occur!

LE370={YES | NO}
Specifies whether LE370 COBOL modules will be executed in CPCS. If your MVS system does **NOT** support the LE370 environment, this parameter should be set to **NO**

START_DKNCOMP={YES | NO}
Specifies whether DKNCOMP should be automatically started during CPCS startup regardless of the CPCS Start parameters.

BUFEXT={0 | *nnnnn***}**
Specifies the Communication Buffer Extended Area Length. This User area is carried along with each buffer generated in CPCS. This area can ONLY be modified by using the User Area Manager. For more details on the User Area Manager, see the UEM documentation in the *CPCS Programming and Diagnostics Guide*, Chapter 2, “User Exit Manager”. The maximum value is 32767.

PARMEXT={0 | *nnnnn***}**
Specifies the PARMLST Extended Area Length. This user area is carried along with the PARMLST area. This area can ONLY be modified by using the User Area Manager. For more details on the User Area Manager, see the UEM documentation in the *CPCS Programming and Diagnostics Guide*, Chapter 2, “User Exit Manager”. The maximum value is 32767.

APTCBEXT={0 | *nnnnn***}**
Specifies the APTCB Extended Area Length. This user area is carried along with each APTCB generated in CPCS. This area can ONLY be modified by using the User Area Manager. For more details on the User Area Manager, see the UEM documentation in the *CPCS Programming and Diagnostics Guide*, Chapter 2, “User Exit Manager”. The maximum value is 32767.

MAXEXIT_PTS={10 | *nnnnn***}**
This keyword specifies the maximum number of user exit points supported by the User Exit Manager Facility. See UEM documentation in the *CPCS Programming and Diagnostics Guide*, Chapter 2, “User Exit Manager”.

The maximum value is 32767.

MAX_EP_EXIT={8 | *nnnnn***}**
This keyword, Maximum Exits Per Exit Point, specifies the maximum number of exit programs that are executed per exit point.

Example

If you code MAX_EP_EXIT=2, then you can specify the following in the SYSTPROF member DKNPEXIT:

MDS_FREE_EXIT=**DKNMDXS1**

MDS_FREE_EXIT=**DKNMDXS2**

GLOBAL_MDSUAL={100 | *nnn***}**
This keyword specifies the global user area length in bytes of extended strings that are not created by the RCVY or ETUT tasks. RCVY and ETUT tasks have the same global user area length they had when they were logged or unloaded.

The keyword’s default value is 100. Its maximum value is 999.

CPCS must be restarted when this value is changed. It is not recommended to decrease the global user area length, because that could result in truncated global user areas by tasks that transfer global user areas from input extended input strings to extended output strings. See the section “Extended Strings - External Storage Considerations” in Chapter 1 of the *CPCS Programming and Diagnostic Guide* for potential implications of large global user area sizes.

CPCS_PASSWORD={*xxxxxxxxxxxx***}**
This keyword is used to set the CPCS password for the CPU on which you are running CPCS. You must obtain a password from IBM support for each CPU on which CPCS is run.

DKNPRCVY Profile Member

The DKNPRCVY profile is the Logging system configuration profile that contains all the Logging system parameters. You must change the logging profile to conform to your system requirements. Use the JCL in member DKNJEDTP of CPCS.V1R11.CTRL to edit the logging system parameters. CPCS also checks all the logging parameters on the initial startup and updates the CPCS system parameter log file (MCMPL) when no errors are detected. The CPCS system parameter log file is used on all subsequent runs to verify all logging parameters. If CPCS detects any parameter changes that require a COLD start, a message is issued and the system shuts down.

The following list describes the recovery system parameters:

LOGDEV={DISK | TAPE}
Specifies whether the MDS is logged to disk or tape. See the description of the DKNLT tape data set on “Optional Tape Data Sets” on page 1-30.

DPXDISK={YES | NO}
Specifies whether the disk log files (DKNLD1, DKNLD2) are duplexed when they are created. This parameter has no effect if LOGDEV=TAPE has been specified.

DPXTAPE={ YES | NO }
Specifies whether the tape log files are duplexed when they are created. If you use the YES option, two identical log tapes are created. When LOGDEV=TAPE has been specified, CPCS performs this duplexing. When LOGDEV=DISK has been specified, the duplexing is performed by the logging backup program (DKNBKUP or DKNBBKUP).

DPXCNTL={ YES |NO}
Specifies whether the logging control files are duplexed when they are updated.

BACKUP={BKUP | SUB member | SUB dsn(member) }
Specifies the start command that is used when a disk backup job is run.

BKUP Causes DKNBKUP to start automatically to perform the backup as a CPCS task.

SUB member
Causes DKNSUB to start automatically and to submit the specified member of the partitioned data set that the DKNSUBMT DD statement defines in the DKNJRUN job stream.

SUB dsn(member)

Causes DKNSUB to start automatically and to submit the specified member of the specified data set to the internal reader for batch processing.

This format of the command can be used only if there is not a DKNSUBMT DD statement in the DKNJRUN job.

To allow for user-specified methods for starting batch jobs automatically, this parameter is not checked for validity. However, failure to specify a valid command prevents the backup jobs from starting automatically.

A sample backup job, DKNJLBKP, exists in the CPCS.V1R11.CTRL library. You can use DKNJLBKP as the target for your BACKUP SUB command once you have customized the job for your installation's environment. This parameter has no effect if LOGDEV=TAPE has been specified.

RCVSIZE={30011 | nnnn}

Specifies the number of records on the string names file (DKNRCVY). A change to this parameter starts the initialization process automatically if the AUTO_INIT=YES parameter is specified. If AUTO_INIT=NO is specified, the DKNGLINT job must be run manually.

This value is calculated as:

```
(number of strings created each day x log tape retention period) x 2
```

To estimate the number of strings created each day, you should perform the following calculations:

1. Estimate the average number of prime entries each day by dividing the daily prime volume by the average entry size.
2. Estimate the average number of strings generated from each prime entry, based on the following:
   - One prime-pass I-string
   - If running high-speed reject re-entry, then one additional I-string
   - Number of pockets on sorter, one D-string in each
   - Repaired reject strings, one or more R-strings
   - Number of rehandle pockets, I-strings
   - Number of rehandle pockets, number of D-string pockets
   - If running high-speed reject reentry, one work M-string
   - Merged string, before balancing
   - Adjustment M-string
   - Adjusted M-string
3. If sub-strings are used, multiply the total strings for each entry by the average number of sub-I strings for each prime-pass entry.

   Note: Take extreme care when calculating this value. Underestimating this value can cause the string names file to reach full capacity.

RCVYBLKZ={11475 | nnnnn}

Specifies the block size of the string names file. This file contains a cross-reference of all strings and the log files to which they were logged. The block size must be a multiple of 153. A change to this parameter starts the initialization process automatically if the AUTO_INIT=YES parameter is

specified. If AUTO_INIT=NO is specified, the DKNGLINT job must be run manually.

Note: Although making the number larger causes a faster initiation of CPCS when reading the whole string names file, a larger number also causes larger blocks to be written whenever CPCS processing updates a logical record. If this number is too large, CPCS operation can be adversely affected during peak processing periods.

VLSRSIZ={50 | nnnn}

Specifies the maximum number of log files tracked by the logging function. A change to this parameter starts the initialization process automatically if the AUTO_INIT=YES parameter is specified. If AUTO_INIT=NO is specified, the DKNGLINT job must be run manually.

This value should be calculated as:

```
(number of tapes created each day x log tape retention period) x 1.5
```

MAXATMP={60 | nnnnn}

Specifies the number of consecutive attempts to find an available record on the string name file (DKNRCVY) before the information on the is lost to the logging index file.

This value should never be less than 50 or more than 65535.

Set this value to the size of the string names file (RCVSIZE), or greater, not exceeding 32767. For more information, see the definition for RCVSIZE earlier in this profile description for System Profile DKNPRCVY.

MAXSTGS={99 | nnnn}

Specifies the maximum number of strings to be recovered in a single pass of DKNRCVY. This value should be large enough to accommodate a full recovery of the mass data set, regardless of how many strings are involved. If this number is exceeded, it causes the RCVY 047 message to appear.

TPRETPD={3 | nnnn}

Specifies the number of days that the string name entries are maintained on the string name file (DKNRCVY). This has no bearing on the recovery of the string, only whether there is room to write a new string to the recovery names file.

BLDTBL={ YES | NO}

Specifies whether the string names file (DKNRCVY) is loaded above the 16MB line during CPCS initialization. Loading the file into memory causes initialization to take longer, but reduces the processing overhead of the Logging function during ongoing runs and also enhanced performance during the selective recovery process.

Multiply the RCVSIZE value by 153 to calculate the required amount of memory.

TG_UPDATE={YES | NO}

Update the tracer group data set during recovery.

RDX={*xxxxxxxx* | blanks}

Override the log data set definition generated by the logging system.

Note: This option is only needed if trying to recover tape data from a system which does *not* contain the log data set definition on the tape, for

instance, V1M9. All CPCS Release 11 log tapes contain the definition on the tapes. IBM recommends this parameter be set to the default.

NUM_STRG_VOLS={200 | nnnn}
Maximum number of volumes that can be displayed during a recovery process.

BYP_SORTTYPE={Y | N}
Bypass the requirement of Sort Type for the external string selection.

AUTO_INIT={YES | No}
Automatically initialize the RCVY files and generate the string names report.

Note: It is advisable to run a backup before bringing up the system with AUTO_INIT=YES. All data in the disk log files is lost when the initialization takes place.

The initialization process of the logging files occurs when *one* of the following recovery parameters is changed in the system profile member DKNPRCVY.

RCVSIZE Number of records contained on DKNRCVY / DKNRCVYD

RCVYBLKZ Block size of DKNRCVY / DKNRCVYD

VLSRSIZ Number of records contained on DKNRCVSR / DKNRCVSD

DPXCNTL When DPXCNTL=NO changed to DPXCNTL=YES

DPXDISK When DPXDISK=NO changed to DPXDISK=YES.

The initialization process of the logging files also occurs when the following CPCS system parameter is changed in the system profile member DKNPCPCS:

MAXOPEN Number of records contained on DKNRCVCK / DKNRCVCD; calculates the block size of these files as 3 times the value.

Note: AUTO_INIT cannot detect the reallocation of the RCVY data sets. If this is done and DKNGLINT is not run and no RCVY parameters are changed, AUTO_INIT will not run. The system detects problems with the logging files and displays error messages and abends with a U004.

DFP_DFSMS={DFP | DFSMS}
The DFP_DFSMS parameter determines whether DKNLJMP can use the facilities provided by DFSMS. If the user does *not* have DFSMS installed, the parameter must be set to DFP_DFSMS=DFP. If DFSMS is installed, the variable may be set to either DFP_DFSMS=DFP or DFP_DFSMS=DFSMS. The default is DFP_DFSMS=DFP.

DKNPEXIT Profile Member

The DKNPEXIT profile member is a system profile data set that contains records that are used to specify user exits to be run at the specific exit point with which they are associated. More than one exit may be specified for a single exit point, and the exits are called in the order specified in this profile member.

DKNPEXIT records must be in the following format:

```
exit-point=user-exit,option1,option2,....
```

Where:

exit-point The exit point name

user-exit The user exit name to be called at this exit point

options One of the following options:

DEACTIVATE=YES|NO
Specifying NO for this option prevents the online deactivation of this user exit and exit point. This option should be specified for exits that are critical to the correct implementation of your CPCS system. The default value for this option is YES, meaning the exit may be deactivated with the online facility.

A DKNPEXIT record may also be a comment record by placing an asterisk (*) in column 1.

Keyword	Description
DKNALLO_EXIT1000_REQ_VALIDATION	This exit allows the further validation of the operator-entered request right after the validations have been successfully performed and before the actual document processor allocation/unallocation operation takes place.
DKNALLO_EXIT2000_MESSAGE	This exit allows the opportunity to interpret the message generated by DKNALLO and to send a more descriptive message explaining the problem or the ability to write a help screen created by the user. This exit is entered each time a message is generated by DKNALLO.
DKNATASK_APPLTSK_START_EXIT01	This exit allows you the opportunity to update any user extended area prior to starting an application task. For more detail, see “CPCS System Exit Points” on page 4-57.
DKNATASK_EXECTSK_START_EXIT01	This exit allows you the opportunity to update any user extended area prior to starting an executive task. For more detail, see “CPCS System Exit Points” on page 4-57.
DKNATSK2_INITIALIZE_EXIT_01	This exit allows you the opportunity to update any user extended area after the APTR DCB has been opened. For more detail, see “CPCS System Exit Points” on page 4-57.

DKNATSK2_TERMINATE_EXIT_01

This exit allows you the opportunity to update any user extended area after the APTR DCB has been closed. For more detail, see “CPCS System Exit Points” on page 4-57.

DKNATSK2_WRITE_APTR_EXIT_01

This exit allows you the opportunity to update any user extended area after each WRITE to the APTR DCB. For more detail, see “CPCS System Exit Points” on page 4-57.

DKNCYCDT_DATE_VALIDATE_EXIT

This exit allows the further validation of the cycle and/or endorse date before a cycle table modification takes place. DKNCYCDT does edit checking and calls DKNDATE; then the user is allowed to do further validation through this exit point.

DKNINIT_POST_DKNITASK_EXIT

This exit allows you the opportunity to initialize any user control block. For more detail, see “CPCS System Exit Points” on page 4-57.

DKNLOADR_EXIT_01

This exit allows you the opportunity to request that a different program be loaded and executed. For more detail, see “CPCS System Exit Points” on page 4-57.

DKNMBEGN_APPLSK_START_EXIT01

This exit allows you the opportunity to update any user extended area after the APTCB is built. For more detail, see “CPCS System Exit Points” on page 4-57.

DKNMDSVC_FREE_STRING=

This keyword specifies the name of a user exit that you want invoked during MDS string FREE processing. You may use these exits to deny a FREE SPACE request.

DKNMDSVC_OPEN_STRING=

This keyword specifies the name of a user exit that you want invoked during MDS string OPEN processing. You may specify up to ten (10) user exits for this exit point.

DKNMDSVC_READ_DIRECTORY=

This keyword specifies the name of a user exit that you want invoked during MDS Read Directory processing. You may specify up to ten (10) user exits for this exit point.

DKNMDSVC_SEARCH_DIRECTORY=

This keyword specifies the name of a user exit that you want invoked during MDS Search Directory processing. You may specify up to ten (10) user exits for this exit point.

DKNMDSV4_COMPRESSION=

This keyword specifies the name of a user exit that you want invoked during item compression and decompression processing. You may specify up to one (1) user exit for this exit point.

DKNMTASK_INITIALIZE_EXIT_01

This exit allows you the opportunity to initialize any user control block after all tasks have been successfully initialized and attached. For more detail, see “CPCS System Exit Points” on page 4-57.

DKNMTASK_TERMINATION_EXIT_01

This exit allows you the opportunity to perform any user clean up before all the major subtasks are terminated prior to shutdown. For more detail, see “CPCS System Exit Points” on page 4-57.

MDS_COMPRESS_EXIT=

This keyword specifies the name of a user exit that you want invoked during Item compression and decompression processing. You may specify up to one (1) user exit for this exit point.

MDS_FREE_EXIT=

This keyword specifies the name of a user exit that you want invoked during MDS string FREE processing. You may use these exits to deny a FREE SPACE request.

MDS_OPEN_EXIT=

This keyword specifies the name of a user exit that you want invoked during MDS string OPEN processing. You may specify up to ten (10) user exits for this exit point.

MDS_RDIR_EXIT=

This keyword specifies the name of a user exit that you want invoked during MDS Read Directory processing. You may specify up to ten (10) user exits for this exit point.

MDS_SRCH_EXIT=

This keyword specifies the name of a user exit that you want invoked during MDS Search Directory processing. You may specify up to ten (10) user exits for this exit point.

DKNPISNG Profile Member

The DKNPISNG profile member is a system profile data set that contains records that may be used to customize the format of item sequence numbers and set item sequence number ranges. This profile is accessed by the Item Sequence Data Set Generation job (DKNGISN).

All parameters must start in column 1. Records with an * in column 1 are ignored. The following parameters may be specified in the DKNPISNG profile:

REGION_NUMBER_LENGTH=l

- This parameter indicates the number of digits (l) used to hold the region number in the left-most part of the 8 lowest order digits of item sequence numbers throughout CPCS.
- This parameter may only be present once.
- If present, this must be the first parameter.

- If the parameter is present, l must be 0, 1 or 2.
- This parameter defaults to 0 (no region number).

SORTER_nn_REGION_NUMBER=*r/rr*

- This parameter specifies the region number (*r/rr*) to be included in sequence numbers corresponding to items captured on a document processor (*nn*).
- This parameter may not be present if no region numbers are used.
- If this parameter is present, *nn* must be a valid document processor number (01 - 99).
- If this parameter is present and 1-digit region numbers are used, *r/rr* must be a 1-digit region number (0 - 9). If the parameter is present and 2-digit region numbers are used, *r/rr* must be a 2-digit region number (00 - 99).
- This parameter may be present up to 99 times, for each of the 99 valid CPCS logical document processor numbers.
- This parameter may only be present once for each document processor.

DEFAULT_SORTER_REGION_NUMBER=*r/rr*

- This parameter specifies the region number (*r/rr*) to be included in sequence numbers corresponding to items captured on document processors not having a region number assigned with the SORTER_*nn*_REGION_NUMBER parameter.
- This parameter may only be present once.
- This parameter may not be present if no region numbers are used.
- If the parameter is present and 1-digit region numbers are used, *r/rr* must be a 1-digit region number (0 - 9). If the parameter is present and 2-digit region numbers are used, *r/rr* must be a 2-digit region number (00 - 99).
- This parameter defaults to region number 0 when 1-digit region numbers are used. The parameter defaults to 00 when 2-digit region numbers are used.

SORTER_*nn*_ROLLOVER_SEQUENCE_NUMBER=*ssssssss*

- This parameter specifies the prime-pass start item sequence number rollover value (*ssssssss*) for a sorter (*nn*). Captures, plus enhanced-prime subsequent entries for XF sort types using the 'setdev at entry breaks' O record option, start with an item sequence number of 00000001/R0000001/RR000001. The value r/rr is the sorter's region number when the next sequence number

to be used (excluding the region number portion) exceeds the value specified by this parameter.

- This parameter affects the first item sequence number assigned in a capture. It also affects the first sequence numbers assigned in enhanced-prime subsequent entries for XF sort types, when the 'setdev at entry breaks' O record option is active. Item sequence numbers do not roll over when the value *ssssssss* is reached while capturing items.
- If this parameter is present, *nn* must be a valid document processor number (01 - 99).
- If this parameter is present, *ssssssss* must be an 8-digit sequence number that does not include a region number. The sequence number must be smaller than 10000000 if 1-digit region numbers are used. The sequence number must be smaller than 01000000 if 2-digit region numbers are used. The sequence number must be greater than 00000000.
- This parameter may be present up to 99 times, for each of the 99 valid CPCS logical document processor numbers.
- This parameter may only be present once for each document processor.

DEFAULT_ROLLOVER_SEQUENCE_NUMBER=*sssssssss*

- This parameter specifies the sequence number rollover value to be used for document processors not having a sequence number rollover value assigned with the SORTER_*nn*_ROLLOVER_SEQUENCE_NUMBER parameter.
- This parameter may only be present once.
- If this parameter is present, *ssssssss* must be an 8-digit sequence number that does not include a region number. The sequence number must be smaller than 10000000 if 1-digit region numbers are used. The sequence number must be smaller than 01000000 if 2-digit region numbers are used. The sequence number must be greater than 00000000.
- This parameter defaults to 09000000 when no region number are used and when 1-digit region numbers are used. The parameter defaults to 00900000 when 2-digit region numbers are used.

REGION_*r/rr*_**SEQUENCE_NUMBER_RANGE=(***llllllll,hhhhhhhh***)**

- This parameter specifies the range of sequence numbers (*llllllll,hhhhhhhh*) to be used as non-MICR task sequence numbers for a region number (*r/rr*). The first sequence number to be used for the region number is the

low-range sequence number. The low-range sequence number is again used for the region number right after the high-range sequence number has been used.

- This parameter may only be present when region numbers are used.
- If the parameter is present and 1-digit region numbers are used, *r/rr* must be a 1-digit (0 - 9) region number. If the parameter is present and 2-digit region numbers are used, *r/rr* must be a 2-digit region number (00 -99).
- If this parameter is present, *llllllll* and *hhhhhhhh* must be 8-digit sequence numbers that do not contain regin numbers. The SEQUENCE numbers must be smaller than 10000000 if 1-digit region numbers are used. The sequence numbers must be smaller than 01000000 if 2-digit region numbers are used. The high range sequence number (*hhhhhhhh*) must be higher than the low range (*llllllll*) sequence number. The sequence numbers must be greater than 00000000.
- This parameter may be present up to 10 times if 1-digit region numbers are used, or up to 100 times if 2-digit region numbers are used, for each of the valid region numbers.
- This parameter may only be present once for each region.

DEFAULT_REGION_SEQUENCE_NUMBER_RANGE=(*llllllll,hhhhhhhh***)**

- This parameter specifies the range of sequence numbers (*llllllll,hhhhhhhh*) to be used as non-MICR task sequence numbers for regions not having a sequence number range assigned with the REGION_*r/rr*_SEQUENCE_NUMBER_RANGE parameter. It also specifies the range of sequence numbers (*llllllll,hhhhhhhh*) to be used as non-MICR task sequence numbers when no regions are used.
- This parameter may only be present once.
- If this parameter is present, *llllllll* and *hhhhhhhh* must be 8-digit sequence numbers that do not contain region numbers. The sequence numbers must be smaller than 10000000 if 1-digit region numbers are used. The sequence numbers must be smaller than 01000000 if 2-digit region numbers are used. The high-range sequence number (*hhhhhhhh*) must be higher than the low-range (*llllllll*) sequence number. The sequence numbers must be greater than 00000000.
- This parameter defaults to (00000001,09000000) when no region numbers are used and when 1-digit region numbers are used. The parameter defaults to (00000001,00900000) when 2-digit region numbers are used. A non-existing profile or a profile with no

parameters result in no region numbers used and an item sequence number range of 00000001 through 09000000.

CAUTION:
The first 2 digits of the 8 lowest-order digits of the starting item sequence number are truncated on non-XF sort type document processor initializations; only the 6 lowest-order digits of item sequence numbers are used in these types of captures. Therefore, if region numbers are used, they are not stamped on checks as part of their sequence number.

Microfilm numbers are the last 6 digits of the corresponding item sequence number.

DKNPATSK Profile Member

The DKNPATSK Profile member is a System Profile data set that contains records that may be used to reduce and/or limit the amount of below-the-line storage used by CPCS.

Task group records may be used to group tasks so they are executed in a controlled manner. One example of how this may be used is to group the manual tasks in one group, and the auto-started tasks in another group. By placing a relatively small value in the group record, you can limit how many auto-started tasks may be active at one time.

A low-limit record may also be used to define the minimum below-the-line large block that must be available before a task can be started by DKNATASK.

Task Grouping Example

```
TASK      TASKGRP=          Task Group Records
DKNDIST     1               GROUP1=003
DKNMRGE     1               GROUP2=100
DKNMDIS     1               .
DKNKILL     1               .
DKNPLST     1               .
DKNOLRR     2
DKNSDIR     2
```

Figure 3-1. Task Grouping Example

Assume that tasks attempt to start in the following order:

DKNDIST
DKNMRGE
DKNKILL
DKNPLST
DKNDIST

For this example, if you had tasks start in quick succession so that all were trying to be active at the same time, the task activity status would be:

```
DKNDIST   active
DKNMRGE   active
DKNKILL   active
DKNPLST   queued
DKNDIST   queued
```

As one of the first 3 active tasks completes and detaches, the next task, DKNPLST would attach and begin execution.

DKNPATSK Record Formats

The individual tasks are assigned a group number within the BLDL. See the *CPCS Programming and Diagnostic Guide* section “Describing an Application Task's Environment” for examples of coding the APCB macro within DKNBLDL to set the group number for a task.

The DKNPATSK data set member is read by DKNATASK upon CPCS startup, and the values for each record are set from the member. If a new value is desired for a particular record, you must edit the DKNPATSK member and restart CPCS.

DKNPATSK records must be in one of the following formats:

DKNPATSK record format 1:

Position	Length	Description
1	5	'GROUP' — A constant that identifies the record.
6	1	*x*=Group number — The group number. Valid values for *x* are 1–9.
7	1	'=' — A constant.
8	3	*nnn* — A 3-digit number that contains the value of how many tasks of that group may be active at one time. Valid values are 001–999.
11	70	Reserved for IBM

Task group records are not required for all possible groups You may only code records for the groups you care about. If a task group record is omitted for a group, the default value for that group is 999.

A task group record for task group 0 is ignored. The number of tasks for group 0 always defaults to 999.

DKNPATSK record format 2:

Position	Length	Description
1	9	'MESSAGES=' — a constant that identifies the record.
10	1	{N|Y} defines if queuing messages should be displayed. The default is N, for no messages.
11	70	Reserved for IBM.

DKNPATSK record format 3:

Position	Length	Description
1	13	'ATASK_LOWLIM=' — a constant that identifies the record.
14	2 - 5 (var.)	xxxxK, where *xxxx* is between 0K and 1024K.
		1024K 1.0M of storage as the low storage limit
		768K .75M of storage as the low storage limit
		512K .50M of storage as the low storage limit
19	70	Reserved for IBM.

If ATASK_LOWLIM is not specified, or if it is specified as 0K, low-limit testing is not performed.

A DKNPATSK record may also be a comment record by placing an '*' in column 1.

Application Profile List

The following is a complete list of all the application profiles and their keywords.

DKNPEMRG

The DKNPEMRG profile controls the behavior of DKNEMRG. The keywords are:

POCKET_MERGE_R_STRING_TABLE_CODELINES=#####

This keyword controls how much space is reserved in the pocket merge table for R-string codelines (other than piggybacks). Replace "#####" with a five-digit number; use leading zeros, if necessary. The larger this number, the larger the R-string EMRG can handle, but also the greater the amount of memory that EMRG consumes.

Note: If EMRG is not asked to perform a Pocket Merge, it does not allocate a pocket merge R-string table.

The actual size of the pocket merge R-string table, in bytes, is given by the formula:

```
44 + ( (28 + length of DKNCRDI2) * (C + P + 10) )
```

Where C is the number of POCKET_MERGE_R_STRING_TABLE_CODELINES, and P is the number of POCKET_MERGE_R_STRING_TABLE_PIGGYBACKS. The value of (28 + length of DKNCRDI2) should be rounded up to the nearest multiple of 4.

The default value for this keyword is 05000. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

POCKET_MERGE_R_STRING_TABLE_PIGGYBACKS=#####

This keyword controls how much space is reserved in the pocket merge table for R-string piggyback codelines. Replace "#####" with a five-digit number; use leading zeros, if necessary. The larger this number, the more piggybacks EMRG can handle, but also the greater the amount of memory that EMRG consumes.

Note: If EMRG is not asked to perform a Pocket Merge, it does not allocate a pocket merge R-string table.

The actual size of the pocket merge R-string table, in bytes, is given by the formula:

```
44 + ( (28 + length of DKNCRDI2) * (C + P + 10) )
```

Where C is the number of POCKET_MERGE_R_STRING_TABLE_CODELINES, and P is the number of POCKET_MERGE_R_STRING_TABLE_PIGGYBACKS. The value of (28 + length of DKNCRDI2) should be rounded up to the nearest multiple of 4.

The default value for this keyword is 01000. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

CONSOLIDATED_MERGE_R_STRING_TABLE_CODELINES=#####

This keyword controls how much space is reserved in the consolidated merge table for R-string codelines (other than piggybacks). Replace "#####" with a five-digit number; use leading zeros, if necessary. The larger this number, the larger the R-string EMRG can handle, but also the greater the amount of memory that EMRG consumes.

Note: If EMRG is not asked to perform a consolidated merge, it does not allocate a consolidated merge R-string table.

The actual size of the consolidated merge R-string table, in bytes, is given by the formula:

```
44 + ( (28 + length of DKNCRDI2) * (C + P + 10) )
```

where C is the number of CONSOLIDATED_MERGE_R_STRING_TABLE_CODELINES, and P is the number of CONSOLIDATED_MERGE_R_STRING_TABLE PIGGYBACKS. The value of (28 + length of DKNCRDI2) should be rounded up to the nearest multiple of 4.

The default value for this keyword is 05000. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

CONSOLIDATED_MERGE_R_STRING_TABLE_PIGGYBACKS=#####

This keyword controls how much space is reserved in the consolidated merge table for R-string piggyback codelines. Replace "#####" with a five-digit number; use leading zeros, if necessary. The larger this number, the more piggybacks EMRG can handle, but also the greater the amount of memory that EMRG consumes.

Note: If EMRG is not asked to perform a consolidated merge, it does not allocate a consolidated merge R-string table.

The actual size of the consolidated merge R-string table, in bytes, is given by the formula:

```
44 + ( (28 + length of DKNCRDI2) * (C + P + 10) )
```

where C is the number of CONSOLIDATED_MERGE_R_STRING_TABLE_CODELINES, and P is the number of CONSOLIDATED_MERGE_R_STRING_TABLE PIGGYBACKS. The value of (28 + length of DKNCRDI2) should be rounded up to the nearest multiple of 4.

The default value for this keyword is 01000. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

HSRR_MERGE_I52M_STRING_TABLE_CODELINES=#####

To complete a HSRR merge, DKNEMRG loads the 52-M string (a temporary string representing the results of codeline data matching between the HSRR I-sting and its prime-pass reject D-string) into the HSRR merge I52M-string table. This keyword controls how large the table can be. Replace "#####" with a five-digit number; use leading zeros, if necessary. The larger this number, the larger the 52-M string that EMRG can handle, but also the greater the amount of memory that EMRG consumes.

Note: If EMRG is not asked to perform a HSRR Merge, it does not allocate a HSRR merge I52M-string table.

The actual size of the HSRR merge I52M-string table, in bytes, is given by the formula.

```
20 + ( (14 + length of DKNCR52M) * (C + 1) )
```

Where C is the number of HSRR_MERGE_I52M_STRING_TABLE_CODELINES. The value of (14 + length of DKNCR52M) should be rounded up to the nearest multiple of 4.

The default value for this keyword is 05000. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

HSRR_MERGE_I50M_STRING_TABLE_CODELINES=#####

To complete a HSRR merge, DKNEMRG loads the 50-M string (a temporary string representing the results of codeline data matching between the HSRR I-sting and R-strings) into the HSRR merge I50M-string table. This keyword controls how large the table can be. Replace "#####" with a five-digit number; use leading zeros, if necessary. The larger this number, the larger the 50-M string that EMRG can handle, but also the greater the amount of memory that EMRG consumes.

Note: If EMRG is not asked to perform a HSRR Merge, it does not allocate a HSRR merge I50M-string table.

The actual size of the HSRR merge I50M-string table, in bytes, is given by the formula.

```
28 + ( (36 + length of DKNCRDI2) * (C + 10) )
```

Where C is the number of HSRR_MERGE_I50M_STRING_TABLE_CODELINES. The value of (36 + length of DKNCRDI2) should be rounded up to the nearest multiple of 4.

The default value for this keyword is 05000. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

IMAGE_MERGE_ONLY_WITH=(01M | I)

This keyword controls with which string EMRG attempts to perform an Option 2 Image-Merge. Specify either the interim M-string (01M) or the I-string (I), whichever you normally send to Key Entry. EMRG attempts to find the indicated string and, if it fails, displays error message EMRG010E.

This keyword is used only by Image-Merge. If you merge with any other EMRG option, this keyword has no effect.

If the keyword is not present in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile, EMRG resorts to its default Image-Merge behavior. In its default mode, EMRG searches first for an Interim M-string, then for an I-string; whichever string it finds first is the string with which EMRG performs an Image-Merge.

BALANCE_HSRR_MERGE_INITIAL_ALLOC_SIZE=#####

This keyword controls how much space, in codelines, is initially reserved for building codeline-linked lists by the Balanced HSRR 99-50 M-string merge

module. This module loads into this area an entire HSRR 50M-string, plus selected codelines from the corresponding Balanced 99M-string (the exact number depends on how many adjustments had been made). If these codelines won't fit, the module allocates additional space as directed by the BALANCE_HSRR_MERGE_SUBSEQUENT_ALLOC_SIZE keyword.

Replace "#####" with a five-digit number; use leading zeros, if necessary. The larger this number, the longer the lists can grow, but also the greater the amount of memory they consume.

Note: If EMRG is not asked to perform a Balanced HSRR 99-50 M-string merge, it does not allocate codeline-linked list space.

The actual size of the allocation, in bytes, is given by the formula:

```
8 + (8 + length of DKNCRDI2 + length of DKNCRZC2) * C
```

Where C is the number of codelines specified by BALANCE_HSRR_MERGE_INITIAL_ALLOC_SIZE.

The default value for this keyword is 05000. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

BALANCE_HSRR_MERGE_SUBSEQUENT_ALLOC_SIZE=#####

This keyword controls how much additional space, in codelines, is allocated by the Balanced HSRR 99-50 M-string merge module whenever it runs out of room for its codeline-linked lists. Should the additional space get used up as well, the module makes another subsequent allocation, then another, and another, until the module finally exceeds its BALANCE_HSRR_MERGE_MAXIMUM_ALLOCS setting.

Replace "#####" with a five-digit number; use leading zeroes, if necessary. The larger this number, the longer the codeline-linked lists can grow, but also the greater the amount of memory they consume.

Note: If EMRG is not asked to perform a Balanced HSRR 99-50 M-string merge, it does not allocate codeline-linked list space.

The actual size of the allocation, in bytes, is given by the formula:

```
8 + (8 + length of DKNCRDI2 + length of DKNCRZC2) * C
```

Where C is the number of codelines specified by BALANCE_HSRR_MERGE_SUBSEQUENT_ALLOC_SIZE.

The default value for this keyword is 01000. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

BALANCE_HSRR_MERGE_MAXIMUM_ALLOCS=##

This keyword controls how many times the Balanced HSRR 99-50 M-string merge module can allocate space for its codeline-linked lists. The count includes both the initial allocation, which takes place when the module starts running, and any subsequent allocations that occur whenever the module runs out of space. Should the merge module exceed this limit, it aborts the Balanced HSRR merge, displays an error message, and quits.

Replace "##" with a two-digit number; use leading zeroes, if necessary.

Note: If EMRG is not asked to perform a Balanced HSRR 99-50 M-string merge, it does not allocate codeline-linked list space and can never exceed this limit.

The default value for this keyword is 16. EMRG uses the default if it cannot find this keyword in the DKNPEMRG application profile, or if the DKNAPPL data set does not contain a DKNPEMRG application profile.

MSG_SUPPRESSION_COUNT=*xxxxx*

This keyword controls the number of EMRG "missing correction" messages that are issued before EMRG suppresses them. If the current count is set to 99999, EMRG suppresses all "missing correction" messages. If the count is set to 00000, EMRG suppresses no messages. At the end of processing, EMRG details how many "missing correction" messages were suppressed. If no suppression parameter is specified, EMRG defaults to displaying all "missing correction" messages.

DKNPETDM Profile Record Format

No restrictions exist on the number of ETDM profiles that an institution can create. Each ETDM profile should be used to describe a unique type of match processing. The ETDM profile name can be anything, as long as it conforms to standard MVS member naming conventions.

Control Record

Position	Description
1–1	**C** Control Record Identifier
2–2	Electronic Match Flag **Y** Perform electronic matching, using master strings from ETIN **N** Electronic matching is not performed
3–3	Reconciliation-string indicator **Y** Create reconciliation string **N** No reconciliation string is created. The reconciliation-file-indicator and the report indicator cannot both be N.
4–20	Reconciliation string name **EEEEPAABBCCDDTSSS** Name of the reconciliation string ***** The corresponding character of the first process string name that is to be used to build the reconciliation string name. **Note:** The reconciliation string name must be different from the process string name. Consequently, the reconciliation string name cannot be all '*' (asterisks).
21–21	Report indicator **Y** Create report

N No report.

The reconciliation-file-indicator and the report indicator cannot both be 0 or blank.

22–22 Electronic Resynch Document. Informs ETDM what control document should be used to resynch on during electronic matching. This control document also acts as a boundary, restricting matching to within the control document boundaries. It is required for electronic matching but is *not* required for non-electronic matching.

6 Resynch files when a divider document is read from the process string

4 Resynch files when a batch ticket is read from the process string

23–23 Control document match flag

Y Match control documents, only control document types 6, 5, 4, 3, and 2 are considered for matching. For electronic matching, the resynch document is always considered for matching.

N Do *not* match control documents.

24–24 Item-sequence-number (ISN) match indicator

Y Match on item-sequence number

N Do *not* match on item-sequence-number

25–25 Codeline data match indicator

Y Match on all fields, digit error threshold is zero; overrides the F record

N For field record match criteria, see “Field Record.”

Note: It is more efficient to set the codeline data match indicator to N and use the F record to specify a match on only those fields defined to the MDS.

26–26 Missing in place flag.

Y Missing items should be kept in place within the matching boundary.

N Missing items are placed at the end of the matching boundary.

27–65 Reserved

66–80 Description name for the matching profile. This is printed on the matching report.

Field Record

Position	**Description**
1–1	**F** Field Record Identifier
2–2	Digit substitutions allowed **Y** If the threshold count can include unmatched digits. **N** Do allow digit substitution.

3–5	Field 01 match indicator
	Fields are matched byte for byte. Two fields are a match if the number of match failures does not exceed the digit-error threshold limit. The digit error threshold is incremented when comparing misread or missing digits during a field match. However, if the digit substitution flat is on, good digits may also be considered in the threshold totals.
	BLANK Do not match on field 01.
	000–256 Digit-error threshold; match on this field.
	Valid digit-error threshold values for fields 1–7 are 000–028. Values for fields 9–15 are 000–256.
	Note: If you specify a digit-error threshold other than 000 for field 01 (amount), you might cause the difference between the current and previous totals and the difference between the missing and free totals to be unequal.
6–8	Field 02 match indicator
9–11	Field 03 match indicator
12–14	Field 04 match indicator
13–17	Field 05 match indicator
18–20	Field 06 match indicator
21–23	Field 07 match indicator
24–26	Field 09 match indicator
27–29	Field 10 match indicator
30–32	Field 11 match indicator
33–35	Field 12 match indicator
36–38	Field 13 match indicator
39–41	Field 14 match indicator
42–44	Field 15 match indicator
45–80	Reserved

User Record

Position	Description
1–1	**U**
	User Options Record Identifier
2–75	User area for user exits
76–80	Reserved

DKNETIN Profile Record Format

ETIN profiles are user-created, partitioned data-set members that are used to control selected internal functions of the ETIN task. No restrictions exist on the number of ETIN profiles that a financial institution can create. Each profile member must contain 12 records.

Record 01: ETIN uses this record to control specific functions and assign values in the creation of strings.

Position	Description
1–4	Constant PF01. Identifies the record.
5–5	Electronic string creation flag **Y** ETIN creates an electronic string. The string is created as extended and has an ETIN user area inserted. **N** ETIN does not create an extended string and does not insert an ETIN user area.
6–6	Extended string creation flag **Y** ETIN creates an extended string. **N** ETIN does not create an extended string. However, an extended string may still be created if the electronic string creation flag is set (see above).
7–38	Description of the type or source of the data being processed. This record has a maximum of 32 characters. This description is printed on the completion report.
39–42	Reserved
43	Item Charge Flag **Y** CPCS charges for transactions introduced to the mass data set using the ETIN task. . **N** CPCS does *not* charge for transactions introduced to the mass data set using the ETIN task. **Note:** It is the customer's responsibility to ensure that transactions introduced to the CPCS mass data set using the ETIN feature are correctly charged for in the CPCS Licensing Agreement. Guidelines for transaction charging are included in the "Accessing the Mass Data Set and Its Index" section of the *CPCS Programming and Diagnostic Guide*.
44–46	Bank number for the processing bank.
47	User Edit Flag. This flag should be set if you wish to perform codeline validation on the items being processed. **Y** ETIN passes items to the host codeline edit routine for editing. **N** Records are not passed to the host codeline edit routine.
48	ETIN completion report flag **Y** A report detailing the processing ETIN performed is produced. **N** No report is produced.

49 ETIN error report flag

Y A report detailing any reject items is produced.

N No report is produced.

50–51 Reserved

52 Update CPCS pass-to-pass control file

Y When importing the electronic data, the CPCS pass-to-pass data set is updated with the string's entry number.

N Do NOT update the CPCS pass-to-pass data set when loading this work.

53 Generate an ESM unit of work

Y A UOW is generated in ESM for the string being loaded.

N No UOW is generated with ESM.

54 Create electronic resynch records. This is needed if work being imported is destined to be used by the ETDM product.

N No electronic resynch record is created.

4 An electronic resynch record is generated for each batch ticket.

5 An electronic resynch record is generated for each block ticket.

6 An electronic resynch record is generated for each divider document.

8 An electronic resynch record is generated for each kill bundle record.

55 Generate kill bundle record. This should be used if, during the import processing, new pocket numbers are to be assigned to items. With this option, existing kill bundle records within the file are not written to the mass-data-set string.

Y Generate new kill bundle records.

N Do NOT generate new kill bundle records.

56–59 Reserved

60–62 Sort pattern number. This value is used by ETIN when creating the string, and it is used to retrieve the user-edit routine name if items are to be edited during import.

63–67 Reserved

68–68 Replace String Flag. This flag is used to tell ETIN if it can replace an existing string the FIRST time a transmission data set is loaded.

Y ETIN replaces an existing string.

N ETIN does not replace an existing string.

69–70 Sorter number. This value is used to allocate item sequence numbers during the import processing. Valid values are 97 to 99 only. Default is 97.

71 Item-sequence-number option flag; default=Y.

Y ETIN assigns new item-sequence numbers to items being imported.

N ETIN writes the items to the mass data set with the item sequence number as received.

72 String type. This value is used to set the string type for strings being created. This is only used if the task cannot determine a string name from any other data source. *

73–80 Reserved

Record 03: If the PF01 profile record specifies that records are to be validated by a user edit routine. Then the PF03 profile record can be used to instruct ETIN to bypass validation for specific control documents.

Position	Description
1–4	Constant PF03. Identifies the record.
5–53	Reserved

Positions 54 through 59 are bypass flags for incoming control documents, where:

Y Yes, bypass indicated control document.
N No, capture indicated control document.

Note: The default for flags 54 - 59 is N.

54	Bypass AST image
55	Bypass sub-batch slip
56	Bypass batch slip
57	Bypass block slip
58	Bypass divider
59	Bypass tracer
60–80	Reserved

Record 05: This record is used to assign a specific string name to the string being created.

Position	Description
1–4	Constant PF05. Identifies the record.
5–21	A specific string name assigned to the input as written to the MDS. This supersedes any string name that ETIN may have built and should only be used where a definite string name is always created by this profile.
22–80	Reserved

Record 07: Records 07 and 08 are used to control creation of a tracer record and to provide the data for fields 1 through 7 of the record, if created.

Position	Description
01–04	Constant PF07. Identifies the record.
05	Generate the tracer flag.

		Y ETIN generates a tracer document, including any data specified for fields 1 through 7.
		N A tracer document is not generated.
	06–20	Data for tracer-record field 1
	21–35	Data for tracer-record field 2
	36–50	Data for tracer-record field 3
		Note: Field 3 data can *not* be present if the update tracer data set option is active on the PF01 profile card *and* the generate tracer option is active on card PF07.
	51–53	Number of tracers to insert. Used only if the generate tracer flag is "Y". Valid values are 001 to 999.
	54–80	Reserved

Record 08

Position	Description
1–4	Constant PF08. Identifies the record.
5–19	Data for tracer-record field 4
20–34	Data for tracer-record field 5
35–49	Data for tracer-record field 6
50–64	Data for tracer-record field 7
65–80	Reserved

Record 09: Records 09 and 10 are used to control creation of a block record and to provide the data for fields 1 through 7 of the record, if created.

Position	Description
1–4	Constant PF09. Identifies the record.
5	Generate block flag.
	Y ETIN generates a block document, including any data specified for fields 1 through 7.
	N A block document is not generated (default).
6–20	Data for block-record field 1
21–35	Data for block-record field 2
36–50	Data for block-record field 3
51–80	Reserved

Record 10

Position	Description
1–4	Constant PF10. Identifies the record.
5–19	Data for block-record field 4
20–34	Data for block-record field 5
35–49	Data for block-record field 6

50–64	Data for block-record field 7
65–80	Reserved

Record 11: Records 11 and 12 are used to control creation of a batch record and to provide the data for fields 1 through 7 of the record, if created.

Position	Description
1–4	Constant PF11. Identifies the record.
5	Generate batch flag.
	Y ETIN generates a batch document, including any data specified for fields 1 through 7.
	N A batch document is not generated (default).
6–20	Data for batch-record field 1
21–35	Data for batch-record field 2
36–50	Data for batch-record field 3
51–80	Reserved

Record 12

Position	Description
1–4	Constant PF12. Identifies the record.
5–19	Data for batch-record field 4
20–34	Data for batch-record field 5
35–49	Data for batch-record field 6
50–64	Data for batch-record field 7
65–80	Reserved

Record 20: ETIN does not use this record, but you can use it to pass data to the user-exit programs.

Position	Description
1–4	Constant PF20. Identifies the record.
5–80	User data area

Record 32: Records 32 and 34 give the user the capability of setting flags as variables in the string directory entry of the string that is created. If the record position is blank, that field is not used. Except as noted, data from the profile record is placed in the string directory entry as received.

Positions 5–11, and 14–33 of record 32 can be turned by setting the flags to a value of one.

Position	Description
1–4	Constant PF32. Identifies the record.
5	ZB-BALANCED
6	ZB-DISTRIBUTED
7	ZB-MERGED
8	ZB-KILL-STATUS
9	ZB-KILL-LISTED

10	ZB-USER-RELSD
11	ZB-TRANSF-INPUT
12-13	Reserved
14	ZB-ADDL-FLAGS
15	ZB-MASS-FLAGS
16	ZB-IMAGE-ENTRY
17	ZB-KEY-ENTRY-DONE
18	ZB-POWER-ENCODE-DONE
19	ZB-IA-REPAIR-DONE
20	ZB-MERGE-FOR-BALANCE
21	ZB-IA-BALANCE-DONE
22	ZB-TYPE-INPUT-FLAG
23	ZB-SIG-VERIFY-FLAG
24	ZB-PRIORITY-CODE
25	ZB-STRING-TYPE-CODE
26	ZB-OLRR-FLAG
27	ZB-REJECT-DIST-FLAG
28	ZB-KILL-PROGRESS-STATUS
29	ZB-CIMS-HSRR-FLAG
30	ZB-SOURCE-LOOKUP
31	ZB-MICR-PROCESSED
32	ZB-REGION-DISPLAY
33	ZB-KILL-SUBSET
34	ZB-CONAINS-ECHECKS
35–80	Reserved

Record 34

Position	Description
1–4	Constant PF34. Identifies the record.
5	Order of Work **0** Debits precede credits **1** Credits precede debits **2** Debits only **3** Credits only **4** Mixed non-POD
6–13	ZB-DIST-STATUS – bit representation. Each byte represents 1 bit; 0 and 1 are the only valid characters.
14–16	ZB-BANK-NUMBER
17–24	ZB-ENDPOINT-ID
25–27	Reserved
28–30	ZB-NEXT-SUBSTRING
31	ZB-KILL-OPTION
32	ZB-DIST
33–36	ZB-COMPL-KB-NUM
37	ZB-DISTR
38–80	Reserved

A sample profile can be found on CPCS.V1R11.SAMPLIB.

DKNETIO Profile Record Format

The ETIO profile can contains a single profile card; you use this card to describe a unique CPCS system name that is used in creating transmission data sets. The profile name you use must be DKNPETIO.

Note: This profile member is only required when a financial institution is exporting and importing transmission data sets on more than one in-house CPCS system. You use it to inform ETIO which CPCS system created the electronic transmission data set.

Record Description:

Position	Description
01–10	Constant ‘CPCS–NAME=’ Identifies the record.
11–18	Unique CPCS system name (user defined).

A sample profile can be found on CPCS.V1R11.SAMPLIB.

DKNPETOT Profile Record Format

DKNPETOT Profile Record Format

ETOT profiles are user-created, partitioned data-set members that are used to control selected internal functions of the ETOT task. Each profile member must contain 1 record and the profile name used must be DKNPETOT.

Record Description:

Position	Description
01–01	Constant 'C'. Identifies the control record.
02–02	Create unique file flag.
	Y ETOT appends a time stamp to the electronic transmission data set name being created. This ensures unique files are always created.
	N Electronic file name is derived exclusively from the DSAT.
03–03	Separate electronic images by sort pattern number.
	Y ETOT separates strings exported to the transmission data set by sort pattern. As work is written to the transmission data set, a sort type delimiter record is written to when the sort type changes. This enables the sort pattern number to be maintained when importing work for multiple sort patterns into CPCS.
	N All work within a single transmission data set is allocated to the same sort pattern number when imported into CPCS.
04–04	Create ETOT report.
	Y A report detailing the transmission data set created is produced.
	N No report is produced.
05–05	Export SDE details. This flag is used to indicate whether the string directory information should be included on the transmission data set as part of the export process.
	Y Export SDE details.
	N Do not export SDE details.
06–06	Export GUA data. This flag is used to indicate whether an item's global user area should be included on the transmission data set when exporting extended strings.
	Y Export GUA details.
	N Do not export GUA details.
07–80	Reserved

A sample profile can be found on CPCS.V1R11.SAMPLIB.

DKNPKILL

The DKNPKILL profile controls the behavior of DKNKILL. The keywords are:

ALLOW_KILL_AND_MCRE_TO_RUN_CONCURRENTLY={YES | NO}

If one or more KILLs run at the same time as MCRE for a given cycle, it is possible for MCRE to fail to extract some of the KILLed bundles. To prevent this from happening, set this keyword to a value of NO. Each started KILL checks to see if an MCRE is running for the same cycle; if it is, KILL waits for the MCRE to finish.

This keyword has no effect if the KILL and the MCRE are being run for different cycles. It does not prevent multiple simultaneous KILLs from running on the same cycle.

Many sites normally schedule their MCREs at a time when no KILLs take place. If your site is one of them, you can set this keyword to YES for a slight improvement in performance.

This keyword should be set to the same value as the "ALLOW_KILL_AND_MCRE_TO_RUN_CONCURRENTLY' keyword in the DKNPMCRE profile. If this keyword is not present in DKNPKILL, it defaults to a value of YES.

DKNPMCRE

The DKNPMCRE profile controls the behavior of DKNMCRE. The keywords are:

ALLOW_KILL_AND_MCRE_TO_RUN_CONCURRENTLY={YES | NO}

If one or more KILLs run at the same time as MCRE for a given cycle, it is possible for MCRE to fail to extract some of the KILLed bundles. To prevent this from happening, set this keyword to a value of NO. When MCRE starts, it checks if any KILLs are being run for the same cycle. Such a KILL causes MCRE to consult with the terminal operator for further instructions. The operator can choose to either cancel MCRE, or make it wait until all the KILLs are finished.

This keyword has no effect if the KILLs and the MCRE are being run for different cycles.

Many sites normally schedule their MCRE runs at a time when no KILLs take place. If your site is one of them, you can set this keyword to YES for a slight improvement in performance.

This keyword should be set to the same value as the "ALLOW_KILL_AND_MCRE_TO_RUN_CONCURRENTLY" keyword in the DKNPKILL profile. If this keyword is not present in DKNPMCRE, it defaults to a value of YES.

DKNPMRGE

The DKNPMRGE profile controls the behavior of DKNMRGE. The keywords are:

USE_SCATTED_SYSTEM_REJECT_R_STRING={YES | NO}

For those merges involving the System Reject pocket, this keyword controls what sort of R-string MRGE looks for. It has no effect on Pocket, Consolidated, Final, or HSRR merges.

A value of YES means that MRGE uses a concatenated System Reject R-string. Before running MRGE, you must run SCAT to combine the various Partial System Reject R-strings into a single concatenated System Reject R-string.

Note that OLRR can be configured to auto-start SCAT for you. See the entry for DKNPOLRR below for more information.

A value of NO means that MRGE uses Partial Reject R-strings and attempts to perform its own concatenation. You can manually start MRGE with the ,M option to gain some control over this process (see DKNMRGE in the *CPCS Programming and Diagnostic Guide* for more information).

Note: The concatenation algorithms used by MRGE are not as sophisticated as those employed by SCAT.

If this keyword is not present in DKNPMRGE, it defaults to a value of NO.

USE_SCATTED_HSRR_SYSTEM_REJECT_R_STRING={YES | NO}

For those HSRR merges involving the system reject pocket, this keyword controls what sort of R-string MRGE looks for.

A value of YES means that HSRR MRGE uses a concatenated reject R-string. Before running MRGE, you must run SCAT to combine the various partial system reject R-strings into a single concatenated system reject R-string. Note that OLRR can be configured to autostart SCAT for HSRR. For more information, see "DKNPOLRR" on page 3-42.

A value of NO means that MRGE uses partial reject R-strings and attempts to perform its own concatenation. You can manually start MRGE with the ,M option to gain some control over this process. For more information, see DKNMRGE in the *CPCS Programming and Diagnostic Guide*.

Note: The concatenation algorithms used by MRGE are not as sophisticated as those employed by SCAT.

If this keyword is not present in DKNPMRGE, it defaults to a value of NO.

TRACEIT={YES | NO}

TRACEIT is a programmer's debug tool and should normally be activated only by CPCS programmers. A value of YES causes MRGE to log to SYSOUT each routine it executes and each key data area that it modifies. A value of NO suppresses this logging.

If this keyword is not present in DKNPMRGE, it defaults to a value of NO.

AUTOSTART_MNOP={YES | NO}

This keyword is useful in installations that use System Manager to schedule and launch tasks. It sometimes can take several executions of MRGE before an M-string is completely built and ready to pass on to other tasks. This keyword helps System Manager determine when it is proper to start those other tasks.

A value of YES causes MRGE to auto-start the dummy DKNMNOP task when it has an M-string ready. System Manager workflows can be tied to the running of DKNMNOP. A value of NO on this keyword suppresses the starting of DKNMNOP.

If this keyword is not present in MREGPROF, it defaults to a value of NO.

RESET_KEY_ENTRY_DONE=*xxx*

Where *xxx* is YES and MRGE turns off the Key Entry done flag so the output 01-M string may be keyed, or NO and MRGE turns on the Key Entry done flag so the output 01-M string may not be keyed.

DKNPOLRR

The DKNPOLRR profile controls the behavior of DKNOLRR. The keywords are:

AUTOSTART_SCAT_WHEN_OLRRING_SYSTEM_REJECT_D-STRING={YES | NO}

This keyword controls how OLRR handles any Partial System Reject R-string it is asked to create. It has no effect on Alternate, Consolidated, or Unencoded Partial R-strings.

A value of YES will cause OLRR to auto-start SCAT and pass it the name of the Partial System Reject R-string. SCAT will accumulate each such Partial it receives until it has enough on hand to concatenate them. MRGE can be configured to read these concatenated R-strings (see the DKNPMRGE profile above for more information).

A value of NO means that OLRR does not auto-start SCAT. You will either have to run SCAT yourself, or configure MRGE to directly process Partial System Reject R-strings. See the DKNPMRGE profile above for more information.

Note: The concatenation algorithms used by MRGE are not as sophisticated as those employed by SCAT.

This keyword is ignored if your installation has installed Enhanced System Manager. In an ESM environment, the only way to auto-start SCAT is to incorporate it into your workflows.

If this keyword is not present in DKNPOLRR, it defaults to a value of NO.

AUTOSTART_SCAT_WHEN_OLRRING_SYSTEM_REJECT_D-STRING_FOR_HSRR={YES | NO}

This keyword controls how OLRR handles any partial system reject R-string it is asked to create for HSRR rejects that are repaired through OLRR.

A value of YES causes OLRR to autostart SCAT and to pass it the name of the partial system reject R-string. SCAT accumulates each such partial it receives until it has enough on hand to concatenate them. MRGE can be configured to read these concatenated R-strings (see “DKNPMRGE” on page 3-40 for more information).

A value of NO means that OLRR does not autostart SCAT. You must either run SCAT yourself or configure MRGE to directly process partial system reject R-strings. (See “DKNPMRGE” on page 3-40 for more information.)

Note: The concatenation algorithms used by MRGE are not as sophisticated as those employed by SCAT.

If this keyword is not present in DKNPOLRR, it defaults to a value of NO.

REQUIRE_PLUS_TO_VALIDATE_INPUT={YES | NO}

If this keyword is set to YES, the operator:

- Enters all corrections on an OLRR screen
- Tabs to the PIGGYBACKS field at the bottom of the screen
- Enters a single plus character (+) to cause the corrections to be accepted.

If the operator does not enter a plus, for example if he or she simply presses ENTER instead, then all corrections are “forgotten” and the screen resets to what had originally been displayed. The operator must then re-enter all the corrections again.

If this parameter is set to NO, it is not necessary for the operator to validate each OLRR screen with a plus. Simply pressing ENTER causes the corrections to be accepted. However, the operator also no longer has a way to make OLRR “forget” corrections and reset its display.

The default value for this keyword is YES.

GENERATE_STATISTICS_ON_OLRR_BYPASS={YES | NO}

If this keyword is set to YES, and OLRR is run in BYPASS mode, an OLRR statistics record is added to the end of the R-string it creates, the same way a normal, non-BYPASS OLRR would.

If this keyword is set to NO, and OLRR is run in BYPASS mode, an OLRR statistics record is not added.

The default value for this keyword is NO.

RTNOTVER_OFF_WHEN_NO_CHECK_DIGIT_ON_PB={NO | YES}

Where: --

NO Means always turn on the RTNOTVER bit.
YES Means do *not* turn on the RTNOTVER bit.

Note: This card only applies to piggybacks.

This card instructs OLRR *not* to turn the Routing Transit Not Verified (RTNOTVER) bit on if the codeline is a piggyback. The default (NO) is to always turn on the RTNOTVER bit before editing the codeline. By changing the setting to YES, the RTNOTVER bit is *not* turned on unless instructed to do so by the Check Digit Routines. The RTNOTVER bit is in DI3BYTE3.

RT_MODCHK_BYPASS=[YES | NO]

This keyword controls how OLRR handles any 9-digit routing and transit. The YES option directs to bypass mod checking, and the NO option directs to not bypass mod checking.

DKNPRLST

The DKNPRLST profile controls the behavior of DKNRLST. The keywords are:

DISPOSITION_OF_SCATTED_SYSTEM_REJECT_R-STRING={KEEP | DELETE}

This keyword controls what RLST does with any concatenated System Reject R-string it is asked to list. It has no effect on RLST’s handling of Partial System Reject R-strings.

A value of DELETE deletes the R-string when RLST is finished; a value of KEEP keeps it. It is not necessarily an error to delete concatenated System Reject R-strings, as the data contained in them are still available in the various Partial System Reject R-strings from which the concatenated was built.

If you choose to KEEP concatenated System Reject R-strings, be aware that they will be automatically deleted anyway by MCRE.

This keyword is ignored if you have installed Enhanced System Manager. In an ESM environment, the only way to delete concatenated System Reject R-strings is to incorporate an STGD step into your workflows.

If this keyword is not present in DKNPRLST, it defaults to a value of KEEP.

```
FIELD from DIDSCT - RELATIVE Location
NAME               BYTE:BIT  in 50-byte MDS
--------           ----:---
DI3BYTE3             45:0
RTCKDIGT             45:0    1010 <-- No check digit
RTNOTVER             45:4    0    <-- RT not verified
SCIPAON              45:5    0
DI33_RES1            45:6    0
DI33_RES2            45:7    0
```

Figure 3-2. Record Type 3

DKNPSCPY

DKNPSCPY provides a utility for copying strings under the control of an application profile. The profile selects properties of the copying process (for example, whether this is a string-to-string copy or a copy-in-place, whether the string is modified in any way during the copy, and so on). There exists a different SCPY profile for each different way a string can be copied.

All SCPY profiles share the same common characteristics:

- Their names all begin with SCPY (for example, DKNPSCPS).
- They all contain the COPY_FROM, COPY_TO, COPY_WORK, and COPY_PROG keywords, which furthermore must appear in that order (that is, COPY_FROM before COPY_TO, COPY_TO before COPY_WORK, etcetera).
- They may contain other keywords unique to that particular SCPY profile.
- They may contain the optional DEBUG keyword.

SCPY Profile Common Keywords

The following is a complete list of all the keywords common to all SCPY profiles:

COPY_FROM=eeee-p-aa-bb-cc-dd-t-sss

Names the string that SCPY is to copy from. Each component (entry number, pass, pocket 1, pocket 2, etcetera), besides containing a legal value for that component, may also contain asterisks (*). In such a case SCPY will plug the component from the Unit of Work (if started by ESM), or from the operator's COPY FROM input, if started manually.

Note: If COPY_FROM is other than ****-*-**-**-**-**-*-***, then the string that is copied won't necessarily be the string passed in the UOW, or entered by the operator.

COPY_FROM is a required keyword. It is an error not to have one present in an SCPY profile. It must appear before the profile's COPY_TO, COPY_WORK, and COPY_PROG keywords.

COPY_TO=eeee-p-aa-bb-cc-dd-t-sss

Names the string that SCPY is to copy to. Each component (entry number, pass, pocket 1, pocket 2, etcetera), besides containing a legal value for that component, may also contain asterisks (*). In such a case SCPY will plug the component from the COPY_FROM string. If SCPY is manually-started, and the operator enters a COPY TO string, then SCPY will plug from the operator's input instead.

A COPY_TO of ****-*-**-**-**-**-*-***, if not overridden by the operator, causes SCPY to perform a copy-in-place.

COPY_TO is a required keyword. It is an error not to have one present in an SCPY profile. It must appear after the profile's COPY_FROM keyword, but before the COPY_WORK and the COPY_PROG.

COPY_WORK=eeee-p-aa-bb-cc-dd-t-sss

Names the work string that SCPY creates as a scratchpad. Each component (entry number, pass, pocket 1, pocket 2, etcetera), besides containing a legal value for that component, may also contain asterisks (*). In such a case SCPY will plug the component from the COPY_FROM string.

COPY_WORK is a required keyword. It is an error not to have one present in an SCPY profile. It must appear after the profile's COPY_FROM and COPY_TO keywords, but before COPY_PROG.

COPY_PROG=xxxxxxxx

Names the program that SCPY is to pass control to after it has validated the common keywords. This sub-module validates all remaining keywords in the profile and may even perform additional validation on COPY_FROM, COPY_TO, and COPY_WORK. Finally, it performs the actual copying of strings.

COPY_PROG is a required keyword. It is an error not to have one present in an SCPY profile. It must appear after the profile's COPY_FROM, COPY_TO, and COPY_WORK keywords.

DEBUG

DEBUG is a programmer's debug tool and should normally be activated only by CPCS programmers. If present, it causes each profile record read from that point forward to be listed on the SCRL log (if SCPY was manually started, the profile records will be listed on the terminal as well). Comment records will not be listed.

If this keyword is not present, SCPY's default action is to list no records.

SCPY Profiles List

DKNPSCPS

The DKNPSCPS profile controls the behavior of the SCPY sub-module DKNSCPSB. DKNSCPSB's first action is to ensure that:

- COPY_FROM names a subpass I-string.
- The COPY_TO string is identical to the COPY_FROM string (that is, COPY_TO specifies copy-in-place).
- The COPY_WORK string is identical to the COPY_FROM string, except for pocket 4, which must be different.

DKNSCPSB also verifies that DKNPSCPS contains at least one PAPER_DOC keyword, or at least one CONTROL_DOC keyword.

The keywords specific to the DKNPSCPS profile are:

PAPER_DOC={nn | nn-nn},{nn | nn-nn},...

Names DITYPEI document type codes and/or ranges of DITYPEI document type codes for non-control documents. Every time DKNSCPSB encounters a non-control document of this type, it checks to see if its sequence number is 9999999999 (that is, whether or not sorter codeline data matching had

decided that this is a free, unmatchable item). If it is, DKNSCPSB replaces the sequence number with a fresh one obtained from DKNIGENS.

The type codes may be specified as single hexadecimal digits, or as ranges of hexadecimal digits separated by dashes. The two forms may be freely intermixed. For example:

```
PAPER_DOC=00,05,1A-1F,70,80-FF
```

You can specify as many type codes as can fit onto an eighty-byte record, and you can specify as many PAPER_DOC keywords as are necessary to list all your codes.

CONTROL_DOC={nn | nn-nn},{nn | nn-nn},...

Names DITYPEI document type codes and/or ranges of DITYPEI document type codes for control documents. Every time DKNSCPSB encounters a control document of this type, it checks to see if its sequence number is 9999999999 (that is, whether or not sorter codeline data matching had decided that this is a free, unmatchable item). If it is, DKNSCPSB replaces the sequence number with a fresh one obtained from DKNIGENS.

The type codes may be specified as single hexadecimal digits, or as ranges of hexadecimal digits separated by dashes. The two forms may be freely intermixed. For example:

```
CONTROL_DOC=C0-C4,C7,D8,F2-F5,FD,FE
```

You can specify as many type codes as can fit onto an eighty-byte record, and you can specify as many CONTROL_DOC keywords as are necessary to list all your codes.

DKNMRGE Codeline Data Matching

This section outlines the preparations required to build the codeline data matching tables used to merge HSRR strings.

Overview

DKNMRGE calls DKNMRG2 to build a match file that contains the prime-pass sequence number and the corresponding HSRR sequence number. The match file is built by comparing the MICR fields in the prime-pass reject D-string to the MICR fields in the HSRR I-string. The resulting file is then merged with the HSRR I-string and the HSRR R-string to create the HSRR file. The HSRR file is sorted into prime-pass sequence number order and merged with the prime-pass I-string to produce the output M-string. For each sort pattern in DKNSPDEF that is used for capturing HSRR strings, build a matching table member, or you may elect to build one for each bank. However, in most cases the default member MRG2S256 provides sufficient results.

The following naming convention is used by DKNMRGEP to locate the HSRR merge profile member in the file specified by the ddname DKNAPPL in the CPCS run JCL.

MRG2*xyyy* The convention:

MRG2
Constant tells DKNMRGEP to use this member

x Where:

S Sort pattern

B Bank

yyy Sort pattern number or bank number

For example, MRG2S001 has the matching table parameters DKNMRG2 uses when matching the HSRR I-string to the prime reject D-string for work captured under sort pattern 001.

The search order is:

1. Look for a member corresponding to the sort pattern using the prime I-string SDE information.
2. If a sort pattern member is not found, look for a member corresponding to the bank number.
3. If neither a corresponding sort pattern or bank member is found, default to the values in MRG2S256.
4. If neither a corresponding sort pattern or bank member and MRG2S256 cannot be found, use the values contained in the COBOL copybook DKNMRGEZ.

For example, you capture a prime I-string using sort pattern 070. You use a device to fix rejects that returns a HSRR I-string to CPCS. Because this device doesn't require a sort pattern, the string directory entry (SDE) is zero filled. In this case, the member name should be MRG2S070.

In the same example but you want to use the sort pattern from the HSRR I-string, you must modify DKNMRGEZ and change the default value

MRGEZ-PRIM-VS-HSRR-ZB to “HSRR”. You are using a device to fix rejects that returns a HSRR I-string to CPCS. This device does require a sort pattern. Because the HSRR string has a sort pattern in the string directory entry (SDE), in other words, sort pattern 240, in this case you would need the member MRG2S240.

In another example, you only occasionally use HSRR. You probably do not need any members because DKNMRG2 uses the default values in DKNMRGEZ.

Use of Parameter Cards by DKNMRG2

The following documentation explains each of the parameters that can be set in the HSRR merge profile member. The sample member DKN2S256 should be used as a skeleton to determine the format of the profile member.

RANGE SWITCH

Where:

ON Indicates DKNMRG2 should not search outside the batch or block boundaries that define the search window.

Note: The batch and/or block switch must be ON.

OFF Indicates DKNMRG2 should search outside of the batch and block boundaries.

Note: When set to OFF, any item from HSRR can be matched to a prime item regardless of its location within control doc groups. Items that are not unique (such as zero amount batch slips) may match to the wrong item, causing other unmatched items to be placed in an incorrect location on the M-string.

BATCH SWITCH

Where:

ON Indicates DKNMRG2 should not search outside the batch boundaries.

OFF Indicates DKNMRG2 should search outside the batch boundaries.

BLOCK SWITCH

Where:

ON Indicate DKNMRG2 should not search outside the block boundaries.

OFF Indicates DKNMRG2 should search outside the block boundaries.

UNMATCHED SWITCH

Where:

Precedes (P)
DKNMRG2 places all unmatched rejects before the last matched control document.

Follows(F) DKNMRG2 places all unmatched rejects following the last matched control document that began the search.

Sequence(M)
DKNMRG2 places all unmatched rejects after the last matched item (default).

Block(B) DKNMRG2 places all unmatched rejects after the last matched block.

Batch(C) DKNMRG2 places all unmatched rejects after the last matched batch within the current block. If there is no matched batch header yet within the current block, DKNMRG2 places unmatched rejects after the last matched block header.

BAD BAD SWITCH
Where:

ON DKNMRG2 masks out digit errors in both the prime and HSRR codelines when both contain errors.

OFF DKNMRG2 does not mask out digit errors.

OVERRIDE SWITCH
Where:

ON If an error occurs while editing, the PDS member overrides the error and allows DKNMRGE to continue with the default settings in DKNMRGEZ.

OFF If an error occurs while editing, the PDS member stops DKNMRGE; no M-string is created.

Control Document Switch
Where:

K Keep unmatched control documents from the HSRR string.

D Discard unmatched control documents from the HSRR string.

Unmatched Reject Report Switch
Where:

1 Print unmatched reject report number DKNMRGE-001

3 Print unmatched reject report number DKNMRGE-003

A Print ALL unmatched reject reports

N Do *not* print any unmatched reject reports

Use of Matching Values by DKNMRG2

Five tables are used by MRG2 during its matching process. Two tables are used to determine if a codeline can be considered eligible for matching (TFLD and DFLD tables), three tables (GFLD, UFLD and BFLD tables) are used to determine if two codelines can be considered matched.

Each table contains four unique rows. There are two rows for documents (one for onus documents and one for transit documents), one row for batch tickets, and a row for block tickets. Each of these rows describes a unique set of values that are used during the matching process for each of the document types.

The following list describes each of the tables used by DKNMRG2 when assigning values during the matching process:

TFLD - Test Field

Each codeline is tested before going through the matching process to determine if there are enough significant digits to use in the matching process. These values are the values assigned to a field when the field has enough significant digits to be used in the matching process. The codeline must have enough points from this initial test to proceed to the next level, which is to actually compare the prime-pass fields to the HSRR pass fields. Both prime-pass and HSRR pass codelines are tested.

GFLD - Good Field

The value assigned to a field when there is a match found on both the prime-pass reject D-string and the HSRR I-string.

UFLD - Unmatched Field

The value deducted from a field when there is a non-match found on both the prime-pass reject D-string and the HSRR I-string.

BFLD - Blank Field

The value deducted when the prime-pass field has significant digits and the HSRR pass has blank fields or vice-versa.

DFLD - Digit Error threshold

The number of digit errors a field is allowed to have before it is considered not to have enough significant digits. When this threshold is exceeded, the field is considered to be blank for matching purposes. When the total digit errors is exceeded, the entire codeline is considered unmatchable. It is possible to set the digit error value to handle fields that are either missing or may not require validation during capture. Setting a digit error field to 998 means that digit errors are *not* accumulated for that field. A value of 999 means that digit errors are *not* accumulated for the field and the field is considered to be blank for matching purposes.

Note: The sample profile MRG2S256 includes a default definition for each of these tables.

How Does the Matching Process Work?

Assume, for the purposes of this exercise, that we have the following scenario. This example assumes that all items are onus items.

```
----------PRIME-------------------------------------------
SEQ.NO.     SER.NO.   RT. TR.  ACCOUNT  NO.   PROC.  AMOUNT
0100473402          0433***9       ***0894  031700      140
0100473403  003282  04330009  410010280894  031700 12104114*
----------HSRR--------------------------------------------
9500473402          04330009                031700      140
9500473403          04330009                      121041146
```

Use the following table to control the matching process.

```
      AMT P/C ACC OPT R/T OPT SER TOT
RECID F1  F2  F3  F4  F5  F6  F7  TOT
----- --- --- --- --- --- --- --- ---
00201 002 001 002 000 003 000 002 004 * ONUS Test
00301 002 001 003 000 003 000 002 004 * ONUS Good
00401 002 000 000 000 002 000 000     * ONUS Unmatched
00501 000 000 000 000 000 000 000     * ONUS Blank
00601 002 001 002 000 002 000 002 006 * ONUS Digit Errors
```

First, look at the first prime item.

```
SEQ.NO.    SER.NO.   RT. TR.  ACCOUNT  NO.   PROC.   AMOUNT
0100473402          0433***9       ***0894  031700      140
```

Step 001 - Test for enough significant digits to use for matching seq. no. 000100473402.

Field 1 has no digit errors: `recid{00201(f1)}` is 002. We add two points to our test weight total.

Field 2 has no digit errors: `recid{00201(f2)}` is 001. We add one point to our test weight total.

Field 3 has three digit errors: `recid{00601(f3)` is 002; since 3 > 2, the field is blanked out for testing purposes. We add no points to our test weight total; we add three points to our digit error total.

Field 5 has three digit errors: `recid{00601(f5)` is 002; since 3 > 2, the field is blanked out for testing purposes. We add no points to our test weight total; we add three points to our digit error total.

Field 7 is blank. No points are added to our test weights.

Results:

```
Test Weight points = 003
Digit Error points = 006
```

Because three is less than four (the total test weight points needed for matching an onus item), this codeline is unmatchable and is missing its correction seq 029500473402, which is a free item. Also, had the test total points been greater than or equal to four, an additional test would have been made to ensure there were no more than six total digit errors in the entire codeline. As there were six digit errors, this codeline would still have been considered unmatchable.

Now, let's consider the next prime item:

```
SEQ.NO.    SER.NO.   RT. TR.  ACCOUNT  NO.   PROC.   AMOUNT
0100473403  003282  04330009  410010280894  031700 12104114*
```

STEP 001 - Test for enough significant digits to use for matching seq no. 000100473403.

Test weight points equal:

```
  {00201(f2) + 00201(f3) + 00201(f5) + 00201(f7)}
=    001     +   002     +   003     +   002
=    008
```

Because eight is greater than four, this codeline is taken to the next step.

STEP 002 - Are there too many digit errors?

```
Digit Error points = digit error count for field one,
                   = 001
```

Because one is less than six, we are OK to proceed.

STEP 003 - Actually do the comparison. We will attempt to match this item with the first unmatched HSRR item, seq no. 0029500473402. Good points equal:

```
 -00401(f1) + 00301(f2) - 00501(f3) + 00301(f5) - 00501(f7)
=  -002     +   001     -   000     +   003     -   000
=   002
```

Because two is less than four (the total points for good fields), these items are considered unmatched.

STEP 003 - Repeat the above process for the next unmatched HSRR item, seq no. 029500473403. Good points equal:

```
00301(f1) - 00501(f2) - 00501(f3) + 00301(f5) - 00501(f7)
= 002     -   000     -   000     +   003     -   000
= 005
```

Because five is greater than four (the total points for good fields), these items are considered matched.

The following pseudocode gives an overview of how the matching process described above actually takes place:

```
tfld-accum = 0
derr-accum = 0

do TEST each FIELD in the CODELINE for SIGNIFCANT DIGITS

do COUNT the number of DIGIT ERRORS

if  tfld-accum >= TOTAL for TFLD
and derr-accum <= TOTAL for DERR
then
    accum = 0
    evaluate true
      when A PRIME FIELD MATCHES A HSRR FIELD
           accum = accum + GFLD
      when A PRIME FIELD IS BLANK AND A HSRR FIELD HAS
             SIGNIFICANT DIGITS
           accum = accum - BFLD
      when A HSRR FIELD IS BLANK AND A PRIME FIELD HAS
             SIGNIFICANT DIGITS
           accum = accum - BFLD
      when BOTH HSRR AND PRIME FIELDS HAVE DIGITS AND
             THEY DON'T MATCH
           accum = accum - UFLD
    end-evaluate
    if  accum >= TOTAL for GFLD
    then
        THIS IS CONSIDERED A MATCH
    else
        THIS IS NOT A MATCH
    end-if
else
    THE CODELINE CANNOT BE MATCHED
end-if
```

Chapter 4. User Programming Requirements and Exits

Overview

This chapter provides general use programming interface and associated guidance information.

This chapter describes the programming you must complete to run CPCS successfully.

User Data Preparation for Sort Programs

This section outlines the preparations required for the sort programs:

- Bank-Control-File Record Formats
- Endpoint Name-and-Address Record Formats
- Sort-Pattern-Definition Record Formats

Bank-Control-File Record Formats

Each bank control file record contains the following information:

- Processing bank number (an internal number between 001 and 999 that the CPCS installation assigns).
- Name and address information that the report tasks use.
- Special kill-list instructions printed on each kill list.
- Customized CPCS option data. This data, if present, overrides the data that the CPCSOPTN macro generates for the CPCS installation. Overriding occurs only for entries that are processed for the financial institution specified by these records.

  The DKNMICR task reads the record for the financial institution before starting the sort. It checks to see whether any of this data is present; if so, it replaces data existing in memory with this data before downloading the document processor instructions.
- Data required for producing electronic cash letters

When you start a new entry, DKNMICR reloads the CPCS option parameters into memory before it reads the financial institution record, ensuring that the default values are always present when needed.

Detailed Bank Control File Input for DKNLOAD

The input data set to the DKNLOAD program (CARD) is device-independent, and it must consist of 80-character logical records. Each processing financial institution requires 10 input records, each of which contains a record code 0 and sequence numbers 001 through 010. An optional 11th input card, with sequence number 011, may also appear. This card is used to specify Electronic Cash Letter data.

The following fields are common to each record.

Position	Description
1	Record code – 0
2	Constant – 0
3–5	Bank number – 001 through 999
6	Constant – 0

The following records describe the data requirements for the card records.

First Record:

Position	Description
7–9	Sequence number – 001
10	Blank
11–36	First line of the processing financial institution's name and address; 26 positions
37–40	Blank
41–66	Second line of the processing financial institution's name and address; 26 positions
67–80	Blank

Second Record:

Position	Description
7–9	Sequence number – 002
10	Blank
11–36	Third line of the processing financial institution's name and address; 26 positions
37–40	Blank
41–66	Fourth line of the processing financial institution's name and address; 26 positions
67–80	Blank

Third Record:

Position	Description
7–9	Sequence number – 003
10	Blank
11–54	First line of the processing financial institution's outgoing kill list data; 44 positions
55–80	Blank

Fourth Record:

Position	Description
7–9	Sequence number – 004
10	Blank
11–54	Second line of the processing financial institution's outgoing kill list data; 44 positions
55–80	Blank

Fifth Record:

Position	Description
7–9	Sequence number – 005
10	Blank
11–54	Third line of the processing financial institution's outgoing kill list data; 44 positions
55–80	Blank

Sixth Record:

Position	Description
7–9	Sequence number – 006
10	Blank
11–54	Fourth line of the processing financial institution's outgoing kill list data; 44 positions
55–80	Blank

Seventh Record:

Position	Description
7–9	Sequence number – 007
10	Blank
11–54	Fifth line of the processing financial institution's outgoing kill list data; 44 positions
55	Blank
56–63	DKNMDIS user-exit program name. The MDIS user exit specifies the name of a user-written routine that is unique for your financial institution. It is left-justified and padded with blanks.
64–80	Blank

Eighth Record:

Notes for Record 8:

1. If dashes occur on the physical document, they should appear in the document identifier.
2. The CPCSOPTN description defines the bank control file for bank 001 **only**. Record 8 must contain only the record type, the bank number, and the record number; columns 11-46 should be blank.
3. Character strings must be left-justified in their document fields and cannot be larger than 16 digits or the maximum length of any field specified, whichever is smaller.

Position	Description
7–9	Sequence number — 008
10	Blank
11–26	The tracer document identifier. It is left-justified and padded with blanks.

27	The primary field for the tracer ID. The number corresponds to the MICR field. 1 = Amount field 2 = Process control field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2 7 = Serial number field
28	The secondary field for the tracer ID. The number corresponds to the MICR field. The field can be blank, but must not be the same as the primary field. 1 = Amount field 2 = Process control field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2 7 = Serial number field
29–44	The divider document identifier. It is left-justified and padded with blanks. See Notes for Record 8, which also apply to this identifier.
45	The primary field for the divider ID. The number corresponds to the MICR field. 1 = Amount field 2 = Process control field 3 = Account number field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2
46	The secondary field for the divider ID. The number corresponds to the MICR field. It can be blank, but it must not be the same as the primary field. 1 = Amount field 2 = Process control field 3 = Account number field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2
47–54	The name for the optional OLRR user exit. This is a user-written exit routine used to resolve non-numeric keys. The first character must be alphabetic; all others can be alphanumeric. If you do not use the exit, you must leave these fields blank.
55–62	The name of the optional Online Adjustments user exit. This is a user-written exit routine used to resolve non-numeric keys. The first character must be alphabetic; all others can be alphanumeric. If you do not use the exit, you must leave these fields blank.
63–80	Blank

Ninth Record:

Notes for Record 9:

1. If dashes occur on the physical document, they should appear in the document identifier.
2. The CPCSOPTN description defines the bank control file for bank 001 **only**. Record 9 must contain only the record type, the bank number, and the record number; columns 11-50 should be blank.
3. Character strings must be left-justified in their document fields and cannot be larger than 16 digits or the maximum length of any field specified, whichever is smaller.

Position	Description
7–9	Sequence number – 009
10	Blank
11–26	The block slip document identifier. It is left-justified and padded with blanks. See Notes for Record 8 on page 4-7, which also apply to this identifier.
27	The primary field for the block ID. The number corresponds to the MICR field. 2 = Process control field 3 = Account number field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2 7 = Serial number field
28	The secondary field for the block slip ID. The number corresponds to the MICR field. The field can be blank, but it must not be the same as the primary field. 2 = Process control field 3 = Account number field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2 7 = Serial number field
29–44	The batch slip document identifier. It is left-justified and padded with blanks. See Notes for Record 8 on page 4-7, which also apply to this identifier.
45	The primary field for the batch slip ID. The number corresponds to the MICR field. 2 = Process control field 3 = Account number field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2 7 = Serial number field
46	The secondary field for the batch slip ID. The number corresponds to the MICR field. The field can be blank, but it must not be the same as the primary field.

2 = Process control field
3 = Account number field
4 = Optional field 1
5 = Routing number field
6 = Optional field 2
7 = Serial number field

47–50 Blank

Tenth Record:

Notes for Record 10:

1. If dashes occur on the physical document, they should appear in the document identifier.
2. The CPCSOPTN description defines the bank control file for bank 001 **only**. Record 10 must contain only the record type, the bank number, and the record number; columns 11-48 should be blank.
3. Character strings must be left-justified in their document fields and cannot be larger than 16 digits or the maximum length of any field specified, whichever is smaller.

Position	Description
7–9	Sequence number – 010
10	Blank
11–26	The sub-batch slip document identifier. It is left-justified and padded with blanks. See Notes for Record 8 on page 4-7, which also apply to this identifier.
27	The primary field for the sub-batch ID. The number corresponds to the MICR field. 2 = Process control field 3 = Account number field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2 7 = Serial number field
28	The secondary field for the sub-batch slip ID. The number corresponds to the MICR field. The field can be blank, but it must not be the same as the primary field. 2 = Process control field 3 = Account number field 4 = Optional field 1 5 = Routing number field 6 = Optional field 2 7 = Serial number field
29–44	The AST pointer document identifier. Used mainly for image processing, it is left-justified and padded with blanks. See Notes for Record 8 on page 4-7, which also apply to this identifier.
45	The primary field for the AST pointer ID. The number corresponds to the MICR field. 1 = Amount field

2 = Process control field
3 = Account number field
5 = Routing number field
7 = Serial number field

46 The secondary field for the AST pointer ID. The number corresponds to the MICR field. The field can be blank, but it must not be the same as the primary field.

1 = Amount field
2 = Process control field
3 = Account number field
5 = Routing number field
7 = Serial number field

47–49 Blank

50 The block sequence. It specifies whether block slips are placed before or after the documents they control.

P = Before (prior)
S = After (subsequent)

51 The batch sequence. It specifies whether batch slips (and sub-batch slips, if used) are placed before or after the documents they control.

P = Before (prior)
S = After (subsequent)

52 Blank

53 Blank

54–61 The user-written MREAD EXIT routine. It specifies the name of a user-written exit routine that is unique for this financial institution. It is left-justified and padded with blanks.

62–69 The MICR codeline data match user exit. It specifies the name of a user-written routine for codeline data matching. If you use this routine, MICR calls it before making final codeline data match decisions.

70 Blank

71 The microfilm option. This option permits a financial institution to have a different microfilm option. If present, the value must be either F (front) or B (both front and back).

72 Blank

73 The process control dash option. Y indicates that dash transmission is a requirement in this field for the financial institution. N means that dash transmission is not a requirement in this field.

74 The account number dash option. Y indicates that dash transmission is a requirement in this field for the financial institution. N means that dash transmission is not a requirement in this field.

75	The serial number dash option. Y indicates that dash transmission is a requirement in this field for the financial institution. N means that dash transmission is not a requirement in this field.
76	The extract M-string option. This option specifies which M-string to extract and whether to delete the 00 M-string. The options are: **0 or blank** Transfers 00 M-strings for subset 000. This is the default. **1** Transfers 00 M-strings for all subsets. **2** Transfers 99 M-strings for subset 000 and keeps 00 M-strings that correspond to the transferred 99 M-strings. **3** Transfers 99 M-strings for all subsets and keeps the 00 M-strings that correspond to the transferred 99 M-strings. **4** Transfers 99 M-strings for subset 000 and deletes the 00 M-strings that correspond to the transferred 99 M-strings. **5** Transfers 99 M-strings for all subsets and deletes the 00 M-strings that correspond to the transferred 99 M-strings.
77	The display extracted string totals option. This option displays extracted string totals on the supervisor screen and on the diagnostic report. It also produces the ICRE file trailer record that shows the extract totals. **0 or blank** Defaults to no totals or trailer records generated. **1** String totals and file trailer records generated. **Note:** Option 1 is only valid if CPCS is running with the online adjustment feature and the extract M-string option in column 76 is any value 2 through 5.
78–80	Blank

Eleventh Record:

Notes for Record 11:

This record is optional. However, if you plan to send Electronic Cash Letters, you must fill out a record 11 for each sending bank.

Position	Description
7–9	Sequence number – 011
10	Blank
11–19	Routing/Transit number for the sending bank, in standard nine-digit ABA format (including self-check digit).
20	Blank
21–34	Contact name. This should be the name of someone at the sending bank whom the receiving bank should call if there are any questions about the Electronic Cash Letter.

	35	Blank
	36–45	Contact phone. This should be the telephone number the receiving bank should use when trying to reach the contact at the sending bank.
	46	Blank
	47–55	Returns Routing/Transit (R/T) number, in standard nine-digit ABA format (including self-check digit). Any items returned from the sending bank's Electronic Cash Letters go to this R/T. The R/T does not necessarily have to be the same as that specified in columns 11–19.
	56–80	Blank

Endpoint Name-and-Address Record Formats

Creating an endpoint name and address record requires one to six 80-byte records, depending on the amount of information to be stored. You can maintain the data set on tape or disk if you observe the card-code format. The input data set to the DKNLOAD program is device-independent, but it must consist of 80-character logical records.

The first column is the record number code. The records are not checked for sequence; if the endpoint numbers match, they are all written to the same record on the data set.

The second through the ninth columns contain the endpoint number and form the field that binds the records into one transaction. When a new endpoint number is read, the stored information from the previous records is written to the data set. All fields are optional. If there is no information to store in a field, you do not need to include the records.

The records are used to create the endpoint name and address data set (DKNAB). Each endpoint that receives items (either physical or electronic) from the processing financial institution must have a record in the DKNAB data set. If a processing financial institution wants to supply kill lists for one or more on-us pockets, a unique endpoint number must be set up for each pocket. The endpoint must start with 7777. See Note 2 on page 4-23 for a description of on-us kill identification.

First Record:

Position	Description
1	Record code – 1
2–9	The endpoint number. It must be the same on all records.
10	Blank
11–36	First line of the receiving financial institution's name and address; 26 positions
37–40	Blank
41–66	Second line of the receiving financial institution's name and address; 26 positions
67	The type of kill list. It defines what data to distribute to the receiving financial institution: P = Printout C = Reserved (do not use)
68	The distribution media. It defines how to distribute the records to the receiving financial institution: P = Paper T = Magnetic tape F = Reserved (do not use) O = Reserved (do not use) D = Reserved (do not use)
69–71	Blank

Position	Description
72–80	Routing/Transit (R/T) number for the endpoint, in standard nine-digit ABA format (including self-check digit). This field is required if you are going to be sending Electronic Cash Letters to the endpoint. Note that the R/T need not necessarily be the same as the endpoint number in columns 2-9.

Second Record:

Position	Description
1	Record code – 2
2–9	The endpoint number. It must be the same on all records.
10	Blank
11–36	Third line of the receiving financial institution's name and address; 26 positions
37–40	Blank
41–66	Fourth line of the receiving financial institution's name and address; 26 positions
67–80	Blank

Third Record:

Position	Description
1	Record code – 3
2–9	The endpoint number. It must be the same on all records.
10	Blank
11–60	First line of special instruction to or from the user site; 50 positions
61–80	Blank

Fourth Record:

Position	Description
1	Record code – 4
2–9	The endpoint number. It must be the same on all records.
10	Blank
11–60	Second line of special instruction to or from the user site; 50 positions
61–80	Blank

Fifth Record:

Position	Description
1	Record code – 5
2–9	The endpoint number. It must be the same on all records.
10	Blank
11–60	Third line of special instruction to or from the user site; 50 positions

61	Blank
62–73	User data—available to the user for any purpose
74–80	Blank

Sixth Record:

Notes for Record 6:

1. Unlike other name-and-address records, you may specify record 6 multiple times–as many times as is necessary to define all cash letter formats desired and all actions to be taken for each format.
2. If you specify more than one action for a cash letter format, all such actions take place simultaneously, meaning you get multiple copies of the same cash letter but with each copy prepared in a different way.
3. It is legal to specify an action with the CLIF Flag and the same action without the CLIF Flag; the cash letter is prepared in both formats simultaneously.
4. If you must specify more actions for a cash letter format than can fit on a single record 6, create another record 6 with the same cash letter format and continue with your action list.
5. Record 6 is optional. If you specify no record 6's for an endpoint, CPCS assumes a single cash letter format of "PAPER" and a single action for that format, "PRINT".

Position	Description
1	Record code – 6
2–9	The endpoint number. It must be the same on all records.
10	Blank
11–18	Cash letter format name, left justified in the field. At present, the only names recognized are "PAPER" and "ECASH". Additional names will be defined in the future.
19	Blank
20–24	Action to take place when processing cash letters that are in this format. At present, the only actions recognized are "PRINT" and "X937". "PRINT" causes the cash letter to be physically printed; "X937" causes the cash letter to be written in ANSI X9.37 format to the data set specified by the DKNETC entry in your DKNDSAT table. Additional actions will be defined in the future.
25	Blank
26	CLIF flag. When preparing an electronic cash letter and this flag is a "Y", the CLIF file prefix information is added to each record so the cash letter can be subsequently recognized and processed by the E-Check server. For any other value, the electronic cash letter is prepared unaltered. Do not specify a "Y" if your action is "PRINT".
27–28	Blank
29–35	Same as with columns 20–26. This allows you to specify a second action for the same cash letter format.
36–37	Blank

38–44	Same as with columns 20–26. This allows you to specify a third action for the same cash letter format.
45–46	Blank
57–53	Same as with columns 20–26. This allows you to specify a fourth action for the same cash letter format.
54–55	Blank
56–62	Same as with columns 20–26. This allows you to specify a fifth action for the same cash letter format.
63–64	Blank
65–71	Same as with columns 20–26. This allows you to specify a sixth action for the same cash letter format.
72–80	Blank

Sort-Pattern-Definition Record Formats

The sort-pattern-definition file contains sort information that MICR retrieves during the initialization of a document processor entry. Name each member of this file in the form SPTYPxxx, where *xxx* (001 through 255) uniquely identifies the type of sort that is defined. The sort-pattern records in each member of this file define the characteristics of each sort type.

A member can consist of multiple P records if there is more than one sorter run in the sort pattern. Each P record may be followed by an R record, one or more K records, a B record, and as many other record types as is necessary to completely describe the sorter pass the P record represents. The records and their contents must conform to the following descriptions. You can specify information for the following record types:

B Bank description record. This record designates the processing financial institution.

E Document-type description record. This record designates the type of document to be captured for a prime pass.

H Hardware-option description record. This record designates OCR or MICR and power encode options for a sort pass. For more details, refer to the *3890/XP Series Programming Guide*.

J Reject or unencoded description record. This record designates reject and unencoded pockets for a sort pass.

I Image-capture description record. This record designates imaging options for a prime pass.

K Kill-pocket description record. This record designates kill pockets for a sort pass.

M Mixed-string combination-key description record. This record enables mixing of rehandle strings for different sort types.

O Run-options description record. This record designates divider-resynchronization and enhanced-prime creation options for a sort pass.

P Pass description record. This record defines the type of sort pass initialization you can perform (for example, standard or expanded format). The 3890 and the 3890 emulator for the 3890/XP Series document processors use the standard format.

R Rehandle-pocket description record. This record designates rehandle pockets for a sort pass.

You can use the following record types only for expanded record formats, except where noted:

BMSG Begin message description record. This optional record enables operator messages to display on the BEGIN screen after SPDEF editing is complete.

FLD*nn* Field description record. This optional expanded format record defines the size of a field. *nn* refers to the field number (01 through 07 or 09 through 15).

FS File-stamp description record. This optional record can contain PROLOG and table information for the module identified in the P record.

RP Run-profile description record. This record provides a run profile name for an expanded-format (XF) sort. It is a requirement for an XF sort. It is acceptable, but not a requirement, on a non-XF sort.

TY Pass-type description record. This optional record identifies sort type (such as 2- or 4-byte) and maximum sort-program size.

B—Bank Description Record (BNKN)

This record designates the processing financial institution.

Note: If a value has been coded in the bank control file for maximum subsets allowed, it only becomes effective if a “B” record is included in the sort pattern definition. The master task value (MAXSUB) is used if the “B” record is omitted.

Position	Length	Description
1	1	‘B’=BNKN identifier
2	3	Reserved
5	3	Sending bank **001=** Site financial institution (default) **002–999=** Non-site financial institution **Note:** If a B record is not explicitly specified for any pass, bank 001 is used for that pass. Including a B record on prime-pass does not mean that bank 001 is automatically used for subsequent passes.
8	65	Reserved
73	8	Sequence number (for user maintenance)

E—Document-Capture-Type Record (DCAP)

This record designates the type of document to be captured for a prime pass.

Note: Spaces are translated to zeros except for records that are all spaces. These records default to a 001 (pre-qualified) document type.

Position	Length	Description
1	1	'E'=DCAP identifier
2	3	Reserved
5	3	Type of document being captured **000=** Unqualified **001=** Pre-qualified (default) **002=** Signature card data **003=** Returns **004=** Cycle/exception **005=** Statement sort **006=** OCR remittance **007=** OCR non-remittance **008=** Wholesale remittance **009=** Drafts (credit card voucher) **010–127=** Reserved for IBM use **128–255=** Reserved for user applications
8	65	Reserved
73	8	Sequence number

H—Document Processor Hardware Options Record (OCR)

This record designates OCR or MICR options for a sort pass, including power encoder options. You cannot turn a PATH off in the sort pattern H record; turn them off or on through the sort operator panel.

Position	Length	Description
1	1	'H'=OCR identifier
2	3	Reserved
5	1	OCR read system 1 *blank***=** Off (default) **N=** Off **Y=** On
6	1	OCR read system 2 *blank***=** Off (default) **N=** Off **Y=** On
7	1	OCR read system 3. Must be off if either read system 1 or read system 2 is on. *blank***=** Off (default) **N=** Off **Y=** On
8	2	Reserved
10	1	MICR read system 1 *blank***=** Off (default) **N=** Off **Y=** On
11	1	MICR read system 2 *blank***=** Off (default) **N=** Off **Y=** On

Position	Length	Description
12	2	Reserved
14	1	Power encode option *blank*= Off (default) **P=** On **Y=** On **N=** Off
15	1	Reserved
16	1	Reserved
17	1	Reserved
18	1	Reserved
19	4	Power encode font *blank*= E13B (MICR default) **E13B=** MICR **CMC7=** OCR
23	1	Reserved
24	7	Fields to encode Digits 1 through 7 are valid. **1=** Amount field **2=** Process control field **3=** Account number field **4=** Extended process control field **5=** Routing number field **6=** Optional field **7=** Serial number field
31	1	Reserved
32	2	Microfilm space *blank*= 0 (default) **0=** No microfilm space option **1=** Option 1 **2=** Option 2 **3=** Option 3
34	39	Reserved
73	8	Sequence number (for user maintenance)

I—Sorter Image-Capture Options Record (IMAGE)

This record identifies the image capture specifications for each document.[1]

Position	Length	Description
1	1	'I'=IMAGE identifier
2	3	Reserved

[1] HPTS Application Library Services Release 1 supports image capture only for the 3890/XP Document Processor.

Position	Length	Description
5	1	Pass *blank*= Prime only (default) **P=** Prime only **H=** HSRR only **B=** Both prime and HSRR
6	2	Reserved
8	2	Front black and white *blank*= Not on (default) **BW** Capture front black-and-white image
10	1	Reserved
11	2	Front gray scale *blank*= Not on (default) **GS=** Capture front gray-scale image
13	4	Reserved
17	2	Back black and white *blank*= Not on (default) **BW=** Capture back black-and-white image
19	1	Reserved
20	2	Back gray scale *blank*= Not on (default) **GS=** Capture back gray-scale image
22	1	Reserved
23	1	Compensation error override *blank*= Not on (default) **Y=** Ignore compensation errors
24	1	Reserved
25	1	Analysis error override *blank*= Not on (default) **Y=** Ignore analysis errors
26	3	Reserved
29	7	Online/offline to host processor *blank*= Online (default) **ONLINE=** Imaging processor online to host processor (CIMS) **OFFLINE=** Imaging processor offline from host processor
36	2	Reserved

Position	Length	Description	
38	10	Overlay/no overlay	
		blank=	No overlay (default)
		NO OVERLAY=	Do not overlay images within the sorter image buffers. The sorter will pause the main feed when the image buffers are full.
		OVERLAY=	Overlay images within the sorter image buffers when the buffers become full. **Note:** This option is not valid if the image processor is online to the host.
48	2	Reserved	
50	3	Location in process buffer for DIDM and document type	
		blank=	None
		000=	None
		001–244=	Specified location for DIDM and document-type information with the document's process buffer. The location is relative to 1 and right to left. For example, if the P/C field is to contain DIDM information and the amount field is 5 bytes plus 12 header bytes, the value would be 018 (6 + 12). This value, the rightmost byte of the P/C field, is in packed-decimal format so that the 4-character value is loaded into 2 bytes.
53	20	Reserved	
73	8	Sequence number	

J—Reject-Pocket-Number Record (RUPKT)

This record designates reject pockets or unencoded pockets for a sort pass.

Position	Length	Description
1	1	'J'=RUPKT identifier
2	3	Reserved
5	1	Reject pocket indicator—pocket 1 *blank*= Not a reject or unencoded pocket **R=** Reject pocket **U=** Unencoded pocket **E=** Enhanced reject pocket
6	1	Reject pocket indicator—pocket 2
7	1	Reject pocket indicator—pocket 3
⋮		
39	1	Reject pocket indicator—pocket 35
40	33	Reserved

Position	Length	Description
73	8	Sequence number (for user maintenance)
		Note: An unencoded pocket must be a rehandle pocket. A reject pocket indicator of **E** indicates that the pocket is eligible for enhanced reject processing. If bad items are sent to a good pocket without the **E** code specified for that pocket, those items are not included for repair in the RC D-string. If a pocket is specified as an alternate reject by having an **R** code for that pocket and no rejects are sent to the pocket, DIST still uses the alternate reject pocket string name.

K—Kill-Pocket Description Record (KDR)

This record designates kill pockets for a sort pass.

Position	Length	Description
1	1	'K'=KDR identifier
2	2	CPCS pocket number of kill pocket
4	8	Endpoint ID (numeric) for kill pocket
12		See Notes
22		See Notes
32		See Notes
42		See Notes
52		See Notes
62		See Notes
72	1	Reserved
73	8	Sequence number

Usage Notes:

1. Additional kill-pocket descriptions can appear in these columns as in columns 2 through 11. (Endpoint ID is an arbitrary 8-digit number that associates a kill pocket with the endpoint name and address record in the DKNAB data set. The endpoint number must be the same as the AB-NUM for a record in the DKNAB data set.)
2. There are two ways to identify an on-us kill pocket:
   a. If you want a kill list for the pocket, you must specify an endpoint that starts with 7777. The remaining four positions can be any digits. You must also add the endpoint to the endpoint table.
   b. If you do not want a kill list for the on-us kill pocket, you must specify an endpoint of 88888888 or 99999999.
      - 88888888 means DKNMCRE automatically releases the D-string.
      - 99999999 means that the D-string is not released until you call DKNSZAP to mark it as released. Once marked, it is released the next time you run DKNMCRE.

M—Mixed-String Combination-Key Description Record (MDR)

This record enables mixing of rehandle strings for different type sorts. For a mixed-string combination, you must specify an M-record. The M-record is a requirement only when you want to mix work of different sort types and different pass pocket histories.

Use the M-record with an R-record on a pass that contains one or more rehandle pockets. By specifying a 2-digit code of 01 through 99 for the correct rehandle pocket, you can mix the rehandle work (when run on the next pass) with other work having the same 2-digit code in a preceding pass M-record. You can include all other records (such as TY, RP, FLDnn, FS, and BMSG) as necessary.

Position	Length	Description
1	1	'M'=MDR identifier
2	2	01 through 99 decimal key for pocket 1*
4	2	01 through 99 decimal key for pocket 2*
6	2	01 through 99 decimal key for pocket 3*
⋮		
70	2	01 through 99 decimal key for pocket 35*
72	1	Reserved
73	8	Sequence number

*Effective only if this is a rehandle pocket on the rehandle record.

O—Run-Option Record (OPTN)

This record designates various options for a sort pass.

Important!

If you want divider resynchronization for a particular pass, you must include this record following the pass descriptor record (P) for the previous and current passes.

Position	Length	Description
1	1	'O'=OPTN identifier
2	3	Reserved
5	1	Divider resynchronization indicator *blank* No options (default) **D** Divider data set created on prime pass. Divider resynchronization on subsequent passes.
6	1	Prime pass type *blank* Conventional prime pass **E** Enhanced prime pass

Position	Length	Description
7	1	Enhanced prime sorter initialization at entry breaks option (this option is only valid when prime-pass type = E; this option is ignored on simulated captures). This option eliminates potential small kill bundles at the beginning of each entry and must be active when the Divider Spray option B (merge-before-main) is active. This option also results in having the item sequence number rolled over after a few items of the new entry have been captured; the item sequence number rolls over if the rollover sequence number has been reached when a new subsequent entry is started. **Note:** The sorter stops and must be restarted when the item sequence number rolls over. *blank* No sorter initialization at entry breaks **Y** Sorter initialization at entry breaks
9	4	Rehandle pocket divider merge count. Used only if the divider resynchronization indicator equals D.
13	1	Tracer-in-kill-pockets option (this option is valid only for enhanced prime passes). **Y** Place a tracer in each kill pocket ahead of each entry. **N** No tracers in kill pockets (default).
14	1	Distribution option. Valid values are 1, 2, 3, 4, 5, 6, 7, 8, 9, A, and B. (For an explanation of these values, see Figure 4-1.) If you want full distribution, you can leave this field blank.
15	1	Enhanced Prime pause option *blank* No pause between entries **N** No pause between entries **Y** Pause between entries
16	8	DKNMIPI user-processing exit name. If this is blank, CPCS uses the name in either the BCF or the CPCSOPTN macro. For more information on this exit routine, see “DKNMIPI User Processing Exit” on page 4-111.
29	1	Enhanced reject processing. The values are: *blank* Enhanced reject processing is not active. **N** Enhanced reject processing is not active. **Y** Enhanced reject processing is active.
30	1	Divider spray type. The values are: *blank* Place dividers after kill bundles. **A** Place dividers after kill bundles. **B** Place dividers before kill bundles (merge-before-main)
31	8	DKNMDIS user-exit name. If this is blank, CPCS uses the name in the bank control file or in the BLDL table.
39	1	Prime-pass main hopper divider identification option. The values are: *blank* Do not identify dividers. **N** Do not identify dividers. **Y** Identify dividers.
40	1	Blank

Position	Length	Description
41	3	Number of tracer groups that comprise an Enhanced Prime entry. Default is 001. This field is only used when column 6 is “E”.
44	29	Reserved
73	8	Sequence number

Distribution Options: Figure 4-1 describes the distribution options for nonsubset processing or for subset processing. For more information on distribution options, and on how DKNDIST behaves under various conditions when confronted with different types of strings and distribution options, see “CPCS Pocket Distribution by DKNDIST” in the *CPCS Programming and Diagnostic Guide*.

Figure 4-1. Distribution Options

Option Number	Description
1	Full string distribution. All pockets are distributed. This is a normal distribution, and option 1 indicates that the I-string is distributed.
2	Full kill distribution. All kill pockets are distributed. DKNDIST selects only the pockets that are marked as good pockets and kill pockets. It does not check for rejects in the good pockets. This is a selective pocket distribution. DKNDIST does not mark the I-string as distributed.
3	Good kill-only distribution. DKNDIST selects only those pockets that are marked as good pockets and eligible kill pockets that contain rejects, if enhanced reject processing is off. This is a selective pocket distribution. DKNDIST does not mark the I-string as distributed.
4	Full reject-pocket distribution. DKNDIST distributes the system reject pocket (pocket 1-1) and the alternate reject pockets. It produces the consolidated reject (RC) string. This is a selective pocket distribution. DKNDIST does not mark the I-string as distributed. It does, however, turn on the flag in the I-string that indicates that the consolidated reject string is distributed.
5	Full enhanced-reject-processing pocket distribution. DKNDIST distributes all reject pockets, including the system reject pocket. This is a selective pocket distribution. DKNDIST does not mark the I-string as distributed.
6	Consolidated reject distribution. DKNDIST distributes reject items that are in good pockets. This is a consolidated reject-pocket distribution. DKNDIST does not mark the I-string as distributed. It does, however, turn on the flag in the I-string that indicates that the consolidated reject string is distributed.
7	Reject and full kill distribution. DKNDIST distributes all pockets except rehandle pockets. This is a selective pocket distribution. DKNDIST does not mark the I-string as distributed.
8	Reject and partial kill distribution. DKNDIST distributes all reject pockets, including the system reject pocket. It also distributes kill and eligible kill pockets that have no rejects. This is a selective pocket distribution. DKNDIST does not mark the I-string as distributed.
9	Suppress automatic distribution.
A	Full rehandle distribution. DKNDIST distributes only rehandle pockets. This is a selective pocket distribution. DKNDIST does not mark the I-string as distributed.
B	Reject and full rehandle distribution. DKNDIST distributes only rehandle pockets and reject pockets. This is a selective pocket distribution. DKNDIST does not mark the I-string as distributed.

P—Pass-Description Record (PDR)—Standard Form

This record defines the type of sort-pass initialization you can perform. The standard format is used for the 3890 Document Processor and a 3890/XP Series document processor emulating a 3890 Document Processor.

Position	Length	Description
1	1	'P'=PDR identifier
2	7	Source pass/pocket history 'PAABBCC'. For example, PAABBCC=2110000 equals pass 2 from pocket 11 on pass 1.
9	8	Primary stacker-select module name (required). Used on prime or subsequent pass.
17	8	Sort pattern table name (optional)
25	2	Number of rehandle pockets resulting from this pass. Used on prime or subsequent pass; defaults to zero.
27	1	Debit/credit order **0 or** *blank* Debits precede credits **1** Credits precede debits **2** All debit work (no credits) **3** All credit work (no debits) Prime pass only; defaults to zero
28	11	Reserved
39	1	Item-numbering option *blank* No item number, invalid on prime pass **1** Position 1 on prime pass, no item number on high speed **2** Position 2 on prime pass, no item number on high speed **3** Position 3 on prime pass, no item number on high speed **4** Position 1 on prime pass, position 2 on high speed **5** Position 2 on prime pass, position 3 on high speed **6** Position 3 on prime pass, position 1 on high speed **7** Position 3 on prime pass, position 2 on high speed
40	4	Kill bundle minimum count
44	4	Unused
48	7	Next run PPH (PAABBCC) (optional)
55	1	Microfilming option *blank* No microfilming **M** Microfilming **J** Microfilming with microfilm jam processing option This option ensures that the MICR operator can reroute to the reject pocket those items that read correctly, but are involved in a jam at or before the microfilm module and are not microfilmed.

Position	Length	Description
56	1	Endorsing option *blank* No endorsing **1** Endorsing position 1 **2** Endorsing position 2 **3** Endorsing position 3
57	16	Available
73	8	Sequence number (for user maintenance)

P—Pass-Description Record (PDR)—Expanded Format (XF)

This record defines the type of sort pass initialization you can perform.

Position	Length	Description
1	1	'P'=PDR identifier
2	7	Source pass/pocket history 'PAABBCC'. For example, PAABBCC=2110000 equals pass 2 from pocket 11 on pass 1.
9	8	Primary stacker-select module name (required). Used on prime or subsequent pass.
17	8	Sort pattern table name (optional)
25	2	Number of rehandle pockets resulting from this pass. Used on prime or subsequent pass; defaults to zero.
27	1	Debits/credit order **0 or** *blank* Debits precede credits **1** Credits precede debits **2** All debit work (no credits) **3** All credit work (no debits) **4** Mixed debit/credit non-POD Prime pass only; defaults to zero
28	11	Reserved
39	1	INF position (prime pass only) and endorse position (prime and subsequent passes) *blank* No item numbering or endorsing; invalid on prime pass **1** Position 1 **2** Position 2 **3** Position 3 **4** Position 4 (3890/XP only) **5** Position 5 (3890/XP only) **6** Position 6 (3890/XP only)
40	4	Kill bundle minimum count
44	4	Expanded format indicator = 'XF '
48	7	Next run PPH (PAABBCC) (optional)
55	1	Microfilming option *blank* No microfilming **M** Microfilming **J** Microfilming with microfilm jam processing option This option ensures that the MICR operator can reroute to the reject pocket those items that read correctly, but are involved in a jam at or before the microfilm module and are not microfilmed.

Position	Length	Description
56	1	Endorsing option *blank* Endorsing off (default) **N** Endorsing off **1** Endorsing on
57	1	INF option *blank* INF off (default) **N** INF off **1** INF on
58	1	Reserved
59	1	INF position and endorse position (HSRR pass only) *blank* Not valid on HSRR pass **1** Position 1 **2** Position 2 **3** Position 3 **4** Position 4 (3890/XP only) **5** Position 5 (3890/XP only) **6** Position 6 (3890/XP only)
60	1	Reserved
61	1	3892/XP front endorse option *blank* Front endorsing off (default) **N** Front endorsing off **1** Front endorsing on **Y** Front endorsing on **C** Front endorse with converge DLL
62	1	Reserved
63	1	3892/XP stamp endorse option *blank* Stamp endorsing off (default) **N** Stamp endorsing off **1** Stamp endorsing on **Y** Stamp endorsing on
64	1	3892/XP stamp endorse position *blank* No stamp endorse **1** Position 1 **2** Position 2 **3** Position 3
65	1	Reserved
66	1	3892/XP back endorse font *blank* Small font (default) **0** Small font **1** Large font
67	1	3892/XP front endorse font *blank* Small font (default) **0** Small font **1** Large font
68	5	Reserved
73	8	Sequence number (for user maintenance)

R—Rehandle-Pocket-Description Record (RDR)

This record designates rehandle pockets for a sort pass.

Position	Length	Description
1	1	'R'=RDR identifier
2	2	CPCS pocket 1 tracer count[2]
4	2	CPCS pocket 2 tracer count[2]
6	2	CPCS pocket 3 tracer count[2]
⋮		
70	2	CPCS pocket 35 tracer count[2]
72	1	Reserved
73	8	Sequence number

Important!

If this is a prime-pass R record, the sum of:

```
(CPCS pocket 1 tracer count + ... CPCS pocket 35 tracer count)
+ 9
+ number of kill pockets
```

must not be more than the MAXRP parameter in the CPCSOPTN macro. If you do not use the tracer-in-kill-pocket option on the O record, you do not need to include the number of kill pockets in the above formula. If the sum is more than the MAXRP parameter, you must reassemble the CPCSOPTN macro, perform the MICR task generation, and perform a cold start.

BMSG—BEGIN Message-Description Record (BMSG)

This optional record enables operator messages to display on the BEGIN screen after SPDEF editing is complete.

Position	Length	Description
1	4	'BMSG'=identifier
5	2	Reserved
7	66	Message text to display on MICR BEGIN screen
73	8	Sequence number (for user maintenance)

FLDnn—Field-Description-Record (FLD) Standard and Expanded Format

You can define the size of a field using this optional record.

Warning: Values and definitions specified by usage of FLDnn cards override any options specified in the CPCS RDR options for this sort type.

[2] For kill pockets, the tracer counts must be blank. If you select the tracer-in-kill-pocket option, CPCS directs one tracer document to each kill pocket.

Position	Length	Description
1	5	'FLDnn'=Identifier **FLD01** Field 1 **FLD02** Field 2 ⋮ **FLD15** Field 15 You cannot change field 8.
6	5	Reserved
11	3	Field length—number of bytes required to store the field. Divide the number of digits by 2 and round up. For information about default byte-lengths for the MICR document fields, see page 2-41.
14	2	Reserved
16	2	Number of digits—the maximum size of the field. Fields 1 through 7 only; maximum=30. For information about digit length defaults for MICR document fields, see page 2-41.
18	10	Reserved
28	2	Field digit-error threshold. Values 0 through 15 (fields 1 through 7). Default value is 0.
30	1	Reserved
31	1	Codeline data match indicator (valid fields 1 through 7 and 9 through 15) **N** Not active **Y** Active **E** Extended codeline data match (EIM) (valid for XF sorts only) *blank* Default **Prime pass** N for all fields **HSRR pass** N for all fields **Sub-pass** Y for fields 1 through 7 N for fields 9 through 15
32	2	Reserved
34	2	High-order zero correction parameter (valid for field 1 only) *blank* Not active **0** Not active **1–15** Number of high-order digits eligible for correction (See the *3890/XP Document Processor Programming Reference*.)
36	1	Symbol error correction indicator for field (valid fields 1 through 7 only) *blank* Active (default) **N** Not active
37	6	Reserved
43	12	Descriptive field name (optional)
55	4	Reserved

Position	Length	Description
59	1	Field 3 closing; the default value is zero. **0** S4 or S5 **1** S4 only This specifies which closing symbol is valid for field 3.
60	2	Reserved
62	1	Field format (fields 1 through 7 only) **F** Fixed (default) **V** Variable
63	10	Reserved
73	8	Sequence number (for user maintenance)

FS—File Date and Time Stamp Description Record (FS) Expanded Format

This optional record contains the date and time data PROLOG and table information for the module identified in the P record.

Position	Length	Description
1	3	'FS '=Identifier
4	5	Reserved
9	8	File name (required)
17	4	Reserved
21	8	Optional date stamp (MM/DD/YY)
29	2	Reserved
31	8	Optional time stamp (HH.MM)
39	34	Reserved
73	8	Sequence number (for user maintenance)

RP—Run-Profile-Description Record (RP) Expanded Format

This record provides a run profile name for an expanded-format (XF) sort. It is required for an XF sort.

Position	Length	Description
1	3	'RP '=Identifier
4	5	Unused
9	8	Run profile name (optional)
17	4	Unused
21	8	Date stamp—YY/MM/DD (optional)
29	2	Unused
31	8	Time stamp—HH/MM/SS (optional)
39	2	Unused

Position	Length	Description
41	5	Document processor model *blank* Any document processor **'XP '** Any XP processor **'3892 '** Any 3892/XP **'PE '** Any power encoder **'3890 '** Any 3890 **3890A** 3890 Model A **3890B** 3890 Model B **3890X** 3890 Model X **IMAGE** Any document processor with image capacity
46	27	Unused
73	8	Sequence number (for user maintenance)

TY—Pass-Type Description Record (TY) Expanded Format

This optional record identifies sort type (such as 2- or 4-byte) and maximum-sort program size.

Position	Length	Description
1	3	'TY '=Identifier
4	27	Unused
31	4	Sort-type indicator (optional) **SCI4** 4-byte SCI **SCI2** 2-byte SCI If set, the sort program can be only SCI2 or SCI4. The default value (blank) is a 2-byte SCI routine.
35	6	Unused
41	6	Maximum sort-size indicator (option) *nnnnnK* indicates the maximum storage size required to load all SCI modules for this pass.
47	26	Unused
73	8	Sequence number (for user maintenance)

CPCS Stacker-Select Routine Operation

The stacker-select prolog (SSP) performs specific MICR processing operations on each document that the document processor reads during stacker-select time. SSP code consists of 3890 stacker control instructions (SCIs) and resides in the first 1504 (X'5E0') bytes of program storage of the document processor. CPCS macros generate this SSP along with the user's SCIs. The SSP performs the following functions:

- Initializing and setting CPCS-reserved switches and areas within the document processor
- Processing I-strings
- Controlling the updating of item number and microfilm number
- Identifying document types
- Selecting and processing CPCS control documents
- Allocating to pockets (tracer spraying)
- Issuing SCI pause instructions for tracer-verification errors and resynchronization of codeline data matching
- Routing all batch, sub-batch, block, and user documents to the user's SCI routine at the label DKNUSER
- Receiving control after user processing at label DKNRTN
- Processing SCI errors.

SSPs are included with SCIs that reside in sorter-program storage. They consist of DKNMPSR and a PROLOG macro, a PROLOG2 macro, or a PROLOGX macro, depending on the format that you selected (standard or expanded). An explanation of the different SSPs follows. For a description of sorter-program storage use, see Figure 4-2 on page 4-35 and Figure 4-4 on page 4-36.

Standard Stacker-Select Prolog

A standard-sort SSP consists of two parts: the ENTR macro (80 bytes long) and PROLOG. The ENTR macro includes a routine for processing SCI errors. When a SCI error occurs, the SCI error routine sets an operator message light on and loops, causing a SCI error header to go to the host processor. For the exact setting of the operator message lights, see the *3890/XP Document Processor Programming Reference*.

The second part of the SSP is the PROLOG macro, which is used in the assembly of user stacker-select routines. It is assembled with user SCIs and is link-edited as reentrant into the CPCS load library. The SSP/user stacker-select routine is loaded into the host computer entry for the specific sort type. DKNMSTRT changes the SSP portion before SETDEV sends it to sorter program storage. A user-provided table can also be loaded with the SSP/user stacker-select routine.

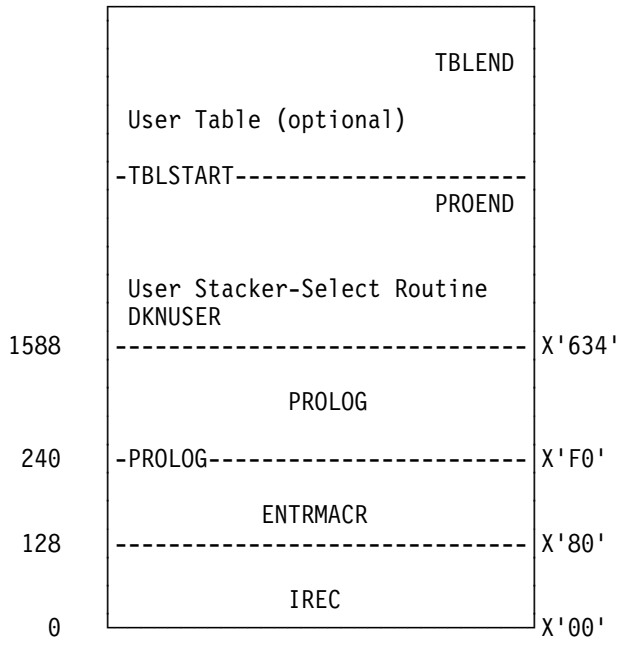

Figure 4-2. CPCS Sorter-Program Storage Map—Standard Sort

When the CPCS standard-sort SSP completes processing both non-control documents and CPCS block and batch control documents, it branches to the user-edit routine at label DKNUSER. Assembling the PROLOG macro in the user SCI assembly supplies user linkage. User-written, stacker-select routines must perform the following functions:

- Pocket selection and field validation for non-control documents
- Field validation for control documents (except tracer slips).

Expanded Stacker-Select Prolog

The PROLOG2 or PROLOGX macro is the first statement coded in expanded-sort primary stacker-selection routines. PROLOG2 uses 2-byte SCIs, program size specification, and a 32-byte prefix. PROLOGX uses 2- or 4-byte SCIs based on a type indicator, a program size specification, a 32-byte prefix, and a table address list.

DKNMLPR*n* (*n*=1, 2, or 4) changes the SSP portion before SETDEV sends it to sorter program storage. A user-provided table can also be loaded with the SSP/user stacker-select routine.

You must use the following convention for SCI routines:

```
NAME     PROLOG
         THI     2,X'80'             IS THIS A CPCS CONTROL DOCUMENT?
         BCS     1,CTLDOC            YES, BRANCH
ONCTL    EQU     *                   PROCESS USER DOCUMENTS
         .
         .   (SCIs)
         .
RETURN   BCS     15,DKNRTN           ALL DONE, RETURN TO PROLOG
CTLDOC   EQU     *
         .
         .   (SCIs)
         .
         BCS     15,RETURN
         PROEND
```

Figure 4-3. Sample Format for SCI Routines

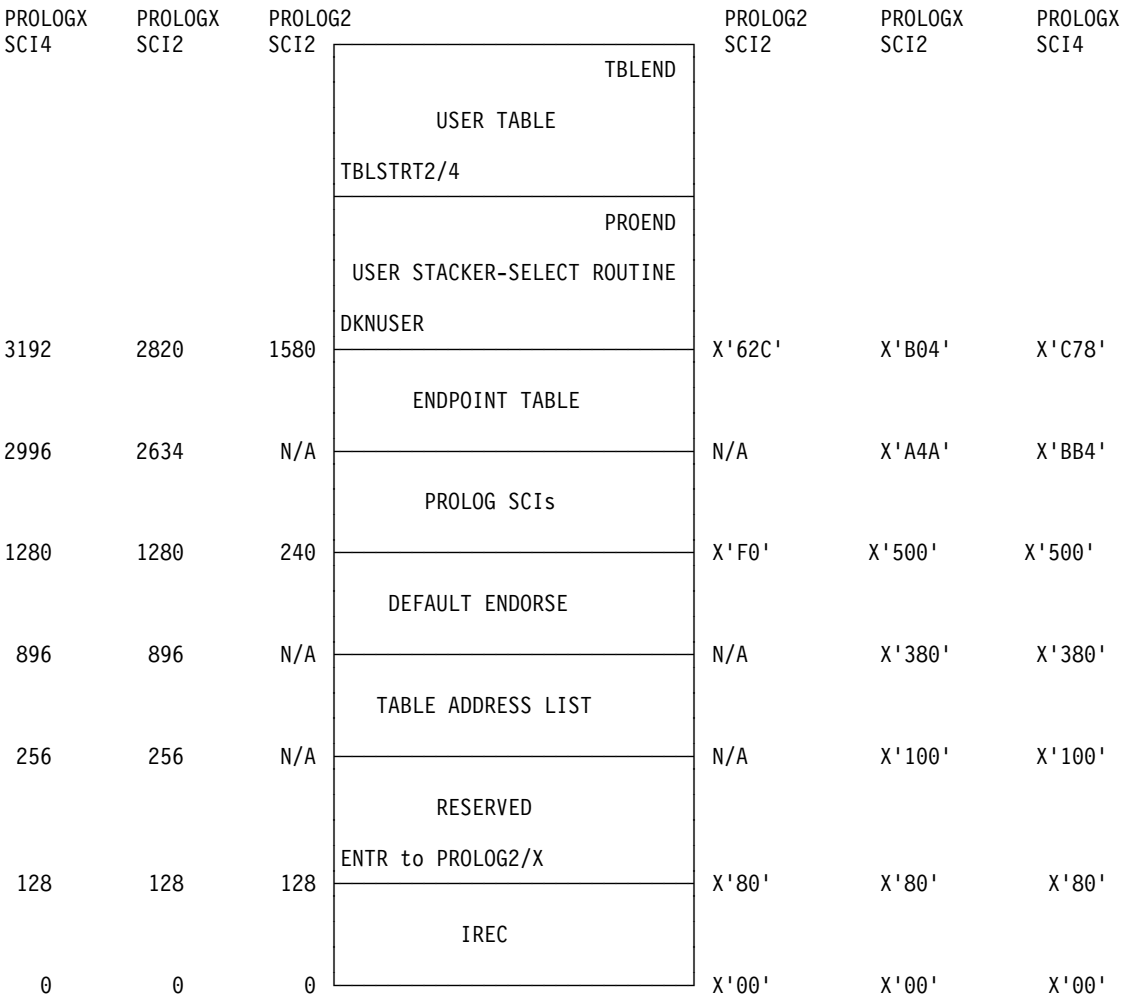

Figure 4-4. CPCS Sorter-Program Storage Map—Expanded Sort

Sort-Control Program Notes

- Your sort-control program routines must ensure that all auto-selected documents go to the reject pocket. Header byte 8 bit 0 indicates to the sort-control program routine that the document is an auto-select.
- You can also specify a sort pattern table in the SPDEF. The table is assembled separately from your stacker-select routine, but SETDEV loads it with your stacker-select routine. The document processor does not provide for relocation of sort-control programs and has no means of resolving sort-control program addresses while sort control programs are running. All sort-control program module linkage must be complete before SETDEV initiation.
- User-written SCIs have access to the codeline data record and header during sort-control program processing; however, specific header bytes and save area locations are reserved for CPCS use. (For a description of header and status bytes and save-area locations, see “CPCS Document Processor Header-Byte, Status-Byte, and Save-Area Usage” on page 4-38.) If your sort control programs set or reset any of these, unpredictable results can occur.
- Your sort control program routines must ensure that all fields are valid and contain correct data.
- You might want to modify fields 1 through 7 in the process buffer. Because a given field’s location in the process buffer is *not* fixed, subroutines enable accurate modification of the process buffer (in program storage, for the 3890 Document Processor and XPs emulating 3890s, and in auxiliary storage for XP processors).

  Load the work register with the modified value for the field. Perform a branch and link (BLS), using link register 1 as the return register and DKNUPF*n* as the target label, where *n* is the number of the field to update. If it is not a defined field, it makes a simple return to the caller. You can determine whether it is a field defined to the document processor by testing the byte at label DKNUPF*n*. If the value of the byte at that label is X'BE'(BUR operation code), the field is *not* defined to the document processor.
- If you define field 4, field 6, sub-batch documents, or AST pointer documents, PROLOG moves the routing-number check digit to the leftmost digit of the process buffer (undefined digit of sequence number). You can access the check digit in the right digit of program storage at label DKNIIRTD+3. You can update the check digit that went to the host by changing the rightmost digit of the byte at label DKNIIRTD+3 and performing a branch and link (BLS), using link register 1 as the return register and DKNUPRTD as the target label.

  If you do not define field 4, field 6, sub-batch documents, or AST pointer documents, you can access and update the check digit in header byte 9, bits 0 through 3.
- The PROLOG marks field errors, indicated by header byte 9, bits 0 and 1 (on is valid), if either of the fields is defined to the document processor. Undefined fields are marked as valid.

  In testing for field 6 validity in header bytes 0 and 1, you should only test byte 1 (length/special symbol error) when the document contains a valid non-blank field 7 (serial number). This is because of the need for a closing symbol to denote the end of a MICR field and because the closing symbol for field 6 is also the opening symbol for field 7. If no field 7 is present, header byte 1

denotes a special symbol error for field 6 and header byte 6 indicates field 6 as not valid.

- In addition to ensuring field 6 validity through the document-data-header bytes, your routine should also check the contents of field 6 to ensure that correct data is present. Give special attention to field 7 (serial number) validity when processing field 6 data. If the serial number field contains a flag that indicates a length or special symbol error (bit 7 of header byte 1), then field 6 might contain data that is not correct. You must check for this possible error and, if the flag is present, reject the document.

CPCS Document Processor Header-Byte, Status-Byte, and Save-Area Usage

You must set the CPCS document processor's header and status bytes, defined in Figure 4-5, where indicated. For more information about header and status bytes, see the *3890/XP Document Processor Programming Reference*.

Figure 4-5 (Page 1 of 2). CPCS Document Processor Header Byte and Status Byte

Byte	Bit	Setting
Header		
2	0	
		0 User document **1** CPCS control document
	1–2	If CPCS control document:
		00 Divider **01** Tracer **10** Block **11** Batch
		Note: Byte 9 designates sub-batch and AST pointer documents.
		If user document:
		00 Transit item **01** Credit **10** On-us **11** On-us and credit
	3–7	Field validity bits for fields 1, 2, 3, 5, and 7
		1 Field valid **0** Field not valid
3		Reserved for the user and carried with the record in the DITYPEU field
5	1	If the microfilm feature is active, the PROLOG turns on this bit. If the user does not want to microfilm a document, this bit must be NHIed (ANDed) off to suppress microfilming the document.
6		Module pocket code must be set for all non-control documents.

Figure 4-5 (Page 2 of 2). CPCS Document Processor Header Byte and Status Byte

Byte	Bit	Setting
9		
	0	1=Field 4 valid
	1	1=Field 6 valid
	2	1=Sub-batch document (byte 2, bits 0 through 2=111)
	3	1=AST pointer document (byte 2, bits 0 through 2=100)
		Note: These bits have meaning only when field 4, field 6, sub-batch, or AST documents are defined. Otherwise, bits 0 through 3 contain the routing number check digit. See "Sort-Control Program Notes" on page 4-37.
Status		
	0–2	Reserved for CPCS
	3–7	Available to the user

The CPCS PROLOG uses the following save areas in the document processor:

0–499 Available to the user
500–503 Tracer count for tracer spraying
504–505 Pocket index for tracer spraying
506–509 Tracer-group number
510–511 Document-type code save area.

CPCS SCI Macros and User SCI Coding

There are a number of macros within CPCS that you can use to code user SCI routines and tables. PROLOG and PROEND delimit the SCI stacker-select routine, and TBLSTART and TBLEND delimit the optional SCI table.

The expanded format PROLOG macros, PROLOG2, PROLOGX, TBLSTRT2, and TBLSTRT4, work with the 3890/XP Series document processors only. The expanded format SPDEF records are required to initialize these sorts (see "Sort-Pattern-Definition Record Formats" on page 4-17). These macros provide SCI routines that process documents at the document processor. CPCS uses the 32-byte prefix area that these macros generate to check for correctness and compatibility of SCI programs during sort initialization. To initialize sorts that use these PROLOGs, you use the *in-core* SETDEV, which does not require loading of DKNMPSR during initialization. The DKNMPSR function is an imbed in the expanded format macros.

Users can code RMODE=ANY and AMODE=31 in the SCI routines, using expanded-format SCI macros to take advantage of MVS storage above 16MB during CPCS loading of the SCI routines. For additional information on SCI routines, see Figure 4-7 on page 4-46.

Loading a table for expanded format sorts requires the TBLSTRT2 and TBLSTRT4 macros. You do not have to code the ENTR macro after these macros; TBLSTRT2 and TBLSTRT4 macros generate the ENTR macro.

PROLOG Macro

Description: The PROLOG macro must be the first statement of the user's SCI program. It generates the CPCS PROLOG SCIs (part 2 of the standard SSP described before). You should code a label; no operands are necessary. The last statement generated by the PROLOG macro is the statement `DKNUSER EQU *`; the user stacker-select routine instructions start immediately after this statement.

Label	Operation	Parameters
LABEL	PROLOG	

PROEND Macro

Description: The PROEND macro must be the last statement of the user's SCI program. It delimits the SCI program so that CPCS can calculate the length of the generated SCI program, including the bytes of PROLOG. The label is optional.

Label	Operation	Parameters
LABEL	PROEND	

TBLSTART Macro

Description: You should code TBLSTART as the first statement of the user's SCI table. The table is a completely different module from the stacker-select routine.

The purpose of TBLSTART is to establish the load point of the table in SCI storage. The coded table usually contains SCI tables, but it can also include SCI routines and executable SCI instructions.

Label	Operation	Parameters
LABEL	TBLSTART	TBLLOAD=XXXX

The required operand TBLLOAD specifies the hexadecimal address in the sorter program storage at which the user wants to load the table. Code it as 3 or 4 hexadecimal characters without quotes. The address must be higher than the address of the end of the user stacker-select routine. It might be necessary to do trial assemblies of both the stacker-select routine and the table modules to determine the sizes and required load addresses. The user's SCIs should follow the TBLSTART macro.

TBLEND Macro

Description: The TBLEND macro must be the last statement of the user's SCI table. It delimits the SCI table so that CPCS can calculate the length of the generated SCI table. The label is optional.

Label	Operation	Parameters
LABEL	TBLEND	

Tables with Code or Binary Tables: If you want to include executable SCIs or to use binary tables in the CPCS SCI table, an ENTR macro must be the second statement of the table immediately following the TBLSTART macro. Code it in the subroutine form with the PGMLEN operand calculated as follows:

1. Subtract X'80' from the hexadecimal address specified in the TBLLOAD operand of the TBLSTART macro.
2. Convert the resulting hexadecimal number to decimal and code in the ENTR macro.

PROLOG2 Macro

Description: This is an expanded format PROLOG that uses 2-byte addressing. You must pair this macro with the PROEND macro for correct address resolution. PROLOG2 permits use of the *in-core* SETDEV by including the DKNMPSR function in the PROLOG. This PROLOG runs only on the 3890/XP Series document processors.

Label	Operation	Parameters
LABEL	PROLOG2	[(PGMSIZE={*nnnnn* | *nnK*})]

PGMSIZE=*nnnnn* **|** *nnK*

PGMSIZE=*nnnnn* is an optional parameter that indicates total SCI program size (PROLOG module + table). This parameter sets the PROLOG PGMSIZE parameter to the hexadecimal equivalent of the specified value. DKNMICR checks this parameter. PGMSIZE defaults to zero. If it is zero, CPCS ignores it. If present, PGMSIZE must be the same in the PROLOG2 or PROLOGX and the TBLSTRT2 or TBLSTRT4 macros. The actual memory used for the table and program cannot be more than the size specified in the operand.

PROLOGX Macro

Description: This is an expanded format PROLOG using 2- or 4-byte addressing. This PROLOG only runs on the 3890/XP Series document processors. The size of sorter storage required for this macro varies, depending on the addressing mode. (See Figure 4-4 on page 4-36.)

You must pair this macro with the PROEND macro for correct address resolution.

This macro performs these additional features:

- Generates a new ENTR macro:

  ```
  DKNMPSR ENTR SCI1=DKNSTART,SCIERR=DKNSCIE,EXT=Y,TYPE=SCI4
  ```

- Performs DKNUSER for divider documents (optional)
- Imbeds the ENTR macro
  - Generates the SCI error routine (SCIERR)
  - Generates the SCI code start address (DKNSTART) assembler
- Creates a table address list

  This macro creates a table that CPCS initialization programs fill with the address of the table. The user's SCI can refer to this table at label DKNTAL to locate the loaded table at sort-program run time.

- Supplies an endpoint table

- Supplies an SBTBL-format table that contains the endpoint ID for each pocket.

Label	Operation	Parameters
LABEL	PROLOGX	[TYPE={**SCI2 | SCI4**}] [PGMSIZE={**0** | *nnnnK* | *nnM*}] [DIVDOC={**YES | NO**}]

TYPE={SCI2 | SCI4}
This is an optional parameter that indicates SCI type. The default is SCI4.

PGMSIZE={0 | *nnnnK* | *nnM*}
This is an optional parameter that indicates total SCI program size (PROLOG module + table). The system uses this parameter to set the PGMSIZE parameter to the hexadecimal equivalent of the value specified. You specify the PGMSIZE parameter in decimal values of kilo (K) bytes or mega (M) bytes. DKNMICR checks this operand for consistency for all modules. PGMSIZE defaults to zero. If it is zero, CPCS ignores this parameter. If present, PGMSIZE must be the same in the PROLOG2 or PROLOGX and TBLSTRT2 or TBLSTRT4 macros. The actual memory used for the table and program cannot be more than the size specified in the operand.

DIVDOC={YES | NO}
This is an optional parameter that may be set to have dividers processed by the SCI routine user code. The default is NO.

PROLOGX does the following for DIVDOC=NO:

1. Flags the item as a merge document (header byte 8 = X'08').
2. Assigns the pocket corresponding to the pocket counter that has reached the limit.
3. Does not pass control to the user control area.

PROLOGX does the following for DIVDOC=YES:

1. Flags the item as a merge document (header byte 8 = X'08').
2. Assigns the system reject pocket (this is a temporary pocket code to be overlaid by the user code).
3. Flags non-merge-before-main dividers (as such, header byte 2 = X'80'). The user code must set this flag for successfully validated merge-before-main dividers.

   The following code could be used to identify merge-before-main documents:

```
          THI   8,X'08'               MERGE FEED DOC ?
          BCS   8,SCISTART            NO - DO NORMAL TEST
          IA    (0,2),498             GET MBM FLAG
          CI    (0,2),X'01'           MBM PROCESSING
          BCS   8,DOMBM               YES - DO MBM
          |
          |      <=========   REGULAR DIVIDER EDITING
          |
BADDIV    EQU   * <========   BRANCH TO IF EDITING FAILS
          FM
          BCS   15,DKNRTN             RETURN
GOODDIV   EQU   * <========   BRANCH TO IF EDITING PASSES
          SPC
          BCS   15,DKNRTN
DOMBM     EQU   *
          |
          |      <=========   MBM DIVIDER EDITING
          |
MBMBAD    EQU   *               SELECT REJECT POCKET IF BAD
          SPI   11
MBMGOOD   EQU   *              LEAVE PKT CODE AS-IS IF GOOD
          BCS   15,DKNRTN             RETURN
SCISTART  DS     0H
```

4. Passes control to the user area.

Endpoint ID Processing

The PROLOGX macro supplies endpoint ID information to the user-sort routine in the form of an SBTBL-format table. The table, which is located at the label DKNEPTBL, contains the endpoint ID for each kill pocket.

Example of Endpoint ID Table: The kill-pocket endpoint ID table (DKNEPTBL) in the PROLOGX portion of the SCI program is in SBTBL format. You can use this table to get the endpoint ID that is associated with a kill pocket. To obtain the endpoint ID, do the following:

1. Load the pocket number into the work register. The SPC instruction selected the pocket number; it is in header-byte 6.
2. Send an SBT instruction for the DKNEPTBL.

The SBT instruction returns a 2-digit pocket number and an 8-digit endpoint ID to the work register.

The DKNEPTBL table is indexed by pocket number in a module/pocket (M/P) format. The 36 table entries range from M/P 11 through M/P 66. For each kill pocket, the endpoint ID section of the table contains the 8-digit (4-byte) endpoint IDs. For each non-kill pocket, the endpoint ID section of the table contains X'AA'. Figure 4-6 is an example of a DKNEPTBL table.

Figure 4-6. Example of DKNEPTBL Table

```
DKNEPTBL    SBTBL    2,11AAAAAAAA,    M/P=1/1
                     12AAAAAAAA,      M/P=1/2
                     13AAAAAAAA,      M/P=1/3
                     1405300019,      M/P=1/4    (Kill pocket)
                     15AAAAAAAA,      M/P=1/5
                     1688888888,      M/P=1/6    (On-us pocket)
                     21AAAAAAAA,      M/P=2/1
                     ⋮
                     6105311004,      M/P=6/1    (Kill pocket)
                     66AAAAAAAA,      M/P=6/6    (Last table entry)
```

TBLSTRT2

Description: This macro is the first statement in the 2-byte SCI program's sort-pattern table routine. It creates a 32-byte prefix area based on user-coded operands. The CPCS initialization programs reference the prefix area to verify that the format and size of the user-generated program are correct.

You must pair this macro with a TBLEND macro. You do not have to code the ENTR macro after this macro; the TBLSTRT2 macro generates the ENTR macro.

Label	Operation	Parameters
LABEL	TBLSTRT2	TBLLOAD=*nnnnn* | *nn*K [,PGMSIZE={**0** | *nn*K}]

TBLLOAD=*nnnnn* | *nn***K**

The load address for the table, expressed in decimal (*nnnnn*) or in decimal increments of kilo (K) bytes. *nnnnn* must be a multiple of 8 (double-word boundary). This operand is required.

PGMSIZE={0 | *nn*K}

The size parameter, indicates total SCI program size (PROLOG module + table) in decimal increments of kilo (K) bytes. This parameter is optional. You must code the same value for each program and table in a sort.

Example:

```
LABEL TBLSTRT2 TBLLOAD=32768,PGMSIZE=48K
```

TBLLOAD=*nnnnn* is the decimal load address for the table and PGMSIZE=*nn*K is the decimal size for the PROLOG module and the table.

TBLSTRT4

Description: This macro is the first statement in the 4-byte SCI program sort-pattern table routine. It creates a 32-byte prefix area based on user-coded operands. The CPCS initialization programs reference the prefix area to verify that the format and size of the user-generated program are correct.

You must pair this macro with a TBLEND macro. You do not have to code the ENTR macro after this macro; the TBLSTRT4 macro generates the ENTR macro.

Label	Operation	Parameters
LABEL	TBLSTRT4	TBLLOAD=*nnnnn* | *nnnnK* | *nnM* [,PGMSIZE={**0** | *nnnnK* | *nnM*}]

TBLLOAD=*nnnnn* | *nnnnK* | *nnM*

The load address for the table, expressed in decimal (*nnnnn*) or in decimal increments of kilo (K) bytes or mega (M) bytes. The load address must be a multiple of 8 (double-word boundary). This operand is required.

PGMSIZE={0 | *nnnnK* | *nnM***}**

The size parameter, indicates total SCI program size (PROLOG module + all table) in decimal increments of kilo (K) or mega (M) bytes. This parameter is optional. If it is coded, it must be the same for each program and table in a sort.

Example: `LABEL TBLSTRT4 TBLLOAD=56000,PGMSIZE=256K`

Prefix Area Description

The expanded format CPCS SCI macros all generate the following 32-byte prefix. The CPCS initialization programs reference the prefix area to verify that the format and size of the user-generated program are correct.

```
+0       DC    XL1'01'          MACRO TYPE (01=TBLSTRTn)
                                           (02=PROLOG2)
                                           (04=PROLOGX)
+1       DC    XL1'01'          SCI INDICATOR (00=SCI2,01=SCI4)
+2       DC    XL2'0000'        RESERVED
+4       DC    AL4(TBLENDX-TBLNAME) LENGTH OF TABLE
+8       DC    AL4'nnnn'        LOAD POINT FOR THIS PROGRAM
+12      DC    AL4'0'           PGMSIZE
+16      DC    CL8'MM/DD/YY'    DATE OF ASSEMBLY (&SYSDATE)
+24TBLEN EQU   *                LABEL FOR TBLEND
+24      DC    CL8'HH.MM   '    TIME OF ASSEMBLY (&SYSTIME)
```

Sample Expanded-Format SCI Routine Using a Table

The following figure shows two different examples: PROLOGX4 and TBLSTRT4. They are separated by a ****** line.

```
          TITLE   'PROLOGX4 SAMPLE'
          AMODE   31
          RMODE   ANY
          PRINT   NOGEN
MX4SAMP1  PROLOGX TYPE=SCI4,PGMSIZE=256K,DIVDOC=YES
          PRINT   GEN
          *
          *
          CST    (0,1),KILLTB
          *
KILLTB    EQU     MX4SAMP1+MEM128K–IRECSI2   ADDRESS OF TABLE
MEM128K   DC      X'131072' VALUE OF 128K OF MEMORY
IRECS12   DC      X'128' VALUE OF SIZE OF IREC
LASTLABL  DC      CL30'128K PROLOGX4 PASS 1- MX4SAMP1'
         PROEND

****************************************************************

          TITLE   'TBLSTRT4 SAMPLE '
          AMODE    31
          RMODE    ANY
MT4SAMP1  TBLSTRT4 TBLLOAD=128K,PGMSIZE=256K
KILLTB    CSTBL    12,0                   CPCS POCKET 1
          CSTBL    13,1                   CPCS POCKET 2
          CSTBL    14,2                   CPCS POCKET 3
          CSTBL    15,3,4,5               CPCS POCKET 4
          CSTBL    16,6,7,8,9             CPCS POCKET 5
          CSTBL    11,T                   CPCS POCKET REJECT
LASTLABL  DC       CL30'128K TBLSTRT4 PASS 1- MT4SAMP1'
          TBLEND
```

Figure 4-7. Sample Expanded-Format SCI Routine and Table

Examples of SCI Routines Using Tables

Figure 4-8 and Figure 4-9 are examples of SCI routines.

Figure 4-8. Main SCI Program

Location		Source Statement
	MAIN	PROLOG
400		*
		*
		(SCIs)
		*
		CST (0,2),KILLTB
		*
		*
600	KILLTB	EQU MAIN+X'530'
		PROEND

Figure 4-9. SCI Table Program

```
                                SOURCE STATEMENT
TABLE                           TBLSTART TBLLOAD=600
                                ENTR PGMLEN=1408
                                *
                                *
                                (SCIs)
                                *
                                *
                                CSTBL 16,01
                                *
                                *
                                TBLEND
```

Host Codeline Edit Routines

A large portion of the user programming effort includes writing host codeline edit routines for MICR and host application processing. These routines determine the pocket to which documents are sent. Host codeline edit routines can also set the field validity bits for rejected documents, and receive control on all items except tracer documents.

Host codeline edit routines append the character R to module names before loading them. You are responsible for creating load modules with the R suffix. You can use aliases for host edit routines that are exactly the same. These aliases are useful if the stacker-select routines are different for each type or pass, but the host codeline edit routine processing is the same across sort-pattern types.

CPCS tasks OLRR and ADJ access the host codeline edit routines.

Your host codeline edit routine must be able to correct items sent to user-defined reject pockets and to the system reject pocket. If you do not want to correct a field in your application task, you must force your application task to accept the field that is not valid.

Document Buffer Area

The input data for host codeline edit routines consists of a document buffer area and a MICR user-parameter area (MUPA copybook). Register 1 points to the document buffer area when entry to the edit routine occurs. To obtain addressability for the document buffer area, code:

```
USING MRDBDSCT,1
:
:
DKNDSCT MRDBUF=YES
ORG   MRDIMG
DKNDSCT DI=YES,DSCTCD=NO
```

The document buffer area for the standard record contains the following information:

- Buffer status indicators

  The first 8 bytes of the document buffer area. Host codeline edit routines use only byte 4. The user's stacker-select routine must place the pocket-select decision into this byte. For example, module 1/pocket 1 is X'11' and module 6/pocket 6 is X'66'. These values are placed in the MRDBS4 field.

- CPCS codeline

  This area is mapped by copybook DIDCSCT.

- BSAM document input area.

If you change the record definition for the MDS, the sizes and displacements for the fields change accordingly.

Figure 4-10 describes the codeline data record of the document buffer area.

Buffer Status Indicators	
Byte	
0–3	Reserved
4	User Pocket Decision
5–7	Reserved
MDS Codeline Record	
8–57	MDS Codeline Record (for a standard MDS record)

Figure 4-10. Document Buffer Area Codeline Data Record

Setting Flags and Valid Field Indicators

The host codeline edit routine must set the following indicators:

1. Field DIFLAG1

   Equate
   DIUNPROC Unprocessed document, codeline data match disabled for this item
   DIMNS On if matched document, out of sequence
   DIMFR On if matched document, character/field repair

DIVAMT	On if amount field valid (user can reset)
DIVPCTL	On if process control field valid (user can reset)
DIVONUS	On if account field valid (user can reset)
DIVABA	On if routing number field valid (user can reset)
DIVAUX	On if serial number field valid (user can reset)

2. Field DIFLAG2

Equate	
DIMTCR	On if this is a control document
DIJAM	On if a jam occurred (user does not set)
DICAN	On if document is a Canadian check
DIOURS	On if document is on-us
DICRD	On if document is a credit (for example, a deposit slip)
DILOW	On if this document has been entered through online reject reentry (user does not set).
DIHIGH	On if this document has been entered through high-speed reject reentry (user does not set).

3. Field DITYPEI

   If flag DIMTCR is ON, you cannot set any flags in this field because this field represents the control document type. If flag DIMTCR is OFF, you can set any flags in this field.

4. Field DITYPEU

   This byte is available to the user.

5. Field DI3FLG3, byte DI3BYTE4

Equate	
DI3FL04V	On if field 4 is valid (user can reset)
DI3FL06V	On if field 6 is valid (user can reset)

6. Field DI3FLG3, byte DI3BYTE5

Equate	
DI3FL09V	On if field 9 is valid (user can reset)
DI3FL10V	On if field 10 is valid (user can reset)
DI3FL11V	On if field 11 is valid (user can reset)
DI3FL12V	On if field 12 is valid (user can reset)
DI3FL13V	On if field 13 is valid (user can reset)
DI3FL14V	On if field 14 is valid (user can reset)
DI3FL15V	On if field 15 is valid (user can reset)

Note: Note CPCS establishes field validity for fields 1, 2, 3, 5, and 7 from DIFLAG1 **only**. CPCS ignores attempts to change field validity for these fields using DI3BYTE4. However, it does maintain the settings for fields 1, 2, 3, 5, and 7 in DI3BYTE4 by copying them from DIFLAG1. For example, if you mark field 1 as not valid by turning DIVAMT off, CPCS turns DI3FL01V off as well.

Setting Pocket Numbers

Your host codeline edit routine should not use the CPCS pocket number. CPCS automatically takes the pocket code placed in buffer status byte 4 (MRDBS4), converts it, and places the result in the CPCS pocket number byte (DIPKT field).

If you identify a pocket as a reject pocket in your SPDEF, CPCS modifies the pocket code by adding X'80' to it. This identifies the pocket as a user-defined

reject pocket. For display purposes, the first digit of the pocket code is converted to an alphabetic character.

Figure 4-11 shows examples of this converted pocket number for the document processor.

Figure 4-11. Converted Pocket Number for the Document Processor

Buffer Status Byte (4)	Document Processor Pocket (Module/Pocket = MP)	Pocket # in CPCS for Good Pockets		Pocket # in CPCS for Reject Pockets	
		SDE	Report	SDE	Report
X'11'	Reject	Not Applicable		X'FF'	' '
X'12'	MP 1-2	X'01'	C'01'	X'81'	C'Z1'
X'13'	MP 1-3	X'02'	C'02'	X'82'	C'Z2'
X'14'	MP 1-4	X'03'	C'03'	X'83'	C'Z3'
X'15'	MP 1-5	X'04'	C'04'	X'84'	C'Z4'
X'16'	MP 1-6	X'05'	C'05'	X'85'	C'Z5'
X'21'	MP 2-1	X'06'	C'06'	X'86'	C'Z6'
X'22'	MP 2-2	X'07'	C'07'	X'87'	C'Z7'
X'23'	MP 2-3	X'08'	C'08'	X'88'	C'Z8'
X'24'	MP 2-4	X'09'	C'09'	X'89'	C'Z9'
X'25'	MP 2-5	X'0A'	C'10'	X'8A'	C'A0'
X'26'	MP 2-6	X'0B'	C'11'	X'8B'	C'A1'
X'31'	MP 3-1	X'0C'	C'12'	X'8C'	C'A2'
X'32'	MP 3-2	X'0D'	C'13'	X'8D'	C'A3'
X'33'	MP 3-3	X'0E'	C'14'	X'8E'	C'A4'
X'34'	MP 3-4	X'0F'	C'15'	X'8F'	C'A5'
X'35'	MP 3-5	X'10'	C'16'	X'90'	C'A6'
X'36'	MP 3-6	X'11'	C'17'	X'91'	C'A7'
X'41'	MP 4-1	X'12'	C'18'	X'92'	C'A8'
X'42'	MP 4-2	X'13'	C'19'	X'93'	C'A9'
X'43'	MP 4-3	X'14'	C'20'	X'94'	C'B0'
X'44'	MP 4-4	X'15'	C'21'	X'95'	C'B1'
X'45'	MP 4-5	X'16'	C'22'	X'96'	C'B2'
X'46'	MP 4-6	X'17'	C'23'	X'97'	C'B3'
⋮	⋮	⋮		⋮	
⋮	⋮	⋮		⋮	
X'61'	MP 6-1	X'1E'	C'30'	X'9E'	C'C0'
X'62'	MP 6-2	X'1F'	C'31'	X'9F'	C'C1'
X'63'	MP 6-3	X'20'	C'32'	X'A0'	C'C2'
X'64'	MP 6-4	X'21'	C'33'	X'A1'	C'C3'
X'65'	MP 6-5	X'22'	C'34'	X'A2'	C'C4'
X'66'	MP 6-6	X'23'	C'35'	X'A3'	C'C5'

Note: The remaining entries in the table are applicable only for electronic or simulated entries. Because there is no document processor, the first two columns are blank.

		X'24'	C'36'	X'A4'	C'C6'
		X'25'	C'37'	X'A5'	C'C7'
		X'26'	C'38'	X'A6'	C'C8'
		X'27'	C'39'	X'A7'	C'C9'
		X'28'	C'40'	X'A8'	C'D0'
		X'29'	C'41'	X'A9'	C'D1'
		⋮		⋮	
		⋮		⋮	
		X'62'	C'98'	X'E2'	C'I8'
		X'63'	C'99'	X'E3'	C'I9'

Host Codeline Edit Routine Register Conventions

Registers on entry to the host codeline edit routine are as follows:

0 Unpredictable
1 Address of the document buffer area
2 Address of the MICR user parameter area
3–12 Unpredictable
13 Address of the standard save area containing the contents of the registers to recover before return from the host codeline edit routine.
Host codeline edit routines must not save their registers in the save area pointed to by register 13. Because CPCS's registers have already been saved in this area, alteration of this save area can cause unexpected results.
You are responsible for recovering registers before return, and you must use register 13 to do so.
14 Return register
15 Host codeline edit routine entry point. You must set this register to zero before return.

All host codeline edit routines **must** be re-entrant. In addition, the linkage editor must assign the re-entrant attribute for the host codeline edit routines.

MICR User-Parameter Area (MUPA)

This parameter area contains information for your host codeline edit routine. Addressability is obtained by coding:

```
USING MUPADSCT,2
:
:
DKNMSCT MUPA=YES
```

The MUPA area is also used to communicate with the DKNMREAD document-processing user exit. Areas of the MUPA specific only to the DKNMREAD user exit are noted in the table below.

Field Name	Field Size	Content Value	Modified by	Field Content Description
MUPAENT	XL02		User	CPCS entry tracer-group number (binary)
MUPACYC	CL02	1 through 9 A through L	User	CPCS entry cycle number (Left justified, alphanumeric)
MUPATYPE	XL01		User	CPCS entry type of work (sort type, binary value 0 through 255, X'00' through X'FF')
MUPAPPH	XL04		User	Pass and pocket history (X'PP112233'), PP=X'00' through X'04'. X'00' if there are high-speed reject re-entry documents in this run. Pocket history: The prior-pass pocket distribution record of the documents in this run. Each byte contains a value of X'01' to X'99' that indicates the pocket that received the document in previous passes. X'11' represents the prime-pass pocket, X'22' represents the second-pass pocket, and X'33' represents the third-pass pocket.

Field Name	Field Size	Content Value	Modified by	Field Content Description
MUPASTEP	Fullword		DKNMOLRI	Contains the address of the optional host codeline edit table specified in MUPASTN. If no table is specified, contains binary zeroes.
MUPAUSER	XL16		User	Area reserved for user (initialized to binary zeros) The contents of this area remain constant from document to document and are available, for example, for program switches and addresses. This area is helpful for all re-entrant user interface routines. If more than 16 bytes of storage are needed, the expanded user work area can be used (see description for MUPAXWA on page 4-53).
MUPAFLGS	XL01			MUPA flag byte
MUPARST	Equate	B'100000000'	DKNMREAD	Restart of an entry in progress (MICR MREAD exit processing) Bit 0 = 1, restart in progress. This means that, on a restart, the MICR document processing routine is still reading from the I-string.
MUPAREX	Equate	B'010000000'	DKNMREAD	Expanded sorter record flag (MICR MREAD exit processing) Bit 1 = 1. The record from the document processor is larger than 44 bytes.
MUPADSPK	Equate	B'001000000'	DKNOLRR (or other user interfaces)	Do not let the host codeline edit routine set the pocket. This indicator is set to signal that the pocket decision must not be changed from the original one made during MICR capture.
MUPAXWA	XL03		CPCS	The address of an expanded user work area. It has the same attributes as the 16-byte area described at label MUPAUSER. This area is available if you coded the SSWORK operand in the CPCSOPTN macro during MICR generation. Loading this address into a register gives you access to this area. You can request up to 2040 bytes. The amount that you specify is allocated during each MICR entry or host codeline edit operation. When allocated, this area initializes to X'00'.
MUPASSN	CL08		DKNMOLRI	Host codeline edit routine name. DKNMOLRI determines this name by appending an R to the name passed passed in a CPCS-defined string header record.
MUPASTN	CL08		DKNMOLRI	Optional host codeline edit table name. DKNMOLRI determines this name by appending an R to the name passed passed in a CPCS-defined string header record.
MUPASSEP	Fullword		DKNMOLRI	Contains the entry-point address of the host codeline edit routine specified in MUPASSN.
MUPASPTR	Fullword		DKNMREAD	Contains the address of the sorter document-buffer area that MREAD uses to communicate with the MREAD user-exit routine.

User Stacker-Select Routine Operation for OLRR

A user stacker-select routine performs the editing of corrected documents during OLRR in a way similar to that of the sorter pass. You should write the user-supplied routines for OLRR document editing as described earlier in this chapter. (For more details, see “Host Codeline Edit Routines” on page 4-47.)

To ensure accurate editing of corrected reject documents, each reject D-string from prime pass, HSRR, or a subsequent pass contains a CPCS string-header record. DKNOLRR uses this information to obtain the following data:

1. The user’s stacker-select routine name for this pass (from the pass description record in the SPDEF library). This name must not contain more than 7 characters.
2. The user’s sort pattern-table name (if any).

DKNOLRR appends the character R to module names before loading them. You are responsible for creating load modules with the R suffix. You can use aliases for OLRR user-edit routines that are exactly the same. These aliases are useful if the stacker-select routines are different for each type or pass, but the OLRR processing is the same across sort pattern types.

Keep in mind that your stacker-select routine must be able to control processing for both your user-defined reject pockets and your system reject pocket.

After being loaded, items 1 and 2 explained above remain the same for the duration of processing. After the OLRR operator corrects a codeline, DKNOLRR branches to the module in item 1. In addition to the document data, the entry point of item 2 passes to the user. For example, you can edit the document as desired and set the validity flags.

Notes:

1. Since DKNOLRR re-displays codeline data records that the user-exit routine does not accept, the operator should have some way of forcing validity on a given field (for example, by filling the account number field with 9s). The user-edit routine must check for these indications. This permits you to control logical rejects that are indicated as reject pocket items.
2. The setting of the field validity bit in each codeline data record determines the corrective action required during online reject reentry. The setting refers to record flag 1 of the CPCS codeline data record, bits 3 through 7, and flag 3 byte 4, bits 4 and 6. Do not flag a field as not valid unless you need to have valid information in that field.
3. If a document amount field contains non-numeric data and it is selected to a non-reject pocket with the field flagged as valid, an abnormal end occurs for the current task. This is true for both the MICR and OLRR user-edit select routines.
4. CPCS reserves the sequence number and the processing unit bytes in the CPCS codeline data record; the user-edit program, either MICR or OLRR, must not change these.

Changing the SETDEV and Diagnostic Operation Time-Out Interval

The MICR program, DKNMPUTC, allows a SETDEV or diagnostic operation to the document processor to end abnormally if it does not complete in a specified time. You can change the time-out interval by modifying DKNMPUTC. The standard time-out interval for the SETDEV is:

Time-Out Interval	Document Processor
1 minute	SETDEV for 3890 Models A, B, and X
3 minutes	SETDEV for channel-attached 3890/XP Series document processors
5 minutes	SETDEV for all LU 6.2-attached 3890/XP Series document processors
5 seconds	Diagnostic operations.

Shown below are the DKNMPUTC constants with current settings. There are separate constants to permit different time-out intervals for different operations that MICR issues. You can set these constants as you want, but you should consider the following factors:

- A SETDEV interval that is too short does not permit normal completion of SETDEV processing.
- A SETDEV interval that is too long delays subsequent SETDEVs to other document processors when a SETDEV problem occurs.

Standard SETDEV Time-Out Interval: Modify the following constant in DKNMPUTC to set the time-out interval for all standard 3890 SETDEVs. All 3890 unexpanded (old format) sort programs use this SETDEV time-out interval.

```
TIME3890 DS    0D                TIME FOR STANDARD SETDEV
         DC    C'00'                HOUR HOUR
         DC    C'01'                MIN  MIN
         DC    C'00'                SEC  SEC
         DC    C'0'                 TENTH
         DC    Z'0'                 HUNDREDS
```

After modification, reassemble DKNMPUTC and link it to the correct CPCS program library.

Expanded SETDEV Time-Out Interval: Modify the following constant in DKNMPUTC for expanded SETDEVs. This SETDEV time-out interval is only for sort programs that are in expanded format (XF) and that process on 3890/XP Series document processors.

```
TIMEXF   DS    0D                TIME FOR EXPANDED (XF) SETDEV
         DC    C'00'                HOUR HOUR
         DC    C'01'                MIN  MIN
         DC    C'00'                SEC  SEC
         DC    C'0'                 TENTH
         DC    Z'0'                 HUNDREDS
```

After you complete the modification, reassemble DKNMPUTC and link it to the corresponding CPCS program library.

Diagnostic Operation Time-Out Interval: Modify the following constant in DKNMPUTC to set a different time-out interval.

```
TIMEDIAG DS    0D                TIME FOR DIAGNOSTIC OPERATION
         DC    C'00'                HOUR HOUR
         DC    C'00'                MIN  MIN
         DC    C'05'                SEC  SEC
         DC    C'0'                 TENTH
         DC    Z'0'                 HUNDREDS
```

After you complete the modification, reassemble DKNMPUTC and link it to the corresponding CPCS program library.

CPCS System Exit Points

CPCS provides several exit points for system related processing. These exit points are detailed in this section, as well as the user exit information relative to the system exit points.

All exits for these exit points are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNINIT_POST_DKNITASK_EXIT=*keyword* is used to specify an exit for this exit point. Up to the maximum number of entry points per exit may be specified for this exit point. See the "DKNPCPCS Profile Member" on page 3-4 located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or no more exist.

DKNALLO_EXIT1000_REQ_VALIDATION

This exit allows the further validation of the operator-entered request right after the validations have been successfully performed and before the actual document processor allocation/unallocation operation takes place.

Environment

This exit is given control within DKNALLO TCB.

Point of Processing

This exit is called after DKNALLO has validated the allocate or unallocate command issued by the operator.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNALLO_EXIT1000_REQ_VALIDATION=*keyword* is used to specify the exit for this exit point. Up to the maximum number of entry points per exit may be specified for this exit point. For details on the MAX_EP_EXIT parameter, refer to the CPCS System Profile (DKNPCPCS) in the SYSTPROF data set; in this manual, refer to "DKNPCPCS Profile Member" on page 3-4.

Each exit is called, in the order specified in the DKNPEXIT profile, until there are no more exits or until one of the exits denies the request with a return code.

Figure 4-12. DKNALLO_EXIT1000_REQ_VALIDATION Point Parameter List

Offset	Description
+00	Control block passed from DKNALLO (DKNCALL1 - DSECT)

Register Contents When Control Is Passed to the Exit Routine

R0	Not applicable
R1	Control block from DKNALLO
R2-R12	Not applicable
R13	Caller's save area address
R14	Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Not applicable

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example:

Member DKNXAL10, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNALLO_EXIT2000_MESSAGE

This exit allows the opportunity to interpret the message generated by DKNALLO and to send a more descriptive message explaining the problem or the ability to write a help screen created by the user. This exit is entered each time a message is generated by DKNALLO.

Environment

This exit is given control within DKNALLO TCB.

Point of Processing

This exit is called each time DKNALLO generates a message to be displayed.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNALLO_EXIT2000_MESSAGE=*keyword* is used to specify the exit for this exit point. Up to the maximum number of entry points per exit may be specified for this exit. For details on the MAX_EP_EXIT parameter for the CPCS system profile (DKNPCPCS) located in the SYSTPROF data set, see “DKNPCPCS Profile Member” on page 3-4.

Each exit is called in the order they were specified in the DKNPEXIT profile, until there are no more exits or until one of the exits denies the request with a return code.

Figure 4-13. DKNALLO_EXIT2000_MESSAGE Point Parameter List

Offset	Description
+00	Control block passed from DKNALLO (DKNCALL2 - DSECT)

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Control block from DKNALLO

R2-R12	Not applicable
R13	Caller's save area address
R14	Return address
R15	User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0	Not applicable
R1	Not applicable
R2-R13	Restored to same values as on user exit entry
R14	Not applicable
R15	Not applicable

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example:

Member DKNXAL20, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNATASK_APPLTSK_START_EXIT01

This exit allows you the opportunity to update any user extended area prior to starting an application task.

Environment

This exit is given control within DKNATASK TCB.

Point of Processing

This exit is called prior to the application task, ATTACH.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNATASK_APPLTSK_START_EXIT01=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-14. DKNATASK_APPLTSK_START_EXIT01 Point Parameter List

Offset	Description
+00	Caller's APTCB Address (DKNAPTCB)
+04	Communication Buffer Address (DKNCBUFF)
+08	Executive Task Information Address(EXINFO)
	Note: The EXINFO address is *NULLS* during the application task, ATTACH.

Register Contents When Control Is Passed to the Exit Routine

R0	Not applicable
R1	Parameter list address
R2-R12	Not applicable
R13	Caller's save area address
R14	Return address
R15	User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0	Not applicable
R1	Not applicable
R2-R13	Restored to same values as on user exit entry
R14	Not applicable
R15	Return codes as described below:
	xx Non-zero - exit denied ATTACH

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNATSKX, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNATASK_EXECTSK_START_EXIT01

This exit allows you the opportunity to update any user extended area prior to starting an executive task.

Environment

This exit is given control within DKNATASK TCB.

Point of Processing

This exit is called prior to the executive task, ATTACH.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNATASK_EXECTSK_START_EXIT01=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS), located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-15. DKNATASK_EXECTSK_START_EXIT01 Point Parameter List

Offset	Description
+00	Caller's APTCB Address (DKNAPTCB)
+04	Communication Buffer Address (DKNCBUFF)
+08	Executive Task Information Address(EXINFO)

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return codes as described below:

xx Non-zero - Exit Denied ATTACH

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNATSKX, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNATSK2_INITIALIZE_EXIT_01

This exit allows you the opportunity to update any user extended area after the APTR DCB has been opened.

Environment

This exit is given control within DKNATASK TCB.

Point of Processing

This exit is called after a successful open of the ATASK Print Log (APTR) DCB.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNATSK2_INITIALIZE_EXIT_01=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-16. DKNATSK2_INITIALIZE_EXIT_01 Point Parameter List

Offset	Description
+00	APTCB with PARMLST address ONLY

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Not applicable

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNATSKI, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNATSK2_TERMINATE_EXIT_01

This exit allows you the opportunity to update any user extended area after the APTR DCB has been closed.

Environment

This exit is given control within DKNATASK TCB.

Point of Processing

This exit is called after a successful CLOSE of the ATASK print log (APTR) DCB.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNATSK2_TERMINATE_EXIT_01=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until there are no more exits.

Figure 4-17. DKNATSK2_TERMINATE_EXIT_01 Point Parameter List

Offset	Description
+00	APTCB with PARMLST address ONLY

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Not applicable

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNATSKT, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNATSK2_WRITE_APTR_EXIT_01

This exit allows you the opportunity to update any user extended area after each WRITE to the APTR DCB.

Environment

This exit is given control within DKNATASK TCB.

Point of Processing

This exit is called after each successful WRITE of the ATASK Print Log (APTR) DCB.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNATSK2_WRITE_APTR_EXIT_01=*keyword* is used to specify an exit for this exit point. You may specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-18. DKNATSK2_WRITE_APTR_EXIT_01 Point Parameter List

Offset	Description
+00	APTCB with PARMLST address ONLY
+04	APTR Output Area Address

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Not applicable

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNATSKW, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNCYCDT_DATE_VALIDATE_EXIT

This exit allows the further validation of the cycle and/or endorse date before a cycle table modification takes place. DKNCYCDT does edit checking and calls DKNDATE; then the user is allowed to do further validation through this exit point.

Environment

This exit is given control within DKNCYCL TCB or the TCB that called the DKNCYCI cycle interface.

Point of Processing

This exit is called after DKNCYCDT has validated the cycle and/or endorse dates when called by the CYCL task or the cycle interface module DKNCYCI.

Activation

All exits for this exit point are specified in the CPCS system profile member (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNCYCDT_DATE_VALIDATE_EXIT=*keyword* is used to specify the exit for this exit point. Up to the maximum number of entry points per exit may be specified for this exit point. See the CPCS system profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter; in this manual, refer to “DKNPCPCS Profile Member” on page 3-4.

Each exit is called, in the order they were specified in the DKNPEXIT profile, until there are no more exits or until one of the exits denies the request with a return code.

Figure 4-19. DKNCYCDT_DATE_VALIDATE_EXIT Point Parameter List

Offset	Description
+00	Control block passed from DKNCYCDT (DKNCYCX1 - DSECT)

Register Contents When Control Is Passed to the Exit Routine

R0	Not applicable
R1	DKNCYCX1 conrol block from DKNCYCDT
R2-R12	Not applicable
R13	Caller’s save area address
R14	Return address
R15	User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0	Not applicable
R1	Not applicable
R2-R13	Restored to same values as on user exit entry
R14	Not applicable
R15	Return code

Other Programming Considerations

Notes:

1. A 20-byte user area is provided for the caller of DKNCYCDT and, subsequently, to the user exit through the DKNCYCX1 control block. Data may be passed by the caller, which the user exit may process. When DKNCYCL is the caller, the 20 bytes of user data are obtained from the last input parameter (must be last) with the keyword of ',USER=.....'. For more information about the CYCL task start parameters, see the DKNCYCL section in the *CPCS Terminal Operations Guide*.
2. The CPCS sample user exit DKNCYCLX uses a user data parameter of OVERRIDE to bypass all date validation processing in the user exit.
3. A return code in R15 of X'14' or decimal '20' shall indicate that the user exit rejected the specified date(s). A 32-character message may be extracted from the DKNCYCX1 control block placed there by the user exit.

Example:

Member DKNCYCLX, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNINIT_POST_DKNITASK_EXIT

This exit allows the user the opportunity to initialize any user control block. Note that, at the time these exits are called, all programs or service associated with CPCS are *NOT* available.

Environment

This exit is given control within DKNINIT TCB.

Point of Processing

These exits are called after DKNITASK has successfully completed.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNINIT_POST_DKNITASK_EXIT=*keyword* is used to specify an exit for this exit point. You may specify up to the maximum number of entry points per exit for this exit point. (See the CPCS system profile DKNPCPCS, located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called, in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-20. DKNINIT_POST_DKNITASK_EXIT Point Parameter List

Offset	Description
+00	APTCB with PARMLST address ONLY

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return codes as described below:

xx Non-zero - system shutdown

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNINIT1, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNLOADR_EXIT_01

This exit allows you the opportunity to request that a different program be loaded and executed.

Environment

This exit is given control within DKNLOADR TCB.

Point of Processing

This exit is called prior to the actual load of the BLDL task.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNLOADR_EXIT_01=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-21. DKNLOADR_EXIT_01 Point Parameter List

Offset	Description
+00	Caller's APTCB Address (DKNAPTCB)
+04	Task Start Parm Address (TCBZB)
+08	Caller's APCB Address (DKNAPCB)
+12	Address of the Alternative Program Name to be Executed

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return codes as described below:

xx Non-zero - Exit denied load

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNLDRX1, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNMBEGN_APPLTSK_START_EXIT01

This exit allows you the opportunity to update any user extended area after the APTCB is built.

Environment

This exit is given control within DKNMICR TCB.

Point of Processing

This exit is called after the APTCB is built in MICR at MBEGN time.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNMBEGN_APPLTSK_START_EXIT01=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-22. DKNMBEGN_APPLTSK_START_EXIT01 Point Parameter List

Offset	Description
+00	Caller's APTCB Address (DKNAPTCB)

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return codes as described below:

xx Non-zero - Exit Denied ATTACH

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNATSKX, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNMTASK_INITIALIZE_EXIT_01

This exit allows you the opportunity to initialize any user control block after all tasks have been successfully initialized and attached.

Environment

This exit is given control within DKNMTASK TCB.

Point of Processing

This exit is called after all major subtasks have been started and prior to DKNVTASK accepting logons.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNMTASK_INITIALIZE_EXIT_01=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS system profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-23. DKNMTASK_INITIALIZE_EXIT_01 Point Parameter List

Offset	Description
+00	APTCB with PARMLST address ONLY

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return codes as described below:

xx Non-zero - system shutdown

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNMTSKI, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNMTASK_MICR_ABEND_EXIT0100

This exit allows you the opportunity to perform any user processing after a MICR abend.

Environment

This exit is given control within DKNMTASK TCB.

Point of Processing

This exit is called after all DKNMTASK MICR abend processing has been completed.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNMTASK_MICR_ABEND_EXIT0100=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-24. DKNMTASK_MICR_ABEND_EXIT0100 Point Parameter List

Offset	Description
+00	APTCB with PARMLST address ONLY

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Not applicable.

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNMTSKA, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNMTASK_TERMINATION_EXIT_01

This exit allows you the opportunity to perform any user clean up before all the major subtasks are terminated prior to shutdown.

Environment

This exit is given control within DKNMTASK TCB.

Point of Processing

This exit is called after System Manager has been posted to shut down and before other major subtasks have been terminated.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNMTASK_TERMINATION_EXIT_01=*keyword* is used to specify an exit for this exit point. You can specify up to the maximum number of entry points per exit for this exit point. (See the CPCS System Profile (DKNPCPCS) located in the SYSTPROF data set for details on the MAX_EP_EXIT parameter.)

Each exit is called in the order specified in the DKNPEXIT profile, until one of the exits denies the request with a return code or until no more exits exist.

Figure 4-25. DKNMTASK_TERMINATION_EXIT_01 Point Parameter List

Offset	Description
+00	APTCB with PARMLST address ONLY

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0	Not applicable
R1	Not applicable
R2-R13	Restored to same values as on user exit entry
R14	Not applicable
R15	Not applicable.

Other Programming Considerations

Note: IBM requires this exit be re-entrant.

Example

Member DKNMTSKT, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

Customizing ETDM User Exits

Two user exits are available for use as part of the electronic data matching function:

1. ETDM User Exit One–Adjust Matching Criteria
2. ETDM User Exit Two–Write Reconciliation Item

ETDM User Exit One–Adjust Matching Criteria

ETDM user exit one is called after an attempt has been made to match a block of process items with a block of master items. The user exit can adjust the matching criteria and can attempt another match of the remaining unmatched items or, if the user exit does not adjust the matching criteria, the matching is considered complete.

It is only possible to attempt up to five match attempts on the items. Consequently, the user exit is called a maximum of four times (one match uses the profile criteria and four further matches may be attempted with adjusted matching criteria).

A different ETDM user exit one routine may be specified for each financial institution. The user exit is called by the ETDM task after each match attempt if there are unmatched items remaining.

To initiate this user exit point, set the following card in the DKNPEXIT member on the system profile data set:

```
DKNETDM_ADJUST_MATCH_CRITERIA=xxxxxxxx
```

If a new user exit is installed, the user exit must be refreshed using the CPCS EXIT task.

The standard OS/390 linkage conventions are followed.

Linkage: Information is passed in the following interface control blocks:

Figure 4-26. ETDM User Exit One Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Application Task Control Block	DKNCATCB	DKNAPTCB
ETDM Profile	DKNETMP2	DKNETMP1
User Exit One Parameter List	DKNETM12	DKNETM11

Restrictions: ETDM user exit one routines must be reentrant and reusable.

Example 1

Operation:

This sample exit does the following processing. If this is the first rematch attempt, it removes the amount field from the matching criteria. If this is not the first match attempt, it leaves the matching criteria as is, resulting in the match being terminated for the current block of work.

User Exit One Routine:

```
       ID DIVISION.
       PROGRAM-ID. DKNETMU1.
       ENVIRONMENT DIVISION.
       DATA DIVISION.

       WORKING-STORAGE SECTION.

       LINKAGE SECTION.

           COPY DKNCATCB.

           COPY DKNETM12.

           COPY DKNETD42.


       PROCEDURE DIVISION USING APTCB
                                DKNETMU1-PARMS
                                ETDM-REC.
         1000-INIT.
              IF ETDM1-MATCH-COUNT = 1
                 MOVE SPACES TO ETDM-FIELD01
              END-IF

         1090-END-DKNETMU1.

              GOBACK.
```

Figure 4-27. ETDM User Exit One - Example 1

ETDM User Exit Two–Write Reconciliation Item

ETDM user exit two is called before a reconciliation item is written to the mass data set. Each item passed is either flagged as matched, free (unmatched item on the process string), or missing (unmatched item on the master string).

The ETDM task constructs a reconciliation item using the data from the process item and the master item (for matched items), from the process item (for free items), and from the master item (for missing items).

If the item is matched, the user exit is passed a copy of the reconciliation item, the process item, and the master item. If the item is free, the user exit only receives a copy of the reconciliation item and the process item; if the item is missing, the user exit is passed a copy of the reconciliation item and the master item.

The user exit can then use the process and master item information passed to modify the reconciliation record before it is written to the mass data set.

Note: If the item is marked as free, the user exit should not attempt to use the master record. If the record is marked as missing, the user exit should not attempt to use the process item.

A different ETDM user exit two routine may be specified for each financial institution. The user exit is called by the ETDM task before each item is written to the mass data set.

To initiate this user exit point, set the following card in the DKNPEXIT member on the system profile data set:

```
DKNETDM_RECON_RECORD_WRITE=xxxxxxxx
```

If a new user exit is installed, the user exit must be refreshed using the CPCS EXIT task.

The standard MVS/ESA linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-28. ETDM User Exit Two Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Application Task Control Block	DKNCATCB	DKNAPTCB
ETDM Profile	DKNETMP2	DKNETMP1
User Exit Two Parameter List	DKNETM22	DKNETM21

Restrictions: ETDM user exit two routines must be reentrant and reusable.

Example 1: **Operation:**

This sample exit demonstrates how to address the various records passed to the user exit, for each of the match conditions.

User Exit Two Routine:

```
          ID DIVISION.
          PROGRAM-ID. DKNETMU2.
          ENVIRONMENT DIVISION.
          DATA DIVISION.

          WORKING-STORAGE SECTION.

          LINKAGE SECTION.

          COPY DKNCATCB.

          COPY DKNETMP2.

          COPY DKNETM22.

          PROCEDURE DIVISION USING APTCB
                                   DKNETMU2-PARMS
                                   ETDM-REC.
```

Figure 4-29 (Part 1 of 2). ETDM User Exit Two - Example 1

```
          0000-MAINLINE.
              IF  ETMU2-MISSING-ITEM
                  PERFORM  2000-MISSING-ITEM-LOGIC
              ELSE
                  IF  ETMU2-FREE-ITEM
                      PERFORM  3000-FREE-ITEM-LOGIC
                  ELSE
                      PERFORM  4000-MATCHED-ITEM-LOGIC
                  END-IF
              END-IF

              MOVE WS-RETURN-CODE TO RETURN-CODE
              GOBACK.
          2000-MISSING-ITEM-LOGIC.

              SET ADDRESS OF DIMAST  TO ETMU2-MASTER-MDS
              SET ADDRESS OF DIRECON TO ETMU2-RECON-MDS

              MOVE 0 TO WS-RETURN-CODE.


          3000-FREE-ITEM-LOGIC.

              SET ADDRESS OF DIPROC  TO ETMU2-PROCESS-MDS
              SET ADDRESS OF DIRECON TO ETMU2-RECON-MDS

              MOVE 0 TO WS-RETURN-CODE.

          4000-MATCHED-ITEM-LOGIC.

              SET ADDRESS OF DIPROC  TO ETMU2-PROCESS-MDS
              SET ADDRESS OF DIMAST  TO ETMU2-MASTER-MDS
              SET ADDRESS OF DIRECON TO ETMU2-RECON-MDS

              MOVE 0 TO WS-RETURN-CODE.
```

Figure 4-29 (Part 2 of 2). ETDM User Exit Two - Example 1

Customizing ETIO User Exits

Six user exits are available for use as part of the electronic transaction input/output function:

1. ETOT User Exit One–Transmission Data Set User Exit
2. ETOT User Exit Two–Write Item User Exit
3. ETIN User Exit One–Read Item User Exit
4. ETIN User Exit Two–Write Item User Exit
5. ETIP User Exit One–Input Processor Messaging
6. ETIP User Exit Two–Input Processor Selection

ETOT User Exit One — Transmission Data Set User Exit

The ETOT user exit one is called after each transmission data set is created. The user can then decide whether or not the transmission data set should be written to the electronic control file.

Important!

Inserting a record into the electronic control file causes the ETIN task to import the associated electronic transmission data set. Therefore, the financial institution should only insert records onto the electronic control file for transmission data sets that are to be imported back into an in-house CPCS system.

Activation

A different ETOT user exit one routine may be specified for each financial institution. The exit's name is DKNETOU1, and it is called by the ETOT task each time a transmission data set is created.

No regenerations, CPCS restarts, or CHAPs are required when a new ETOT user exit one routine is installed.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-30. ETOT User Exit One Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Application Task Control Block	DKNCATCB	DKNAPTCB
Call Parameters List	DKNETO12	DKNETO11
ETOT Profile Record	DKNETOP2	DKNETOP1

The exit routine must decide on the action that DKNETOT should perform by setting the ETO1-USER-RESPONSE flag within the call parameters control block.

Restrictions

ETOT user exit routines must be reentrant and reusable.

Example 1

Operation:

This sample exit uses the capture information supplied within the call parameter list to build the import information. The ETOT task uses this information to construct the electronic control file record that is used to import the transmission data set.

Processing Description

- Set the import profile name as a constant.
- Set the import information using the supplied capture information.
- Instruct ETOT to write a record to the electronic control file.

User Exit One Routine:

```
       ID DIVISION.
       PROGRAM-ID. DKNETOU1.
       ENVIRONMENT DIVISION.
       DATA DIVISION.

       WORKING-STORAGE SECTION.

       01  WS-WORK-DATA.
           03  WS-PROFILE-MEM            PIC X(08)     VALUE
               'ETINSAMP'.

       LINKAGE SECTION.

           COPY DKNCATCB.

           COPY DKNETO12.
           COPY DKNETOP2.

       PROCEDURE DIVISION USING APTCB,
                                ETO1-USER-EXIT-PARMS,
                                ETOT-PROFILE-RECORD.

       0000-MAIN SECTION.

           PERFORM 1000-SET-IMPORT-DATA

           SET ETO1-WRITE-CONTROL-RECORD TO TRUE

           GOBACK.
```

Figure 4-31 (Part 1 of 2). ETOT User Exit One - Example 1

```
       1000-SET-IMPORT-DATA SECTION.

           MOVE WS-PROFILE-MEM        TO ETO1-IMP-PROFILE-MEM
           MOVE ETO1-CAP-SYSTEM       TO ETO1-IMP-SYSTEM
           MOVE ETO1-CAP-CYCLE-NUM    TO ETO1-IMP-CYCLE-NUM
           MOVE ETO1-CAP-BANK-NUMBER  TO ETO1-IMP-BANK-NUMBER
           MOVE ETO1-CAP-SORT-TYPE    TO ETO1-IMP-SORT-TYPE

           MOVE SPACES                TO ETO1-IMP-STRING-NAME.
```

Figure 4-31 (Part 2 of 2). ETOT User Exit One - Example 1

ETOT User Exit Two — Write Item User Exit

The ETOT user exit two is called before each item is written to the transmission data set. The user exit has the opportunity to instruct ETOT to delete the item, to write the item, to write the item and resend it (insert an item), or to terminate the processing.

Any modifications the user exit may make to an item is permanent and is reflected on the transmission data set.

Note

It is only possible to 'write and resend' item records; any attempt to 'write and resend' header or trailer records results in a bad response message, and the task terminates.

Activation

A different ETOT user exit two routine may be specified for each financial institution. The exit's name is DKNETOU2, and it is called by the ETOT task each time an item is to be written to the transmission data set.

No regenerations, CPCS restarts, or CHAPs are required when a new ETOT user exit two routine is installed.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-32. ETOT User Exit Two Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Application Task Control Block	DKNCATCB	DKNAPTCB
Call Parameters List	DKNETO22	DKNETO21

The exit routine must decide on the action that DKNETOT should perform by setting the ETO2-USER-RESPONSE flag within the call parameters control block.

Restrictions

ETOT user exit routines must be reentrant and reusable.

Example 1

Operation:

This sample exit demonstrates how to interpret the various record types the user exit can receive. The sample user exit writes the default header and trailer records, but only writes certain item records.

Processing Description

- Ensure we have a valid record address.
- Evaluate the type of record we have.
- If the record is a header record, then write it.
- If the record is a trailer record, then write it.
- If the record is an item, then only write MICR documents, dividers, tracers, and kill bundle records.

User Exit Two Routine:

```
       ID DIVISION.
       PROGRAM-ID. DKNETOU2.
       ENVIRONMENT DIVISION.
       DATA DIVISION.

       WORKING-STORAGE SECTION.

       01  WS-WORK-DATA.
           03  WS-PROFILE-MEM            PIC X(08)     VALUE
               'ETINSAMP'.

       LINKAGE SECTION.

           COPY DKNCATCB.

           COPY DKNETO22.
           COPY DKNCRZC.
           COPY DKNETFL2 REPLACING == TAG == BY == ETIO ==.
```

Figure 4-33 (Part 1 of 2). ETOT User Exit Two - Example 1

```
       PROCEDURE DIVISION USING APTCB,
                                ETO2-USER-EXIT-PARMS,

       0000-MAIN SECTION.

           IF ETO2-RECORD-ADDRESS = NULL THEN
              SET ETO2-TERMINATE-PROCESSING TO TRUE
           ELSE
              EVALUATE TRUE
                WHEN ETO2-ITEM-RECORD
                     PERFORM 1000-PROCESS-ITEM-RECORD
                WHEN ETO2-HEADER-RECORD
                     PERFORM 2000-PROCESS-HEADER-RECORD
                WHEN ETO2-TRAILER-RECORD
                     PERFORM 3000-PROCESS-TRAILER-RECORD
              END-EVALUATE
           END-IF

           GOBACK.

       1000-PROCESS-ITEM-RECORD.
           SET ADDRESS OF ZC TO ETO2-RECORD-ADDRESS

      *
      * IF THE ITEM IS A MICR DOCUMENT, TRACER OR KILL BUNDLE
      * DOCUMENT THEN WRITE IT OTHERWISE, DELETE IT.
      *

           IF ZC-DOCU-TYPE = (X'00' OR X'F6' OR X'F7' OR X'F8')
              SET ETO2-WRITE-RECORD  TO TRUE
           ELSE
              SET ETO2-DELETE-RECORD TO TRUE
           END-IF.

           2000-PROCESS-HEADER-RECORD.
               SET ADDRESS OF ETIO-ELECFILE-HEADER TO ETO2-RECORD-ADDRES

               SET ETO2-WRITE-RECORD TO TRUE.

          ***************************************************************
          **
          ** FUNCTION = PROCESS THE ELECTRONIC TRANSACTION FILE TRAILER
          **
          ***************************************************************
           3000-PROCESS-TRAILER-RECORD.
               SET ADDRESS OF ETIO-ELECFILE-TRAILER TO ETO2-RECORD-ADDRE

               SET ETO2-WRITE-RECORD TO TRUE.
```

Figure 4-33 (Part 2 of 2). ETOT User Exit Two - Example 1

ETIN User Exit One — Read Item User Exit

The ETIN user exit one is called after each item is read from the transmission data set. The user exit has the opportunity to instruct ETIN to delete the item, to write the item, or to terminate the processing.

The exit gives the institution both the opportunity to reformat any non-standard transmission data set records into compressed MDS records and also to perform additional processing on an item before it has been modified by any of ETIN's internal processing.

If the ETS product is installed, the user exit receives the address of the global user area into which user-specific data can be inserted. If the ETS product is not installed, the global user area address passed is zero.

Activation

A different ETIN user exit one routine may be specified for each financial institution. The exit's name is DKNETIU1, and it is called by the ETIN task each time an item is read from the transmission data set.

No regenerations, CPCS restarts, or CHAPs are required when a new ETIN user exit one routine is installed.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-34. ETIN User Exit One Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Application Task Control Block	DKNCATCB	DKNAPTCB
Call Parameters List	DKNETI12	DKNETI11

The exit routine may return the following condition codes in the RETURN-CODE special register (COBOL programs) or register 15 (assembler programs).

Figure 4-35. ETIN Exit One Routine Return Codes

Code (dec)	Request
+00	Process this record
+04	Do NOT process this record
+08	Terminate ETIN processing

Restrictions

ETIN user exit one must be reentrant and reusable.

Example 1

Operation:

This sample exit only accepts MICR items for processing.

Processing Description

- Get the control blocks passed by ETIN.
- Set up addressability for all the relevant records.
- If this is the first time, set the first-time flag.
- Check if the item is not a MICR record, then delete it; otherwise, accept the item for processing.

User Exit One Routine:

```
DKNETIU1 CSECT
DKNETIU1 AMODE 31
DKNETIU1 RMODE ANY
         SAVE  (14,12),,DKNETIU1*V01R01*??/??/??*&SYSDATE*&SYSTIME
         LR    R12,R15                 LOAD BASE REGISTER
         USING DKNETIU1,R12            TELL ASSEMBLER
```

Figure 4-36 (Part 1 of 3). ETIN User Exit One - Example 1

```
*********************************************************************
*   PICK UP INPUT PARMS                                             *
*********************************************************************
         LM    R6,R7,0(R1)             LOAD PARM ADDRESS
         USING PARMAREA,R7             MAP PARAMETER AREA
*
         L     R8,ETIN1HD@             GET HEADER RECORD ADD
         L     R9,ETIN1DI@             GET MDS    RECORD ADD
         L     R10,ETIN1PF@            GET PROFILE CONTROL BLK ADD
         LA    R11,ETIN1WKA            ADDRESS USER EXITS WORK AREA

         USING ETINHDR,R8              MAP HEADER RECORD
         USING DIDSCT,R9               MAP MDS RECORD
         USING PCBDSCT,R10             MAP PROFILE CONTROL BLOCK
         USING USERWORK,R11            MAP USER EXIT WORK AREA
*
         ST    R6,APTCB@               SAVE APTCB ADDRESS
*
         CLI   FIRST_TIME,X'00'        IS THIS THE FIRST TIME
         BNE   B10000                  NO, SO CONTINUE
*
         MVI   FIRST_TIME,X'FF'        SET FIRST TIME OFF

B10000   EQU   *
*
         CLI   DITYPEI,X'00'           IS THE RECORD A MICR DOC
         BNE   B99004                  NO - SO DELETE IT
*********************************************************************
*   EXIT WITH RETURN CODE ZERO - PROCESS SUCCESSFUL - CONTINUE      *
*********************************************************************
B99000   DS    0H
         SLR   R15,R15                 ZERO RETURN CODE
         B     B99999                  GO TO RETURN
*********************************************************************
*   EXIT WITH RETURN CODE FOUR - ERROR DETECTED - CONTINUE RUN BUT  *
*   DO NOT WRITE THIS ITEM TO OUTPUT STRING                         *
*********************************************************************
B99004   DS    0H
         LA    R15,4                   SET RETURN CODE
         B     B99999                  GO TO RETURN
*********************************************************************
*   EXIT WITH RETURN CODE EIGHT - ERROR DETECTED - CANCEL RUN       *
*********************************************************************
B99008   DS    0H
         LA    R15,8                   SET RETURN CODE
*********************************************************************
*   RETURN TO DKNETIN                                               *
*********************************************************************
B99999   DS    0H
*
         L     R13,WKSVAREA+4
         XRETURN (14,12),,RC=(15)      ;END OF ETIN PROCESSING
```

Figure 4-36 (Part 2 of 3). ETIN User Exit One - Example 1

```
*************************************************************************
*   C O N S T A N T S                                                   *
*************************************************************************
*
USERWORK DSECT
WORKAREA DS    0CL150                 MAP FOR USER EXIT WORK AREA
*
FIRST_TIME DS  CL001
APTCB@   DS    F
         DS    CL145
*
ETINHDR  DSECT
         COPY  DKNETFL1               MAP FOR HEADER RECORD
*
DIDSCT   DSECT
         COPY  DIDSCT                 MAP FOR MDS RECORD
*
PCBDSCT  DSECT
         COPY  DKNETIP1               COPY FOR PROFILE CONTROL BLK
PCBDSCTL EQU   *-PCBDSCT
*
PARMAREA DSECT
         COPY  DKNETI11
         END
```

Figure 4-36 (Part 3 of 3). ETIN User Exit One - Example 1

ETIN User Exit Two — Write Item User Exit

The ETIN user exit two is called before an item is written to the CPCS mass data set string. The exit has the opportunity to instruct ETIN to delete the item, to write the item, or to terminate the processing.

The exit gives the institution the opportunity to modify and check any item before it is written to the mass data set.

If this function is installed, the user exit receives the address of the global user area into which user-specific data can be inserted. If this function is not installed, the global user area address passed is zero.

Activation

A different ETIN user exit two routine may be specified for each financial institution. The exit's name is DKNETIU2, and it is called by the ETIN task each time an item is about to be written to the mass data set.

No regenerations, CPCS restarts, or CHAPs are required when a new ETIN user exit one routine is installed.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-37. ETIN User Exit Two Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Application Task Control Block	DKNCATCB	DKNAPTCB
Call Parameters List	DKNETI22	DKNETI21

The exit routine may return the following condition codes in the RETURN-CODE special register (COBOL programs) or register 15 (assembler programs).

Figure 4-38. ETIN Exit Two Routine Return Codes

Code (dec)	Request
+00	Process this record
+04	Do NOT process this record
+08	Terminate ETIN processing

Restrictions

ETIN user exit two must be reentrant and reusable.

Example 1

Operation:

This sample exit accepts all MICR items for writing to the mass data set.

Processing Description

- Get the control blocks passed by ETIN.
- Set up addressability for all the relevant records.
- If this is the first time, set the first-time flag.
- Mark the item for writing to the mass data set.

User Exit Two Routine:

```
DKNETIU2 CSECT
DKNETIU2 AMODE 31
DKNETIU2 RMODE ANY
         SAVE  (14,12),,DKNETIU2*V01R01*??/??/??*&SYSDATE*&SYSTIME
         LR    R12,R15                 LOAD BASE REGISTER
         USING DKNETIU2,R12            TELL ASSEMBLER
```

Figure 4-39 (Part 1 of 3). ETIN User Exit Two - Example 1

```
*********************************************************************
*   PICK UP INPUT PARMS                                             *
*********************************************************************
         LM    R6,R7,0(R1)             LOAD PARM ADDRESS
         USING PARMAREA,R7             MAP PARAMETER AREA
*
         L     R8,ETIN2HD@             GET HEADER RECORD ADD
         L     R9,ETIN2DI@             GET MDS    RECORD ADD
         L     R10,ETIN2PF@            GET PROFILE CONTROL BLK ADD
         LA    R11,ETIN2WKA            ADDRESS USER EXITS WORK AREA

         USING ETINHDR,R8              MAP HEADER RECORD
         USING DIDSCT,R9               MAP MDS RECORD
         USING PCBDSCT,R10             MAP PROFILE CONTROL BLOCK
         USING USERWORK,R11            MAP USER EXIT WORK AREA
*
         ST    R6,APTCB@               SAVE APTCB ADDRESS
*
         CLI   FIRST_TIME,X'00'        IS THIS THE FIRST TIME
         BNE   B10000                  ;NO, SO CONTINUE
*
         MVI   FIRST_TIME,X'FF'        ;SET FIRST TIME OFF

B10000   EQU   *
*********************************************************************
*   EXIT WITH RETURN CODE ZERO - PROCESS SUCCESSFUL - CONTINUE      *
*********************************************************************
B99000   DS    0H
         SLR   R15,R15                 ;ZERO RETURN CODE
         B     B99999                  ;GO TO RETURN
*********************************************************************
*   EXIT WITH RETURN CODE FOUR - ERROR DETECTED - CONTINUE RUN BUT  *
*   DO NOT WRITE THIS ITEM TO OUTPUT STRING                         *
*********************************************************************
B99004   DS    0H
         LA    R15,4                   SET RETURN CODE
         B     B99999                  GO TO RETURN
*********************************************************************
*   EXIT WITH RETURN CODE EIGHT - ERROR DETECTED - CANCEL RUN       *
*********************************************************************
B99008   DS    0H
         LA    R15,8                   SET RETURN CODE
*********************************************************************
*   RETURN TO DKNETIN                                               *
*********************************************************************
B99999   DS    0H
*
         L     R13,WKSVAREA+4
         XRETURN (14,12),,RC=(15)      ;END OF ETIN PROCESSING
```

Figure 4-39 (Part 2 of 3). ETIN User Exit Two - Example 1

```
*************************************************************************
*   C O N S T A N T S                                                   *
*************************************************************************
*
USERWORK DSECT
WORKAREA DS    0CL150                 MAP FOR USER EXIT WORK AREA
*
FIRST_TIME DS  CL001
APTCB@   DS    F
         DS    CL145
*
ETINHDR  DSECT
         COPY  DKNETFL1               MAP FOR HEADER RECORD
*
DIDSCT   DSECT
         COPY  DIDSCT                 MAP FOR MDS RECORD
*
PCBDSCT  DSECT
         COPY  DKNETIP1               COPY FOR PROFILE CONTROL BLK
PCBDSCTL EQU   *-PCBDSCT
*
PARMAREA DSECT
         COPY  DKNETI21
         END
```

Figure 4-39 (Part 3 of 3). ETIN User Exit Two - Example 1

ETIP User Exit One — Input Processor Messaging

The ETIP user exit one is called at the end of ETIP processing; it is called to give the financial institution the opportunity to issue an end-of-task message detailing any error condition that may have been identified. The user exit is supplied a set of counts detailing the number of eligible, complete, and queued records processed during this run of ETIP. The task is also passed a start switch detailing the type of ETIP run that has occurred.

Activation

A different ETIP user exit one routine may be specified for each financial institution. The exit is called by the ETIP task at the end of processing.

No regenerations, CPCS restarts, or CHAPs are required when a new ETIP exit one routine is installed.

To initiate this user exit point, set the following card in the DKNPEXIT member on the system profile data set:

```
DKNETIP_ANALYSE_RECS_PROCESSED=xxxxxxxx
```

If a new user exit is installed, the user exit must be refreshed using the CPCS EXIT task.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-40. ETIP User Exit One Communication Control Blocks

Program counts	Control Block Copybook	Assembler Macro/ Copybook
Control Record Counts	N/A	N/A
Program Start Switch	N/A	N/A
System Message Area	N/A	N/A

The exit routine may return the following condition codes in the RETURN-CODE special register (COBOL programs) or register 15 (assembler programs).

Figure 4-41. ETIP Exit One Routine Return Codes

Code (dec)	Request
+00	Do not issue a system message
+04	Issue a system message using the message built in the message area

Restrictions

ETIP user exit one routines must be reentrant and reusable.

Example 1

Operation:

This sample exit does not set a system message, but the sample code does contain the code necessary to set up and issue a message.

Processing Description

- Address the control information passed
- Set the return code to tell ETIP not to issue a message.

User Exit One Routine:

```
DKNETPU1 CSECT
DKNETPU1 AMODE 31
DKNETPU1 RMODE ANY
         SAVE  (14,12),,DKNETPU1
         LR    R12,R15           ;LOAD BASE REGISTER.
         USING DKNETPU1,R12      ;TELL ASSEMBLER.

*********************************************************************
*   PICK UP INPUT PARMS                                             *
*********************************************************************
*
         LM    R8,R10,0(R1)      LOAD PARM ADDRESSES.
         USING COUNTERS,R8       ADDRESS
         USING STARTSW,R9        ADDRESS
         USING SMSGPARM,R10      ADDRESS
```

Figure 4-42 (Part 1 of 3). ETIP User Exit One - Example 1

```
*************************************************************************
*   USER CODE GOES HERE                                                 *
*************************************************************************
*
         B     B99000            EXIT WITHOUT DISPLAYING A MESSAGE.
*
*     ROUTE MESSAGE TO BOTH TERMINAL AND TO THE SUPV TUBE.
*
         MVC   DESTCODE,=AL2(SMPDRTN+SMPDSUPV)
*
*     THIS IS AN ETPU1 MESSAGE, NOT AN ETIP MESSAGE.
*
         MVC   PROGNAME,=CL8'DKNETPU1'
*
*     SET MESSAGE NUMBER USING STANDARD SNNNN FORMAT.
*
         MVC   MSGNUMBR,=CL5'30001'    SEVERITY 3, MESSAGE 0001
*
         LA    R3,INSERTBF        LOCATE THE INSERT BUFFER.
*
*     CREATE AND TERMINATE THE FIRST SUBSTITUTION FIELD.
*
         MVC   0(L'INSERT1,R3),INSERT1  ADD THE FIRST INSERT AND
         LA    R3,L'INSERT1(0,R3)         STEP AROUND IT.
         MVI   0(R3),X'00'              ADD THE TERMINATOR.
         LA    R3,1(0,R3)               STEP AROUND IT.
*
*     CREATE AND TERMINATE THE SECOND SUBSTITUTION FIELD.
*
         MVC   0(L'INSERT2,R3),INSERT2  ADD THE SECOND INSERT AND
         LA    R3,L'INSERT2(0,R3)         STEP AROUND IT.
         MVI   0(R3),X'00'              ADD THE TERMINATOR.
         LA    R3,1(0,R3)               STEP AROUND IT.
*
*     CREATE AND TERMINATE THE SECOND SUBSTITUTION FIELD.
*
         MVC   0(L'INSERT2,R3),INSERT2  ADD THE SECOND INSERT AND
         LA    R3,L'INSERT2(0,R3)         STEP AROUND IT.
         MVI   0(R3),X'00'              ADD THE TERMINATOR.
         LA    R3,1(0,R3)               STEP AROUND IT.
*
*     MARK LAST FIELD IN BUFFER.
*
         MVI   0(R3),X'00'        NO MORE INSERT FIELDS.
*
         B     B99004             COMMAND ETIP TO DISPLAY THE MESSAGE.
*************************************************************************
*   EXIT WITH RETURN CODE ZERO - CONTINUE WITH NORMAL ETIP PROCESSING *
*************************************************************************
B99000   DS    0H
         SLR   R15,R15            ZERO RETURN CODE
         B     B99999             GO TO RETURN
```

Figure 4-42 (Part 2 of 3). ETIP User Exit One - Example 1

```
*************************************************************************
*   EXIT WITH RETURN CODE FOUR - DISPLAY MESSAGE FROM SMSG AREA       *
*************************************************************************
B99004   DS    0H
         LA    R15,4              SET RETURN CODE
*************************************************************************
*   RETURN TO DKNETIP                                                 *
*************************************************************************
B99999   DS    0H
         XRETURN (14,12),,RC=(15)   END OF ETPU1 PROCESSING

*************************************************************************
*   LITERALS AND CONSTANTS                                            *
*************************************************************************
INSERT1  DC    CL39'THIS TEXT WILL OVERLAY THE MESSAGE''S %1'
INSERT2  DC    CL21'TEXT FOR REPLACING %2'
*************************************************************************
*   DSECTS AND VARIABLE STORAGE                                       *
*************************************************************************

COUNTERS DSECT                    COUNT OF RECORD TYPES PROCESSED:
ELIGIBLE DS    H                  ... ELIGIBLE RECORDS
COMPLETE DS    H                  ... COMPLETE RECORDS
QUEUED   DS    H                  ... QUEUED RECORDS

STARTSW  DSECT                    ETIP START SWITCH --
ETIPSTRT DS    X                    HOW DKNETIP WAS STARTED:
OPERATOR EQU   B'00000001'          ... MANUAL START BY CPCS OPERATOR
ESMECYC  EQU   B'00000010'          ... BY ESM DURING END CYCLE
ESMTIMER EQU   B'00000100'          ... BY ESM START TIMER
INITIAL  EQU   B'00001000'          ... THIS IS 1ST RUN SINCE CPCS IPL
ESMCOLD  EQU   B'00010000'          ... BY ESM DURING COLD START
ESMANUAL EQU   B'00100000'          ... MANUAL START VIA ESM
SMSGPARM DSECT                    OPTIONAL MESSAGE FOR DISPLAY BY ETIP
DESTCODE DS    AL2                ... DESTINATION CODE(S) (NOTE:
*                                     THESE MAY BE OR'ED TOGETHER):
SMPDRTN  EQU   X'0001'               ... RETURN MESSAGE TO DKNETIP
SMPDSUPV EQU   X'0002'               ... SEND MESSAGE TO SUPV TUBE
SMPDSCRL EQU   X'0004'               ... SEND MESSAGE TO SCROLL
SMPDWTO  EQU   X'0008'               ... DISPLAY MESSAGE VIA WTO
SMPDMICR EQU   X'0010'               ... SEND MESSAGE TO MICR TUBE
SMPDINSC EQU   X'0020'               ... SEND MESSAGE TO INSC TUBE
*
PROGNAME DS    CL8                ... SUBSYSTEM / PROGRAM NAME
MSGNUMBR DS    CL5                ... MESSAGE NUMBER
INSERTBF DS    CL120              ... INSERT BUFFER
         END
```

Figure 4-42 (Part 3 of 3). ETIP User Exit One - Example 1

ETIP User Exit Two — Input Processor Selection

The ETIP user exit two is called each time an eligible record is selected from the electronic control file. The user exit can then determine whether the eligible record should be passed to ETIN for processing.

Activation

A different ETIP user exit two routine may be specified for each financial institution. The exit is called by the ETIP task each time an eligible control file record is found.

No regenerations, CPCS restarts, or CHAPs are required when a new ETIP user exit two routine is installed.

To initiate this user exit point, set the following card in the DKNPEXIT member on the system profile data set:

```
DKNETIP_IDENTIFY_RECS_TO_PROCESS=xxxxxxxx
```

If a new user exit is installed, the user exit must be refreshed using the CPCS EXIT task.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-43. ETIP User Exit Two Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Control file description	DKNETP22	DKNETP21
ETIP type of run switch	N/A	N/A

The exit routine may return the following condition codes in the RETURN-CODE special register (COBOL programs) or register 15 (assembler programs).

Figure 4-44. ETIP Exit Two Routine Return Codes

Code (dec)	Request
+00	Pass the control file record to ETIN for processing
+04	Do NOT pass the control file record to ETIN

Restrictions

ETIP user exit one routines must be reentrant and reusable.

Example 1

Operation:

This sample exit instructs ETIP to always import the transmission file being checked.

Processing Description

- Address the control information passed
- Set the return code to tell ETIP to initiate the import for the transmission data set.

User Exit Two Routine:

```
DKNETPU2 CSECT
DKNETPU2 AMODE 31
DKNETPU2 RMODE ANY
         SAVE  (14,12),,DKNETPU2
         LR    R12,R15           ;LOAD BASE REGISTER.
         USING DKNETPU2,R12      ;TELL ASSEMBLER.

         LM    R8,R9,0(R1)       LOAD PARM ADDRESSES.
         USING DKNETCF1,R8       ADDRESS CONTROL BLOCK
         USING STARTSW,R9        ADDRESS SWITCHES
*
         B     B99000            ACCEPT RECORD
*
*********************************************************************
*   EXIT WITH RETURN CODE ZERO - ACCEPT CONTROL FILE RECORD          *
*********************************************************************
B99000   DS    0H
         SLR   R15,R15           ZERO RETURN CODE
         B     B99999            GO TO RETURN
*********************************************************************
*   EXIT WITH RETURN CODE FOUR - DO NOT PROCESS CONTROL FILE RECORD  *
*********************************************************************
B99004   DS    0H
         LA    R15,4              SET RETURN CODE TO 4
```

Figure 4-45 (Part 1 of 2). ETIP User Exit Two - Example 1

```
*************************************************************************
*   RETURN TO DKNETIP                                                   *
*************************************************************************
B99999   DS    0H
         XRETURN (14,12),,RC=(15)   END OF ETPU2 PROCESSING
*
*************************************************************************
*   DSECTS AND VARIABLE STORAGE                                         *
*************************************************************************
STARTSW  DSECT                    ETIP START SWITCH --
ETIPSTRT DS    X                    HOW DKNETIP WAS STARTED:
OPERATOR EQU   B'00000001'          ... MANUAL START BY CPCS OPERATOR
ESMECYC  EQU   B'00000010'          ... BY ESM DURING END CYCLE
ESMTIMER EQU   B'00000100'          ... BY ESM START TIMER
INITIAL  EQU   B'00001000'          ... THIS IS 1ST RUN SINCE CPCS IPL
ESMCOLD  EQU   B'00010000'          ... BY ESM DURING COLD START
ESMANUAL EQU   B'00100000'          ... MANUAL START VIA ESM
*
         DKNETCF1 DSECT=YES       GET COPY OF CONTROL FILE RECORD
         END
```

Figure 4-45 (Part 2 of 2). ETIP User Exit Two - Example 1

Customizing ETCSH and X937 User Exits

Four user exits are available for use as part of the Electronic Transaction Cash Letter function:

- DKNETCSH_FORMATS=DKNETX01
- DKNETCSH_0100_NEW_RECORD=DKNETX02
- DKNETCSH_0200_FILE_CREATED=DKNETX03
- DKNX937_0100_NEW_RECORD=DKNETX04

DKNETCSH_FORMATS

DKNETCSH calls this program during initialization time to determine if a format other than X9.37 is required and, if so, to pass back the file length. It is also called for every detail item to determine if a file format other than X9.37 is required.

Initialize Work Areas

In the work areas you can check if a special type of record is to be created. These records may be specified in the kill-bundle record that can be accessed through the ETPM dsect. If the special records are to be processed, and the record length is greater than 80 (size of X9.37 record), the new length must be placed in the DSCRECLN field. The DSCSWCH1 flag can be set to identify special records and can be used in future calls of this exit.

Detail Record Processing

If we are processing a format other than X9.37, and if it is applicable, produce the necessary data, such as file headers. Also adjust record pointers, if necessary, so that records are written out correctly.

Environment

This exit is given control within DKNCLSM TCB.

Point of Processing

This exit is called during the initialization process of DKNETCSH to determine if special records are to be created. If special records are to be processed, this exit is called prior to creating X9.37 records for each record processed.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNETCSH_FORMATS=keyword is used to specify an exit for this exit point.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-46. DKNETCSH_FORMATS Point Parameter List

Offset	Description
+00	CPCS Parameter List (DKNPARM)
+04	DKNETCSH Parameter Block (DKNETPM1)
+08	DKNETCSH Data Set Control Area (DKNETDSC)

Register Contents When Control is Passed to the Exit Routine:

R0	Not applicable
R1	Parameter list address
R2-R12	Not applicable
R13	Caller's save area address
R14	Return Address
R15	User exit entry point address

Register Contents When Control is Passed Back to CPCS:

R0	Not applicable
R1	Not applicable
R2-R13	Restored to same values as on user exit entry
R14	Not applicable
R15	Return codes as described below
	0 Good return code
	4 Warning level error
	8 Error
	12 Severe Error

Other Programming Considerations

Note: IBM requires this exit to be re-entrant.

Example:

Member DKNETX01, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNETCSH_0100_NEW_RECORD

This program is called by DKNETCSH after an X9.37 record has been created. This exit allows the operator to modify the X9.37 record.

Environment

This exit is given control within DKNCLSM TCB.

Point of Processing

Each X9.37 record is passed to this exit and the operator can modify the record as necessary.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNETCSH_0100_NEW_RECORD=keyword is used to specify an exit for this exit point.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-47. DKNETCSH_0100_NEW_RECORD Point Parameter List

Offset	Description	
+00	X9.37 Record	
	Cobol	**Assembler**
	DKNCSX92	DKNCSX91

Register Contents When Control is Passed to the Exit Routine:

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control is Passed Back to CPCS:

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not Applicable

R15 Return codes as described below

- **0** Good return code
- **4** Warning level error
- **8** Error
- **12** Severe Error

Other Programming Considerations

Note: IBM requires this exit to be re-entrant.

Example:

Member DKNETX02, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNETCSH_0200_FILE-CREATED

This program is called by DKNETCSH after the electronic cash letter file has been created.

Environment

This exit is given control within DKNCLSM TCB.

Point of Processing

This exit, named DKNETX03, is called after the electronic cash letter file has been created. The exit receives the data set name of the cash letter file.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNETCSH_0200_FILE_CREATED=keyword is used to specify an exit for this exit point.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-48. DKNETCSH_0200_FILE-CREATED Point Parameter List

Offset	Description	
+00	DKNETCSH Parameter Block	
	Cobol	**Assembler**
	DKNETPM2	DKNETPM1
04	Cash Letter Data Set Name	
08	Application Task Control Block	
	Cobol	**Assembler**
	DKNCATCB	DKNAPTCB

Register Contents When Control is Passed to the Exit Routine:

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control is Passed Back to CPCS:

R0 Not applicable

R1 Not applicable

R2-R13 Restored to same values as on user exit entry

R14 Not applicable

R15 Return codes as described below

- **0** Good return code
- **4** Warning level error

8 Error

12 Severe Error

Other Programming Considerations

Note: IBM requires this exit to be re-entrant.

Example

Member DKNETX03, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNX937_0100_NEW_RECORD

This program is called by DKNX937 after an X9.37 record has been created. This exit allows the operator to modify the X9.37 record.

Environment

This exit is given control within DKNCLSM TCB.

Point of Processing

Each X9.37 record is passed to this exit and the operator can modify the record as necessary.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNX937_0100_NEW_RECORD=keyword is used to specify an exit for this exit point.

Linkage

The standard OS/390 linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-49. DKNX937_0100_NEW_RECORD Point Parameter List

Offset	Description	
+00	X9.37 Record	
	Cobol	**Assembler**
	DKNCSX92	DKNCSX91

Register Contents When Control is Passed to the Exit Routine:

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller's save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control is Passed Back to CPCS:

R0	Not applicable
R1	Not applicable
R2-R13	Restored to same values as on user exit entry
R14	Not applicable
R15	Return codes as described below
	0 Good return code
	4 Warning level error
	8 Error
	12 Severe Error

Other Programming Considerations

Note: IBM requires this exit to be re-entrant.

Example

Member DKNETX04, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNLDMY and DKNLJMP Exit — Logging Job MVS Posture

DKNLDMY (Dummy executive task)

DKNLDMY is an executive task that can be automatically started each time CPCS is started. DKNLDMY's only function is to autostart DKNLJMP at CPCS start-up.

DKNLJMP

DKNLJMP reports the status of the logging files and gives control to a user exit that can then tell DKNLJMP to automatically start a backup. It has three distinct levels.

1. DKNBKUP and DKNBBKUP turn on a backup in progress flag (BIP), and they send a message to indicate a backup has started. When the backup completes successfully an additional message is sent to indicate the backup task (either DKNBKUP or DKNBBKUP) ran successfully.
2. DKNLJMP can be run manually by the CPCS supervisor or started automatically by the executive task DKNLDMY, to check the status of the BIP flag. If no user exit is specified DKNLJMP will send messages to the CPCS supervisor and the CPCS SCROLL data set indicating the status of the DASD logging files. If a user exit is specified then the user exit can instruct DKNLJMP to autostart either DKNBBKUP or DKNBKUP. Using this technique and LOGCRASH, you can now have a backup run each time CPCS is started and stopped.
3. LOGCRASH is a batch job that when placed in the CPCS run JCL before or after the CPCS JOBSTEP will run and backup the logging files. DKNLJMP's primary purpose is to provide the CPCS supervisor a way within CPCS to determine if either DKNBKUP or DKNBBKUP completed successfully.

DASD logging will log to either of two files whose DDNAMES are DKNLD1 and DKNLD2. Logging will log to one of these two files and when the file gets full logging will flip-flop the files and continue logging to the other file. Logging then autostarts either DKNBKUP or submits an MVS job to start DKNBBKUP, when either DKNBKUP or DKNBBKUP complete the recovery status file is updated to reflect the status of the DASD logging files.

DKNLJMP reads the recovery status file (DDNAME DKNRCVTD) to determine if DKNBKUP or DKNBBKUP has successfully completed.

If DKNBKUP or DKNBBKUP do not complete successfully then DKNLJMP will attempt a restart of either DKNBKUP or DKNBBKUP, if told to do so by the DKNLJMP user exit. The user should however verify that DKNBKUP or DKNBBKUP are not currently running (they may be waiting for a tape mount or other system resources). This capability has not been included in the user exit; the default in the user exit is that a backup is started using the parameters in the system profile member DKNPRCVY.

The LJMP user exit receives control at three times during the execution of DKNLJMP.

- During Initialization
- After the logging status file has been read. The user exit may be able to determine the status of the logging files by testing to see if at least one of the

files is being logged to and the other has been backed up. This scenario would not hold true if the user had just run DKNGLINT to initialize the logging files.

- During final processing

Activation

A CPCS system-wide LJMP user exit routine can be activated by specifying its name in the LJMP DKNBLDL entry via the USREXIT parameter.

DKNBLDL must be re-assembled and CPCS must be restarted when LJMP exit is activated/deactivated via the USREXIT parameter.

Linkage

The standard MVS/ESA linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-50. LJMP Exit Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Call Parameters • LJMP processing modification requests are made via the reason-code field in this control block.		Hardcoded in DKNLJMPU
Logging/Recovery status file	RGENDSCT	
APTCB	DKNCAPCB	DKNAPTCB
PARMLST	DKNCPARM	DKNPARM

Restrictions

The LJMP exit routine is called only if Logging is activated.

The following restrictions apply when DKNBBKUP is used as the backup task:

- The Backup in Progress flag can be set only if the batch job actually starts. If the job is canceled prior to the program starting (i.e. waiting for a tape mount) then the backup in progress flag will not be set.
- The user exit may be able to determine the status of the logging files by testing to see if at least one of the files is being logged to and the other has been backed up. This scenario would not hold true if the user had just run DKNGLINT to initialize the logging files.

Writing Your Own User Exit for DKNLJMP

The sample user exit (DKNLJMPU) has test scenarios which are executed by DKNLJMPU. You should read and study the code in the sample user exit DKNLJMPU, and then write a user exit to suit your environment.

DKNLOGU: Logging User Exit

DKNLOGU is the default name for the logging user exit.

DKNRCVY_EXIT1000_BEFORE_WRITE

The primary purpose of this exit is to provide your CPCS system a point at which to examine each item and to adjust the item record for your individual requirements. This user exit routine is entered once for each item processed by DKNRCVY.

Environment

This exit is given control within DKNRCVY.

Point of Processing

This exit is called before the write to the mass data set. Each item is passed to the exit after DKNRCVY has performed its conversion process.

Activation

All exits for this exit point are specified in the CPCS system profile (DKNPEXIT), which is located in the CPCS.V1R11.SYSTPROF partitioned data set. The DKNRCVY_EXIT1000_BEFORE_WRITE=*keyword* is used to specify an exit for this exit point. Up to the maximum number of entry points per exit may be specified for this exit point. See “DKNPCPCS Profile Member” on page 3-4 for this profile, located in the SYSTPROF data set, for details on the MAX_EP_EXIT parameter.

Each exit is called, in the order they were specified in the DKNPEXIT profile, until there are no more exits or until one of the exits denies the request with a return code.

Figure 4-51. DKNRCVY_EXIT1000_BEFORE_WRITE Point Parameter List

Offset	Description
+00	Caller’s APTCB Address (DKNAPTCB)
+04	Tape Record Address
+08	MDS Record Address
+12	RDX Address (RDXREC)
+16	XREC Address (MDXREC)

Register Contents When Control Is Passed to the Exit Routine

R0 Not applicable

R1 Parameter list address

R2-R12 Not applicable

R13 Caller’s save area address

R14 Return address

R15 User exit entry point address

Register Contents When Control Is Passed Back to CPCS

R0	Not applicable
R1	Not applicable
R2-R13	Restored to same values as on user exit entry
R14	Not applicable
R15	Return codes as described below:
	xx Non-zero - do not write item to MDS

Other Programming Considerations

Notes:

1. IBM requires this exit be re-entrant.
2. MDSREC is the updated record written to the mass data set.

Example

Member DKNLOGU, in the CPCS.V1R11.SAMPLIB partitioned data set, may be used as an example when coding assembler exit routines for this exit point.

DKNMBEGN User Processing Exit

The CPCSOPTN macro specifies the DKNMBEGN user processing exit. It permits exit-routine inspection of the information that the operator enters on the MICR BEGIN screen and determines whether to return a diagnostic message to the operator or to permit the entry to start. If specified, this user-supplied routine is entered after the operator presses **ENTER** to verify the BEGIN screen, but before the start of the entry.

When accepting an entry, the exit might cause 4 bytes of user data, for use by application programs, to go into the string-header record of the entry I-string. This option also permits the operator to enter up to 12 bytes of additional information, in addition to the normal BEGIN screen input, so that the DKNMBEGN user exit can inspect it and process accordingly, based on user criteria. An example of this exit appears on the following pages.

To use the option, the CPCSOPTN macro operand BEXIT=*symbol* must be specified. *Symbol* is the name of the user-written exit routine that loads during MICR task initialization. After the routine loads, it is resident throughout CPCS running. You must ensure that the exit routine is reentrant. The same copy of the program is used for all active document processors.

The user-exit routine receives control from the MICR BEGIN command processor after the operator verifies the screen by pressing **ENTER**, but before the start of the entry. On entry to the user-exit routine, register 1 contains the address of a 16-byte field. The first 12 bytes contain information that the operator entered on the option line of the screen. The first 2 of these bytes must be U=. The rest of the field is user-dependent. You must key in U=*X...X* so that the exit routine will process this information. If the data entered is less than 12 bytes, it is padded with blanks. The user-exit routine can use the last 4 bytes of the 16-digit field. When DKNMBEGN regains control from the user-exit routine, it puts the 4 bytes into RSCBUSPA and this field is later inserted into the string header (DIDSCT) for

processing by user application programs. The definition for this field is DIABA + 7(4). Register 1 serves as a parameter register only if the program is to pass operator-entered information (U=*X...X*).

For more extensive processing by the user-exit routine, use BEGNDSCT. This is the save area and the work area of DKNMBEGN. You can edit and inspect all information that the operator entered for the start of the entry. On entry to the begin-user-exit routine, register 13 contains the address of the information that the DSECT represents.

A copy of BEGNDSCT is in the source library. Code `USING WKAREA,13` to get access to BEGNDSCT. The label BGNUSEX is the 12-byte field of information that the operator enters. Label BGNUSPA is the 4-byte field for information that passes to the string-header record. Any other changes can cause unpredictable results.

If the user program determines that the data the operator entered is wrong, it can, on returning to the BEGIN MICR command processor, send a 32-byte message to the operator by inserting the address of the message in register 15. The displayed message is numbered 31. If the operator does not then enter the correction that the message specifies, the entry does not start. If the user program determines that the information that you originally supplied is correct, it should, on return to the BEGIN MICR command processor, clear register 15.

Note: You must observe standard save-area conventions.

You should not use any supervisor or data management macro instructions except for the TIME macro, or unpredictable results might occur.

Register 15 should contain the following information at completion of begin-user-exit processing:

00000000 Normal return. Processing continues.

00XXXXXX Bits 8 through 31. Address of diagnostic message. Processing does not continue.

Register Contents on Entry to DKNMBEGN Processing Exit Routine

The following registers are used on entry to the user exit:

Register Content

1 Address of 16-byte parameter field that the user-exit routine processes:

Bytes 0–11 Operator-entered information
Bytes 12–15 User-exit string header information to transfer

13 Address of standard save area and BEGNDSCT (save area and work area for BEGIN MICR command processor)

14 BEGIN MICR command processor return address

15 Address of user-exit processing routine entry point

Sample DKNMBEGN User Processing Exit Routine

A sample routine is in CPCS.V1R11.SAMPLIB under the name BEGNEXIT. This sample routine serves as a guide for the user. It performs the following functions:

1. Checks to see whether the operator entered `U=run1` on the option line of the BEGIN command processor.

   If yes, the exit-processing routine continues. If the operator entered blanks, the exit processing is bypassed. Otherwise, the routine inserts the address of the message `WRONG EXIT INFORMATION ENTERED` in register 15 and returns to the BEGIN command processor.

2. Checks to see whether the cycle that the operator entered is 8.

   If it is, exit processing continues. If not, the routine inserts the address of the message `THE CYCLE ENTERED IS WRONG` in register 15 and returns to the BEGIN command processor.

3. Checks to see whether the sort type that the operator entered is 1, 4, or 60.

   If it is one of these, the routine moves RUN1 into BGNUSPA, clears register 15, and returns to the BEGIN command processor.

   If the sort type is not one of the above numbers, the routine inserts the address of the message `THE SORT TYPE ENTERED IS WRONG` and returns to the BEGIN command processor.

DKNMDIS–ESM Task Profile User Data Area

If DKNMDIS is automatically started with the Enhanced System Manager feature of CPCS, one of the following options should be entered in the user data area of the task profile:

ESM User Option	Corresponding Manual Start Option
null	1 (No option specified defaults to option 01)
01	1 (Normal distribution - the default)
02	2 (Force distribute enhanced reject pockets)
03	3 (Force distribute kill pockets)
04	4 (Force redistribute enhanced reject pockets)

DKNMDIS User Exit — Modify M-String Distribution

The MDIS user exit receives each codeline record being processed, once MDIS has made a distribution decision about the record. The exit allows you to:

- Distribute codeline records to D-strings other than the ones selected by MDIS.
- Exclude codeline records from being distributed.
- Change the pocket number of codeline records.
- Distribute non-distributable records.
- Deactivate the exit routine for the remaining MDIS processing.
- Terminate MDIS processing.

The exit routine can perform first or last call processing based on the value of a passed call-sequence indicator.

The string directory UD flag is set for each MDIS distributed D-string that has been altered by the exit routine.

The 'distribute non-distributable record' request is satisfied without affecting the current status of the MDIS record save buffers.

No output is generated when the 'terminate MDIS' option is requested (distributed records are only written when all the input has been processed).

Message MDIS 30013, which includes the exit routine name, is displayed on the CPCS supervisor's terminal and written to the scroll log when MDIS processing is terminated due to an exit routine request (see MDIS messages in *CPCS Messages and Codes* for more information).

Note: Members DKNMDIX and DKNMDIX2 of CPCS.V1R11.SAMPLIB may be used as templates when coding MDIS user exit routines in assembler and COBOL respectively.

Activation

A different MDIS exit routine can be specified for each financial institution and sort type. The exit is active when one or more MDIS exit routine names are specified in the bank control file, or when an MDIS exit routine name is specified in the sort pattern definition for the distribution, with the sort pattern definition taking precedence over the bank control file when both specify exit routines (see "Bank-Control-File Record Formats" on page 4-5 and "Sort-Pattern-Definition Record Formats" on page 4-17 for more information).

A CPCS system-wide MDIS user exit routine can also be activated by specifying its name in the MDIS DKNBLDL entry via the USREXIT parameter. An MDIS exit routine specified in the DKNBLDL table gets activated only when the bank control file and sort pattern definition do not specify any MDIS exit routine for the distribution. See "Describing an Application Task's Environment" on page 2-15 for more information on the USREXIT parameter.

No re-gens, CPCS restarts or CHAPs are required when MDIS exits are activated/deactivated via the bank control file or sort pattern definitions, or when new MDIS exit routine versions are installed. Module DKNBLDL must be re-assembled and CPCS must be restarted when MDIS exits are activated/deactivated via the USREXIT parameter.

Linkage

The standard MVS/ESA linkage conventions are followed.

Information is passed in the following interface control blocks:

Figure 4-52 (Page 1 of 2). MDIS Exit Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
Call Parameters • MDIS processing modification requests are made via the reason-code field in this control block.	DKNMDIXC	DKNMDIXP

Figure 4-52 (Page 2 of 2). MDIS Exit Communication Control Blocks

Control Block	COBOL Copybook	Assembler Macro/ Copybook
MDS Codeline Buffer	DKNCRDI	ZDDSCT
	DKNCRZC	DEDSCT
MDIS Stacks	DKNMDISQ	DKNMDIXP
String's Expanded Directory Record Description (ZE)	DKNCRZE	ZEDSCT
MDIS Candidate List	DKNMDISS	DKNMDIXP

- This list contains the pockets that are valid for distribution.

Restrictions

MDIS exit routines should preferably be written in COBOL, but may also be written in assembler.

The candidate list should not be modified by the MDIS exit routine.

Codeline records that have their pocket number or distribution pocket number changed to an invalid (not in the candidate list) pocket number, do not get distributed.

Codeline records that have their pocket number changed, get distributed to the D-string corresponding to the new pocket number.

DKNMIPI User Processing Exit

This exit is an option in CPCSOPTN, in the bank control file, or in the SPDEF. It gives you control over the records sent to the document processor for codeline data matching. If specified, the user-supplied routine is presented for each document that the document-processing work element of MICR (DKNMIPI) processes.

To select this option, you must specify the CPCSOPTN macro operand MEXIT=*nnnnnnnn*, where *nnnnnnnn* is the name of a user-written exit routine that loads during MICR task initialization. In addition, you can specify a different DKNMIPI exit in the BCF for each financial institution or for each run type in the SPDEF.

DKNMIPI user exit routines must be re-entrant. Results will be unpredictable if this requirement is not met.

DKNMIPI uses the user-exit routine for each codeline data record or group of records whose item sequence numbers are the same. DKNMIPI passes the record or records along with the entire MDS codeline data record to the exit routine in a buffer that contains a flag byte for each record. The default buffer contains slots for 16 codeline data records.

The user-exit routine must indicate, through the flag byte, which records to send to the document processor for codeline data matching. The flag byte is preset with X'00', indicating that normal document records should be sent to the document processor. Unused buffer slots are filled with X'00'.

Reject codeline records (DITYPEI=X'95'), change control records (DITYPEI=X'85' and DI3CHG on), and power-encoding records (DITYPEI=X'86') are preset with X'FF', indicating that they should not be sent to the document processor. The user-exit routine can prevent a record from being sent to the document processor by placing any value other than X'00' in the associated flag byte. For more information about these records, see “Balancing and Power-Encoding Considerations” on page 4-145.

DKNMIPI provides the prior pass D-string records in groups of 16 to the document processor. These groups of 16 documents are sequential in the order that they are on the prior pass D-string. Resynchronization may be required for conditions such as a Divider Slip, a sorter pause, or an out-of-synch condition. This will cause DKNMIPI to back up in the prior pass D-string, and process some records again. Records that are processed again will be sent to the MIPIEXIT if one is specified, and then sent to the document processor.

Notes:

1. A resynchronization is done after the first two buffers are sent following new tracers due to a possible sorter pause. This results in most of the first buffer, and all of the second buffer being repeated.
2. When divider resynchronization is active, the divider slips must be encoded with unique serial numbers and with the field that identifies the control document. When codeline data matching is active, the input of D-string data records must be synchronized on a restart. The D-string records must also be resynchronized for each new tracer group, divider slip, or string combination.

User-Exit Parameters and Conventions

At entry, register 1 contains a pointer to a four-fullword parameter list that contains the following:

- Fullword address of the end of the buffer (end pointer)
- Fullword address of the first slot of the buffer that contains a codeline data record (head pointer)
- Fullword address of the next available empty slot within the buffer (tail pointer)
- Halfword length of the MDS record
- Halfword length of the buffer slot (MDS length plus one).

The following table shows the setting of the pointers for a 16-slot buffer into which MICR has placed five MDS records.

Head pointer	Slot 1	Flag byte, MDS record 1
	Slot 2	Flag byte, MDS record 2
	Slot 3	Flag byte, MDS record 3
	Slot 4	Flag byte, MDS record 4
	Slot 5	Flag byte, MDS record 5
Tail pointer	Slot 6	Empty
	Slot 7	Empty
	⋮	⋮
	Slot 14	Empty
	Slot 15	Empty
	Slot 16	Empty
End pointer		

You can insert codeline data records in the stream of documents sent to the document processor by:

1. Formatting an MDS record in the slot to which the tail pointer points
2. Placing an X'00' in the associated flag byte
3. Moving the tail pointer by the length of the buffer slot.

The document processor does not return inserted records to MICR because they will not match within the document processor.

Warning: When you insert records, use caution to prevent the tail pointer from going beyond the end pointer. Changing storage beyond the end pointer can cause unpredictable results. The user-exit routines should not change any pointers other than the tail pointer.

If a 16-slot buffer is not enough, you can allocate additional slots by modifying the &NOBUFF parameter in the CPCSOPTN macro. Change the parameter value to the number of slots that you require and generate a new MICR task to make the change effective.

Although you can change the MDS records, modifying a field used by the document processor to perform codeline data matching might cause the codeline data match on that document to fail. Changing field 8, the item sequence number (DISEQ12), can cause missing and free documents in later CPCS processing.

DKNMREAD Document-Processing User Exit

You can specify this exit in either the CPCSOPTN macro or as an option in the bank control file. It permits additional user processing over and above that which the user stacker-select routine has already performed in the document processor. If specified, the user-supplied routine is used for each document processed by the document-processing work element of MICR (DKNMREAD). Such additional processing might include:

- Additional item validation that was not necessary to determine pocket selection.
- Analysis of rejected items and setting of communication flags to affect processing of the item when received by the user-edit (stacker-select) routine during online reject reentry.
- Determination of document sequence errors that should cause entry ending or restart or other operator intervention. For example, it could check for too many rejects in a row or for control documents that are not valid for the type of run.
- Preparation of user-defined control records for insertion into the I-string.
- Checking for incorrect programmable endorse when MODEL=XP was coded in CPCSRDR generation.
- User data inserted into expanded MDS fields.

To select this option, you must specify the CPCSOPTN macro operand, REXIT=*nnnnnnn*, where *nnnnnnn* is the name of the user-written exit routine that loads during MICR task initialization. You can also specify in the bank control file an MREAD exit name for banks other than bank 001.

DKNMREAD user exit routines must be re-entrant. Results will be unpredictable if this requirement is not met.

The user-exit routine receives control from the MICR task document processing routine after a codeline data record is retrieved from the document processor, but before its capture on the I-string. All documents pass to the routine, with the exception of divider documents. In addition, the routine is entered before the first document and again after the last document completes processing.

You can inspect the codeline data record and control information; however, you should change only the following information:

- User flag byte
- Flag 3 bytes 4 and 5
- Expanded MICR fields (9 through 15).

Unpredictable results might occur if you change other portions of the record. The routine has the ability to suspend document processor operations and communicate with the MICR operator. If this is done, no more document processing is attempted, and the operator must either end or cancel the entry. Operator communication in this case consists of displaying a message on the MICR Status display. Messages supplied by the exit routine for this purpose can be up to 40 characters long and can contain any EBCDIC characters except % (X'6C') and ? (X'6F').

The user-exit routine should not send any supervisor or data-management macro instructions, except the TIME macro. If it does, unpredictable results might occur.

The sample exit routine has addressability to both the MDS record to be written (passed in register 1 on entry) and the document processor record (in the field MUPASPTR in the MUPA DSECT).

MDS Record Insertion

The user exit can also insert records to be written to the I-string. On return from the user exit, DKNMREAD inspects the contents of register 15 and, if an address is present, either a user stop request or a user message insertion request is expected. The routine honors a record insertion request if bit 1 is on in the first byte of the 2-byte user area immediately in front of the codeline data record to be inserted. It is addressed by register 15. The record to be inserted must start after the second byte of the area addressed. The record is written to the MDS immediately.

You must code any insert record with bit 0 on X'80' in DIFLAG2 and any code from X'00' to X'7F' in DITYPEI. The pocket code must be valid for the document processor on which the entry takes place. The first byte of the record must not be X'FF' because this signals end-of-string to DKNMREAD. There are no system restrictions on the remainder of the record or on the DITYPEU byte. DIFLAG2 and DITYPEI are restricted by convention. You should put an X'80' into DIFLAG2 for every record to be inserted, but you can overlay the byte before DKNMREAD writes to an I-string.

The user exit receives control on restarts when the MICR task document-processing routine reads records from the restarted I-string, and reaches the point at which previous processing ended. At this point, the user exit gains control. This occurs before document processing resumes. Any records previously inserted into the I-string are not available to the user exit, and the user exit does not honor document insertion requests while the entry is in the process of reading

the I-string. The MUPARST flag in MUPAFLGS of the MUPA in the MUPA DSECT indicates expanded document processor records.

The user exit receives control on the first tracer slip of each tracer group, including the entry group on HSRR. All other tracer groups in HSRR are not passed. The tracer slip cannot be changed. However, records can be inserted on the I-string before the tracer is written to the string. An X'80' bit in DIFLAG2 and an X'F7' character in DITYPEI identify a tracer slip.

DKNMREAD writes the inserted record to the MDS immediately when it receives the insertion request from the exit routine. DKNMREAD does not validate an inserted record. Also, DKNMREAD does not process the inserted record. The inserted record precedes the document that had a record passed to the user exit. DKNMREAD also permits insertions at the end of the string. When this occurs, a codeline data record that contains all X'FFs' passes to the exit. (This record is also received when MICR SUSPEND, caused by canceling the MICR run after documents have been captured, occurs.) Because insertions must precede the document that passes to the user exit, DKNMREAD permits no insertions on the initialization call to the exit. If you want to insert data after a specific document, you must make the request on the presentation of the next document to the user exit.

A loop of repetitive calls to the user-exit routine can be used to insert more than one record before a given document. This is done if the user exit turns on the X'80' bit in the first byte of the 2-byte user area that immediately precedes the codeline data record to be inserted. It is addressed by register 15. You can also use this technique to make a user stop-processing request following a record insertion.

User-Exit Register Conventions

The user-exit routine must set a return code in register 15 before returning to DKNMREAD.

Register 15 Return Codes

0 Normal return, continue processing

Address Address of a user area

If bit 1 of the byte pointed to by register 15 is on, the user exit is requesting that MREAD insert a user control record into the I-string.

If bit 1 of the byte pointed to by register 15 is off, the user exit is requesting that MREAD suspend (disengage) the document processor and display the message supplied.

Figure 4-53 shows the format for the record insertion area. Figure 4-54 on page 4-116 shows the format for the user-supplied message area. Figure 4-55 on page 4-116 shows the register contents on entry to the user-exit routine.

Figure 4-53 (Page 1 of 2). Format of Record Insertion Area

Bytes	Bits	Contents
0	0	If on, perform insertion that bit 1 indicates and return to user exit for additional inserts
	1	If on, perform record insertion

Figure 4-53 (Page 2 of 2). Format of Record Insertion Area

Bytes	Bits	Contents
	2–7	Reserved
1	0–7	Reserved
2–xx		MDS formatted record for insertion into the I-string. The defined length of the MDS determines xx.

Figure 4-54. Format of User-Supplied Message Area

Bytes	Bits	Contents
0	0–7	Reserved; must be 0
1	0–1	Reserved; must be 0
	2–7	Binary length of user-supplied message. Must be less than or equal to the binary equivalent of decimal 40.
2–41		User-supplied message

Figure 4-55. User Exit Register Contents

Register	Bits	Contents
0	0–23	Zero
	24–31	Document processor (X'01'=document processor 1) (X'02'=document processor 2)... (X'63'=document processor 99)
1	8–31	Address of codeline data record
2	8–31	Address of MICR user-parameter area (MUPA)
13	8–31	Address of a standard save area that lets the user save DKNMREAD's registers. Any registers that the exit routine uses must be saved in this area and restored before returning to DKNMREAD.
14	8–31	MICR task return address
15	8–31	Address of user-exit routine entry point

Sample Document-Processing User-Exit Routine

Sample routines are in CPCS.V1R11.SAMPLIB under the names SAMPEXIT and SEXIT770.

This sample routine serves as a guide for the user. It performs the following functions:

1. After initialization, clears a work area for the indicated document processor
2. Checks to see whether the first document is a block slip (not done on subsequent pass). If YES, does a normal exit. If NO, suspends the entry and displays message 1
3. Checks to see whether the batch slip always follows the block slip (not done on subsequent pass). If YES, does a normal exit. If NO, suspends the entry and display message 2.
4. Checks to see whether more than 20 successive rejects occur. If NO, does the next step. If YES, suspends the entry and displays message 3.

5. Checks whether more than 20 successive programmable endorse errors occur (SEXIT770 routine only–used with the 3890/XP Series document processors). If NO, does a normal exit. If YES, suspends the entry and displays message 4.
6. Clears the work area after ending the entry.

Number	Message
1	BLOCK SEQUENCE ERROR : CANCEL/RESTART
2	BATCH SEQUENCE ERROR : CANCEL/RESTART
3	EXCESSIVE REJECTS : CANCEL/RESTART
4	EXCESSIVE ENDR ERRORS : CANCEL/RESTART

DKNCSBU—Sort Program Build Task

The DKNCSBU task provides the capability for a workstation-based application to load the sorter version of the sort program from CPCS and run it on a workstation. This includes the initialization data (IREC), user stacker-select routine, and the user table.

Task Initiation

DKNCSBU should have the following APCB entry added to DKNBLDL:

```
APCB NAME=DKNCSBU,AUTO=1
```

To initiate DKNCSBU, the calling program posts DKNATASK with the following information in the start parameter:

First fullword Address of the ECB that DKNCSBU posts after it is attached by DKNATASK

Second fullword Address of the ECB that DKNCSBU waits on for the post from the calling program

Third fullword Address of the DKNCSBUB control block.

DKNEMRG User Exits

DKNEMRG makes a number of calls to various user exits. These exits are all controlled by the CPCS user exit manager (UEM); they are listed in the DKNPEXIT System Profile Control Card and can be activated, deactivated, or refreshed using the CPCS EXIT task.

For each user exit, IBM has provided a sample in the SAMPLIB distribution library. These samples are not type 1 supported code; rather, they are provided as examples of how the end user should go about coding his or her user exit logic.

Figure 4-56 shows a complete list of all DKNEMRG user exits, along with the sample program that IBM ships for each one:

Figure 4-56. DKNEMRG user exits

EMRG User Exit Name	Sample User Exit Program
DKNEMRG_MAIN_EXIT	DKNEMRG1
Provides overall control over EMRG's behavior.	

Figure 4-56. DKNEMRG user exits

EMRG User Exit Name	Sample User Exit Program
DKNEMRG_USER_INPUT_EXIT	DKNEMRG2
Controls how EMRG's input (whether from a screen, from an autostart buffer, or from ESM) is edited.	
DKNEMRG_POCKET_MERGE_EXIT	DKNEMRG3
Provides control over EMRG's Pocket Merge behavior.	
DKNEMRG_CONSOL_REJ_MERGE_EXIT	DKNEMRG4
Provides control over EMRG's Consolidated Reject Merge behavior. The Consolidated Reject Merge module is, in fact, quite dependent on its user exit; it cannot function without it. For this reason, DKNEMRG_CONSOL_REJ_MERGE_EXIT has been set as DEACTIVATE=NO.	
DKNEMRG_HSRR_MERGE_EXIT	DKNEMRG6
Provides control over EMRG's HSRR merge behavior.	
DKNEMRG_KEY_ENTRY_MERGE_EXIT	DKNEMRG7
Provides control over EMRG's Key Entry merge behavior.	

DKNEMRG User Exit Calling Parameters

Although each different DKNEMRG user exit is called in a different way, they all share some common properties. Each receives a common control block, EMRGX-CB, which is mapped by the copybook DKNEMRGL. Among the fields contained in the control block is EMRGX-CB-FUNC. This is a four-byte character function code indicating what operation the exit is expected to perform. Typical of such functions include:

INT The calling program is starting up and initializing; it is asking the user exit to do the same.

PRC This is a process call; most often the calling program issues such a call for each CPCS codeline it handles.

TRM The calling program is ending and is asking the exit to perform whatever wrap-up is necessary.

Depending on the exit, other more specialized functions may have to be handled.

Figure 4-57 is a complete list of all function codes each user exit is expected to support:

Figure 4-57 (Page 1 of 4). EMRG user exits, function, and purpose

User Exit Name
DKNEMRG_MAIN_EXIT
Function: INT
Purpose: Initialize. This function is invoked when DKNEMRG starts running.

Figure 4-57 (Page 2 of 4). EMRG user exits, function, and purpose

User Exit Name
Function: TRM **Purpose:** Terminate. This function is invoked when EMRG has finished its work and is shutting down.
DKNEMRG_USER_INPUT_EXIT
Function: AINT **Purpose:** Initialize. This function is invoked when EMRG first calls on its internal DKNEMRGA module to edit ESM or autostart input or, in the case of a manual start, to display an input screen.
Function: APRC **Purpose:** Process. This function asks the exit to decide which process option EMRG should use -- the one implied by the string name, the one entered by the user (assuming the user keyed a valid option, of course), or one selected by the exit itself.
DKNEMRG_POCKET_MERGE_EXIT
Function: BINT **Purpose:** Initialize. This function is invoked when EMRG first calls on its internal DKNEMRGB module to start a Pocket Merge.
Function: BMAT **Purpose:** Matched item. This function is called each time Pocket Merge successfully matches an input string codeline to one in the R-string. Possible responses include suppressing one, or both, codelines from being written to the M-string.
Function: BFCL **Purpose:** Free codeline. This function is called for each R-string item left over at the end of a Pocket Merge. These are not necessarily free items; they might be control documents, such as the OLRR statistics record, that appear only in R-strings. For each item, the exit can choose to write the item to the end of the M-string, or to bypass it.
Function: BFPB **Purpose:** Free piggyback. This function is called for each piggyback left over at the end of a Pocket Merge. For each item, the exit can choose to write the item to the end of the M-string, or to bypass it.
Function: BPRC **Purpose:** Process. This function is called for each codeline the Pocket Merge module DKNEMRGB is about to write to the M-string. The exit can choose to permit the write, or to order DKNEMRGB to bypass it.
Function: BTRM **Purpose:** Terminate. This function is invoked when EMRG's internal module DKNEMRGB has completed a Pocket Merge and has written an M-string.

Figure 4-57 (Page 3 of 4). EMRG user exits, function, and purpose

User Exit Name
DKNEMRG_CONSOL_REJ_MERGE_EXIT **Function:** CINT **Purpose:** Initialize. This function is invoked when EMRG first calls on its internal DKNEMRGC module to start a Consolidated Reject Merge. **Function:** CEXP **Purpose:** Expect correction. This function is called for each input string codeline read by the Consolidated Reject Merge module. The user exit is to indicate whether the item can be immediately written to the M-string, or whether it has to be tabled up for matching to a corresponding R-string item. **Function:** CMAT **Purpose:** Matched items. This function is called each time Consolidated Reject Merge successfully matches an input string codeline to one in the RC-string. Possible responses include suppressing one, or both, codelines from being written to the M-string. **Function:** CFNC **Purpose:** Find next codeline. The Consolidated Reject Merge module DKNEMRGC has read the next codeline in the input string; however, this still may not be the correct point at which to insert an RC-string piggyback. Batch statistics records, for example, should never be separated from the batch records with which they are associated. The user exit can either tell DKNERMGC to stop here, or to continue scanning for the next input string paper item. **Function:** CFCL **Purpose:** Free codeline. This function is called for each R-string item left over at the end of a Consolidated Reject Merge. These are not necessarily free items; they might be control documents, such as the OLRR statistics record, that appear only in R-strings. For each item, the exit can choose to write it to the end of the M-string, or to bypass it. **Function:** CFPB **Purpose:** Free piggyback. This function is called for each piggyback left over at the end of a Consolidated Reject Merge. For each item, the exit can choose to write it to the end of the M-string, or to bypass it. **Function:** COUT **Purpose:** Out. This function is called for each codeline the Consolidated Reject Merge module DKNEMRGC is about to write to the M-string. The exit can choose to permit the write, or to order DKNEMRGC to bypass it. **Function:** CSDE **Purpose:** SDE change. This function is called if, at the end of Consolidated Reject Merge, there are still input string codelines that have not been repaired and/or repaired pockets that still contain rejects. The Consolidated Reject Merge module DKNEMRGC normally displays error message(s); however, the user exit can choose to suppress the error instead by changing the appropriate pocket flags in the M-string SDE.

Figure 4-57 (Page 4 of 4). EMRG user exits, function, and purpose

User Exit Name
Function: CTRM **Purpose:** Terminate. This function is invoked when EMRG's internal module DKNEMRGC has completed a Consolidated Reject Merge and has written an M-string.
DKNEMRG_HSRR_MERGE_EXIT **Function:** EINT **Purpose:** Initialize. This function is invoked when EMRG first calls on its internal DKNEMRGE module to complete a HSRR merge.
Function: EPRC **Purpose:** Process. This function is called for each codeline the HSRR Merge module DKNEMRGE is about to write to the M-string. The exit can choose to permit the write, or to order DKNEMRGE to bypass it.
Function: EFCL **Purpose:** Free codeline. This function is called for each 50M-string item left over at the end of a HSRR merge. These are not necessarily free items; they might be control documents, such as the OLRR statistics record, that transferred in from the R-string. For each item, the exit can choose to write it to the end of the M-string, or to bypass it.
Function: ESTG **Purpose:** Override output string name. This function allows the user to change Pass Pocket 4 of the output M-string name. This allows users of systems, such as HPTS KEY, to create an intermediate 01-M string until all passes of HPTS KEY have been completed. At that time, Option 7, Force 00-M string, can be run to create the final 00-M string.
Function: ETRM **Purpose:** Terminate. This function is invoked when EMRG's internal module DKNEMRGE has completed a HSRR merge and has written an M-string.
DKNEMRG_KEY_ENTRY_MERGE_EXIT **Function:** KINT **Purpose:**Initialize. This function is invoked when EMRG first calls on its internal DKNEMRGK module to start a Key Entry merge.
Function: KRST **Purpose:** R-string. This function is invoked for each Key Entry R-string codeline that is selected for writing to the M-string. The exit's options are limited to modifying the codeline before it is written.
Function: KTRM **Purpose:** Terminate. This function is invoked when EMRG's internal module DKNEMRGK has completed a Key Entry merge and has written an M-string.

Using the Sample User Exits

Most (though not all) of the sample user exit code has been placed under the control of U-TEST switches, located at the top of each exit's WORKING-STORAGE area. By changing the appropriate switch, recompiling, and refreshing the exit with DKNEXIT's "F" function, you can quickly activate and deactivate various sample user exit features. Some of these features may be of use at your installation. If so, you can browse the sample source code, look up all the references to the appropriate switch, and from this discover everything you need to import to your own user exit to achieve the same effect.

The listings of DKNEMRG user exits below show all the U-TEST switches located in each sample user exit, and discusses the effects they have when turned on. Unless otherwise specified, these switches are shipped turned off:

DKNEMRG_MAIN_EXIT—DKNEMRG1

U-TEST-SWITCH-1-01

If set to "2", this switch causes EMRG's HSRR codeline data match module DKNEMRGD to allocate a CPCS spool for its unmatched reject report. If set to "3", DKNERMGD allocates a JES spool instead. The shipped setting is "D" (default), which causes DKNEMRGD to take its default action of allocating a CPCS spool.

U-TEST-SWITCH-1-02

If on, this switch causes EMRG's Pocket Merge module DKNEMRGB to use the DI-CPU-FLAG1 tracking byte, as well as the item sequence number, when comparing input string codelines against R-string codelines to see if they match. Note that this feature can reduce the chance of EMRG mistaking an autoselect for a neighboring good item.

U-TEST-SWITCH-1-03

If on, this switch causes EMRG to autostart RLST at the end of a pocket or Consolidated Reject Merge. It also autostarts PLST if the result of the merge is a prime-pass 00M-string. Note that no program autostarts if you are running ESM. In such an environment, use ESM workflows to start the desired programs.

U-TEST-SWITCH-1-04

If this switch and U-TEST-SWITCH-1-03 are both on, EMRG always autostarts an RLST at the end of its run, even if it had not been conducting a pocket or Consolidated Reject Merge. Note that no program autostarts if you are running ESM. In such an environment, use ESM workflows to start the desired programs.

U-TEST-SWITCH-1-05

If this switch and U-TEST-SWITCH-1-03 are both on, EMRG always autostarts a PLST after writing a prime-pass 00M-string, even if it had not been conducting a pocket or Consolidated Reject Merge. Note that no program autostarts if you are running ESM. In such an environment, use ESM workflows to start the desired programs.

U-TEST-SWITCH-1-06

If this switch and U-TEST-SWITCH-1-03 are both on, EMRG always autostarts an MDIS after writing a prime-pass 00M-string. Note that no program autostarts if you are running ESM. In such an environment, use ESM workflows to start the desired programs.

U-TEST-SWITCH-1-07

If on, this switch causes informational messages (message numbers 00000 through 09999) to be sent only to the SCROLL data set, rather than to both the SUPV terminal and to SCROLL. Note that this switch is shipped turned on.

U-TEST-SWITCH-1-08

If on, this switch causes EMRG's Pocket Merge module DKNEMRGB to always create an 01M-string, even when all rejects have been merged.

DKNEMRG1 also contains the following hard-coded behavior:

- On an INT call, selects certain DITYPEs recorded in the DKNEMRGT table and alters their properties (is this a DKNCRZC *uncompressed* or a DKNCRDI *compressed* codeline; was there an actual paper document read for it; is it correctable; does it contain an item sequence number). Please do not use these alterations as-is; they are provided only as examples of how you can make similar changes for custom documents in your own institution.
- On an INT call, sends messages to the SCROLL data set describing how the U-TEST switches have been set up.

DKNEMRG_USER_INPUT_EXIT—DKNEMRG2

U-TEST-SWITCH-2-01

If set to "U", this switch causes EMRG to accept the merge option the user had keyed on the input screen (that is, after editing it to make sure it is a 1, 2, or 3). If set to "X", this switch causes EMRG to always perform option 3, Consolidated Reject Merge, no matter what string or option the operator had entered. The shipped setting is "D" (default), which causes EMRG to decide what option to apply based on the name of the string the operator had entered.

DKNEMRG2 also contains the following hard-coded behavior:

- On an AINT call, this switch tests to see if EMRG had been manually started by an operator named JED, or if it had been manually started by an operator named KEY. For JED, it sets U-TEST-SWITCH-2-01 to "U" (that is, it causes EMRG to accept JED"s keyed option rather than compute one from the string name); for KEY, it not only sets U-TEST-SWITCH-2-01 to "X" (EMRG always performs option 3), but it also causes EMRG to release the terminal the instant it begins the actual merge, rather than wait until the merge is complete.

DKNEMRG_POCKET_MERGE_EXIT—DKNEMRG3

U-TEST-SWITCH-3-02

If on, this switch causes EMRG to keep each free paper-document codeline passed to DKNEMRG3 as part of a BFCL call. The default action is to bypass the free codelines.

U-TEST-SWITCH-3-03

If on, this switch causes EMRG to keep each free piggyback passed to DKNEMRG3 as part of a BFPB call. The default action is to bypass the free piggybacks.

U-TEST-SWITCH-3-04

If on, this switch causes the EMRG Pocket Merge module DKNEMRGB to issue ENQ and DEQ calls to prevent other programs from trying to create the same M-string (or work M-string) that it is in the process of writing. Note that this switch is shipped turned on.

DKNEMRG3 also contains the following hard-coded behavior:

- On a BINT call, displays a message indicating that it had started.
- On a BMAT call, marks R-string items that have been keyed or reconciled, but are not in the system reject pocket, as low-speed (corrected) items.
- On a BPRC call, bypasses X'95' original reject items that are either low-speed (corrected), keyed, or reconciled. In other words, the only original reject passed into the M-string is the one from the input string.
- On a BTRM call, displays a message summarizing the total number of times the user exit had been called.

DKNEMRG_CONSOL_REJ_MERGE_EXIT—DKNEMRG4

U-TEST-SWITCH-4-01

If on, this switch causes DKNEMRG4 to accept CSDE function calls from EMRG's consolidated reject repair module, DKNEMRGC. DKNEMRG4 will then suppress unrepaired codelines/rejects in repaired pocket errors by forcing off the appropriate pocket flags. DKNEMRG4's normal behavior is to ignore CSDE calls, with the result that any such errors generate error messages.

U-TEST-SWITCH-4-04

If on, this switch causes EMRG to keep each free paper-document codeline passed to DKNEMRG4 as part of a CFCL call. The default action is to bypass the free codelines.

U-TEST-SWITCH-4-05

If on, this switch causes EMRG to keep each free piggyback passed to DKNEMRG4 as part of a CFPB call. The default action is to bypass the free piggybacks.

U-TEST-SWITCH-4-06

If on, this switch causes the EMRG Consolidated Reject Merge module DKNEMRGC to issue ENQ and DEQ calls to prevent other programs from trying to create the same M-string (or work M-string) that it is in the process of writing. Note that this switch is shipped turned on.

DKNEMRG4 also contains the following hard-coded behavior:

- On a CINT call, displays a message indicating that it had started.
- On a CINT call, if the input string is an I-string, DKNEMRG4 resets all the repaired flags so as to make it seem as though no pocket has yet been repaired.
- On a CEXP call, for each paper input string codeline, DKNEMRG4 hunts the R-string buffer for an unmatched codeline that has identical sequence number and DI-CPU-FLAG1 values, and which has at least one field different (that is, corrected). If such an R-string codeline is found, DKNEMRG4 instructs

DKNEMRGC to expect an R-string correction; otherwise, it instructs DKNEMRGC to go ahead and write the input string codeline out.

- On a CMAT call, marks R-string items that have been keyed or reconciled, but are not in the system reject pocket, as low-speed (corrected) items.
- On a CFNC call, DKNEMRG4 instructs the Consolidated Reject Merge module DKNEMRGC to continue reading through the input string until it encounters either a paper document (that is, a codeline corresponding to actual paper read through the sorter), or at the very least a paper control document that was deleted in balancing.
- On a COUT call, bypasses X'95' original reject items that are either low-speed (that is, corrected), keyed, or reconciled. In other words, the only original reject passed into the M-string is the one from the input string.
- On a CTRM call, displays a message summarizing the total number of times the user exit had been called.

DKNEMRG_HSRR_MERGE_EXIT—DKNEMRG6

U-TEST-SWITCH-6-01

If on, this switch causes the EMRG HSRR merge module DKNEMRGE to issue ENQ and DEQ calls to prevent other programs from trying to create the same M-string (or work M-string) that it is in the process of writing. Note that this switch is shipped turned on.

DKNEMRG6 also contains the following hard-coded behavior:

- On a EINT call, displays a message indicating that it had started.
- On a BPRC call, bypasses high-speed (captured on HSRR) uncompressed X'95' original reject items. The only original reject passed into the M-string is the one from the prime-pass capture string.
- On an ETRM call, displays a message summarizing the total number of times the user exit had been called.
- On an ESTG call, the code as delivered is commented out. It is up to the user to update or change the code to follow the institution's requirements. This changes Pass Pocket 4 of the output 00-M string to 01.

DKNEMRG_KEY_ENTRY_MERGE_EXIT—DKNEMRG7

DKNEMRG7 also contains the following hard-coded behavior:

- On a KINT call, displays a message indicating that it had started.
- On a KINT call, selects certain DITYPEs recorded in the DKNEMRGT table and alters their properties (is this a DKNCRZC *uncompressed* or a DKNCRDI *compressed* codeline; was there an actual paper document read for it; is it correctable; does it contain an item sequence number). Please do not use these alterations as-is; they are provided only as examples of how you can make similar changes for custom documents in your own institution.
- On a KRST call, marks R-string items that have been keyed or reconciled, but are not in the system reject pocket, as low-speed (corrected) items.
- On a KTRM call, displays a message summarizing the total number of times the user exit had been called.

DKNMRGE Programmable Switches

This programmable switch controls whether a batch or block slip containing field errors on prime pass will be marked automatically as an original reject document (X'95') if it is not matched on a HSRR pass. The default is that the batch or block slip will not be marked as an original reject document. If you want this option, change the switch value to 0.

```
77  BAD-PRIME-CTL-DOC-ALLOWED-SW        PIC X VALUE ‘1’
    88  BAD-PRIME CTL-DOC-ALLOWED             VALUE ‘1’
    88  BAD-PRIME-CTL-DOC-NOT-ALLOWED         VALUE ‘0’
```

DKNMRGK–Document-Processing User Exit

You can specify this exit in the USREXIT parameter of the DKNBLDL macro. It permits you to control the setting of the DILOW and DI34KEY bits in the mass data set codeline. Unpredictable results might occur if you change other portions of the record.

DKNMRGK performs the codeline matching on High Performance Transaction System image-repaired strings. DKNMRGK passes each codeline from the 00-R-string to the DKNMRGK user exit for processing.

A sample document-processing user-exit routine, DKNMRGKX, is provided for this use and may be found in CPCS.V1R11.SAMPLIB. The prologue of DKNMRGKX contains a README file that contains the most up-to-date information about the user exit.

DKNMRGK–Selectable Functions for DKNMRGK

The DKNMRGE task calls DKNMRGK to build the 00-M-string in a High Performance Transaction System image environment when DKNMRGE detects the presence of a string header record (X'80' x X'80'); this string header record must be first in the 00-R-string. You can set flags in DKNMRGK to eliminate original reject documents made by DKNMRGE.

The purpose of the S-USER-DELETE-MULT-X95 switch is to delete interim original reject codelines that were produced by MRGE. The default is to delete the interim original reject (X'95') codelines. The values for this switch are:

```
S-USER-DELETE-MULT-X95          PIC  X(01) VALUE 'Y'
88  S-DONT-DELETE-MULT-X95                 VALUE 'N'
88  S-DELETE-MULT-X95                      VALUE 'Y'
```

For example, consider the following steps:

1. Run MICR and capture an UNENCODED entry.
2. Run DIST to create the system reject D-string.
3. Run OLRR to create an 01-R-string.
4. Run MRGE to create an 01-M-string.
   (This step produces original rejects from OLRR.)
5. KEY the 01-M string.
6. Run MRGE to create a 00-M string.
   The switch, S-USER-DELETE-MULT-X95, becomes effective.

A comparison of the resulting M-string follows:

```
DON'T DELETE                DELETE
----- ------                 ------
X'95' - from prime          X'95' - from prime
X'95' - from OLRR           X'00' - from Key
X'00' - from Key
```

DKNPLST, DKNSLST, and DKNXLST: Controlling Entry Master-List Reports

The entry master-list task (DKNPLST) controls the format of the entry master-list reports. These formats are selected by flags that you can specify in the DKNPLST program and by the format of the mass data set (standard or expanded record length). You can set the flags by changing the affected fields in the program, compiling the new program, and using CHAP PLST to load a new version in the BLDL table.

For a standard MDS record size (50 bytes), DKNPLST provides two types of report layouts:

Type A This layout provides for a 3-column report in which each column consists of the following fields:

- Sequence number
- Routing number (field 5) or, for on-us items, account number (field 3)
- Amount (field 1)
- Pocket number.

You can select this report type by setting the 300-TYPE-OF-REPORT field to a value of A and the 300-PLST-NO-OF-COLMS field to a value of 3 for a 3-column report.

Type B This layout provides a 2-column report in which each column consists of the following fields:

- Sequence number
- Serial number (field 7)
- Routing number (field 5)
- Account number (field 3)
- Process control (field 2)
- Amount (field 1)
- Pocket number.

You can select this report type by setting the 300-TYPE-OF-REPORT field to a value of B and the 300-PLST-NO-OF-COLMS field to a value of 2 for a 3-column report.

For a nonstandard MDS record length, DKNPLST provides the following report layouts:

Type A or B Each of these report layouts uses DKNMDXR to print a 1-column report that contains the following fields:

- Sequence number
- Routing number (field 5)

- Account number (field 3)
- Amount (field 1)
- Flag 2
- Pocket number.

You can select one of these layout types by setting the 300-TYPE-OF-REPORT field to a value of A or B. DKNMDXR formats the report in one column, so it is not necessary to set the value for the 300-PLST-NO-OF-COLMS field.

DKNXLST, as supplied with CPCS, automatically causes the printing of both prime-pass and subsequent-pass exception listings for all entries throughout the day. For prime-pass entries, DKNXLST produces a detailed item listing only for out-of-balance batches. For subsequent-pass entries, DKNXLST produces a detailed item listing only for out-of-balance tracer groups.

CPCS provides a skeleton format, which you can change or expand according to your financial institution's requirements. The skeleton format can determine the type of master list to print, based on the following:

- Time of day
- Cycle
- Sort pattern type.

You can add other parameters. DKNXLST is accessed on either automatic or manual starts.

Before assembling DKNXLST, you can insert the values of the sort patterns. The code supplied is essentially a skeleton in which you can insert the sort-pattern numbers with the proper cycles and set the return to indicate the type of report produced by DKNPLST or DKNSLST. You can also cause the program to default and print the exception lists at all times.

Sort patterns should be set by EQU statements to an equivalent 1-byte binary value. They should be checked for under the correct cycle, and the type of report you want should be branched to on an equal condition. The branches are:

PLSTSFLL	Produces full listing only for DKNPLST
PLSTSBTH	Produces full and summary listings for DKNPLST
PLSTSXCP	Produces exception print by batch for DKNPLST (default)
PLSTSEXT	Causes DKNPLST to end
SLSTSDET	Produces full listing only for DKNSLST
SLSTSSUM	Produces summary listing only for DKNSLST
SLSTSXCP	Produces exception print by batch for DKNSLST (default).

Processing Description

DKNXLST contains two separate entry points, DKNXLST and DKNYLST. A common processing section, DKNZLST, is accessed through each entry point. DKNPLST calls entry point DKNXLST and DKNSLST calls entry point DKNYLST. Each entry point sets a switch (LISTSW) that DKNZLST uses to determine the calling program.

DKNPLST or DKNSLST passes two parameters, the cycle and the sort type, to DKNXLST. (You can add additional parameters.) DKNZLST accesses the time-of-day and bases the processing decision on these three parameters. DKNXLST passes the decision back to DKNPLST or DKNSLST.

The first check is based on the time of the run. If, because of time restraints, a certain type of printout (for example, exception) is the only type wanted after a certain hour, make the hour comparison value the value of PLSTLATE or SLSTLATE.

If the branch on the time is not taken, DKNXLST checks the value of the cycle parameter and a branch on a valid cycle of 0 to L is taken.

On valid cycles, DKNXLST branches to the proper cycle routine. Under each cycle, DKNXLST makes the comparison on the sort-pattern type. DKNXLST sets the sort patterns as EQUs of a 1-byte binary value converted to the hexadecimal values of the sort-pattern numbers.

DKNKILL and DKNPLST: Modifying Report Layouts

You can modify the report layouts for the DKNKILL and DKNPLST programs to meet your installation's requirements. To modify the layout, change the values documented after the 300-USER-SUPPLIED-INPUT field in the working storage section of the DKNKILL and DKNPLST programs. Research the modified layout to ensure that the printer and paper used are still appropriate for the data.

For the DKNKILL report, you can also select an expanded format that shows portions of MICR fields (1, 2, 3, 5, and 7) as delivered. This layout provides for a 2-column report. To select the report, set the value of the 310-NEW-REPORT-TYPE-SW field (immediately precedes the 340-FIELD-STOR field) to YES. If the value of the 310-NEW-REPORT-TYPE-SW field is set to YES, also set the value of the 300-KILL-NO-OF-COLMS field to 2. The expanded report layout can also be modified. Research the modifications to ensure that the printer and paper used are appropriate for the data.

DKNPRIOX–System Manager Priority User Exit

System manager provides a method of prioritizing units of work with the user exit, DKNPRIOX. For more information, see the *CPCS System Manager User's Guide*.

DKNSCATX - String Concatenation User Exit

You can specify a user exit in the USREXIT parameter of DKNSCAT's APCB entry in DKNBLDL. This exit allows you to customize and control the behavior of SCAT in many different ways. For example, you could:

- Control which codelines are selected from the Reject D-string and the partial R-string(s) for sequence-matching against each other
- Prepare a report listing uncorrected rejects, duplicate corrections, missing and free codelines, and any other anomalies that may be uncovered during concatenation
- Control how duplicate corrections are handled: Which duplicates (if any) pull into the final R-string, and which are suppressed
- Decide final disposition of any interim R-strings created during concatenation. You can decide to delete them, or retain them for archival purposes.

The user exit is called out of the DKNSCAT subroutine DKNSCA3. This means it is invoked only during concatenation of the System Reject Pocket.

Terminology

Duplicate Corrections
A codeline that has been corrected two or more times. This could happen, for example, if two or more OLRR operators inadvertently processed the same portion of a reject D-string.

Final R-string
The target R-string that SCAT tries to create. Contains one correction for every codeline in the original reject D-string. Typically identified by a `00` in the fourth pocket of its string name.

Free Codelines
Records found in partial R-strings that could not be sequence-matched back to Reject D-string codelines.

Interim R-string
If SCAT determines that it cannot construct a final R-string, it builds an interim R-string instead. This represents various partial R-strings that are still missing. Typically identified by a `99` in the fourth pocket of its string name.

Missing Codelines
Reject D-string records that were never successfully sequence-matched to any partial R-string codeline.

Operator ID Record
One or more control documents at the top of final and interim R-strings that name the partial R-strings they are composed of. These records have a DITYPEI of `X'99'`, and each byte after the DISEQ field contains an operator ID in binary format. The last operator ID byte is followed by a `X'FF'`.

Partial R-string
The individual R-strings prepared out of OLRR (or other reject repair tool used in your installation). It is SCAT's job to construct a final R-string by assembling all the partial R-strings in the correct order. Typically, the fourth pocket of the string name for an partial R-string contains the operator ID (01 through 98) of whomever it was that created the string.

Uncorrected Rejects
Codelines in partial R-strings that still contain invalid fields (that is, fields whose validity bits are turned off).

String Concatenation User Exit Parameters

The following parameters are passed to the String Concatenation user exit on every invocation. Note that not all parameters are useable on all calls. For example, during user exit INIT, only the Document Use Table is available for manipulation. The Codeline Table has just been cleared, and the Operator, Match, and Corrections Tables have not yet been loaded.

CONTROL BLOCK

Mapped by: DKNSCATU

This is the main control block for communicating between SCAT and its user exit. It contains the following fields:

FUNCTION-CODE

The specific call SCAT is making to its user exit. This is a PIC X(04) field set to one of the following values:

INIT SCAT is starting for the first time. The Codeline Table has been cleared and the Document Use Table loaded. No other tables are available for manipulation at this time.

MAIN SCAT has added a partial R-string record to the Codeline Table. The record could be in the DOC, NDOC, DUP, or PIGY regions of the table (CDLN-CURRENT SUB tells you exactly where it is). If this is a DOC record, MATCH-TABLE-SUB points to the Match Table entry to which it matched. RESULT-CODELINE-MATCH and RESULT-CODELINE-CORR are set; if the reject is uncorrected, the Correction Table tells you exactly what is wrong. (If the record came from an OLRR bypass run, RESULT-CODELINE-CORR always indicates the item is 100% correct; however, its errors are still available for inspection in the Corrections Table.) At this time, the DOC, NDOC, and DUP regions of the Codeline Table are threaded together, but piggybacks have been stored in the PIGY region as short independent chains of items. Refer to the description of the Codeline Table below for more information.

PIGY With all R-string codelines now in the Codeline Table, SCAT begins linking in the piggyback chains. A PIGY call is made every time SCAT successfully inserts a chain. SCATC-CDLN-CURRENT-SUB and SCATC-CDLN-DOC-SUB-2 point to, respectively, the first and last items in the piggyback chain; SCATC-CDLN-PREV-SUB and SCATC-CDLN-NEXT-SUB point to, respectively, the items before and after the inserted chain. A user exit PIGY call is made for each chain successfully relinked.

BYPS BYPS calls are made only when SCAT is run in bypass mode. All available partial R-string codelines have been added to the Codeline Table; all piggyback chains have been linked in. SCAT now begins filling in missing codelines by copying Reject D-string codelines from the Match Table. A BYPS call is made for each reject D-string record added. SCATC-CDLN-CURRENT-SUB points to the Codeline Table slot where the record was inserted.

LAST SCAT has made its last MAIN, PIGY, and/or BYPS call. The Codeline Table is now complete and ready to be read out into the final (or interim) R-string. The read-out begins as soon as this user exit call returns to SCAT.

DELE SCAT makes this call only under the following circumstances:

1. It is ready to create a final R-string;
2. Some earlier run of SCAT created an interim R-string.

Most users will want at this time to delete the interim R-string. Some users might want to retain it for archival purposes, or even copy it to another string or file. Note that SCAT does NOT call this

exit if it is creating an interim R-string and some earlier SCAT created a previous interim R-string. In such circumstances SCAT on its own deletes the old interim R-string and replaces it with the new.

FINL SCAT has completed all processing and is preparing to quit. The FINL call is made so that the user exit, too, can complete processing and prepare to quit.

RETURN-CODE

Before returning, the user exit should set this PIC S9(04) COMP field to either 0, 4, or some value 8 or greater.

0 Normal return.

4 Requests SCAT not to make any further calls to the user exit.

8 or greater

Requests SCAT to abend. It ends with an error message containing the returned RETURN-CODE.

D-STRING-NAME

The Reject D-string name, in condensed 17-byte format. That is, all dashes are stripped out, and the System Reject Pocket, if present, is represented as two bytes of HIGH-VALUES.

OP-R-STRING-NAME

The partial R-string name, in condensed 17-byte format. That is, all dashes are stripped out, and the System Reject Pocket, if present, is represented as two bytes of HIGH-VALUES.

R-STRING-NAME

The final (or interim) R-string name, in condensed 17-byte format. That is, all dashes are stripped out, and the System Reject Pocket, if present, is represented as two bytes of HIGH-VALUES.

D-STRING-SSB

Pointer to the SSB for an open Reject D-string.

OP-R-STRING-SSB

Pointer to the SSB for an open partial R-string.

R-STRING-SSB

Pointer to the SSB for an open interim or final R-string.

RESULT-CODELINE-MATCH

During MAIN calls, this PIC X(01) field indicates what happened when SCAT tried to sequence match the partial R-string codeline to the Reject D-string sequence numbers in the Match Table:

M This codeline was successfully matched. SCAT has filed it in the DOC area of the Codeline Table.

Space This is a free codeline; it failed to match any Match Table sequence. SCAT has filed it in the NDOC area of the Codeline Table.

D This is a duplicate correction; it matched to a sequence number that was already matched. SCAT has filed the item in the DUP area of the Codeline Table.

The user exit is free to change this field if there are grounds to countermand SCAT's decision. If it does so, the exit should also consider moving the item to a different area of the Codeline Table.

RESULT-CODELINE-CORR

During MAIN calls, this PIC X(01) field summarizes SCAT's analysis of the partial R-string record's validity:

Y This is either a 100% valid codeline, or this codeline came from a partial R-string created out of OLRR bypass mode.

N This is an uncorrected reject; some fields are invalid. The Corrections Table will describe the error in greater detail. Please see the description of the Corrections Table below for more information.

The user exit is free to change this field if there are grounds to alter SCAT's analysis. If it does so, the Exit should also consider editing the offending document, or moving it to a different location in the Codeline Table.

TRACKING-BYTE-SW

This switch allows the user to disable DKNSCAT using the CPU tracking to uniquely match an autoselect with the same sequence number.

ON SCAT compares DICPU(1:1) in the R-string to the DICPU(1:1) in the reject D-string.

OFF SCAT compares the ISN of the R-string to the ISN in the reject D-string. This may cause a false match when matching autoselects with the same sequence number.

SCRATCH-AREA

100 bytes free for use by the user.

OPTIONS

Mapped by: DKNSCATO

SCAT incorporates a copybook, DKNSCATK, that lists eighteen different combinations of options controlling how it is to run. Each combination includes a twelve-byte key holding such information as the reject D-string pass code, whether the string came from a normal or a HSRR capture, and how SCAT was started. SCAT selects one of the options combination records based on its key, and uses it for the duration of its processing. It also passes it to its user exit via copybook DKNSCATO:

Position	Length	Value	Meaning
1	2		Unused.
3	1	1, 2, or 3	The pass the reject D-string codelines were captured under.
4	1	SPACE	Normal capture.
		H	HSRR capture.
5	1		Unused.
6	6	MANUAL	SCAT was manually started.
		MULTI	SCAT was manually started with the **,**M option.
		AUTO	AUTO was autostarted, either by a CPCS task or through the Enhanced System Manager.
12	1		Unused.

Position	Length	Value	Meaning
13	1	U	Use interim R-string. If there is a pre-existing interim R-string, SCAT reads it for operator ID records and includes the partial R-strings named therein in its concatenation. This option is used during MANUAL and ESM starts.
		I	Ignore interim R-string. The only partial R-strings concatenated are those explicitly supplied with the ,M input screen.
14	1		Unused.
15	1	Y	Required existence. It is an error if SCAT fails to find an partial R-string it is looking for. This option is used during ,M and ESM starts, when the exact partial R-strings to concatenate are known in advance.
		N	SCAT concatenates all possible partial R-strings in succession until it fails to find one. It then creates its final (or interim) R-string and quits. This option is used during MANUAL starts, when it is not known in advance how many partial R-strings exist to concatenate.
16	1		Unused.
17	1	Y	SCAT uses exact sequence matching. If an partial R-string codeline matches to a reject D-string sequence already matched to, then the codeline is considered a duplicate correction. This option is used only when it is known with certainty that all reject D-string sequence numbers are unique.
		N	SCAT uses probable sequence matching. If an partial R-string codeline matches to a reject D-string sequence already matched to, then SCAT continues scanning the Match Table looking for a better match. This option is used when it is possible one or more reject D-string codelines could have the same sequence number.
18	1		Unused.
19	1	D	Reject D-string ordering. The partial R-string codelines are assembled in the order of the reject D-string codelines to which they match. This option is legal only with exact sequence matching, and is used only if SCAT has not been given a different assembly order with the ,M screen.
		O	Operator ID ordering. The partial R-string codelines are assembled in the order they are encountered in the various partial R-strings.
20	1		Unused.
21	2	00	The code used in the pass pocket of System Reject partial R-strings. This is the pocket that, in reject D-strings and in final or interim R-strings, is represented by an R (24-byte string name format) or by HIGH-VALUES (17-byte string name format).

Position	Length	Value	Meaning
23	1		Unused.
24	2	99	The code used in the fourth pocket of interim R-strings.
26	1		Unused.
27	2	00	The code used in the fourth pocket of final R-strings.
29	1		Unused.
30	1	Y	If a piggyback is encountered that has its DI30DOCH flag turned on (bit X'80' in DIFLAG3 Byte 0), abort the concatenation with an error message.
		N	If a piggyback is encountered that has its DI30DOCH flag turned on (bit X'80' in DIFLAG3 Byte 0), ignore it and carry on with the concatenation.
31	1		Unused.
32	1	Y	If a piggyback is encountered that has its DI30DOCH flag turned on (bit X'80' in DIFLAG3 Byte 0), reset the flag.
		N	If a piggyback is encountered that has its DI30DOCH flag turned on (bit X'80' in DIFLAG3 Byte 0), do nothing to the flag.
33	48		Unused.

OPERATOR TABLE

Mapped by: DKNSCATI

This table names all of the partial R-strings that SCAT is to access, and, in the case of partial R-string ordering, also lists the order in which they are to be assembled. There are 99 entries. Each entry consists of the following:

Position	Length	Value	Meaning
1	2	01–98	An operator ID. SCAT tries to concatenate in a partial R-string with this ID in its name.
		HIGH-VALUES	Indicates the end of the table.
3	1	Y	It is an error if this partial R-string does not exist.
		N	If this partial R-string cannot be found, the end of the concatenation has been reached. Create a final (or interim) R-string and quit, just as if the HIGH_VALUES end-table marker had been encountered.

The table is constructed in two phases, both of which occur between the user exit INIT call and the first User Exit MAIN call.

In the first phase, if the Use Interim R-string option has been selected, SCAT looks for one and, if it finds it, scans it for operator ID records. The operator IDs extracted from those records are entered into the table with an error code of `Y` (since these operator IDs participated in some earlier SCAT, they obviously must exist).

In the second phase, SCAT merges in operator IDs supplied with its start parameter and/or **,**`M` screen. These IDs are given either an error code of `Y`, if the required existence option is selected; or `N`, if it is not.

In the case of MANUAL starts, the Operator Table consists of every possible operator ID, from 1 to 98. SCAT attempts to concatenate each partial R-string

in succession until it fails to find one; then, it creates its final (or interim) R-string and quits.

The end of the table is always indicated by HIGH-VALUES in an operator ID slot.

DOCUMENT USE TABLE

Mapped by: DKNSCATD

The Document Use Table contains 256 entries, one for each DITYPEI value that can possibly exist. Each entry, in turn, consists of two pairs of flags – one pair for control documents, and one pair for non-control documents.

Position	Length	Value	Meaning
1	1	C	This control document is stored compressed (ZC format).
		U	This control document is stored uncompressed (DI format).
2	1	Y	This control document is to be used in sequence-matching.
		N	This control document is not to be used in sequence-matching.
3	1	C	This non-control document is stored compressed (ZC format).
		U	This non-control document is stored uncompressed (DI format).
4	1	Y	This non-control document is to be used in sequence-matching.
		N	This non-control document is not to be used in sequence-matching.

The compressed/uncompressed flag indicates how a codeline of this type is stored within the Codeline Table – either in ZC format, or in DI format. The sequence-matching flag, meanwhile, controls how SCAT treats the various items it reads initially from the reject D-string, and, later on, from the partial R-string(s).

String	Flag	Behavior
Reject D-string	Y	This codeline is added to the Match Table.
	N	This codeline is ignored.
Partial R-string	Y	SCAT attempts to sequence-match this codeline against codelines from the Match Table. If the match is successful, SCAT stores the partial R-string codeline in the `DOC` region of the Codeline Table. Otherwise, the item goes into the `NDOC` or `DUP` region, as appropriate.
	N	SCAT attempts no sequence-matching. The codeline is instead stored directly into the `NDOC` region of the Codeline Table.

An item should have a `Y` sequence-match flag only if:

1. It has a sequence number
2. It can appear inside an partial R-string.

A DOC item is one that has a `Y` Document Use Table sequence-matching flag, and which successfully sequence-matched against a Match Table codeline. Where in the DOC table a partial-R codeline goes depends on the ordering

option in effect. If we are using Operator ID ordering, then DOC slots are allocated, one after the other, linearly from the top of the table. If we are using Reject D-string ordering, however, then the slot a DOC item is placed in corresponds to the Match Table entry that it matched against. Either way, once all partial R-strings have been read, constructing a final (or interim) R-string is simply a matter of scanning the DOC table from top to bottom and, of course, following pointer chains in and out of the PIGY, DUP, and NDOC tables as necessary.

During the SCAT user exit INIT call, You may modify the Document Use Table to suit your needs.

MATCH TABLE

Mapped by: DKNSCATH

The Match Table is an OCCURS DEPENDING ON table whose size is controlled by MATCH-TABLE-SIZE. Up to 50000 entries can be added to the table, each organized as follows:

Position	Length	Value	Meaning
1	4	Zero	This entry has not yet been sequence-matched.
		Non-zero	Identifies the Codeline Table entry to which this Match Table entry has been sequence-matched.
5	1	C	This codeline has been stored in compressed (ZC) format.
		S	This codeline has been stored in standard, uncompressed (DI) format.
6	3		Unused
9			The codeline itself is stored, beginning here.

SCAT loads the Match Table between the user exit INIT and first MAIN calls by opening up and scanning the reject D-string. For each codeline that has a Y sequence-match flag in the Document Use Table, SCAT adds it to the Match Table. SCAT also increments a MATCH-TABLE-COUNT variable. Later, as SCAT reads and sequence-matches partial R-string codelines, it counts them off by decrementing MATCH-TABLE-COUNT. When this variable reaches zero, SCAT knows it is ready to create a final R-string. If it does not reach zero, then the concatenation is incomplete and SCAT creates an interim R-string instead.

CODELINE TABLE

Mapped by: DKNSCATC

The Codeline Table is an OCCURS DEPENDING ON table whose size is controlled by CDLN-DOC-END. It lies at the heart of all SCAT processing. Every partial R-string codeline read is lodged somewhere in this table, so that constructing a final (or interim) R-string is merely a matter of reading the table out in the proper order.

The Codeline Table actually consists of four sub-tables, or “regions”:

PIGY Piggyback Table

Starts at:	CDLN-PIGY-START
Ends at:	CDLN-PIGY-END - 1
Next available entry:	CDLN-PIGY-SUB

Bidirectionally-linked lists of piggyback items are built here. Initially, each list is anchored by its X'E5' piggyback control record, which is also filed in the PIGY table. That is, each group of piggybacks forms a separate list wholly contained within PIGY. Later, after the last user exit MAIN call but before the LAST call, SCAT scans PIGY and relinks the chains into their appropriate places in the DOC table.

DUP Duplicate Corrections Table

Starts at: CDLN-DUP-START

Ends at: CDLN-DUP-END - 1

Next available entry: CDLN-DUP-SUB

Bidirectionally-linked lists of Duplicate Corrections are built here. Each list is anchored by the DOC table entry that its list elements duplicate. In fact, in any DOC table chain, the duplicates always come first; if the DOC table entry has any NDOC or PIGY items attached to it, they are always threaded on after the DUP entries. All duplicates are initially stored with their CDLN-USED flag turned off, which means normally they don't get written to the final (or interim) R-string. Of course, the user exit is free at any time to turn on or off any CDLN-USED flag it pleases.

NDOC Non-Matchable Document Table.

Starts at: CDLN-NDOC-START

Ends at: CDLN-NDOC-END - 1

Next available entry: CDLN-NDOC-SUB

Bidirectionally-linked lists of Non-Matchable Documents are built here. These are partial R-string records which either had an N Document Use Table sequence-matching flag, or which could not be successfully matched against the Match Table. Each list is anchored by the DOC table entry it was chronologically read after (if the DOC table entry has duplicates, they will be inserted into the list ahead of the NDOC entries).

DOC Document Table

Starts at: CDLN-DOC-START (dummy entry)

Ends at: CDLN-DOC-END

Next available entry: CDLN-DOC-END + 1

This is the main Codeline Table region; it lies at the end so as to take advantage of the Codeline Table's OCCURS DEPENDING ON organization. There are never more DOC table slots allocated than is necessary to hold the DOC items read to date.

A DOC item is one that has a Y Document Use Table sequence-matching flag, and which successfully sequence-matched against a Match Table entry. Where in the DOC table a codeline goes depends on the ordering option in effect. For partial R-string ordering, DOC slots are allocated, one after the other, linearly from the top of the table. For Reject D-string ordering, however, the slot a DOC item is placed in corresponds to the Match Table entry that it matched against. Either way, once all partial R-strings have been read, constructing a final (or interim) R-string is simply a matter of scanning the DOC table

from top to bottom – and, of course, following pointer chains in and out of the PIGY, DUP, and NDOC tables as necessary.

See Figure 4-58 for an illustration of the relationship between the Codeline Table and its companion, the Match Table.

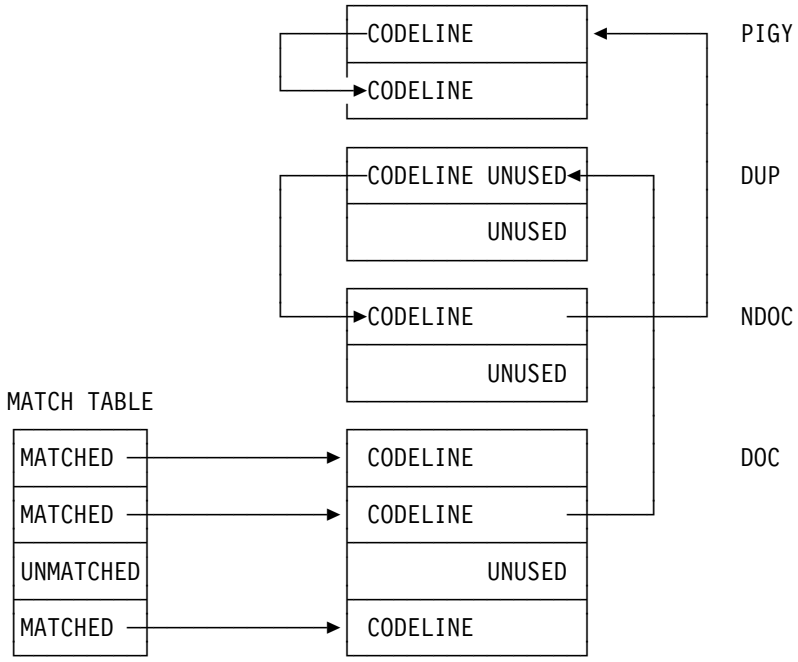

Figure 4-58. Codeline and Match Tables

This illustrates the Codeline Table as it looks after the PIGY chains have been linked in. Note that most DOC entries do not actually anchor link chains; also, that there are many entries with their CDLN-USED flags turned off. Unused entries never get written out to final (or interim) R-strings. Finally, note that at least one Match Table entry remains unmatched. Because of this, SCAT creates an interim rather than a final R-string.

The organization of each Codeline Table entry is as follows:

Figure 4-59. Codeline Entry Organization

Position	Length	Contents
1	4	Fore Pointer — locates the next Codeline Table entry in this chain. If this is a DOC table entry that anchors no chain, then the Fore Pointer points to the DOC table entry itself.
5	4	Back Pointer — locates the previous Codeline Table entry in this chain. If this is a DOC table entry that anchors no chain, then the Back Pointer points to the DOC table entry itself.
9	4	Likely piggyback location. Used only by PIGY table Piggyback Control records, this points to the DOC table entry the record was chronologically found after. In all likelihood this is the probable place where the piggyback chain is to link in. However, if this location does not pan out, then SCAT is prepared to scan the entirety of the Codeline Table to find the true piggyback insert point.
13	1	CDLN-USED flag. If U, this Codeline Table entry is in use and should be written to the final (or interim) R-string. If SPACE, this Codeline Table entry is not in use and should not be written.
14	1	Compressed/Uncompressed flag. Copied from the Document Use table, this flag indicates how the codeline is stored: A C means in compressed (ZC) format, a U means in uncompressed (DI) format.
15	2	Operator ID. Names the partial R-string from which this codeline was read.
17		The codeline itself is stored beginning here.

CORRECTION TABLE

Mapped by: DKNSCATF

For each partial R-string codeline read, SCAT builds a Correction Table showing the status of each of its fields (valid, invalid, or unused). Normally, if an R-string codeline contains invalid fields, it is an uncorrected reject, and SCAT sets the RESULT-CODELINE-CORR flag to N to alert the user exit to this fact. However, if the codeline came out of OLRR Bypass processing, the presence of rejects is probably intentional. In such a case SCAT leaves RESULT-CODELINE-CORR set to Y. The validity flags in this table are never changed, however, so the user exit can always interrogate them to learn the true status of any codeline.

This table is fifteen bytes long, with each byte corresponding to MICR fields 1 through 15, in that order. Each byte can have one of three values (refer to Figure 4-59 on page 4-139).

Value	Meaning
U	This field is not used, as per your MDX generation.
SPACE	This field is valid.
I	This field is invalid.

Altering these fifteen bytes has no affect on SCAT's behavior. To indicate that an uncorrected reject is really corrected (or vise-versa), the user exit must change the setting of RESULT-CODELINE-CORR, and, optionally, edit the offending codeline and/or change its location in the Codeline Table.

APTCB

Mapped by: DKNAPTCB

Application Program Task Control Block, for use in interfacing with various CPCS subtasks and service routines.

Sample String Concatenation User Exit

A sample String Concatenation user exit may be found in CPCS.V1R11.SAMPLIB under the name DKNSCATX. It performs the following functions:

- During initialization, calls on DKNMDXR to set up and open a SCAT Uncorrected Rejects Report.
- For every uncorrected reject, duplicate correction, and free codeline encountered, writes the offending record to the Uncorrected Rejects Report, along with a message documenting the error.
- After SCAT reads the last partial R-string codeline, scans the Match Table for missing codelines and adds them to the Uncorrected Rejects Report.
- Deletes the interim R-string, when SCAT decides it's ready to create a final R-string but discovers that an interim R-string already exists.
- Calls on DKNMDXR to close and write the Uncorrected Rejects Report during SCAT shutdown.

Note: DKNSCATX prepares a report. If you use it as-is on the USREXIT parameter of DKNSCAT's BLDL entry, then you will also need to equip the entry with a PRINT=1 and a CLASS parameter.

Extract Programming

After CPCS captures and processes input documents, the next step is to extract the codeline data records, usually by M-string, from CPCS. You can use these records as input to your posting systems. The coding to do this is referred to as ***extract programming***, which you must supply.

Important!

Power-encoding records must be bypassed by the extract application to prevent double posting. For a description of these records, see "Balancing and Power-Encoding Considerations" on page 4-145.

Extracting Mass Data Set Records

DKNICRE extracts all completely processed M-strings for a specific cycle, and for either a specific bank or all banks, and transfers them to an output tape (or disk) data set. You can use the output data set as input to an archive retrieval system. If the output data set is unreadable when you use it as input, you must perform a selective string recovery from one or more CPCS log tapes to retrieve the M-strings. (See "Data-Set Recovery Procedures" on page 1-33.) Otherwise, use the DKNICRE program as a base or guideline to create your extract programs. You can change the output format to meet the requirements of your posting system.

If the user specifies the MSTRING=1 option in the CPCS system profile member DKNPCPCS parameters (see “DKNPCPCS Profile Member” on page 3-4), the extract program can flag extracted M-strings as being user-processed.

User Application Requirements

CPCS lets you refer to the MDS records in several different ways:

- You can change application tasks to refer to the maximum allowable field sizes. The advantage of this method is that you only need to change the program once to use these enlarged fields. The disadvantage of doing this is that each field has to be parsed to allow for the internal pad characters used for formatting the expanded fields. For additional information on fields and their sizes, see the *CPCS Programming and Diagnostic Guide*.
- You can change application tasks to refer to the redefinable copybooks and DSECTs that describe the MDS record layout. You must recompile or reassemble these programs whenever you redefine the MDS record with the MDX macro. This is the best option for COBOL programs.
- You can change your application tasks to use the MDXREC DSECT; specifically, use the field lengths in the DSECT as parameters to change instructions. For example, an application task can move the length (–1) of a field into a valid move-character instruction.

```
          USING TSTDSECT,R10
*
          XR    R5,R5
          IC    R5,MDXFLD03               GET FIELD LENGTH
          BCTR  R5,0                      DECREMENT FOR EXECUTE
          EX    R5,EXECUTE                MOVE FIELD
          B     WOW
*
EXECUTE   MVC   SAVEACCT(*-*),VARFIELD
*
WOW       DS    OH
*
TSTDSECT  DSECT
SAVEACCT  DS    XL256
VARFIELD  DS    XL256
R5        EQU   5
R10       EQU   10
R11       EQU   11
R12       EQU   12
          COPY  MDXREC
```

If the MDXREC is defined in DKNMTASK during system generation and is addressed through a pointer in the PARMLST (AMDXREC), you do not have to reassemble programs that refer only to the MDXREC. However, this method is not recommended because it is difficult to maintain and debug.

Installing the Refreshed Enhanced Jam Option

You can choose to have CPCS display a refreshed enhanced jam screen at the end of a runout condition on the 3890/XP. In addition, you can choose to have all items displayed on the refreshed jam screens selected to the reject pocket when the operator enters DONE, as well as have all items selected to the reject pocket set as user control documents. You can also flag suspect items as having been imaged.

To invoke these options, follow these steps:

1. Make a user change to copy member EJAMCTRL, changing:
   ```
   &REFRESH  SETC  'N' to
   &REFRESH  SETC  'Y'
   ```
2. Make a user change to copy member EJAMCTRL, changing:
   ```
   &ALLDONE  SETC  'N' to
   &ALLDONE  SETC  'Y'
   ```
3. Make a user change to copy member EJAMCTRL, changing:
   ```
   &USERSET  SETC  'N' to
   &USERSET  SETC  'Y'
   ```
4. Make a user change to copy member EJAMCTRL, changing:
   ```
   &HSIBACT  SETC  'N' to
   &HSIBACT  SETC  'Y'
   ```
5. Assemble and LKED modules DKNMGET, DKNMJAM, DKNMCDRJ, DKNMCMDS, DKNMREAD, DKNMCQC, DKNMCRST, and DKNMSTAT.
6. Changing these modules requires you to generate a new version of the DKNMICR module. The JCL is in member DKNGMICR of CPCS.V1R11.CTRL.

   For information about generating DKNMICR, see “MICR Task Generation” on page 2-33.
7. Copy DKNJAMPR.DLL to \OS2\DLL on the sorter PC.
8. Update the run profiles to include the following:

   ```
   USERDLL    DKNJAMPR
   SPSINIT    JAMPROC_INIT_EVNT
   SPSMS      JAMPROC_MS_EVNT
   SPSDOC     JAMPROC_DOC_EVNT     Note: Must be LAST DOC event.
   SPSPAUSE   JAMPROC_PAUSE_EVNT
   ```

Notes:

1. Control program code R15F10, or higher, must be running on the 3890/XP.
2. The &ALLDONE option is valid only if the &REFRESH option is also active.
3. The DKNJAMPR DLL should not be used unless the &REFRESH value is set to Y. The function of the DKNJAMPR DLL is to direct the “inflight” items between the hopper and the read head to the reject pocket and to disengage the sorter at the end of the runout.
4. The DKNJAMPR DLL uses the first 10 bytes of the New Save Area in AUX storage on the 3890/XP. If you have any user DLL’s that use these 10 bytes, you must modify these user DLL’s.

Installing the Feature Disable/Disengage Option

If selected features have been initialized in the IREC and are subsequently disabled, you can have the sorter issue a disengage:

1. Download DKNFEATD in binary format from the SDKNDAT1 data set to a diskette.
2. Copy DKNFEATD.DLL to the \OS2\DLL directory on your sorters.
3. Add the following statements to your run profiles for which you want this feature:

```
USERDLL DKNFEATD
SPSINIT DKNFEATD_INIT_EVNT
* Use the following SPSPOST event for all features.
* (INF, END, and IMG)
SPSPOST DKNFEATD_POST_EVNT
* Use the following SPSPOST event for IMG feature only.
SPSPOST DKNFEATD_POST_IMGONLY_EVNT
```

Notes:

1. If you are installing this DLL on a 3890/XP, you must be at control program fix level R15F34 or greater. (There is no specific fix level requirement for the 3890/XPE.) You must also be at fix level R271F04 or greater on the 3897, if applicable.
2. This DLL disengages the sorter when any of the following features have been initialized in the IREC and are subsequently uninitialized:
   - INF (Item Numbering Feature)
   - END (Endorse)
   - IMG (Image Scanner)

   After the disengage, the operator *must* cancel/restart the entry run. If the entry is not canceled/restarted, the sorter sorts 6 (plus or minus) more checks and disengages again.
3. This DLL uses bytes 6 - 9 (relative to 0) of the NewArea of auxiliary storage. If you have any other DLLs that use this area, they must be modified.

DKNMREAD Assembly Option

In the event that the number of images captured by CIMS does not match the number of items for which imaging was requested, you can have CPCS display a message on the CPCS SYST supervisor terminal; and you can specify a number of times to repeat the message.

To do this:

1. Edit CPCS module DKNMREAD and change the value specified for "&MSGCNT" from 0 to the number of times that you want the message displayed.
2. Reassemble DKNMREAD, followed by a MICR generation.
3. Restart CPCS to invoke the new copy of MICR.

A message will be written to the SCROLL data set at the end of each image capture entry, showing the count for captured and requested images.

Balancing and Power-Encoding Considerations

This section describes High Performance Transaction System balancing adjustment records, power-encoding records that are associated with adjustments, and the DKNMIPI and DKNMDIX2 user-exit routines.

ImagePlus High Performance Transaction System Application Library Services Version 1 Release 2 (or higher) is required for balancing applications and for power encoding.

Balancing Adjustment Records

High Performance Transaction System balancing applications can set several balancing flags to identify the type of adjustment record. The flags also provide an audit trail of the balancing operations performed on the record. For COBOL modules, see the DI-FLG3-BYTE2 field in the copybook DKNCRDI. For assembler modules, see the DI3BYTE2 field in the copybook DIDSCT.

DKNCRDI contains the uncompressed, unexpanded COBOL format. DIDSCT contains the compressed assembler format. Figure 4-60 shows the mapping of some of the fields in the CPCS mass data set:

Figure 4-60 (Page 1 of 2). Mapping of fields in the CPCS mass data set

Type	Length	Value	Name	Description
BITSTRING	1		DI3BYTE2	Flag byte (3:2) relative to zero
1...			PEDRSION	PINE Endorse is initialized ON
.1..			PINFION	Pine INF is initialized ON
..1.			PENOTDON	Power encode requested, not done
...1			DI3MVD	Moved by High Performance Transaction System Balancing
.... 1...			DI3INS	Inserted by High Performance Transaction System Balancing
.... .1..			DI3DEL	Deleted by High Performance Transaction System Balancing
.... ..1.			DI3CHG	Changed by High Performance Transaction System Balancing
....1			DI3PE	Place holder for power encode

Figure 4-60 (Page 2 of 2). Mapping of fields in the CPCS mass data set

Type	Length	Value	Name	Description
CHARACTER	1		DITYPEI	Hexadecimal document type
		X'85'	DIBALDCH	Change or Delete
		X'86'	DIBALPE	Power/Encode Balancing control record
		X'87'	DIBALINS	Insertion Balancing control record
		X'95'	DIOREJ	Original reject control record
BITSTRING	1		DIFLAG2	Flag byte 2
1...			DIMTCR	CPCS control document
.1..			DIJAM	Jam document
..1.			DICAN	Canadian document
...1			DIOURS	On-us document
.... 1...				Reserved for IBM
.... .1..			DICRD	Credit document
.... ..1.			DILOW	Document entered with on-line entry
....1			DIHIGH	Document entered on high-speed pass

Adjusted M-strings can contain the following types of High Performance Transaction System balancing adjustments:

INSERT The example below shows Balancing has inserted one item into the 99-M string. The offsets are based on a standard 50-byte mass data set.

```
Tag nm  DISEQ12      DIAMT      DIMTCR   DITYPEI  DI3INS   DI3PE
Size    6/HEX        5/HEX      0:1/BIT  1/HEX    0:1/BIT  0:1/BIT
Loc     (5-10)       (31-35)    (39-38)  (40-40)  (45-44)  (45-44)
Fld num 2----------- 8--------- 22------ 30------ 56------ 59------
        100000001278 0000000200 1        87       1        0
        100000001278 0000000200 0        00       1        0
```

1. Insertion control record.

Field	Flag	Flag Description
DIFLAG2	DIMCTR	Control record
DITYPEI	DIBALINS	Insertion
DI3BYTE2	DI3INS	Insert

2. Insertion detail record. This record contains the inserted data used for posting. It has the same DISEQ12 field sequence number as the associated insertion control record, with the exception of the first digit, which is always 1.

Field	Flag	Flag Description
DI3BYTE2	DI3INS	Insert

DELETE An example of a delete is shown below:

```
DISEQ12      DIAMT      DIMTCR   DITYPEI  DI3DEL   DI3CHG
6/HEX        5/HEX      0:1/BIT  1/HEX    0:1/BIT  0:1/BIT
(5-10)       (31-35)    (39-38)  (40-40)  (45-44)  (45-44)
2----------- 8--------- 22------ 30------ 57------ 58------
000700003357 0000000000 1        95       0        0
000700003357 0000102297 1        85       1        0
000700003358 0000000000 1        95       0        0
000700003358 0000017457 0        00       0        0
```

The first 3357 is the original rejected item from the prime-pass I-string. The next 3357 indicates this record was deleted by Balancing.

This adjustment is indicated by a deletion control record that contains the following flags:

Field	Flag	Flag Description
DIFLAG2	DIMCTR	Control record
DITYPEI	DIBALDCH	Deletion or change
DI3BYTE2	DI3DEL	Delete

CHANGE An example of a changed record is shown below:

```
DISEQ12      DIAMT      DIMTCR   DITYPEI  DI3DEL   DI3CHG
6/HEX        5/HEX      0:1/BIT  1/HEX    0:1/BIT  0:1/BIT
(5-10)       (31-35)    (39-38)  (40-40)  (45-44)  (45-44)
2---------- 8--------- 22------ 30------ 57------ 58------
000700001994 0000754544 1        85       0        0
100700001994 0000954544 0        00       0        1
```

Note: A DITYPEI record is either a delete or a change. There are two ways to distinguish them. The changed record consists of a before-and-after pair, whereas the delete only has one record. Additionally, the flag bytes DI3DEL and DI3CHG uniquely define a delete from a change.

This adjustment is indicated by the following record pair:

1. A change control record that contains the original data and the following flags:

Field	Flag	Flag Description
DIFLAG2	DIMCTR	Control record
DITYPEI	DIBALDCH	Deletion or change
DI3BYTE2	DI3CHG	Change

2. A change detail record that contains the new data. This record has the same DISEQ12 field sequence number as the associated change control record, with the exception of the first digit, which is 1.

Field	Flag	Flag Description
DI3BYTE2	DI3CHG	Change

MOVE A move is a combination of a delete record and an insert record. However, the power encode-only record has a DITYPEI of X'86', not X'87', and DI3DEL is never turned on for a move.

```
DISEQ12      DIAMT      DIMTCR   DITYPEI  DI3MVD   DI3INS   DI3PE
6/HEX        5/HEX      0:1/BIT  1/HEX    0:1/BIT  0:1/BIT  0:1/BIT
(5-10)       (31-35)    (39-38)  (40-40)  (45-44)  (45-44)  (45-44)
2----------- 8--------- 22------ 30------ 55------ 56------ 59------
000800279538 0000000000 1        95       0        0        0
000800279538 0000000012 1        85       1        0        0
000800279538 0000000012 1        86       1        0        1
100800279538 0000000012 0        00       1        0        1
000000117828 0000021683 1        87       0        1        0
100000117828 0000021683 0        00       0        1        0
000800279538 0000000012 1        87       1        0        0
100800279538 0000000012 0        00       1        0        0
000800279539 0000000000 1        95       0        0        0
000800279539 0000000012 1        85       0        0        0
100800279539 0000000112 0        00       0        0        0
```

The records are identified by the flags shown for the delete and insert adjustments, except the DI3BYTE2 field contains the following flag and flag description:

Field	Flag	Flag Description
DI3BYTE2	DI3MVD	Move

Power-Encoding Records

An example of the power encode control record is shown below.

Note: These records are paired and a X'86' should always be followed by a X'00' in the 99-M-string. By default, DKNMDIS *does not* distribute the X'86' record. DKNMDIS distributes only the X'00' record that follows the X'86' record, and ICRE does not extract either the X'86' or X'00' when DI3PE is on.

```
DISEQ12      DIAMT      DIMTCR   DITYPEI  DI3MVD   DI3INS   DI3PE
6/HEX        5/HEX      0:1/BIT  1/HEX    0:1/BIT  0:1/BIT  0:1/BIT
(5-10)       (31-35)    (39-38)  (40-40)  (45-44)  (45-44)  (45-44)
2----------- 8--------- 22------ 30------ 55------ 56------ 59------
000800279538 0000000012 1        86       1        0        1
```

For insert and move adjustments, a power-encoding record pair can also be included as follows:

- A record is inserted and the corresponding physical item is also inserted in the correct location in the entry.
- A record is moved and the corresponding physical item is either left in its original location or moved to the correct location in the entry.

Power-encoding records have the same relative location as the physical items they represent. If they are associated with an insert, they follow the insert record pair. If they are associated with a move and the item is left in its original location, they follow the delete record. If they are associated with a move and the item is also moved, they follow the insert record pair.

Power-encoding record pairs have the following format:

1. Power-encoding control record. This record has the inserted item's data or the moved item's original data (if it has also been changed). The DISEQ12 field sequence number is the same as the associated insert control record or move control record.

Field	Flag	Flag Description
DIFLAG2	DIMCTR	Control record
DITYPEI	DIBAPE	Power encoding
DI3BYTE2	DI3PE	Power encode

2. Power-encoding detail record. An example of a power encode-only codeline is shown below:

```
DISEQ12      DIAMT      DIMTCR   DITYPEI  DI3MVD   DI3INS   DI3PE
6/HEX        5/HEX      0:1/BIT  1/HEX    0:1/BIT  0:1/BIT  0:1/BIT
(5-10)       (31-35)    (39-38)  (40-40)  (45-44)  (45-44)  (45-44)
2----------- 8--------- 22------ 30------ 55------ 56------ 59------
100800279538 0000000012 0        00       1        0        1
```

This record contains the inserted data or final moved data (if the moved item data is changed before or after the move). The DISEQ12 field sequence number is the same as the associated power-encoding control record, except that the first digit is 1.

Field	Flag	Flag Description
DI3BYTE2	DI3PE	Power encode

Adjusted M-String Processing

Power-encoding records are used only for power-encoding purposes and are never used for posting purposes. Because they do not have any posting value and contain duplicate detail data, power-encoding records must be bypassed by extract tasks for the same reason they are bypassed by the DKNICRE and DKNPLST tasks.

Important!

When using extract applications, a double posting can occur if you do not bypass power-encoding records (setting flag DI3PE on in field DI3BYTE2).

The M-string distribution task (DKNMDIS) distributes only one adjustment record for each adjusted item (with the final adjusted data) in the same relative location it was captured.

- Adjustment control records and insert detail adjustment records are not distributed.
- Change detail records, power-encoding detail records, and individual delete records (not part of a move) are distributed.

Power-Encoding Subsequent Passes

When the DKNMIPI user-exit routine (MIPIEXIT) is not active, the D-string adjustment records are treated as follows:

- Detail records are passed to the sorter for codeline data matching.
- Delete control records are passed to the sorter for codeline data matching, with power encoding disabled.

Power-Encoding Subsequent-Pass Reconciliation

The subsequent pass balancing list task (DKNSBAL) treats delete control records in the same way as detail records and includes them in the matching process. As a result, the deleted items are listed as missing items.

To eliminate missing item conditions, you can activate a DKNMDIS user-exit routine to stop distribution of the delete control records. The user-exit routine can also be used to avoid other problems, such as codeline data matching failures caused by groups of contiguous deletes. The IBM-supplied sample DKNMDIS user-exit routines (DKNMDIX and DKNMDIX2) include part of the code needed to do this.

Matching Codeline Data with Original Data

The following steps provide a method to match codeline data with original data and to power encode from adjusted data. Using this method can help you avoid codeline data matching failures that result from changed data.

1. Write and activate a DKNMDIS user-exit routine to force the distribution of change and power-encoding adjustment control records.
2. For the power-encoding pass sort-pattern definition, define a DKNMIPI exit work field in the O-record. The byte length must be greater than or equal to the total sorter-record byte length of fields 0 through 8. The field must not exist in the MDS record.
3. Write and activate a MICR DKNMIPI user-exit routine that merges each change or power-encoding record pair into one record. This single record contains the old and new data (in fields 1 through 8) for codeline data matching. Use the data in the work field (flagged for use in codeline data matching) for power encoding.

   When the DKNMIPI user-exit routine is active, CPCS processes the D-string adjustment records as follows:

   - Records that are normally passed to the sorter for codeline data matching are marked for codeline data matching and passed to the user-exit routine.
   - Change and power-encoding control records are marked as “not available for codeline data matching” and are passed to the user-exit routine.
4. Add logic to the sorter edit routine used in the power-encoding pass to move the power-encoding data from the work field to the power-encoding buffer. When codeline data matching is successful, set the appropriate field selection in the power-encoding mask byte before calling the power-encoding SPX services.

Security Options

CPCS lets you control, through IDs and passwords, which operators can log on CPCS. You can also define the tasks that the operator is allowed to start.

Resource Access Control Facility Security Option

The CPCS system profile parameter, SCRTY=RACF, indicates that the MVS resource access control facility (RACF) security subsystem is the security option for CPCS. If enabled, the master task loads and initializes a task named DKNSECR from the load library specified in the step library and places its address in PARMLST.

When RACF security is specified, DKNSGO3 and DKNSGOF control access to CPCS through IDs, passwords, and other parameters defined in RACF for the respective CPCS system. Each time DKNATASK is requested to run an application task, it accesses RACF data to determine whether the terminal operator can access the task. RACF definitions for each CPCS operator can include:

- The time of day and days of the week when an operator can log on
- The number of invalid logon attempts that can occur before ID access to CPCS is revoked
- ID password retention period
- The format of the IDs and passwords.

DKNMAYI is the transaction-level interface that enables applications to perform more detailed verification. For example, an operator might have access to CYCL, but not have access to CYCLU. The operator can start the CYCL task and display information defined for each cycle but cannot change it. DKNATASK verifies the CYCL command before starting the DKNCYCL task for the operator; but, if the operator tries to change any of the definitions, DKNCYCL checks the CYCLU transaction, finds it not valid, and permits no changes. For more information on the MVS RACF security interface, see the *CPCS Programming and Diagnostic Guide*.

Data Security Option

The CPCS system profile parameter, SCRTY=DASC, indicates that the data-access security control option of CPCS is called. It causes DKNMTASK to load a table called DKNDASC and places its address in PARMLST. If you omit the parameter, or specify SCRTY=NONE, the data security option is ignored. If you specify the data security option incorrectly, the CPCS system editor generates the appropriate message in the CPCS system parameter edit report (MPTR).

When you specify SCRTY=DASC in the system profile member DKNPCPCS, you must generate an operator logon table (DKNDASC) by coding a DKNBSCY macro for each operator entry in the table. An operator password that is not in the table cannot log on CPCS.

DKNMTASK controls whether data security is called. If it is called, data security compares the information in DKNBLDL and DKNDASC for each application task logon request. SGO3, SGOF, and tasks that automatically start are exempt and are not subject to data security testing; if they can be initiated under existing criteria, they will be.

You must arbitrarily assign authority classes to the application programs by the AUTH parameter of the DKNBLDL generation. For more information on the BLDL table, see “Describing an Application Task’s Environment” on page 2-15. These assignments should be consistent with the CPCS philosophy. For example, the SUPV task given any authority class still controls tasks requiring a supervisory terminal, no matter what authority classes are given to those tasks. An application task can be of only one authority class.

In creating the operator logon table, the user supplies an operator password, an operator logon ID, and one or more authority classes that the operator is **restricted** from running. For example, if all authority classes are specified, the operator can run only SGO3 and SGOF; there is no way to start any other application task or MICR. For more information on the coding requirements for this table, see “DASC Table.”

The macro DKNBSCY creates the operator logon table (DKNDASC); it must have an entry for each operator who can operate CPCS. Unknown operator passwords entered into SGO3 are not accepted and the terminal is not logged on. All logon attempts through SGO3 are logged to the ATASK log, showing the results of the logon attempt. Once a display terminal logs on with SGO3, the terminal takes on the security characteristics of the operator signing on.

These characteristics remain in effect until the terminal logs off through SGOF, even if the operator identification in the terminal changes by other means (for example, MSGON).

BLDL Table: The parameter AUTH=x in the APCB macro, where *x* is 1, 2, 3, 4, or 5, assigns an application task to an authorization class of tasks. The default authorization class is 1 if no authorization class is specified (unless the application task can only be started automatically). Automatically started tasks have the authorization class set to zero, regardless of what is coded in the AUTH parameter. An application task can only belong to one authorization class.

DASC Table: You can create the data-security table (DKNDASC) by coding the macro DKNBSCY with correct parameters: one macro for each CPCS operator. The assembled table consists of 8-byte entries, each containing an operator password, operator ID, and, if applicable, application authorization classes and MICR that this operator is **not** permitted to start.

Important!

You specify the restrictions. Omitting authorization classes for an entry in the data-security table means that operator can run all tasks in the unspecified authorization classes.

After you code the table, assemble it and link-edit it with the name DKNDASC into the CPCS load library. The first occurrence of DKNBSCY generates a CSECT record labeled DKNDASC. Coding DKNBSCY without any parameters generates the end of the table.

DKNBSCY Macro Format

The macro format is:

Label	Operation	Parameters
label	DKNBSCY	,*password* ,*operator ID* [*authorization class*]

password
Specifies a 3-character password with no embedded blanks. The operator must enter the password to log on. SGO3 reads it.

operator ID
Specifies a 3-character operator identification with no embedded blanks. It is used for display purposes only.

authorization class
Specifies the authorization class that the operator is **not** permitted to run. It is either a single-digit number (1, 2, 3, 4, or 5), the letter M, or a combination of both. If two or more classes are coded, the operator must enclose them in parentheses and separate them with commas. The codes are expressed in any order to a maximum of six authorization classes. If you omit this parameter, the operator can start all authorization classes.

DKNBSCY Examples

Macro Examples	Explanation
`DKNBSCY PW1,AAA`	Operator password PW1 signs on as operator AAA and can run any authorization class.
`DKNBSCY PW2,BBB,1`	Operator password PW2 signs on as operator BBB and can run any authorization class tasks except class 1.
`DKNBSCY PW3,CCC, (M,3)`	Operator password PW3 signs on as operator CCC and can run any authorization tasks except MICR and class 3.
`DKNBSCY`	Denotes the end of the DKNDASC table.

Note: Along with the CPCS SGON security option, VTAM users can specify the password parameter (PASSWD) in the VNODE macro. This requires the operator to enter a password when logging on CPCS by a standard VTAM LOGON request. For more information, see “DKNVTASK Node-Name Table Description” on page 2-56.

RACF Security Installation Procedure

You must complete the following tasks when using RACF facilities for security with CPCS.

1. Define a RACF class and class group to protect your CPCS resources.
2. Connect your CPCS resources to the RACF class and/or class group structure.
3. Define a RACF group for your CPCS user IDs.
4. Change the CPCS system profile parameter SCRTY= to activate your RACF security procedure (see “DKNPCPCS Profile Member” on page 3-4).

The following procedures illustrate, in greater detail, how to implement RACF security within CPCS.

1. In order to define a RACF class and class group, you must do the following:
   - Define a RACF class with the RACLIST=YES option. The naming convention for RACF classes is defined in the *System Programming Library: Resource Access Control Facility* manual. The DKNGRACF JCL in CPCS.V1R11.CTRL shows an example of how to define a resource class and resource class group.
   - The default RACF class is $CPCS. To change it, substitute your choice in the JCL.
   - To use a group of resources, code a group in the JCL. CPCS uses $GCPCS as the default class group name.
   - You must have system-level authority to run this JCL because you are updating SYS1.LINKLIB.
   - Issue the RACF command SETROPTS RACLIST (class) for the CPCS class. Every time you modify the RACF CPCS class tables, you must issue this command.
2. In order to protect your resources, you must connect them to a class.
   - Use the ISPF display panels for RACF or you may do this by submitting the CPCSRACF JCL in CPCS.V1R11.CTRL. You must have RACF special authority to use either method to connect your resources to the class.

     You can connect each of the resources to the class $CPCS and connect groups of resources to the CLASS GROUP ($GCPCS). For an example, see “Establishing Your RACF Structure” on page 4-155.
3. In order to provide access to CPCS and its resources, you should define a RACF group for the user IDs.

   A RACF group may be defined through ISPF RACF panels. CPCS does not include JCL to complete this task. You must have RACF special authority to define a group of user IDs.

   Note: This step may be optional if you use an existing RACF group, such as one for TSO users. Please read Step 4 carefully concerning the CPCS system profile member DKNPCPCS parameter `SCRGP=`, if you choose to eliminate this step.
4. The following CPCS system profile member DKNPCPCS parameters may be coded in order to activate your RACF security procedure for CPCS.
   - Change the CPCS system profile member DKNPCPCS parameter to `SCRTY=RACF`.
   - Specify the RACF class in the DKNPCPCS parameter `SCRCL=` for the resources that you protected. The default class name for CPCS is $CPCS. If you used a different class name in the DKNGRACF and CPCSRACF JCL, change the parameter to reflect the proper class name.
   - Specify the RACF group name for the CPCS user IDs in the DKNPCPCS parameter as SCRGP=. The default RACF group name is CPCS. If you defined a different RACF group or are using an existing RACF group, change the parameter to reflect the proper RACF group name for user IDs

that may log on (SGON) to CPCS. Only one RACF group of user IDs may be defined to CPCS, unless you specify SCRGP=0.

Note: If the value of SCRGP is 0, TSO users defined to any RACF group can log on to CPCS. The users may then access any resources that are not protected or not defined in your resource class group structure. However, they may not access any resources or groups of resources they have not been permitted to access. For more information about the SCRGP keyword, see "DKNPCPCS Profile Member" on page 3-4.

- The system profile member DKNPCPCS has a default parameter, CHGPSWD=0, that allows you to maintain a password within the rules established in the definition to RACF for CPCS. If you do not want an operator to be able to change the password, code CHGPSWD=1 in the system profile member DKNPCPCS.

Note: The default RACF security lets an ID log on only one terminal at a time. To permit multiple logons, change the program DKNSG03.

Establishing Your RACF Structure

Before you establish your RACF structure, plan your class/group structure to protect your resources. If you use a class/group structure, you will be able to authorize new users more easily. See Figure 4-61 for a sample group structure.

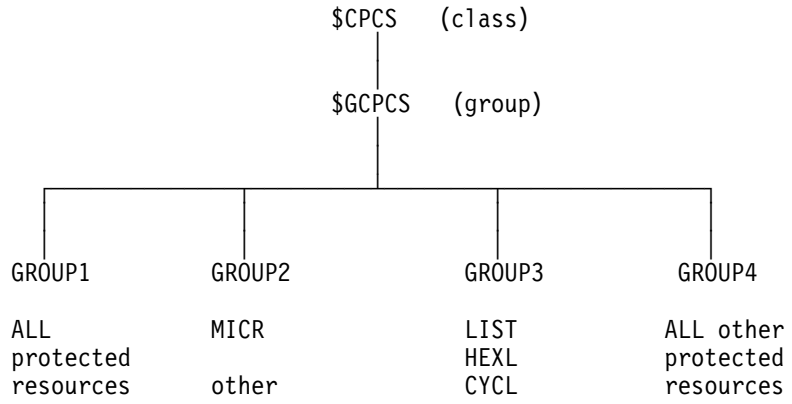

Figure 4-61. Sample RACF Class/Group to User ID Structure

The groups would be joined to resource class group $GCPCS.

If you do not want the resource class group structure, define each resource separately to the resource class.

Permitting Access to Resources

When you use the resource class group structure, you can let user IDs access resources by giving the IDs access to the group itself. Without the resource class group structure, you must give each ID access to each resource. Here is an example of the resource class group structure:

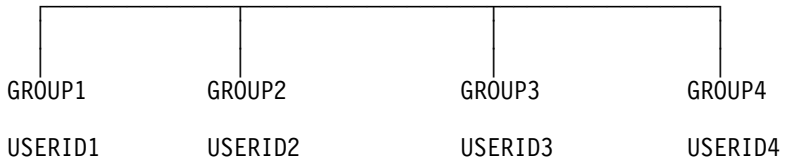

Figure 4-62. Sample RACF Class/Group to User ID Structure

This example lets USERID1 access all resources. USERID2 can access only resources in GROUP2. Any resource not defined to the class $CPCS or class group $GCPCS is not protected. Any USERID defined to the RACF group CPCS can access undefined resources.

You must understand this resource class structure. It is important to the security of the CPCS system. Consider your plan carefully.

Here are some other RACF system parameters that you can use:

- Time of day restrictions
- Days of the week restrictions
- Password length and limitations.

For more information on defining an application system to RACF, see the following RACF manuals:

- *Resource Access Control Facility Security Administrator's Guide*
- *Resource Access Control Facility Command Language Reference*
- *System Programming Library: Resource Access Control Facility*.

CPCS Commands That You Can Protect with RACF

After you create the RACF class CPCS($CPCS), determine which CPCS commands you want to protect. Use the following guidelines to make your decision:

- RACF can protect all tasks that are started manually and are listed in DKNBLDL.
- RACF can also protect the following commands that are not in DKNBLDL:

  AHCPY
  ANODE
  DHCPY
  FNODE
  MICR
  PBUFF
  PDUMP
  STOP

- RACF cannot protect the following commands:

  SGON
  SGO3
  SGOF
  ASGF
  CANCEL
  STRTMICR
  HALTMICR.

Non-RACF Security Option

CPCS includes an option in the CPCS system profile member DKNPCPCS that lets you use a product other than RACF to control which operators can log on CPCS.

To establish your non-RACF security installation you must change the SCRTY parameter in the CPCS system profile member DKNPCPCS to SCRTY=USER. This causes the following events to occur:

- CPCS places the code X'03' into the PARMLIST field, SCRTYFLG, indicating user security.
- CPCS loads the user-security exit, DKNSECRX, and places its address in the PARMLIST field.

  The DKNPARM copybook provides addressability to a new field, ADKNSECX, as specified in the following examples, to identify the user-exit module DKNSECRX:

  ADKNSECX Defines a non-RACF user-security interface

  SCRTUSER Equates to the value of X'03' to indicate the user-security exit system.
- If the load fails, CPCS issues a write to the operator (WTO) error and sets SCRTYFLG=SCRTNONE (no security) and writes a message to the SCROLL data set.
- CPCS calls the user-security exit system initialization through the DKNSECR module, which, in turn, calls the module DKNSECRX.
- CPCS analyzes the return code from the exit and issues error messages. If a non-zero return code is encountered, CPCS issues a WTO, sets SCRTYFLG=SCRTNONE, and writes a message to the SCROLL data set.

You must supply the user-exit DKNSECRX.

When you log on CPCS, the DKNSGO3 module performs the following tasks:

1. Checks whether the value of the SCRTUSER field is equal to the value of the SCRTYFLG field (X'03'), located in the PARMLIST field SCRTYFLG
2. Passes control to DKNSECR, which then gives control to DKNSECRX
3. Uses the return code to route control for further signon processing.

DKNSECR provides support in a CPCS environment for a user-specified security system and for a single control point in CPCS processing through which all calls to the user's security package are passed.

DKNSECR does the following:

1. Checks whether SCRTY=USER.
2. If SCRTY=USER, it calls DKNSECUI, which then calls the user-security module, DKNSECRX.
3. DKNSECR passes the return code from DKNSECRX back to the calling routine (DKNSGO3).

DKNSGOF provides sign-off and logoff support for a user-specified security system. It does the following:

1. Checks whether SCRTYFLG=SCRTUSER
2. Calls DKNSECR with the sign-off transaction.

DKNCYCLX - Validate Cycle and Endorse Dates Exit

The CYCL user exit DKNCYCLX may be used to perform additional cycle and endorse date validations; for example, to prevent user holidays from being specified as cycle date or endorse date. Before control is passed to DKNCYCLX, the dates are validated by DKNCYCDT for valid month, day and year.

Activation

DKNCYCLX is invoked if it exists in the load library concatenation. No re-gens, CPCS restarts, or CHAPs are required when a new CYCL exit routine version is installed.

Linkage

DKNCYCLX is called by DKNCYCDT whenever a cycle is activated by the CYCL task or the DKNCYCI interface. For the parameter list passed, see the assembler copybook CYCXDSCT or the COBOL version DKNCCYCX. The calling program must specify a cycle date and a one-byte indicator of the format, and an endorse date with a one-byte indicator of the format. The values of the format indicators are the same as the values for PARMDATE in DKNPARM, the CPCS parameter list. DKNCYCLX returns a zero return code if the dates are valid, or a nonzero return code with a message if either date is invalid.

Example

The SAMPLIB PDS contains an assembler example of DKNCYCLX to prevent January 1 from being specified for either cycle date or endorse date.

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*, SC20-1699.

A

ABA. See *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 document, containing the financial institution identification number.

account number field. A MICR-encoded field, on a check or a deposit slip, that represents the account number of the item. The account number field is MICR field 3.

active task status. This indicates that this task is currently processing the string associated with this UOW. See also *task status*.

active UOW status. This indicates that a task in the task list is currently processing the string associated with this UOW. See also *UOW status*.

adjustment. A change or a description of a change that has been made to reflect a detected error in work that has been processed.

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.

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

amount field. A MICR-encoded field on an item that represents the amount of that item. The amount field is MICR field 1.

Application Library Services. See *ImagePlus HPTS Application Library Services*.

application program task control block (APTCB). A CPCS 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 executive programs.

APTCB. See *application program task control block*.

AST. Assist document.

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

auxiliary on-us field. See *serial number field*.

B

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

batch. The lowest required level that has dollar control established by a control document.

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

batch slip. A level of control for balancing items. See also *batch*.

block. (1) A prime-pass control level consisting of one or more batches. In CPCS, 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 *data preparation*.

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

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

bypass task status. This indicates that this task should not process for the string associated with this UOW. See also *task status*.

C

cash letter. The group of items to be delivered to an endpoint. Grouping of the items is usually by kill bundle.

cash-letter detail. A listing of all kill bundles. See also *kill list*.

cash-letter summary. A listing that summarizes all of the kill bundles in a cash letter by giving monetary and item controls for each kill list.

check. A draft drawn on a financial institution and payable on demand any time on or after the date indicated.

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

CIMS. See *Check Image Management System*.

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

code-line data matching. 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. Code-line data matching occurs on subsequent operations.

code-line data record. See *data record*.

cold start. An initiation of the CPCS 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. See also *UOW status*.

concurrent kill. Producing kill lists for kill pockets in an entry before the whole entry has been 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 kill printing) begins before completing capture for the whole entry.

control slip. A MICR-encoded document that contains control information, including the amount of the items that the document controls, the source of the items, and a code that describes the level of the control.

control total. The total dollar 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 programs.

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

credit. The opposite of a debit. In normal check collection terminology, deposit slips are credits.

cursor. A small horizontal line on a computer screen that indicates the position to which the next character is transmitted from either the keyboard or the CPU.

cutoff. The financial institution's designated point for balancing or releasing work before processing continues. Also, the designated time after which the financial institution cannot accept work for processing.

cut slip. A control document used to separate and identify one kill bundle from another.

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

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

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

data set. A single collection of data that can be stored on cards, a tape, or one or more disks (for example, a kill-bundle data set).

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).

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

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

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 single pocket of the document processor.

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

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

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

D-string. See *distributed string*.

E

eligible task status. This indicates that this task is waiting for processing. Other tasks must process before this task. See also *task status*.

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 list or inscriber tape.

encode. To imprint a MICR field on a check.

encoder. A machine that encodes.

endorsement. The signature of the endorser; 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 a check.

entry. A variable whole number of blocks that are processed as a single group of work.

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

EPC. See *extended process control field*.

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

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

exchange charge. A charge made by the drawee financial institution for its services in paying checks and other instruments presented to it. The Federal Reserve Act forbids drawee financial institutions to make such charges against Federal Reserve Banks on checks in process of collection. The purpose is to assure that checkbook money will be payable throughout the country at its face value.

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

F

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

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 and coupons) 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 string with a task list through the specification of sort type, pass pocket history, string type, and flow code.

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 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 and coupons) 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 dollar 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 must complete first. See also *task status*.

held UOW status. This indicates that the task to process next for this UOW has a held status. See also *UOW 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 (1-1).

holdover. Items that were not processed in time to meet their deadline.

HPTS. High Performance Transaction System.

I

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.

inclearings. Checks drawn on your financial institution that are sent in for collection by another financial institution, the Federal Reserve Bank, or a clearing house.

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.

informational unit of work. This usually represents a physical group of checks that CPCS has not yet processed. It represents work to be done. One or more informational units of work are ultimately associated with a functional unit of work when CPCS captures the physical check documents.

input string (I-string). This is 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.

inscriber. A machine that encodes.

I-string. See *input string*.

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

item number. A number that is associated uniquely with a document throughout the processing cycle.

J

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

joggler. 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.

K

kill. To process items to a point where no further distribution is required.

kill bundle. A group of killed items, indicated by divider slips. With concurrent kill, this group can span strings.

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 killed items.

L

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

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

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

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 check. The MICR system uses 10 specially coded digits and four special symbols.

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

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. See *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. See *magnetic ink character recognition*.

MICR field 1. See *amount field*.

MICR field 2. See *process control field*.

MICR field 3. See *account number field*.

MICR field 4. See *optional field 1*.

MICR field 5. See *ABA number*.

MICR field 6. See *extended process control field*.

MICR field 7. See *serial number field*.

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

MIS. See *Management Information System*.

misread. A condition that occurs when a document processor interprets a MICR character as a good character other than that which was actually encoded on the document.

missort. An item that is found in a pocket other than the pocket to which it was sorted.

M-string. See *merged string*.

O

online fine sort. A computer-controlled sorting of on-us checks by either or both the account number and the serial number sequence for filing. This process can use image-processing techniques.

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

on-us checks. Checks that are drawn on the financial institution that is processing them.

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

optional field 2. See *extended process control field*.

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.

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 dollar and item control of a group of MICR 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 string. See also *flow code* and *flow control*.

pending request queue. A first-in-first-out System Manager queue through which CPCS 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*.

prime pass. The first pass of an entry on a document processor.

printing after the fact. See *deferred printing*.

process control field. (1) A MICR-encoded field on a document, usually representing the type of document. The process control field is MICR field 2.
(2) Transaction code.

proof. Receives checks that come from tellers, mail and night depository, and internal departments of the financial institution. Proof proves and inscribes the dollar amount in MICR.

proof of deposit. The act of totaling items at the deposit level and ensuring that the total of the credits equals the total of the debits.

Q

queued task status. This indicates that this task is ready and waiting to process. See also *task status*.

queued UOW status. This indicates that no task is currently active for this UOW; however, one task has a queued status. See also *UOW status*.

R

RACF. See *Resource Access Control Facility*.

RBA. See *relative block address*.

reconcile. To find and correct the cause of a difference between two sets of totals.

reconciliation. See *balancing*.

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 MICR-encoded document that cannot be read in its entirety by a document processor or that fails certain editing checks. This document is directed to a special pocket called a reject pocket.

reject string (R-string). Strings 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, the calculated location of a specific record.

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 further 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.

restart. An initiation of the CPCS 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 XP reader/sorter area where records are stored during online operations until they are sent to the host. The buffer is accessed during automatic restart.

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 number. A MICR-encoded check field that represents the financial institution on which the check is drawn. The routing transit field is field 5.

routing transit field. See *ABA number*.

R-string. See *reject string*.

S

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.

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 number field. A MICR-encoded check field that represents the serial number of that check. Synonymous with *auxiliary on-us field*. The serial number is MICR field 7.

settlement. The act of bringing sets of related figures from two financial institutions into agreement. Adjustments are made to offset the differences.

SMOF. System Manager Online Functions.

sort pattern. A table used by the sort routine to determine the pocket to which a check is to be directed.

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 application task and is read by the CPCS output writer as it is being printed.

SSB. See *string status block*.

statistics. The processing of Unit of Work 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.

SSM. See *string segment map*.

string. The data records representing a group of items entered through a physical or simulated document processor or through OLRR. Can be 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. Each string in the system is associated with a string segment map. Each bit in a map represents a segment of direct access storage. The bit settings for the string segment map are (1) 0=segment available or (2) 1=segment allocated.

string status block (SSB). This CPCS control block is maintained by the MDS programs for every open string.

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.

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.

suspended task status. This indicates that this task processed, but it did not complete successfully. See also *task status*.

suspended UOW status. This indicates that the last task that processed for this UOW did not complete successfully. See also *UOW status*.

System Manager. A subsystem of CPCS that directs and controls the operations of the IBM 3890/XP Series document processors.

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 to hold machine and user-selected rejects.

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 application or function. A task name must be DKNMICR or it must be in the CPCS 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 (1) active, (2) bypass, (3) complete, (4) eligible, (5) held, (6) queued, or (7) suspended. 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. An arbitrary grouping of items for control purposes.

transit. The sorting of checks to external destinations.

U

unit of work (UOW). A logical entity that the System Manager uses to track a piece of work through CPCS. It can be informational or functional. See also *informational unit of work* and *functional unit of work*.

UOW. See *unit of work*.

UOW status. This status represents the state of a unit of work and its associated string. Can be (1) active, (2) queued, (3) suspended, (4) complete, (5) deleted, or (6) held. See definitions for details.

W

warm start. An initiation of the CPCS system, causing the contents of the MDS and the control data sets to be retained. A warm start is generally used for restarting CPCS after a normal ending.

work. Any document or group of documents that CPCS processes.

work flow. An ordered list of tasks for a specific CPCS string. Each CPCS 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.

Bibliography

The publications in this bibliography contain information related to CPCS 1.11.

ACF/VTAM Publications

The following publications are related to the ACF/VTAM[00] 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

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Communicating Your Comments to IBM

Check Processing Control System
Customization Guide
Release 11

Publication No. SC31-2853-06

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a readers' comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:

  United States & Canada: 1-800-955-5259
- If you prefer to send comments electronically, use this network ID:

  IBM Mail Exchange: USIB1362 at IBMMAIL

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies.

Readers' Comments — We'd Like to Hear from You

Check Processing Control System
Customization Guide
Release 11

Publication No. SC31-2853-06

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	□	□	□	□	□

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	□	□	□	□	□
Complete	□	□	□	□	□
Easy to find	□	□	□	□	□
Easy to understand	□	□	□	□	□
Well organized	□	□	□	□	□
Applicable to your tasks	□	□	□	□	□

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? □ Yes □ No

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Name _____ Address _____

Company or Organization _____

Phone No. _____

Readers' Comments — We'd Like to Hear from You
SC31-2853-06

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: 5734-F11

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-2853-06