Check Processing Control System
International MVS/ESA™

Customization Guide

Release 1

SC31-2943-03

Check Processing Control System
International MVS/ESA™

Customization Guide

Release 1

SC31-2943-03

Note!

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

| **Fourth Edition (November 1999)**

This edition applies to Release 1 Modification 0 of the IBM Check Processing Control System International MVS/ESA™ (CPCS-I) licensed program (Program No. 5799-FKT). This publication is current as of PTF number UW61613.

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, Department 58G, MG96/204, 8501 IBM Drive, Charlotte, NC 28262-8563, U.S.A. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Contents

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 Licensing, IBM Corporation, North Castle Drive, Armonk, New York, 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 International MVS/ESA (CPCS-I).

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 International MVS/ESA.

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

Year 2000 Readiness

This IBM program, when used in accordance with its associated documentation, is capable of correctly processing, providing and/or receiving date data within and between the twentieth and twenty-first centuries, provided that all products (for example, hardware, software, and firmware) used with this IBM program properly exchange accurate date data with it.

Trademarks

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

IBM®, ACF/VTAM®, AD/Cycle®, COBOL/370™, Hiperspace™, ImagePlus®, MVS/ESA™, MVS/SP™, RACF™, SAA®, System/370™, Systems Application Architecture®, VTAM®.

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

All other trademarks denoted by a double asterisk (**) in this publication, are the property of their respective owners.

About This Book

This book supplies customization information for personnel who install and maintain the IBM Check Processing Control System International MVS/ESA (CPCS-I).

This book provides information about:

- CPCS-I operating requirements in an MVS environment
- Installation and generation procedures for tailoring CPCS-I
- Creating user-exit routines and other user programming requirements.

Who Should Read This Book?

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

Programmers and operational personnel responsible for ImagePlus High Performance Transaction System applications 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-I configurations, software requirements, data set allocation, and operational considerations.

Chapter 2, “Installation and Generation Procedures,” presents information about generating the Master task (DKNMTASK) and MICR task, DSAT macros, activating Logging, creating an expanded MDS environment, and requirements for the Virtual Telecommunication Access Method (VTAM) program.

Chapter 3, “System and Application Profiles,” describes application profile names and profile records, and shows you how to use them.

Chapter 4, “User Programming Requirements,” describes MICR and Online Reject Re-entry (OLRR) stacker-select routines, record formats for the bank control file, endpoint name-and-address file, and sort-pattern definitions, extract programming, and CPCS-I security options.

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

Related Publications

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

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

  This publication gives a general introduction to CPCS-I. It describes various features and advantages of CPCS-I and the hardware and software requirements for operating this system. It also discusses CPCS-I support of the IBM 3890 Document Processor and the IBM 3890/XP Series document processors, along with some of the features of these processors.

- *IBM Check Processing Control System International MVS/ESA: Installation Guide*, GC31-2942
  Short Title: *CPCS-I Installation Guide*

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

- *IBM Check Processing Control System International MVS/ESA: Terminal Operations Guide*, SC31-2946
  Short Title: *CPCS-I Terminal Operations Guide*

  This guide explains how to perform CPCS-I tasks and is written for the CPCS-I operators. Included in this guide are terminal operations for the MICR restart procedures and sample reports.

- *IBM Check Processing Control System International MVS/ESA: Programming Guide*, SC31-2948
  Short Title: *CPCS-I Programming Guide*

  This guide contains guidelines for CPCS-I programmers. It includes information about application-program processing, problem analysis and documentation procedures.

- *IBM Check Processing Control System International MVS/ESA: Programming Reference*, GC31-3997
  Short Title: *CPCS-I Programming Reference*

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

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

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

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

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

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

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

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

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

Summary of Changes for SC31-2943-03.

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

ALLO User Exit: The ALLO user exit allows additional document processor allocation/unallocation request and user validations.

Chapter 1. System Programming and Operations

Overview

This chapter contains information about establishing and using the IBM Check Processing Control System International MVS/ESA (CPCS-I) operating environment and includes information about:

- System programming requirements, including system configuration
- CPCS-I startup and shutdown
- Operational considerations
- External storage requirements
- Data-set recovery procedures
- Supplied command procedures
- CPCS-I control data sets
- CPCS-I internal data sets.

Important!

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

System Programming Requirements

CPCS-I 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-I operates only on Multiple Virtual Storage (MVS) systems with Enterprise System Architecture (ESA) configured with expanded storage.

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

Minimum System Configuration

CPCS-I requires, as a minimum hardware configuration, a System/370* processor operating in ESA mode with a minimum of 8 megabytes of main storage. You should ensure that there is enough real and expanded storage to meet the combined storage requirements of the host operating system, access methods, and CPCS-I. In addition to the ESA requirement, the following features and input/output (I/O) devices are necessary:

- Two direct access storage devices, with sufficient space for the CPCS-I data sets.

* Trademark of IBM

- 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.

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-I Startup and Shutdown

This section describes how to start and stop CPCS-I. 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.

Starting CPCS-I

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

Figure 1-1 on page 1-5 summarizes the effect of various start effect on critical data sets parameters on CPCS-I data sets. "CPCS-I Startup JCL Definition" on page 1-6 provides a detailed description of the CPCS-I startup JCL.

Figure 1-1. Effect of CPCS-I 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**	**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	Does not clear the data set	Does not clear the checkpoint record	Does not clear the data set
COLD	FMT*	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 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	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 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	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	Does not clear the data set	Clears the checkpoint record	Clears the data set
WARM	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	Does not clear the data set	Does not clear the checkpoint record	Clears the data set
WARM	NOFMT*	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	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	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	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 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 clear the data set	Clears the checkpoint record	Does not clear the data set
REST	NOFMT*	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 clear the data set	Does not clear the checkpoint record	Does not clear the data set
REST	NOFMT*	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 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	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	Clears the data set	Clears the checkpoint record	Does not clear the data set

* 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-I Startup JCL Definition

The JCL for starting CPCS-I (member DKNJCPCS in CPCSI.V01R01.SDKNSAM1) is the basis for the descriptions of the CPCS-I start parameters, device specification, file allocation and initialization, and recovery procedures in this section.

The startup JCL assumes a configuration of three tape drives with virtual I/O (VIO) data sets used for temporary files. 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-I 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,NAME,ARST,CMID,ESM
//      TIME=1439
```

DKNMTASK is the load module name from the master task generation. You can use a different load module name if you have generated the CPCS-I master task under a name other than DKNMTASK. The CPCS-I EXEC statement must contain the following positional parameters:

STYPE Start type defines how CPCS-I 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.

NAME Unique job identifier.

ARST Automatic restart job identifier.

CMID Check Image Management System subsystem identifier.

ESM Start type for Enhanced System Manager.

Figure 1-1 on page 1-5 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-I process, as shown on the following pages. Be sure you understand the implications of each option and establish your installation procedures accordingly. Because CPCS-I is often a continuously running task, set the TIME operand value large enough to prevent 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

- 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.
- The FTYPE setting determines whether to clear the MDS and the scroll data set. (See Figure 1-1 on page 1-5.)
- 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 MDS for I-strings or R-strings that were open when CPCS-I last ended and sets them to restart mode.
- Does not change the MDS, the MDS index, the cycle data set, or the pass-to-pass control tracer data set.
- Calls DKNCOMP to compress fully processed records from the kill-bundle 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 MDS for I-strings or R-strings that were open when CPCS-I was last stopped and sets them to restart mode.
- Does not change the pass-to-pass control tracer data set or the cycle data set.
- Calls DKNCOMP to scan the kill-bundle data set to determine the next available key. DKNCOMP does not compress these files. If you use REST consistently, you should manually schedule DKNCOMP after end-cycle processing.
- Permits use of previous information in the scroll data set.
- Gives the quickest restart of CPCS-I without loss or change of data.

RECV

- Recovery of the MDS and index should be complete before CPCS-I 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-I was last brought down and sets them to restart mode.
- Calls DKNCOMP to scan the kill-bundle 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 1-29.

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-related parameters in the MDEF macro, 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 1-29.

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

INDEX CPCS-I recovers the MDS index from the MDS.

BOTH CPCS-I recovers the MDS and the MDS index. CPCS-I 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-I 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-I. 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.

NAME Parameter

XXXX Represents a unique job identifier (four positions) that CPCS-I can use to verify the existing installation level of operation or to identify test version operation. The default value is VTSK.

ARST Parameter

X Specifies a character that identifies the CPCS-I job that captured the last items that passed through a document processor. This value is used during automatic restart processing.

Usage Notes:

1. Different CPCS-I jobs that access the same document processors must have different values specified for the ARST parameter.
2. If more than one character is specified, only the first one is used. The remaining characters are ignored.

CMID Parameter

XXXX Represents the four-position identifier of the unique CIMS subsystem that processes CPCS-I document images. For information about defining the CMID, see the *CPCS-I Programming Guide*.

ESM Parameter

XXXX Specifies the type of startup that ESM should perform. See the *CPCS 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-I load library and any user-load libraries required by the installation. AD/Cycle COBOL/370 programs might require the correct system library if the AD/Cycle COBOL/370 and sort libraries are not in the MVS link list.

SORTLIB Data-Set Statements: CPCS-I requires this library when running CPCS-I tasks or user-written COBOL programs that use the SORT verb.

Document Processor Statements

These DDs are a requirement if you use channel-attached document processors and if allocation is necessary at startup. They refer to the DDRDRIN parameters of the CPCSRDR macros. 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-I. 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-I Data Sets

Permanent CPCS-I data sets should be catalogued 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 1-18.

CPCS-I temporary data sets can be virtual I/O (UNIT=VIO) data sets. They 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-I, based on the number specified by the SPOOL parameter in the MDEF macro. 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-I 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 DASD space. You can allocate a minimum of two and a maximum of 255 spool data sets.

High-Volume Spool Data-Set Statements

CPCS-I supports tasks that generate a large amount of spooled output (for example, DKNRMIT). 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-I Programming 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.

Tape Data-Set Statements

For most installations, the data sets created by the input create task (DKNIN), the master create task (DKNMD), and the end-cycle task (DKNSK) can be allocated to a single tape unit. These tapes can be dynamically allocated.

The recovery tape (DKNRT) is used for both selective and full MDS recoveries. The MDS log tape (DKNLT) should have its own drive and, if possible, a secondary unit 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. 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-I. 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-I application program returns with a nonzero system or user return code.

Normal Shutdown of CPCS-I

The MVS system operator can remove CPCS-I from the system through the normal operating system facilities by cancelling the job. However, the operator should not cancel the CPCS-I job unless directed to do so by a CPCS-I supervisor. The supervisor can shut down CPCS-I 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-I has been shut down because of a supervisor request. For more information on the STOP command, see Appendix A of the *CPCS-I Terminal Operations Guide*.

Operational Considerations

This section describes system operation recommendations for CPCS-I. These recommendations are intended to help you run CPCS-I 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, data-space considerations, terminal and sorter attachment, automatic restart, printing, job entry subsystem (JES), and system management facilities (SMF).

MVS Operations

You should run CPCS-I with the highest available job priority on your system. You must run CPCS-I as non-swappable. Include an entry for the CPCS-I 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-I 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 Reference*.

Data-Space Considerations

The merged string (M-string) distribution module (DKNMDIS) uses data-spaces for the IBM licensed program Multiple Virtual Storage/Enterprise Systems Architecture (MVS/ESA) to temporarily store D-strings that DKNMDIS builds. The following formula determines the size of the data space:

```
DS size = {(D-max * L'SDE) + (D-Recs (L'MDSREC + 8)) + 4095} / 4096
```

DS Size is the number of 4K pages of dataspace required, where:

D-max	Maximum number of D-strings
L'SDE	Length of the SDE
D-Recs	Number of MDS records per D-string
L'MDSREC	Length of MDS record.

For example, given a 50,000 item entry, the size of the data space would be, given the following values:

D-max	99 D-strings
L'SDE	736 bytes
D-Recs	50,000 records in the entry
L'MDSREC	158 bytes

```
DS size = {(99 * 736) + (50,000 (158 + 8)) + 4095} / 4096
```

results in 2046 pages of dataspace, which is approximately 8MB. The sample BLDL allows three copies of MDIS to be active at the same time, which means MDIS could use 24 MB of dataspace for this example.

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-I configuration needs a region of at least 6144K bytes; however, the working set is seldom more than 700K bytes. For concurrent sorting, the region size must increase 40K bytes for each concurrent sort coded in the MDEF macro MAXSORT operand.

```
Region size = 6144K + (40K  x  MAXSORT value)
```

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-I 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 MDEF macro. Specifying a number greater than the actual number of physical terminals creates extra terminal entries in the terminal table. CPCS-I uses VTAM functions for allocation of the terminals.

Document Processor Considerations

CPCS-I 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 2-37.

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-I 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. An operator uses the DKNALLO program to define the physical document processor channel-unit address. 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 2-22. For information about using DKNALLO, see the *CPCS-I Terminal Operations Guide*.

LU 6.2 Attachment

An LU 6.2-attached document processor requires 3890/XP MVS Support Release 2 or higher and ACF/VTAM Version 3 Release 4.1. 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*.

For information about using DKNALLO, see the *CPCS-I Terminal Operations Guide*.

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 MDEF macro 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-I 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 DKNJPROC in the CPCSI.V01R01.SDKNSAM1 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-I 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-I processes exception or SCI error headers. These control records, which are identified by X'80' in the DIFLAG2 field and X'E3' in the DITYPEI field, 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-I 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 4245 and 3203 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-I is started. They can be allocated to CPCS-I, when needed, by using the DKNALLO task.

CPCS-I can also use JES output facilities; see “JES Considerations with Dynamic Allocation” on page 1-16.

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 printers 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 MDEF macro. For more information on the DKNWTR task, see the *CPCS-I Programming 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-I Terminal Operations Guide*.

JES Considerations with Dynamic Allocation

Use DSAT macros that specify JESPRT*nc* to generate CPCS-I printed output when you use dynamically allocated SYSOUT data sets and JES writers. These macros provide a pool of SYSOUT data sets for use by the CPCS-I writer task. For more information on dynamic allocation for SYSOUT data sets, see “Example of Coding a DKNDSAT Entry for a JES Printer” on page 2-28.

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 2-22 for a description of using the DSAT macro. For a description of DKNADCB3, see the *CPCS-I Programming Guide*.

Enhanced Prime Capture Considerations

Enhanced Prime captures have the following characteristics:

- You only enter the MICR capture begin information once, before you start the run, with the PRIME field containing the tracer group number corresponding to the first entry to capture. All stacked entries run under the same cycle, the same MICR user parameters, and exit routines.
- Enter the MICR capture end commands only once after you capture the last stacked entry. The capture end screens show counts and totals corresponding to the last entry captured.
- Each tracer group identifies an entry.
- Output I-strings follow the same naming conventions as non-subset I-strings.
- You can optionally display MICR status message 18 before each new entry is captured; this allows the MICR operator to pull the pocketed items before the next entry is captured. You can suspend the capture or end it at this point instead of continuing to capturing more entries.
- If you do not pull pocketed items after each capture entry, kill items belonging to different entries can be separated by activating the “tracers in kill pockets option”.
- All non-subset 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 potential small bundles at the beginning of the entries.
- The function and its options are activated with the sort pattern definition options (O) record.
- As in conventional prime passes, captures are cancelled on end/suspend operator requests for the first entry when no data items were captured. Subsequent entries are cancelled, and the capture ends on the end/suspend operator requests when no data items were captured for the current entry.
- No empty I-strings are created. The “Null Tracer Group” MICR status error message is displayed when a tracer group with no data items is read.
- On capture restarts, the MICR Begin screen shows the information pertaining to the entry being restarted.

- "PRIOR ENTRY" is included on the manual restart pocket screens when the items shown belong to a previous entry.
- Dividers are only fed between entries when both merge before main and sorter initialization if entry breaks are active.
- If the MICR exit 2 (customize document processing) is active on an enhanced prime pass, it is also entered at the beginning of each stacked entry before the current string is closed. The exit receives the same data it gets (end of string dummy record) when it is entered at the end of conventional prime passes. On conventional prime passes, the exit is not entered if no data items were captured for the current entry (the current I-string is not created when this "null tracer group" error condition occurs).
- The Enhanced Prime function has the following restrictions:
  - The function can only be active on non-HSRR and non-CDMR prime passes.
  - You must use the same sort type, cycle and capture user parameters, and user exit routines to capture all the stacked entries in a pass.
  - Entries can only contain one tracer group.

Data Facility Storage Management Subsystem Considerations

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

1. SMS can not control the CPCS-I Mass data set. The Mass data set is allocated as follows:

   ```
   DD DSN=CPCSI.V01R01.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 can not have space released.
3. The duplex data sets can not 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 can not 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-I 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-I 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-I 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-I. It includes information about direct access storage, magnetic tape storage, and data-set preparation.

Direct Access Storage Device Requirements

The following is a list of those CPCS-I data sets whose space allocation and/or DCB characteristics depend on how you customize and use CPCS-I You should review this section and make appropriate adjustments to your allocation JCL prior to submitting it.

CPCS-I features, such at the Logging Subsystem, Propagation of Adjustments, and Enhanced System Manager, use other data sets that must also be carefully sized. Please refer to the appropriate documentation or appropriate sections of this guide for each feature you plan to install.

The following data sets are listed by ddname. Unless otherwise specified, these ddnames must be coded in the JCL when you start CPCS-I.

CDIF (Temporary Work Data Set for DKNCDIF)

This file is used by the DKNCDIF module to hold the Mass Data Set records for all strings that DKCDIF was asked to extract. It should therefore be sized to contain enough records to accomodate all items in each string that DKNCDIF was instructed to process during a single invocation.

This data set is dynamically allocated. It is coded in your DKNDSAT table, rather than in your JCL.

CKPTDS (Writer Checkpoint Record)

This data set contains status information about the spools and printers assigned to CPCS-I. 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 preallocated. It requires no special formatting; DKNWTRIF formats it, based on the PARM operand values in the JCL EXEC statement.

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

CLDSI, CLDSO (Temporary Work Data Sets for DKNCLSM)

DKNCLSM uses these data sets to store kill bundle records. It then sorts the records using the following keys:

Endpoint
Sending financial institution
Cycle code

When you request cash letter summaries, DKNCLSM writes those records from these data sets that meet your selection criteria.

These data sets are dynamically allocated. They are coded in your DKNDSAT table, rather than in your JCL.

CREFIN, CREFOUT (Work Data Sets for DKNCREF)

You can allocate these data sets as virtual I/O (for example, UNIT=VIO). They contain one active record for each kill bundle that meets the criteria for the

current run of DKNCREF. The recommended track allocation is VIO with the primary allocation equal to that required for the average run.

These data sets are dynamically allocated. They are coded in your DKNDSAT table, rather than in your JCL.

CYCLDS1 (Cycle Table Data Set)
This data set contains records for status, date, endorse date, name, and cycle quantity. You must preallocate 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=YES. The storage requirements are the same. Allocate it on a different DASD volume than the one used for CYCLDS1. (This data set is required only if DUPLEX=YES.)

DCVXI (Temporary Work Data Set for DKNDCVX)
DKNDCVX (the main DCVS program) uses this data set to store kill bundle records. It then sorts the records using the following keys:

Endpoint
Sending financial institution
Cycle code

When you request cash letter summaries, DKNDCVX writes those records from this data set that meet your selection criteria.

This data set is dynamically allocated. It is coded in your DKNDSAT table, rather than in your JCL.

DKNAB (Endpoint Name and Address Data Set)
The CPCS-I installation creates the endpoint name and address data set for reference by DKNRMIT, DKNKILL, DKNDCVS, and DKNCLSM. There is one record for each sort code entered in the file. DKNLOAD creates this file each time it runs (see “Data-Set Preparation” on page 1-28). The data set is unblocked and the logical record length is 305. You should allow 20% additional space for expansion.

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. It must be permanently allocated, because it contains the only complete copy of the kill-bundle data set during compress operations.

DKNDIV (Divider Slip Data Set)
This is a key-sequenced data set (KSDS) that provides the location of a divider slip in the MDS. DKNDIST updates this data set when prime-pass distribution completes.

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 per M-string processed during an input creation run. The recommended allocation for this data set is VIO.

This data set is dynamically allocated. It is coded in your DKNDSAT table, rather than in your JCL.

DKNKB (Kill-Bundle Data Set)

DKNRMIT and DKNKILL write one record to the kill-bundle data set for each printed kill bundle. These records are deleted by cycle when DKNECYC runs. DKNCOMP preformats 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 this data set:

$K = (N \div A) \times 1.1$

where:

K = Number of kill-bundle records required
N = Number of items remitted
A = Average number of items per kill bundle.

Base the calculation on peak-day volume.

Note: The value 1.1 is an allowance for kill bundles for high-speed reject re-entry (HSRR) runs and for the partial bundles at the end of each prime and subsequent pass for kill pockets.

DKNKD (Duplex Kill-Bundle Data Set)

This data set is a duplicate of DKNKB, and its storage requirements are the same. It is required only if DUPLEX=YES. Allocate it on a different DASD volume from the volume used for DKNKB.

If you specify DUPLEX=YES 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. The recommended allocation for this data set is VIO.

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.

This data set is dynamically allocated. It is coded in your DKNDSAT table, rather than in your JCL.

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. DKNMICR writes a record for every batch and block slip.

Consider how often you run DKNFILM 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-I.

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. It is required only if DUPLEX=YES.

If you specify DUPLEX=YES 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.

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 "MDEF Parameters" on page 2-30)

R = MAXRP + 9 (defined under "MDEF Parameters" on page 2-30)

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

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-I cold start is executed. This BLKSIZE may differ from the above chart depending on your CPCS-I 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. It is required only if DUPLEX=YES.

If you specify DUPLEX=YES 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.

FSCNTEMP (Temporary Work Data Set for DKNFSCN)

This file is used by the DKNFSCN module as a temporary dataset to hold a record for each Mass Dataset String that is being extracted for fine sorting. It should therefore be sized to contain space to hold the maximum number of strings that will be extracted for a single Fine Sort Group.

This data set is dynamically allocated. It is coded in your DKNDSAT table, rather than in your JCL.

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 (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.

Example:

String Type	Number of Physical Records
NDIRBLK	41
MDIREND	1
IDIREND	4
RDIREND	5
DDIREND	20

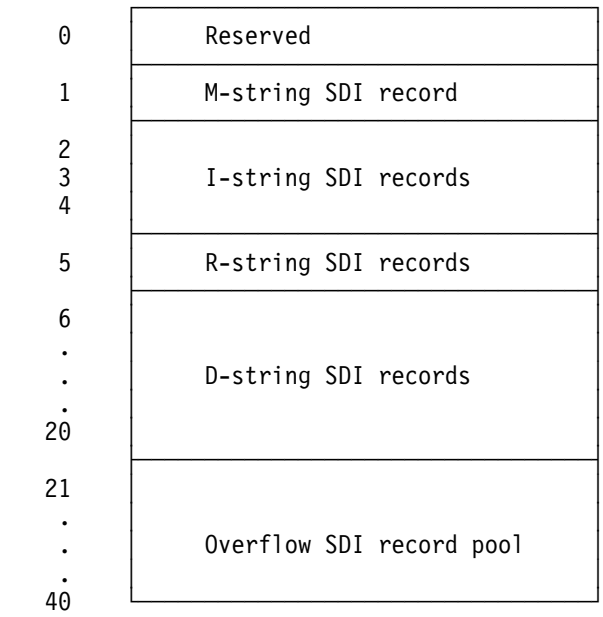


Figure 1-3. MDS Index Block Allocation

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.

KLDSI, KLDSO (Work Data Sets for DKNKILL)

You can allocate these data sets as virtual I/O (for example, UNIT=VIO). They contain one record for each string that will be kill-listed during this run of DKNKILL. These data sets are input and output to an internal sort of the string names that DKNKILL uses to generate the kill lists.

This data set is dynamically allocated. It is coded in your DKNDSAT table, rather than in your JCL.

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 MDEF macro parameters.

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 DKNMCRE or DKNDELE to free MDS data set space. For information about using DKNMCRE and DKNDELE, see the *CPCS-I Programming Guide* and the *CPCS-I Terminal Operations Guide*.

MFSORTED (Microfilm Work File)

This data set contains the sorted output from the two sorts 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 the work file can contain a complete data set of microfilm records.

This data set is dynamically allocated. It is coded in your DKNDSAT table, rather than in your JCL.

MRGEIN (MRGE Input Data Set)

DKNMRG1 creates this temporary data set. It contains all items for the prime I-string, HSRR I-string, and the R-string.

The following variables must be calculated by the user:

&DYNPSPA

&DYNSSPA

The default calculations are based on a 3350 DASD device.

Below is a sample extract of the member DSATUPDT variables calcuated when GENCLIB was run.

```
&MRINREC    SETC   '190'        MRGEIN   LRECL
&MRINBLK    SETC   '15390'         MRGEIN   BLKSIZE
&MROTREC    SETC   '190'        MRGEOUT  LRECL
&MROTBLK    SETC   '15390'         MRGEOUT  BLKSIZE
&MRHSREC    SETC   '200'        MRGHSRR  LRECL
&MRHSBLK    SETC   '15400'         MRGHSRR  BLKSIZE
&SCATREC    SETC   '200'        SCATIN1  LRECL
&SCATBLK    SETC   '15400'         SCATIN1  BLKSIZE
&ICREREC    SETC   '51'         ICRE LRECL
&ICREBLK    SETC   '15453'         ICRE BLKSIZE
&MCREREC    SETC   '59'         MCRE LRECL
&MCREBLK    SETC   '15458'         MCRE BLKSIZE
&MDSLREC    SETC   '50'        MDS  LRECL
&MDSBLKS    SETC   '15450'         MDS OUTP BLK
&PRADREC    SETC   '384'        PRAD LRECL
&PRADBLK    SETC   '15360'         PRAD BLKSIZE
```

Given a 3390 DASD device and using an optimum blocking factor of two blocks per track, the optimum blocking factor would be 27,200 bytes per track which includes an allowance for different 3390 models. We can recalculate the BLKSIZE, DYNPSPA (primary space allocation and the DYNSSPA (secondary space allocation).

Where:

&MRINBLK Optimum blocksize to use.

AIS Average items in a string.

&MRINREC Length of the DKNCRDI.

IPB Optimum items per block.

OBB Optimum bytes per block (27,200).

DYNPSPA Optimum primary blocks to allocate.

DYNSSPA Optimum secondary blocks to allocate.

The formula to calculate is:

```
&MRINBLK =  IPB  *  &MRINREC
IPB =  Integer value of {27,200/&MRINREC}
DYNPSPA =  Integer value of {AIS/IPB}
DYNSSPA =  Integer value of {100}
```

The following is an example:

```
&MRINREC =     190 bytes.
     AIS =  45,000 items.

&MRINBLK =  143  *  190  =  27,170
     IPB =  143
 DYNPSPA =  314
 DYNSSPA =  100
```

The resultant DSAT entry for the ddname MRGEIN would look similar to the following example:

```
MRGEIN   DSAT  DYNDEV=3390,                    x
               DYNDDN=MRGEIN,                 x
               DYNDSN=&HILVL..MRGEIN,         x
               DYNSTAT=NEW,                   x
               DYNNORM=DELETE,                x
               DYNCOND=DELETE,                x
               DYNUNIT=SYSDA,                 x
               DYNSTYP=BLK,                   x
               DYNPSPA=314,                   x
               DYNSSPA=100,                   x
               DYNLRCL=190,                   x
               DYNBSIZ=27170
```

The same formula should be used for the following DDNAMES in DKNDSAT:

MRGEOUT
SCATIN1
SCATIN2
MRGHSSR
MRGMATCH

Note: The LRECL for MRGMATCH and SCATIN2 is constant. Therefore the LRECL will not be calculated by running GENCLIB.

MRGEOUT (MRGE Output Data Set)

DKNMRG1 creates this temporary data set. It contains all items for the prime I-string, HSRR I-string, and the R-string in item sequence number order.

To calculate the size, see the formula under MRGEIN.

MRGHSRR (MRGE High Speed Reject Re-entry Data Set)

DKNMRG1 creates this temporary data set. It contains all items for the HSRR I-string.

To calculate the size, see the formula under MRGEIN.

MRGMATCH (MRGE Match Data Set)

DKNMRG3 creates this temporary data set. It contains all items for the prime reject D-string.

To calculate the size, see the formula under MRGEIN.

SCATIN1 (SCAT Match Data Set)

DKNSCA2 creates this temporary data set. It contains all items for all R-strings referenced.

To calculate the size, see the formula under MRGEIN.

SCATIN2 (SCAT Match Data Set)

DKNSCA2 creates this temporary data set. It contains all items for all R-strings referenced, in item sequence order.

To calculate the size, see the formula under MRGEIN.

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 SCRL online 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-I. It can optionally format the data set during a warm start of CPCS-I.

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-I modules acquire the work data sets dynamically.

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-I dynamic allocation requirements, see “Dynamically Allocating Data Sets” on page 2-22 and the *CPCS-I 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-I 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-I task that issues a write to the MDS goes into a wait state and this has a serious effect on all CPCS-I operations. For instructions on assigning two tape drives to one data set, see the *MVS/ESA JCL User's Guide*.

When CPCS-I opens a tape data set, it sends a message to the system operator indicating that it is creating a data set. CPCS-I identifies the external label for the tape. When CPCS-I 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 catalogued 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 generation data group, DKNICRET appends .T*dddhhmm* to the end of the data-set name, 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-I.

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

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.

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.

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.

DKNLT (Log Data Set)
This data set is required for logging if LOGDEV=TAPE is selected in the options module. DKNLT and DKNLTD replace the DKNLD and DKNLDD statements.

DKNRT (Recovery-Tape Data Set)
CPCS-I uses this data set to restore the MDS. CPCS-I 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-I initialization. The DCB parameters are obtained from the header label of the tape. You supply the volume serial numbers for the data set.

TAPEOUT (Printer Tape Output)
When specified as a tape in the DD statement, this tape substitutes for a dedicated printer. It contains CPCS-I output writer information from data written to spool data sets. For more information on tape-drive substitution, see "Printer Considerations" on page 1-15.

Data-Set Preparation

These data sets are listed by ddname and you must format them, with data, before running CPCS-I. 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 each time it runs. The data set contains:

- The sort code
- The receiving financial institution's name and address
- Four bytes 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. DKNRMIT and DKNDCVS use the information as a reference during the processing cycle.

You must run DKNLOAD to load the endpoint name-and-address data set. Use the JCL in the member GENBCFAB of CPCSI.V01R01.SDKNSAM1.

DKNBCF (Bank-Control-File Data Set)
DKNBCF is a partitioned data set (PDS) that contains members with the name and address of each financial institution that has work being processed. It also contains special processing requirements.

The JCL member GENBCFAB allocates a PDS for DKNBCF. Rerun DKNBCFLD if the default bank number or the processing options are changed.

Note: The default bank number is set to 000 when you install CPCS-I. (The COBOL copybook DKNCRBCF contains an 88-level field called BCF-DEFAULT-BANK.) You can change the default value in this definition and override the default bank processing. If you change this value, recompile the DKNBCFIO program and place the new version in your load library. For additional information about changing the default bank, see "Changing the Default Bank" on page 4-26.

If the DKNBCFLD 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. DKNDCVS and DKNRMIT accept member names as input. CPCS-I uses the endpoints in the table you select to determine what to include in the kill list or to summarize in a DCV.

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.

You should use the JCL in member GENEPT of CPCSI.V01R01.SDKNSAM1 to create the endpoint tables data set.

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*.

SMSPDEF (Enhanced System Manager Sort-Pattern Definition Library)
This DD statement points to the sort patterns used by Enhanced System Manager for the work-flow generation process. As shipped, the ddnames DKNSPDEF and SMSPDEF identify the same data set (CPCSI.V01R01.DKNSPDEF).

DKNSPDEF (Sort-Pattern Definition Library)
Sort patterns are defined to CPCS-I 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 value of *xxx* can be 1 to a maximum of 255.

Use the JCL in member GENSPDEF of CPCSI.V01R01.SDKNSAM1 to set up the sort-pattern definitions used during CPCS-I operation.

For a detailed description of the sort-pattern definition format, see “Sort-Pattern-Definition Record Formats” on page 4-30.

Data-Set Recovery Procedures

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

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-I JCL. For the suggested recovery parameters, see Figure 1-4 on page 1-30.

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

Whenever you cannot access data on the MDS or the index, run an MDS or index recovery to restore the data. CPCS-I 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-I 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 then 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 1-4 contains the suggested recovery procedures for the CPCS-I 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 1-4, 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, the answer for the RCVY Support Files is *NO*.

DKNRCVY DD name for the logging recovery index
DKNRCVS DD name for the logging VOL-SER file
DKNRCVT DD name for the logging status file

Figure 1-4. 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 1-4 on page 1-30. For information about using the RCVY task, see *CPCS-I Terminal Operations Guide*.

1. Reallocate and initialize logging files.

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

   Note: All files are allocated as empty. 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 “Mass Data Set and Index Recovery” on page 1-34.

4. If existing MDS data is destroyed, change the PARM statement to `PARM='RECV,NBTH'` or change the PARM statement to `PARM='RECV,RBTH'`.

   For more details about these parameters, see “BOTH Recovery (PARM=‘RECV,BOTH’)” on page 1-35 and “Restart MDS and Index Recovery (PARM=‘RECV,RBTH’)” on page 1-36.

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

   For more details about these parameters, see “Mass Data Set Recovery without Recovery Support Files” on page 1-33.

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

   For more details about these parameters, see “Mass Data Set Index Recovery” on page 1-32.

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

   For more details about these parameters, see “Selective Recovery” on page 1-36.

8. Change the CPCS-I startup JCL in the CPCS-I PROC.

   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. You can also use a DKNDUMP tape.

9. Copy the duplex tracer group file over to the newly allocated tracer group file (DKNTG).
10. Start CPCS-I.
11. Sign on to CPCS-I 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-I 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-I Programming 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-I recovery parameters. These parameters are valid only when LOG=YES is specified in the MDEF macro.

Mass Data Set Index Recovery

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-I 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-I 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 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 without Recovery Support Files

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 1-4 on page 1-30.

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-I startup JCL by specifying the recovery tape serial numbers for the recovery data set (DKNRT). If you use DKNDUMP regularly, ensure that the first tape serial number corresponds to the first tape that DKNDUMP created during its last run. Specify all tapes created since then, in the order of their creation. If DKNDUMP is not being used, you must specify tape serial numbers in the JCL for all tapes that contain a log of data that is now active.

Each time CPCS-I 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 cancelling 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 1-35). 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. 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” on page 1-36.
- 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 1-35.

- 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 1-35.

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.

CPCS-I does not process any incomplete 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-I 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-I 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.

Mass Data Set and Index Recovery

This section explains the BOTH recovery process and the Restart BOTH recovery process.

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 Sort Control Block (RSCB) in the tracer data set. Without the RSCB information for an entry, MICR cannot restart the entry.

Delete the old mass data set and index and allocate new ones, preferably to different devices. 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-I 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-I 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 do exist:

  DKNRCVY DD name for the logging recovery index
  DKNRCVS DD name for the logging VOLSER file.
  DKNRCVT 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-I 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-I 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-I Terminal Operations Guide* and the *CPCS-I Programming 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 without Recovery Support Files" on page 1-33 and "Mass Data Set Index Recovery" on page 1-32). 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 MDS and Index 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-I 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.

MDS and Index Recovery with Intact Tracer Data Set

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

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 without Recovery Support Files" on page 1-33). 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. For examples of recovery options, see page 1-34.

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-I 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.

Suggestions

- Use the CPCS-I program DKNDUMP on a regular basis (see "Log Data-Set Utilities" on page 2-20). This program generates a tape data set that contains all the active data on the MDS. In effect, DKNDUMP gives an origin point for the data set and simplifies the maintenance and accumulation of tape volumes.

- Mark the log tapes with the date, time, and sequence number as they are created and then dismounted. If you use DKNDUMP, write the serial number of the existing dump tape on the label.
- 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-I. All data sets that are duplexed can be recovered by:

1. Ending CPCS-I 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-I.

The following CPCS-I data sets are duplexed:

CYCLDS1 The cycle-table data set. The duplex data set name is CYCLDS2.
DKNTG The pass-to-pass control data set. The duplex data set name is DKNTGD.
DKNKB The kill-bundle data set. The duplex data set name is DKNKD.
DKNMF The microfilm data set. The duplex data set name is DKNMFD.

You can use IDCAMS to restore the lost data for the kill-bundle and microfilm data sets. 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 IDCAMS, see *DFSMS/MVS V1R2.0 AMS for the ICF*.

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

CPCS-I-Supplied Command Procedures

Each of the following supplied command procedures is a series of job steps, each containing EXEC statements for assembly, compile, and link-edit. The load library is named CPCSI.V01R01.SDKNMOD1. The assembly and link-edit parameters for the assembler programs are coded in accordance with the table of link-edit parameters. The data sets CPCSI.V01R01.SDKNSRC1 and CPCSI.V01R01.SDKNMOD1 are assumed to be cataloged. If the data sets are not cataloged, the volume identification must be coded.

Use the member CMPL0050 in CPCSI.V01R01.SDKNSAM1 to compile and link-edit all SAA AD/Cycle COBOL/370 source modules necessary for CPCS-I. You can change this member to meet your installation standards.

Use the member CMPL0030 in CPCSI.V01R01.SDKNSAM1 to assemble and link-edit those source modules that are called by other source modules.

Use the member CMPL0040 in CPCSI.V01R01.SDKNSAM1 to assemble and link-edit all other assembler source necessary for CPCS-I, except the MICR modules. This JCL places load modules in CPCSI.V01R01.SDKNMOD1. You can change this member to meet your installation standards.

Use the member CMPL0060 in CPCSI.V01R01.SDKNSAM1 to assemble and link-edit all MICR assembler source modules necessary for CPCS-I. This JCL places load modules in CPCSI.V01R01.SDKNMOD1. 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.

Assembly Considerations

- Concatenate the CPCS-I source libraries CPCSI.V01R01.SDKNSRC1 and CPCSI.V01R01.SDKNMAC1 to your SYSLIB in the assembly step. Ensure that the first DD statement is for the largest block size.
- Concatenate your SYS1.AMODGEN library to the SYSLIB of your assembly step.
- In the SYSIN statement, refer to the source library, CPCSI.V01R01.SDKNSRC1.

Link-Edit Considerations

- Use the IEFBR14 utility to allocate your load library.
- 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-I 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 and associated user exits *must* be link-edited with RENT attributes.
- OLRR user-edit routines *must* be re-entrant and be link-edited with RENT attributes. DKNMOLRI *must* be included with DKNOLRR.

  DKNLOADR has a conditional assembly parameter of &LE370. The default value is 1, which indicates that you have LE/370 installed. The data set name for CLIB3 must contain the correct name for the LE/370 macro library.
- SCI and table modules *must* be link-edited as RENT, but need not be re-entrant.

  You should change the link-edit parameters specified for your user modules to include the RENT parameter:

```
//X  EXEC COBOL,   (or whatever proc you are using)
//     PARM.LKED=' ................,RENT'
```

  You should do this only for source modules that are changed, as described above, using the CBL statement.

Changing CPCS-I Control Blocks

Certain CPCS-I processing modules maintain the integrity of the CPCS-I Object Code Only (OCO) programs. The following table lists these token-table modules and their corresponding control blocks.

Token Table	Control Block
DKNAPCGT	DKNAPCB
DKNAPTGT	DKNAPTCB
DKNBCFGT	BCF
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-I 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. The parameter DNAME should be specified as APTCB. 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 CMPL0040 in CPCSI.V01R01.SDKNSAM1 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-I, except the CPCS-I OCO modules, when additions are made to these control blocks.

CPCS-I Control Data Sets

CPCS-I control data sets contain customized information used to control the execution of some CPCS-I tasks.

Bank Control Data Set

The bank control (BCF) partitioned data set describes each processing bank. It contains name and address information and unique processing requirements for each financial institution in a multi-bank facility.

The data set contains a member for each processing bank and one member for the default bank. Members are named BANKxxx, where xxx is the bank number.

At capture initialization time, the MICR task accesses the bank control data set to retrieve processing options pertaining to the processing bank. Other CPCS-I tasks (like KILL, RMIT and CLSM) access the data set to retrieve the name and address information and special instructions to be included in their reports.

See “Bank Control File Record Formats” on page 4-15 and “Sort-Pattern-Definition Record Formats” on page 4-30, for information on how to set up the Bank Control Data Set.

Endpoint Table Data Set

The endpoint table (EPT) partitioned data set contains sets of endpoint IDs, grouped by user established criteria. The data set contains a member for each endpoint ID group (or table).

Endpoint tables facilitate the invocation of some CPCS-I tasks (like KILL, RMIT and CLSM) by entering endpoint table names for selected endpoints.

Endpoint Name and Address Data Set

The endpoint name and address (ABA) data set consists of the endpoint ID, name, address, and kill bundle type and format of each financial institution to which items are sent.

Some CPCS-I tasks (like KILL, RMIT and CLSM) access the Endpoint Name and Address data set to retrieve the receiving financial institution’s mailing information.

Sort Pattern Definition Data Set

The sort pattern definition (SPDEF) partitioned data set specifies the MICR options and document processor features associated with each sort type.

The data set contains a member for each existing sort type. Members are named SPTYPxxx, where xxx is the sort type.

At capture initialization time, the MICR task retrieves the information associated with the operator-entered sort type. It also uses the information to initialize the document processor. This initialization controls the execution of the capture.

See “Sort-Pattern-Definition Record Formats” on page 4-30, for information on how to set up the Sort Pattern Definition data set.

CPCS-I Internal Data Sets

The CPCS-I internal data sets reflect the status of the items being processed by CPCS-I and control CPCS-I processing.

Divider Data Set

The divider data set (DDS) contains information used to control the optional divider resynchronization process. The CPCS-I distribution task updates the DDS after each of the first three passes in which divider resynchronization is active. During subsequent passes, on divider out-of-match situations, the CPCS-I MICR task uses DDS MDS location data to resynchronize record matching.

The DDS records are prefixed with an 18-character key in the following format:

```
CCxxxxxxxxyyyyyyyy
```

where:

CC Two-character cycle identifier.

xxxxxxxx Field number 5 (the rightmost eight positions of the field data)

yyyyyyyy Divider Serial Number field from BCF (the rightmost eight positions of the field data)

DDS records are deleted by the CPCS-I end-cycle task.

Kill Bundle Data Set

The kill bundle data set (KBDS) contains kill bundle summary information. The CPCS-I KILL/RMIT tasks create a kill bundle record for each kill-bundle they process. Cash letter creation tasks receive kill-bundle summary information from the KBDS. The MCRE task uses the KBDS information to identify killed D-strings.

KBDS records are deleted by the CPCS-I end-cycle task.

Mass Data Set

The mass data set (MDS) contains the codeline records being processed by CPCS-I. Codeline records are logically grouped into strings. Identification, control and status information is placed by CPCS-I tasks in the string directory entry (the first record of each string) as strings get created, processed and deleted. New strings are created as items are captured, rejects corrected, and entries balanced. Strings are deleted that are not required for the execution of subsequent tasks. The end cycle task (last CPCS-I task run for each cycle) deletes all the remaining strings for the cycle.

The MDS has an associated index data set that contains the names and relative block addresses (RBAs) of all active strings. The index data set is used to locate active strings in the MDS.

Microfilm Data Set

The microfilm data set (MFDS) contains microfilm summary information. The CPCS-I MICR task creates an I-string microfilm record whenever an end-of-cartridge condition occurs on prime passes that use the microfilm option. The CPCS-I distribution task copies the microfilm records to the MFDS when I-strings are distributed. The MFDS is used as input to the FILM reporting task.

MFDS records are deleted by the CPCS-I end-cycle task.

Tracer Data Set

The tracer data set (TDS) contains tracer group control and summary information. The CPCS-I MICR task creates a TDS record for each tracer group captured on prime passes and updates the records during subsequent passes. The MICR task accesses the TDS data to control pass-to-pass processing and enable resynchronization of the codeline data match function during subsequent passes. Other CPCS-I tasks access the TDS data to provide pass-to-pass reconciliation data.

The CPCS-I MICR task also maintains TDS control information used for the MICR restart process.

The TDS is initialized by CPCS-I cold starts.

Cycle Data Set

The cycle data set (CDS) contains the cycle status, cycle date, endorse date, and end-prime status information for each cycle ID. CPCS-I tasks access the CDS to obtain cycle related data. CPCS-I operators and CPCS-I tasks update the contents of the CDS to control the execution of tasks that are only startable for certain cycle and end-prime status codes.

The CDS is initialized by CPCS-I cold starts.

Item-Sequence Data Set

The Item-Sequence data set (ISDS) contains the next item sequence number to be assigned to each of 100 logical document processors. Logical processors 1 through 96 are for MICR-controlled document processors. Logical document processors 97 through 99 are for electronic input. Logical document processor 0 is for applications requiring small groups of unique item sequence numbers. CPCS-I tasks access ISDS to obtain unique codeline record sequence numbers.

The ISDS maintains the last eight digits of the twelve digit sequence number. Applications are responsible for assigning the other four digits. Sequence numbers are automatically reset to zero when sequence number 99000000 is assigned.

The ISDS is initialized by batch job GENISEQ.

DKNSMSG–System Message Handler

The system message handler (DKNSMSG) module gets, formats, and sends CPCS-I messages. DKNIMENU is a member of CPCSI.V01R01.SDKNSAM2 that can be customized for US system messages. DKNIMENG is a member of CPCSI.V01R01.SDKNSAM2 that can be customized for UK system messages. After customization, use the DKNGSMSG JCL in CPCSI.V01R01.SDKNSAM1 to create the system message file used by DKNSMSG.

See the *CPCS-I Programming Guide* for additional information on the DKNSMSG system message handler.

Chapter 2. Installation and Generation Procedures

Overview

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

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

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

1. Use SMP/E to unload the distribution tape to your system, and run the installation job streams as described in the *CPCS-I Installation Guide*.
2. Customize the CPCS-I 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 2-37.) Link the MICR modules together in a workable DKNMICR module.
3. Perform an MDEF generation to describe the program features, user volumes, and operational environment. (For information on generating DKNMTASK, see “Master Task Generation” on page 2-29.) Use this generation as input to link the DKNMTASK module.
4. Allocate and initialize all CPCS-I data sets in accordance with the requirements of your system.
5. Run the sample problems to verify that CPCS-I has been correctly installed. For instructions on how to run the sample problems, see the *CPCS-I Installation Guide*.
6. Design, code, and test user document-processing routines. Chapter 4, “User Programming Requirements,” describes the requirements for creating user document-processing routines.
7. Generate data security tables and routines as required for your system. For information about controlling CPCS-I resource access, see “Security Options” on page 4-83.
8. Code user-exit routines as required for your system.
9. Customize CPCS-I reports by changing the endpoint name-and-address data set (DKNAB) and the bank-control-file data set (DKNBCF).

Macro Descriptions

The CPCS-I 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 a group of items.

Vertical bar (|) A vertical bar between operands indicates that you must code one parameter from among the values separated by the 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-I uses if you omit the parameter.

Braces { } Braces indicate mutually exclusive parameters.

Never code brackets, the vertical bar, underlines, braces, or subscripts.

Accessing the VSAM Table

The CPCS-I 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-I resource, provides minimum system overhead in file processing.

Note: There is no requirement that files used during CPCS-I processing be listed in DKNVSTBL.

The CPCS-I 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-I cold start occurs.
- For each VSAM table-file identification, the table entry can specify exit routines that run during CPCS-I startup, CPCS-I 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-I 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-I 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-I VSAM Table

The following is a sample of CPCS-I VSAM table entries:

```
DIV      DKNVSENT DDN=DKNDIV,      - CPCS-I 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-I 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-I START-UP
                  ENDEXIT=SAMPTERM   - CALL SAMPTERM AT CPCS-I 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-I.

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-I 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-I startup.

ENDEXIT=*exit name*
This 8-character name for an optional termination-exit module. This module is called during CPCS-I termination.

Processing Description

This section describes the CPCS-I VSAM Manager DKNVSMGR operations.

During CPCS-I 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-I (a flag on the entry indicates *empty* on CPCS-I 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-I VSAM operations. Event control blocks that are incorporated in the VSAM table make requests and signal the completion of the requests.

During CPCS-I 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-I 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-I 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-I 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-I End Processing

DKNVSMGR scans the VSAM table for files that were opened during CPCS-I 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.

Using Entry-Sequenced Data Sets

The VSAM input/output module (DKNVSMIO) and related modules support the manipulation of entry-sequenced data sets (ESDS). The modules provide for writing, reading, deleting, and updating ESDS records. Since VSAM does not support ESDS record deletion, a record header that includes a delete flag is provided. The record header is available as a macro named VSESDS in the DKNVSMAC source module. The following coding generates the header:

```
(label) VSESDS   DSECT=NO
```

The following coding generates a DSECT of the header:

```
(label) VSESDS   DSECT=YES
```

This control block is also available as a COBOL copybook named DKNCVHDR. To generate a VSMVOFL control block, use the following COBOL code:

```
       .
       .
DATA DIVISION.
WORKING-STORAGE SECTION.
    COPY DKNCVHDR.
       .
       .
       .
```

The record header has the following fields:

RBA Relative byte address of the record. After a successful write operation, this value is inserted into the header and the record is rewritten.

DELETED_FLAG
Set to a value of DELETED_RECORD after the record is deleted. Data read from a deleted record is returned to a caller's read buffer, but the record length (VSMVOFL label VSMLRECL) is not updated.

UNTOUCHABLE_FLAG
Set to a value of UNTOUCHABLE by a user of the data set. Data read from an untouchable record is returned to a caller's read buffer, but the record length (VSMVOFL label VSMLRECL) is not updated.

SECONDARY_PTR
Provided for the caller's use. The caller can use the flag and its associated field SEC_RBA_PTR to establish RBA pointers for backup data sets. The caller can use an equate, SEC_PTR_VALID, to set SECONDARY_PTR to indicate that the SEC_RBA_PTR contains valid information.

SEC_RBA_PTR
Provided for the caller's use. The caller can store a cross-reference to the RBA of the record in a backup file here.

ESDS Application Guidelines

The following guidelines apply to entry-sequenced data sets:

- The file can be allocated using the VSAM access method services utility IDCAMS.
- Records used to initialize the file can be contained in a TSO data set that has a fixed record length (no longer than the maximum length supported by the file). The TSO command REPRO or the IDCAMS utility option REPRO can be used to insert the TSO data set record into the ESDS.
- DKNVSOPN writes an UNTOUCHABLE record to an empty ESDS during open processing.
- The ESDS record header, VSESDS, must prefix each record written to the data set.
- The ESDS record header, VSESDS, and other record data are returned after a read operation.
- To make record updates easier, the caller should use fixed-length records in an ESDS.

DKNVSMAC Source File

The DKNVSMAC source file contains several macros that assist VSAM application developers who use the DKNVSMIO interface. These macros are described in the following text. To use the macros, the source module, DKNVSMAC, must be copied into the source program preceding the macros specification.

- VSMVOFL is a macro that generates the VSAM open file list control block. The macro is used as shown here in assembler language.

```
(label)  VSMVOFL DSECT=NO     .This syntax allocates a VSMVOFL control
*                             . block
(label)  VSMVOFL DSECT=YES    .This syntax generates a dummy control
*                             . section which maps the VSMVOFL
```

- VSGENERR is a macro that generates EQUATES for error codes set by all DKNVSxxx routines. These values are complemented (set to a negative (–) value) before returning to the caller to indicate non-VSAM errors. The VSAM error codes have a positive (+) sign. VSAM return codes are returned in register 15, as documented in the *MVS/ESA VSAM Administration: Macro Instruction Reference*. The reason code for a VSAM failure is returned in the VSMVOFL at the label VSM_REASON. The macro VSGENERR is coded as shown here.

  ```
  (label)  VSGENERR             .This syntax generates the error equates
  ```

- VSESDS is a macro that generates the entry-sequenced data set (ESDS) record header. This header should prefix each record in the ESDS. The macro can be used as shown here in assembler language.

  ```
  (label)  VSESDS  DSECT=NO     .This syntax allocates a VSESDS record
  *                             . header
  (label)  VSESDS  DSECT=YES    .This syntax generates a dummy control
  *                             . section which maps the VSESDS header
  ```

- VOFLBASE is a macro that generates a VOFLBASE control block or provides a DSECT for mapping the control block.

  ```
  (label)  VOFLBASE DSECT=NO     .This syntax allocates a VOFLBASE
  *                              . control block
  (label)  VOFLBASE DSECT=YES    .This syntax generates a dummy control
  *                              . section which maps the control block
  ```

- Other macros contained in DKNVSMAC and used in the DKNVSxxx modules are not described here.

COBOL Copy Definitions

The COPY definitions DKNCRPKT, DKNCRTRK, and DKNCRTG 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 CPCSI.V01R01.SDKNSRC1 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). DKNDFTO, DKNRMIT, and DKNDCVS 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. Whenever you change this member in the CPCS-I source

	library, you must recompile DKNRMIT, DKNLOAD, DKNDFTOX, and DKNDCVX.
DKNCRTG	You must change this data description to contain the number of pockets of the largest sorter, minus 1. Change the OCCURS statement for the number of pockets.

Extended Architecture Considerations

This section outlines the special considerations for installing CPCS-I assembler and COBOL modules in a System/370 Enterprise Systems Architecture environment.

Assembler Modules

CPCS-I assembler modules contain code that indicates the addressing mode (AMODE) and the residency mode (RMODE) of the CSECT. If a CPCS-I 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
```

SAA AD/Cycle COBOL/370 Modules

You must install SAA AD/Cycle COBOL/370 Release 1, or higher, to compile CPCS-I COBOL modules that can run above the 16MB line. The COBOL modules that IBM ships with the CPCS-I system are all designed to run above the 16MB line (AMODE 31, RMODE ANY).

If you write SAA AD/Cycle COBOL/370 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)
```

For the latest requirements concerning JCL statements for compiling SAA AD/Cycle COBOL/370 modules, see the *CPCS-I Installation Guide*.

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-I relies on the data-set duplexing functions to re-establish the database and to continue processing existing work under monetary control. You can select the duplex option by specifying DUPLEX=YES in the master task generation options.

Data sets that represent monetary 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-I 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.

Capture of Duplexed Data Sets

Each time CPCS-I 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-I data set that is duplexed, you must stop CPCS-I. 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 1-37.

Logging CPCS-I Data

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

The CPCS-I system is shipped with the Logging function inactive (LOG=NO in the MDEF macro). You can use the sample problems described in the *CPCS-I 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-I processing, changing JCL statements, and restarting CPCS-I.

CPCS-I Logging supports a number of options that enhance the CPCS-I 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 MDEF parameters to activate Logging.
2. Verify that RGENDEF specifies the Logging options.
3. Verify that RCVUNTBL specifies the device names and types.
4. Verify that DKNROPTS specifies default settings for DKNRCVY.
5. Assemble and link-edit the DKNMTASK module.
6. Change the DKNDSAT parameters.
7. Assemble the Logging options module.
8. Assemble the Logging program modules.
9. Allocate and initialize the Logging data sets.
10. Create the generation data group (GDG) for the log data-set backups.
11. Update the CPCS-I startup JCL.
12. Restart CPCS-I with PARM='COLD' in the EXEC statement of the JCL.
13. Back up the Logging files (optional).

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

Step 1: Change the MDEF Macro to Activate the Logging Functions

To activate the Logging functions of CPCS-I you must change the MDEF macro parameters and generate a new DKNMTASK module. Change the following MDEF macro parameters:

- Set the buffer ratio (BFRAT) to a non-zero value.
- Set the number of tape buffers (TPBFNM) to a non-zero 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 MDEF macro.
- Set the LOGGEN parameter to DKNRCVGN. This is the name of the Logging options module that controls all functions of Logging.

  Note: DKNRCVGN is the default module name.
- The block sizes for the logging files are calculated using the formula:

  `(BFRAT×(BLKSIZE + 12)) + ((BFRAT × 4) + 16)`

  This value is then reported as an MNOTE in the assembly listing.

For details about the MDEF macro parameters, see “MDEF Parameters” on page 2-30.

To deactivate Logging, set the LOG parameter in the MDEF macro to NO and generate a new DKNMTASK module.

Step 2: Use the RGENDEF Macro to Specify the Logging Options

The Logging options module controls the environment for Logging. The RGENDEF macro assigns no default values; you must specify all parameters. A sample RGENDEF is in CPCSI.V01R01.SDKNSAM2 member DKNRCVGN (see Figure 2-1 on page 2-13). The default module name is DKNRCVGN.

Note: The following parameters require that the initialization job is run after changes have been generated. These parameters affect file changes:

RCVSIZE Number of records contained on DKNRCVY/DKNRCVYD

BLKSIZE Block size of DKNRCVY/DKNRCVYD

MAXOPEN Number of records contained on DKNRCVCK/DKNRCVCD calculates the block size of these files as 3 x value.

VLSRSIZ Number of records contained on DKNRCVSR/DKNRCVSD

```
         TITLE 'CPCS-I-SAMPLE LOGGING DEF'                      X
DKNRCVGN RGENDEF LOGDEV=DISK,                                   X
               DPXLOGG=YES,                                     X
               DPXBKUP=NO,                                      X
               DPXCNTL=NO,                                      X
               BACKUP='BKUP',                                   X
               RCVSIZE=30011,                                   X
               BLKSIZE=11475,                                   X
               VLSRSIZ=50,                                      X
               MAXATMP=60,                                      X
               MAXOPEN=99,                                      X
               MAXSTGS=99,                                      X
               TPRETPD=3,                                       X
               BLDTBL=YES                                       X
          RGENDSCT DSCTCD=YES,GENPRMS=YES                       X
          END                                                   X
```

Figure 2-1. Sample Logging Options Definition

The parameters are specified as follows:

LOGDEV= {DISK | TAPE}
Specifies whether the MDS is logged to disk or tape.

DISK Specifies that the MDS is logged to disk log files DKNLD1 and DKNLD2. MDS strings are logged to either of these two files until it is filled. Logging then switches to the other file while the first file is backed up.

TAPE Specifies that the MDS is logged directly to tape file DKNLT. DKNLTD is also created, if duplexing is specified. For more information, see the description of the DKNLT tape data set in “Optional Tape Data Sets” in “Optional Tape Data Sets” on page 1-27.

DPXLOGG= {YES | NO}
Specifies whether the disk log files (DKNLD1, DKNLD2) are duplexed when they are created.

If you have specified LOGDEV=TAPE, you must use NO for this parameter.

YES If LOGDEV=DISK, this option specifies that the disk log files (DKNLD1, DKNLD2) are duplexed when they are created.

NO If LOGDEV=DISK, this option specifies that the disk log files (DKNLD1, DKNLD2) are not duplexed when they are created.

DPXBKUP= {YES | NO}
Specifies whether the Logging backup files are duplexed when they are updated.

YES If LOGDEV=DISK, this option specifies that two disk log backup files (DKNLD, DKNLDD) be created. The Logging backup program performs this duplexing.

If LOGDEV=TAPE then two identical log tapes (DKNLT, DKNLTD) are created. CPCS-I performs this duplexing.

NO If LOGDEV=DISK, this option specifies that only one disk log backup file (DKNLD) is created.

If LOGDEV=TAPE, this option specifies that only one log tape (DKNLT) be created.

DPXCNTL= {YES | NO}

Specifies whether the Logging control files are duplexed when they are updated.

YES Specifies that the Logging control files be duplexed when they are updated.

NO Specifies that the Logging control files not be 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-I task.

'SUBM *member***'**
Causes DKNSUBM to start automatically and to submit the specified member of the partitioned data set that the DKNSUBMT DD statement defines in the DKNJPROC job stream.

'SUBM *dsn(member)***'**
Causes DKNSUBM 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 DKNJPROC 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.

RCVSIZE={*nnnn***}**

Specifies the number of records on the string names file (DKNRCVY). This value is calculated as:

(number of strings created each day × log tape retention period) × 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 upon 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 re-entry, 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.

BLKSIZE={*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 169.

Note: Although making the number larger causes a faster initiation of CPCS-I when reading the whole string names file, a larger number also causes larger blocks to be written whenever CPCS-I processing updates a logical record. If this number is too large, CPCS-I operation can be adversely affected during peak processing periods.

VLSRSIZ={*nnnn*}
Specifies the maximum number of log files tracked by the Logging function. This value should be calculated as:

```
(number of tapes created each day × log tape retention period) × 1.5
```

MAXATMP={*nnnn*}
Specifies the number of consecutive attempts to find an available record on the string names file (DKNRCVY) before the information on the string is lost to the Logging index file.

This value should never be less than 50 or more than 1000. A general guideline for this value is to set it to approximately 1/500th (0.002) of the RCVSIZE value.

MAXOPEN={*nnnn*}
Specifies the maximum number of concurrently open strings. Set this parameter to the same value as MAXOPEN in the MDEF macro. For more information about the MDEF macro parameters, see “MDEF Parameters” on page 2-30.

MAXSTGS={*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.

TPRETPD={*nnnn*}
Specifies the number of days that the string name entries would be maintained on the string name file (DKNRCVY).

BLDTBL={YES | NO}
Specifies whether the string names file (DKNRCVY) is above the 16MB line during CPCS-I initialization. Loading the file into memory causes initialization to

take longer, but reduces the processing overhead of the Logging function during on-going runs and also provides enhanced performance during the selective recovery process.

Multiply the RCVSIZE value by 169 to calculate the required amount of memory.

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 2-2). If you modify this table after installing CPCS-I, activate the changes by generating a new version of the DKNLOGX 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    CL6'2400  ',XL2'8001'
         DC    CL6'3400  ',XL2'8003'
         DC    CL6'3480  ',XL2'8080'
         DC    CL6'      '             END OF TABLE MARKER
```

Figure 2-2. RCVUNTBL Copybook Example

Step 4: Verify the DKNROPTS Macro Definition

DKNROPTS is a macro that contains the default settings for program DKNRCVY. The current default values as shipped are NO for Tracer Group Updating (&TGUP) and blank for &RDX. Spaces in this field indicate that the definition is in the first record on the log tape. Modify DKNROPTS according to your installation requirements.

```
         SPACE 3
         GBLC  &TGUP,&RDX
****
****     ASSIGN VALUES TO RCVY UPDATE SCREEN VARIABLES
****
&TGUP    SETC 'NO'
&RDX     SETC
         EJECT
```

Figure 2-3. DKNROPTS Example

Step 5: Assemble and Link-Edit DKNMTASK

The assembly and link-edit JCL for the CPCS-I master task (DKNMTASK) is in member GENMTASK of CPCSI.V01R01.SDKNSAM1. This job assembles the MDEF macro and builds the DKNMTASK module.

File Allocation Note: The Logging file block size is reported by an MNOTE during the assembly of DKNMTASK. The following files, which are allocated in steps 6, 9, and 10, require this block size:

Defined in LOGBKUP, LOGCOPY, LOGDCOPY, and by DKNDSAT entry:

- DKNLD
- DKNLDD

Defined in LOGALLO:

- DKNLD1
- DKNLD1D
- DKNLD2
- DKNLD2D

Defined in LOGGDG:

- DKNLDX.

Step 6: Modify DKNDSAT

The DKNDSAT data set describes dynamically allocated data sets. Change the DYNBSIZ and DYNLRCL parameters within the DKNLDD and DKNLD entries, using the block-size value that was reported from the GENMTASK job.

Note: If the LOGDEV parameter in the Logging options macro RGENDEF is changed to a value other than the original value as delivered from IBM, you must add entries for the tape data sets DKNLT (and DKNLTD if duplexing). You may also delete the DKNLD and DKNLDD entries if you do not plan to use disk logging in the future.

For information about using the DSAT macro to create DKNDSAT entries, see “Dynamically Allocating Data Sets” on page 2-22.

Step 7: Assemble the Logging Options

Assemble and link-edit the Logging options module before assembling the application programs. Submit the JCL from member LOGGEN in CPCSI.V01R01.SDKNSAM1.

Step 8: Assemble the Logging Program Modules

Assemble and link-edit the Logging assembler source modules, using member CMPLLOG in CPCSI.V01R01.SDKNSAM1.

Step 9: Allocate and Initialize the Logging Data Sets

Allocate and initialize the Logging data sets by running the LOGALLO job from CPCSI.V01R01.SDKNSAM1. The JCL in member LOGINIT initializes Logging data sets that were previously allocated.

The following data sets (with high-level qualifier CPCSI.V01R01) are allocated and initialized:

DKNRCVY
DKNRCVS
DKNRCVK
DKNRCVT

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

DKNRCVYD
DKNRCVSD
DKNRCVKD
DKNRCVTD.

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

DKNLD1
DKNLD2.

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

DKNLD1D
DKNLD2D.

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 LOGGDG of CPCSI.V01R01.SDKNSAM1. Set the `LIM(NNN)` parameter in the GDG statements to the number of catalogue entries you want to maintain for each data set.

Step 11: Update the CPCS-I Startup JCL

Uncomment the Logging DD statements in the CPCS-I startup procedure (DKNJPROC of CPCSI.V01R01.SDKNSAM1). The JCL contains detailed instructions about which DD names to uncomment.

Step 12: Restart CPCS-I

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

For a description of EXEC statement operands and their effects on CPCS-I startup, see “CPCS-I Startup JCL Definition” on page 1-6.

Step 13: Batch Backup of Logging Files

The JCL in member LOGBKUP 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-I SUPERVISOR
```

or

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

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 MDEF macro parameter BFRAT in the DKNMTASK generation. You also specify the number of available buffers for DKNLOGX (MDEF 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 MDS by starting CPCS-I 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-I 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-I 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 on recovering data sets, see the recovery procedures described in “Data-Set Recovery Procedures” on page 1-29.

Log-Tape Synchronization

Whenever CPCS-I 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-I 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 correct log data-set definition in the first record of each log data set. RCVY normally uses this system-generated definition record to define the log data set being recovered.

However, in early releases of CPCS, this log data set definition record did not exist. To recover from such log files, you must assemble an external log data set definition module. This “DKNRDX” module can then be specified as an override on RCVY's Recovery Cycle Specification screen.

Use the following steps to create a new definition to override the DKNRDX50 definition:

1. Change the RBFRAT field to match the MDEF macro BFRAT parameter value that was in effect when the data set was created.
2. Change the RTPBFN field to match the MDEF macro TPBFNM parameter value that was in effect when the data set was created.
3. Change the RBLKSZ field to match the MDEF macro BLKSZ parameter value that was in effect when the data set was created.
4. If the log data set was created under CPCS (Program No. 5734-F11) 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.

Log Data-Set Utilities

There are two data-set duplexing utilities. DKNDUMP dumps all active data from the MDS. DKNCOPY copies the log tape to a second tape.

DKNDUMP

While CPCS-I is running, the CPCS-I supervisor can close the log tape and dump all active data on the MDS. The dump tape contains all active strings at the time of the dump. You should not perform this task when CPCS-I application tasks that update the MDS are processing. For information about which tasks update the MDS, see the *CPCS-I Programming Guide*. In addition, DKNDUMP does not let any of these application tasks start.

DKNCOPY

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

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*)}]

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-I 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-I 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	Uncatalogue the data set when the program ends.
CATLG	Catalogue 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	Uncatalogue the data set when the program ends.
CATLG	Catalogue 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.

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-I 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 is either a 5-digit number (YYDDD) or a 7-digit number (YYYYDDD), depending on the settings in copybook DKNEXDT. 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-I 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.

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 an optional parameter. It defaults to the specified tape drive attributes.

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 5 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-I run JCL. Each name can be 1 through 8 characters in length.

Example of Coding the DSAT Parameter DYNOUT

The following examples show you how to code OUTPUT records for the CPCS-I 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-I 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-I run JCL:

```
//CPCSP   OUTPUT COPIES=1,CLASS=E,DEST=PRT1   REGULAR CPCS-I PRINT
//FICHE   OUTPUT COPIES=1,CLASS=F,DEST=FICHE  CPCS-I 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=CPCS-I01,     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.” Use 01E as the device address in the DKNALLO commands. Increase the NUMPTR parameter in the MDEF macro, 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-I internal class specification. This option lets application tasks that use spools have an assigned CPCS-I 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 the *CPCS-I Programming Guide*.

  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.

Master Task Generation

DKNMTASK supervises CPCS-I initialization and shutdown, provides a central location for tables and control blocks, and attaches or detaches executive tasks. Most tables and lists that other CPCS-I tasks use are resident within this task. The MDEF macro is the DKNMTASK source code. This macro specifies installation generation parameters. MDEF also supports LU 6.2 applications and executive tasks in the areas of PARMLST parameter declarations, ECB list declarations, and executive task startup and shutdown during CPCS-I initialization.

After you assemble this macro, known as the master task generation, you must link the CPCS-I executable module. The name of the module in the link is used as the *name* field in the CPCS-I JCL statement `EXEC PGM=`*name*. For consistency, IBM recommends the name DKNMTASK. The master task is activated only if a shutdown occurs or if the CPCS-I supervisor wants to stop the MICR tasks or restart a MICR task. For additional information on DKNMTASK, see the *CPCS-I Programming Guide*.

Use the JCL in member GENMTASK of CPCSI.V01R01.SDKNSAM1 to assemble the MDEF macro used to install DKNMTASK for CPCS-I. It places the module in CPCSI.V01R01.SDKNMOD1.

MDEF Parameters

The following table lists the parameters for MDEF.

Name	Operation	Parameters
[symbol]	MDEF	[MAXTERM={**12** | *n*}]
		[,MAXTASK={**50** | *n*}]
		[,MAXBUF={**255** | *n*}]
		[,MAXTG={**50** | *n*}]
		[,MAXDPKT={**35** | *n*}]
		[,MAXRP={**25** | *n*}]
		[,NUMPTR={**14** | *n*}]
		[,SPOOL={**12** | *n*}]
		[,MSTRING={**0** | **1**}]
		[,SCRTY={**NONE** | **RACF** | **USER**}]
		[,CHGPSWD={**0** | **1**}]
		[,SCRCL=**$CPCSI**]
		[,TIMECK={**0** | *nn*}]
		[,BLKSIZE={**1450** | *nnnn*}]
		[,MAXDA={**1** | *n*}]
		[,BLKSEG={**16** | *nnnnn*}]
		[,SEGDEV={**200** | *nnnnn*}]
		[,NDIRBLK={**40** | *nnnnn*}]
		[,MDIREND={**1** | *n*}]
		[,IDIREND={**4** | *n*}]
		[,RDIREND={**5** | *n*}]
		[,DDIREND={**20** | *n*}]
		[,MAXOPEN={**100** | *nnn*}]
		[,DUPLEX={**YES** | **NO**}]
		[,BFRAT={**0** | *n*}]
		[,TPBFNM={**0** | *n*}]
		[,MAXSORT={**NO** | *nn*}]
		[,CLASS={**A** | *n*}]
		[,LNTNAME={*name*}]
		[,LOG={**NO** | **YES**}]
		[,RCVY={**NO** | **YES**}]
		[,LOGGEN={**DKNRCVGN** | *name*}]
		[,LANG={**ENG** | **ENU** | **ENA**}]
		[,DATE={nnnnnnnnnn}]

The following list describes the MDEF parameters:

MAXTERM={12 | *n*}
Specifies the maximum number of display terminals that CPCS-I supports. This number must also include the keyboard-printer terminal if one is used to print supervisor terminal messages under CPCS-I control. This parameter determines the size of the terminal table.

MAXTASK={50 | *n*}
Specifies the maximum number of application tasks that can run concurrently under CPCS-I. MAXTASK should be at least twice the size of MAXTERM to account for automatically started tasks. The MAXTASK total is added to the number of all user executive tasks (that have entries in the CPCS-I BLDL list) to calculate the amount of storage needed for the ECB list.

MAXBUF={255 | *n*}
Specifies the maximum number of communication buffers. You can allocate a maximum of 2500 communication buffers. The default is 255. 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-I task is unable to obtain a communication buffer, it can continue processing, retry the request for a communication buffer, or end.

MAXTG={50 | *n*}
Specifies the maximum number of tracer groups permitted in the system concurrently. When the number reaches the maximum number, no more tracer groups can enter the system until there is an end-cycle operation. The default value for MAXTG is 50. Changing this parameter requires a CPCS-I cold start to initialize the pass-to-pass control file. For more information on CPCS-I cold start, see “STYPE Parameter” on page 1-6.

Note: This parameter has a direct effect on the Enhanced System Manager’s use of MVS Hiperspace storage. The exact space requirement can be calculated using the equations in the *CPCS Enhanced System Manager User’s Guide*.

MAXRP={25 | *n*}
Specifies the number of tracer slips required for the sort pattern with the most rehandle pockets. You can determine this number by totalling the number of slips specified in the R-records of each sort pattern and selecting the maximum. For example, a specific sort pattern has five rehandle pockets, defined as two rehandle pockets on the prime pass and three rehandle pockets on pass 2. Each rehandle pocket receives two tracer slips. MAXRP is 10.

If you specify TRKP (tracers in kill pockets), include additional tracer slips to allow one tracer slip for each prime-pass kill pocket. The default value for MAXRP is 25.

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-I to effectively maintain pass-to-pass controls.
2. Changing this parameter requires a CPCS-I cold start to initialize the pass-to-pass control file. To avoid cold starts you should make this number larger than your current requirements.

MAXDPKT={35 | *n*}
Specifies the maximum number of electronic pockets. CPCS-I can use this value to determine the block size for the pass-to-pass control data set (DKNTG). If you change this parameter, you might need to re-initialize or re-allocate the DKNTG data set. For more information about the DKNTG data set, see page 1-21.

NUMPTR={14 | *n*}
Specifies the maximum number of printers that CPCS-I 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-I spool data sets that CPCS-I 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

0 M-string not required.
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.

SCRTY={NONE | 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.

RACF Specifies the MVS RACF security option. You must supply RACF definitions for this option.

USER Specifies a user-supplied security product.

See “Security Options” on page 4-83 for details concerning the coding of security options and the minimum requirements for the security user exit routine.

Note: Specify the next three keywords (CHGPSWD and SCRCL) only when SCRTY=RACF.

CHGPSWD={0 | 1}
Enables you to change passwords during signon, where

0 Permits you to change passwords during signon.
1 Prohibits you from changing passwords during signon.

SCRCL=$CPCSI
Specifies the RACF class to which you define the CPCS-I resources.

TIMECK={0 | *nn*}
Specifies the status of the auto sign-off function for inactive terminals, where

0 Disables this function.

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).

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-I cold start. For more information on CPCS-I cold start, see "STYPE Parameter" on page 1-6.

BLKSIZE={1450 | *nnnn*}
Specifies block size for the MDS.

- BLKSIZE (standard 50-byte record)

  `50 × 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 32,750. 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) ≤ 32750`, where R indicates the MDS record length.

  BLKSIZE must be a multiple of the record size. You must change the MDSLRECL value to reflect the new MDS record size. MDSLRECL + 12 is the actual block size on the MDS.

MAXDA={1 | *n*}
Specifies the number of direct access devices that the MDS uses. MAXDA × SEGDEV must be less than or equal to 65535.

Note: All DASD that the MDS uses must be the same type.

BLKSEG={16 | *nnnnn*}
Specifies the number of physical blocks for each segment. 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={200 | *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={40 | *nnnnn*}
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.

MDIREND={1 | *n*}
Specifies the relative block address of the last physical record block of the index data set that contains M-strings.

IDIREND={4 | *n*}
Specifies the relative block address of the last physical record block of the index data set that contains I-strings.

RDIREND={5 | *n*}
Specifies the relative block address of the last physical record block of the index data set that contains R-strings.

DDIREND={20 | *n*}
Specifies the relative block address of the last physical record block of the index data set that contains D-strings.

MAXOPEN={100 | *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-I cold start. For more information on CPCS-I cold start, see "STYPE Parameter" on page 1-6.

DUPLEX={YES | NO}
Specifies whether the duplex function is active.

YES Duplex function is active
NO Duplex function is not active.

For more information about the duplex function, see "Using Data-Set Duplexing" on page 2-10.

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.

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-I system. For more information about concurrent sorting, see "Concurrent-Sort Generation" on page 2-36.

CLASS={A | *n*}
Specifies the default CPCS-I 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 the *CPCS-I Programming Guide* for more information on the BLDL table.

LNTNAME={*name***}**
Specifies the name of an LU 6.2 node table module that loads during CPCS-I startup. The LU 6.2 node table module must reside in the CPCS-I 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-I system (see "Installing the Logging Subsystem" on page 2-12).

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 2-12.

NO Does not perform the Logging initialization required for recovery during CPCS-I startup. Set this parameter to NO if you have installed CPCS-I without the Logging function.

YES This Logging initialization required for recovery of strings occurs during CPCS-I 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.

LOGGEN={DKNRCVGN | *name*}
Specifies the name of the Logging generation module. For more information about using Logging with CPCS-I, see "Installing the Logging Subsystem" on page 2-12.

LANG={ENG | ENU | ENA}
Specifies the language used to present system messages, based on whether or not LANG=ENG, ENU, or ENA.

DATE={nnnnnnnnnn}
Specifies the CPCS default date format. The dates on all reports and screens are 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

The default DATE format for a system with LANG=ENU is MM/DD/YY, and the default date format for a system with LANG=ENG is DD/MM/YY. In

either case, the specification of DATE={valid date format} in the MDEF macro overrides the default value.

Concurrent-Sort Generation

When you specify concurrent sorting in the MDEF macro, you must create a SORT load module by assembling the CCSDEF macro.

Important!

If you code any SORTWK data sets in the CPCS-I run JCL, CPCS-I 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*

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.

MICR Task Generation

You perform the CPCS-I 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-I Installation Guide*. You can use the CPCSRDR macro to select keyword parameters.

You must specify the CPCSRDR macro for each 3890 or 3890/XP Series document processor that CPCS-I operates. The keyword parameters for the CPCSRDR macro are defined in "CPCSRDR Macro Parameters" on page 2-38.

Important!

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

  If the additions or alterations do not change the maximum number of stackers, you must either perform a CPCS-I cold start or use the HALTMICR and STRTMICR commands. For more information on CPCS-I cold start, see "STYPE Parameter" on page 1-6. For a description of the HALTMICR and STRTMICR commands, see the *CPCS-I Terminal Operations Guide*.

Use the JCL in member GENMICR of CPCSI.V01R01.SDKNSAM1 to generate DKNMICR for CPCS-I. Figure 2-4 on page 2-38 is an example of the member SAMPMICR, which is used as the SYSIN data set in the ASMMGEN step of the GENMICR job.

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

Figure 2-4. Example of Macros for DKNMICR Generation

CPCSRDR Macro Parameters

Figure 2-5 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 2-5 (Page 1 of 2). 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*	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	**CHANNEL** | LU62 | SIM	**LU62** | SIM	**LU62** | SIM
LUNAME	NA	luname	luname	luname
EXT	NA	YES | **NO**	YES | **NO**	YES | **NO**
FLD1–7	(Page 2-43)	(Page 2-43)	(Page 2-43)	(Page 2-43)
FLD9–15	(Page 2-43)	(Page 2-43)	(Page 2-43)	(Page 2-43)
LDPN	(Page 2-44)	(Page 2-44)	(Page 2-44)	(Page 2-44)

* For LU 6.2-attached document processors, the ddname is NA.

Figure 2-5 (Page 2 of 2). CPCSRDR Parameter Values

Option	3890	3890/XP	3891/XP	3892/XP
CODELINE	**US** | UK	**US** | UK	**US** | UK	**US** | UK

* 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}]
		[,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*}]
		[,CODELINE={**US** | UK}]

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}
This optional operand defines how the document processor attaches to the host.

CHANNEL
Specifies channel-attached document processors.

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

SIM
Specifies host-simulated document processors. The default value is CHANNEL. 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. Host simulation 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 supports codeline data matching in subsequent passes. The codeline data match record, which CPCS-I supplies to the simulated document processor, returns to CPCS-I as document read records. For

more information on the simulator, see the *3890/XP MVS Support and 3890/XP VSE Support Program Reference*.

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-I Programming Guide*. For information on dynamically allocating document processors, see "Dynamically Allocating Data Sets" on page 2-22. You must add entries to the DKNDSAT table and then assemble and link DKNALLO.

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.

For a complete description of the sequence number field, see "MDS Record Description" in the *CPCS-I Programming Guide*.

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	10KB
B	26KB
X	42KB
XP	Between 64KB and 6MB

Note: In all models, you have 3KB 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.

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.

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.

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: You must define field 1 as 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.

Note: You should be careful about using this option. 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 maximum number of acceptable digit errors permitted in the field when codeline data matching occurs. This number indicates the threshold limit, and it must be equal to or less than the number of digits that the first setting (d) specifies, or it must be 15 (whichever is smaller).

If you do not include a parameter for fields 1 through 7 or 9 through 15, the default values in Figure 2-6 are applied:

Figure 2-6. CPCSRDR Macro Default Values for FLD Parameters

Parameter	CODELINE=US
FLD1=	(*d* | **10**,*b* | **5**,*o* | **FSI**,*i* | **0**)
FLD2=	(*d* | **6**,*b* | **3**,*o* | **SI**,*i* | **0**)
FLD3=	(*d* | **14**,*b* | **7**,*o* | **SI**,*i* | **0**)
FLD4=	(*d* | **0**,*b* | **0**,*o* | **blank**,*i* | **0**)
FLD5=	(*d* | **9**,*b* | **5**,*o* | **FDSI**,*i* | **0**)
FLD6=	(*d* | **2**,*b* | **1**,*o* | **SI**,*i* | **0**)
FLD7=	(*d* | **6**,*b* | **3**,*o* | **SI**,*i* | **0**)
FLD9=	(,*b* | **0**,,)
FLD10=	(,*b* | **0**,,)
FLD11=	(,*b* | **0**,,)
FLD12=	(,*b* | **0**,,)
FLD13=	(,*b* | **0**,,)
FLD14=	(,*b* | **0**,,)
FLD15=	(,*b* | **0**,,)

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-I 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: Record 00 is reserved for CPCS-I 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	DPN 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

CODELINE={US | UK}

Specifies the MICR codeline to use and sets the default MICR fields for the selected country codeline.

The CPCSRDR macro uses the following parameters internally to assemble the initialization record (IREC). You should not code these parameters as part of your CPCSRDR macro. The macro gets its input for these parameters from the sort-pattern definition data set (DKNSPDEF). These descriptions are for information purposes only.

RUNPROF — Specifies the name of the run profile that the 3890/XP Series document processors loaded at run initialization time. It can also specify whether to bypass run profile processing.

EIM — Specifies whether you are requesting extended codeline data matching

SCIAUS — Specifies whether the SCI program should process automatically selected documents

RPNO — Specifies the number of documents to select into MPnn-defined rehandle pockets before feeding a merge document

KPNO — Specifies the number of documents to select into MPnn-defined kill pockets before feeding a merge document

VSYMB — Defines the closing symbol for field 3, the account number field. If you code this parameter, field 3 must close with the beginning symbol for field 4. If you omit this parameter, field 3 can close with either the beginning symbol for field 4 or the beginning symbol for field 5.

HOZ — Specifies whether to do high-order zero correction for field 1, the amount field. High-order zero correction ignores digit errors that occur for high-order positions where zeros are expected and assumes the digits to be zero if all digits to the left are errors or zero. You must specify field 1 as a fixed-length field and it must be the correct length.

ITNO — Specifies the condition of the item number feature (INF) during the run. It supports the 3890/XP Series document processors' programmable, item-number endorse feature.

MCF — Specifies the condition of the microfilm feature during the run

ENDRS — Specifies the condition that the endorse feature has during the run. It supports the programmable item number endorse feature of the document processor.

OPT — Specifies the options used for the run. The options are BPO, DBG, MBM, PBF, RPC, and RUS.

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-I 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 in your MDS record a field that contains kill data, you can define one of the optional fields (fields 9 through 12) for this purpose.

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 2-38.

The following is the syntax for the MDX macro:

Name	Operation	Parameters
[symbol]	MDX	BASE={**US** | UK} **MDXF***yy* ,[*m*] ,['*descriptor*'] ,[*displen*] ,[*justification*] ,[*truncation*] ,[*fill character*] ,[*extrlen*]

BASE={US | UK}
This parameter is specified only once for all fields following the MDX keyword. The value UK must be specified for United Kingdom systems. The default value is US.

MDXF*yy*
Specifies the field that you are defining. *yy* can be field numbers 01 through 15, except 08. (You must include the 0.)

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 kills. If you specify this parameter as `'ENDPOINT'`, your reports will have an ENDPOINT label on any information retrieved from field 9. For examples, see “MDX Macro Examples” on page 2-48.

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. This field size 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.

This parameter must be specified on the last field definition.

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. 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.

MDX Macro Examples

Example 1

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 field length and display length remain at the default setting.

```
MDX MDXF01,06,,11,R,L,00,MDXF02,03,2,R,L,AA
```

Example 2

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 7 use the default settings.

```
MDX MDXF09,75,'ENDPOINT',
```

Example 3

CPCS-I is supplied with two mass data set definitions:

- Member MDXENU of CPCSI.V01R01.SDKNSAM2 contains the MDX macro definition for the U.S. standard 50 byte MDS.

- Member MDXENG of CPCSI.V01R01.SDKNSAM2 contains the MDX macro definition for the United Kingdom 85 byte MDS.

MDXENU has the following fields defined by the MDX macro:

```
MDX                              <=All fields take default values
```

Another way of coding the MDX macro for a 50 byte standard MDS is:

```
MDX  BASE=US,                                  <=MDS base
     MDXF01,05,'AMOUNT',10,R,L,00,,            <=Amount field
     MDXF02,03,'P/C',6,R,L,AA,,                <=Process Control field
     MDXF03,07,'ACCOUNT NBR',14,R,L,AA,,       <=Account number field
     MDXF05,04,'R/T',8,R,L,AA,,                <=Routing Transit
     MDXF06,01,'EXTEND P/C',2,R,L,AA,,         <=Extended P/C field
     MDXF07,05,'SERIAL',10,R,L,00,             <=Serial number field
```

MDXENG has the following fields defined by the MDX macro:

```
MDX  BASE=UK,
     MDXF01,06,'AMOUNT',11,R,L,00,,            <=Amount field
     MDXF02,03,'TRAN CODE',2,R,L,AA,,          <=Transaction Code field
     MDXF03,07,'ACCOUNT',08,R,L,AA,,           <=Cheque Account number
     MDXF05,04,'SORT CODE',6,R,L,AA,,          <=Sort Code XX-XXXX
     MDXF06,01,'FLD6     ',0,R,L,AA,,          <=Field 6 (Not displayed)
     MDXF07,06,'AMOUNT DUE',11,R,L,00,,        <=Amount Due
     MDXF12,10,'BAL OPRID',10,R,L,AA,,         <=Balance Operator ID
     MDXF13,04,'END POINT',4,R,L,AA,,          <=Document Endpoint
     MDXF14,01,'DDIM',1,L,R,AA,,               <=OCR Document indicator
     MDXF15,18,'SERIAL/REF',18,L,R,40,         <=Serial/Reference
```

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-I modules and reinstall CPCS-I. This includes a cold start of CPCS-I after the install. For more information about a CPCS-I cold start, see “STYPE Parameter” on page 1-6.

“Expanded MDS Installation Procedure” describes the procedure for generating a new CPCS-I system with an expanded mass data set (MDS). You cannot recover data captured before the install.

Expanded MDS Installation Procedure

CPCS-I is delivered with an 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-I with an MDS record length that is larger than the default size.

Step 1: Expand the Copybooks

If you change the MDS record in any way from the standard size, change the member CMPL0010 so that it points to the library that contains the CPCS-I source code. This member assembles the MDX macro to create copybooks reflecting the expanded fields. Change the MDX macro statements in CMPL0010 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 MDXBUILD of CPCSI.V01R01.SDKNSAM1.

Run job CMPL0010 before you run this job. If necessary, change CPCSI.V01R01 to a qualifier you have selected.

If you have the High Performance Transaction System Application Library Services installed, you must re-assemble the DSS user exit. For additional information, see *IBM ImagePlus High Performance Transaction System Application Library Services Operations Guide*, SC31-2710.

Step 3: Change MDS Block Size

Change the MDS block-size parameter to reflect the new block size plus 12.

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 GENMICR of CPCSI.V01R01.SDKNSAM1.

For information about generating DKNMICR, see “MICR Task Generation” on page 2-37.

Step 5: Generate the DKNMTASK Module

Changing the MDS record length also requires that you generate a new version of the DKNMTASK module. The JCL is in member GENMTASK of CPCSI.V01R01.SDKNSAM1. Set the MDEF BLKSIZE parameter to the new block size.

For information about generating DKNMTASK with the MDEF macro, see “Master Task Generation” on page 2-29.

Modifying the DKNRDX50 Module

The standard MDS record size is 50 bytes. If your standards require an MDS record size other than 50, make the following changes to DKNRDX50:

- Change RDXRECL to the correct record length for your system.
- Change RDXTRUN to define the truncated length that you want, if the MDS record length is longer than 50 bytes.
- Review and change all field definitions according to your requirements.
- Change the RBLKSZ value to the MDS block size that you are using.

Assemble and link-edit the module into your load library.

For more information, see “Overriding the Log Data-Set Definition” on page 2-20.

VTAM Installation Requirements for CPCS-I

To run CPCS-I, 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-I load library and name it DKNVNODE.
2. Define the CPCS-I application and all CPCS-I 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-I as a VTAM application program, use the APPL statement in SYS1.VTAMLST. For user flexibility, the distributed DKNVTASK program takes the CPCS-I application program name from the job-step name of the EXEC statement in the CPCS-I job stream. Therefore, the CPCS-I 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-I application, you can change two SETC variables within the DKNVOPTS copybook to cause the EXEC statement to be ignored. Variable &ACBLAB can be changed from `X'0'` to `SETC VCVTASK`. This causes the VTAM products to use the name specified in the &ACBNAME variable as the application name.

As distributed, variable &ACBNAME contains the application name "CPCS". You can change this name by changing the 1- to 8-character alphanumeric character name, and it must be the same as the name specified in the APPL statement within SYS1.VTAMLST.

DKNVTASK Installation

The DKNVTASK copybook includes a number of global variables that let you:

- Omit the abend processor (CSECT VPSTA000)
- Omit the issuing of some or all write-to-operators (WTOs)
- Omit the expansion of the CPCS-I, 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-I task or use the job-step name from the CPCS-I job stream
- Set session protocol definitions for the session BIND
- Assign program function keys to either CPCS-I commands or tasks or both.

If you change any value in DKNVOPTS, you must reassemble DKNVTASK in order to incorporate the changes into the load module. Figure 2-7 on page 2-52 lists the DKNVOPTS variables, their initial settings, and the functions they perform.

Figure 2-7 (Page 1 of 2). 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.
&PCPCS	SETB 0	Expand CPCS-I 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-I stops with an abend 119 (see &STAABRT).
&VCIORTC	SETA 3	Session I/O error retry counter.
&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).
&PFKYMEM	SETC “VTSKPFKS”	SYSTPROF PF key table.

Figure 2-7 (Page 2 of 2). 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-I APPLID to be set:		
&ACBNAME	SETC 'CPCSI'	APPLID name if &ACBLAB is not set to 0.
&ACBLAB	SETC '0'	If this variable is set to 0, the CPCS-I APPLID is the job-step name from the CPCS-I EXEC statement. If this variable is **not** set to a 0, the CPCS-I 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'.

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-I as *auxiliary* terminals. If you specify a terminal as a primary CPCS-I terminal, it automatically logs on CPCS-I during initialization, and CPCS-I writes its logo to the screen. If you specify a terminal as an auxiliary CPCS-I terminal, it requires the standard VTAM command LOGON to log on CPCS-I. The LOGON command causes CPCS-I to write its logo to the screen and to identify the terminal as a standard CPCS-I terminal.

If you do not specify any primary CPCS-I terminals in the node-name table, all terminals that CPCS-I uses must log on the VTAM products as auxiliary terminals. In this case, terminals are not automatically assigned to CPCS-I during CPCS-I initialization. CPCS-I does not write logos to any terminal, and any terminal in the VTAM network can log on CPCS-I. If you do not specify any terminals as auxiliary CPCS-I terminals in the node-name table, any node in the VTAM network can log on CPCS-I and function as a CPCS-I 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-I and a later attempt is made to log it back on CPCS-I 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”). 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-I 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 2-53. 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={**CPCSI** | *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-I terminal and is put into session with CPCS-I when CPCS-I starts.

NO Specifies that the node-name entry is in the auxiliary node-name table and must be put into session with CPCS-I 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-I 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={CPCSI | *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-I terminals and which are not. You generate the node-name table by coding VNODE macro statements as described in "VNODE Macro Description" on page 2-54. To associate terminals with CPCS-I, you must specify (in CPCS-I VNODE macro statements) whether the node names on the LU and LOCAL statements are primary or auxiliary CPCS-I terminals, as follows:

Primary Terminals: Primary terminals log on CPCS-I automatically during CPCS-I initialization. If a primary terminal is powered off, initialization continues. However, if that terminal becomes available, then it automatically logs on CPCS-I. When a primary terminal logs on CPCS-I, the CPCS-I logo appears, which indicates that entering the SGON command will start CPCS-I processing. For information about specifying a primary terminal, see "DKNVTASK Node-Name Table Description" on page 2-53.

Auxiliary Terminals: Unlike CPCS-I primary terminals, CPCS-I auxiliary terminals do not automatically log on CPCS-I. An auxiliary CPCS-I terminal requires the VTAM LOGON request to log on CPCS-I.

You must specify auxiliary terminals only if you want to limit which terminals within the VTAM network can or cannot log on the CPCS-I application.

Note: If you do not specify an auxiliary terminal, any terminal within the VTAM network can use the CPCS-I application. If you specify at least one terminal as a CPCS-I auxiliary terminal, only those terminals included in the node-name table as primary or auxiliary terminals, or both, can use the CPCS-I application.

After an auxiliary CPCS-I terminal or a primary CPCS-I terminal (one that has been logged off CPCS-I) logs on CPCS-I, the CPCS-I logo appears and you can use the SGON command to log on CPCS-I. For information on specifying an auxiliary terminal, see “DKNVTASK Node-Name Table Description” on page 2-53.

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-I Programming Guide*.

VNODE Macro Error Messages

See the *CPCS-I 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-I as a VTAM product and to define CPCS-I terminals to VTAM. VTAM configuration statements must define all local terminals (non-SNA and SNA) and the CPCS-I 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-I 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-I 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-I

Terminal or printer device node names that you specify as primary CPCS-I terminals in the node-name table automatically log on CPCS-I when CPCS-I is started. During initialization, there are no password checks and all terminals that you specified as primary CPCS-I terminals are reserved for the CPCS-I application.

If a terminal is powered off at the time of initialization, it is assigned to CPCS-I and becomes active when the device is powered on. If a primary CPCS-I terminal is not online to the VTAM products when CPCS-I 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-I terminal, the terminal requires a VTAM LOGON command to log on CPCS-I again. If a CPCS-I DHCPY command releases a printer device to the VTAM products, the printer requires a CPCS-I AHCPY command to log on CPCS-I again.

Terminal or printer devices that you specify as auxiliary CPCS-I terminals in the node-name table require a manual operation to log on CPCS-I. For terminal

displays, you can log on CPCS-I with a VTAM LOGON request from the display device. For terminal printers, you can log on CPCS-I with an AHCPY command.

If you specify any devices in the node-name table as CPCS-I auxiliary terminals, only those devices whose node names you specify in the node-name table as either primary or auxiliary CPCS-I terminals can log on CPCS-I. If you do not specify any devices as auxiliary CPCS-I terminals, any terminal in the complete VTAM network can log on CPCS-I.

The optional user-specified password parameter in the node-name table is a requirement only for a VTAM LOGON request. For primary CPCS-I terminals, it is a requirement only if the primary terminal logs off and then logs back on CPCS-I with a VTAM LOGON request. When a printer logs on CPCS-I with an 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-I terminal logs on CPCS-I.

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-I rejects any node containing the optional password.

Releasing a CPCS-I Terminal

Terminals that CPCS-I uses are available to other VTAM products at any time. You must first use the SGOF command to log off CPCS-I. When the CPCS-I 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-I commands, see the *CPCS-I 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-I 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-I writes commands and messages on a screen from the top to the bottom. The screen wraps to the first line and CPCS-I 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-I writes all supervisor terminal messages in white for color terminals and in high-intensity for non-color terminals.

Program Function (PF) Keys

You can use the global variable &PFKYMEM to select a PF key table from the System Profile Data Set (ddname SYSTPROF). If you use this option and press a PF key assigned to a CPCS-I 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-I 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-I 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-I. 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.

Chapter 3. System and Application Profiles

Profiles provide a simple and efficient means for customizing CPCS-I. They consist of members in a PDS, each of which contains one or more keywords. The values assigned to these keywords control how CPCS-I behaves.

System profiles, stored in the SYSTPROF data set, affect the CPCS-I system as a whole. They are loaded once whenever CPCS-I is started and are used thereafter for the remainder of the CPCS-I run.

System Profile Data Set

The System Profile data set, CPCSI.V01R01.SYSTPROF, is a data set that contains information used to define parameters used by CPCS-I. These parameters may be changed by editing the appropriate member within the data set.

Listed within this systems profile section are the following profile members:

- EXIT
- VTASK

DKNPEXIT—User Exit 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 for 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-name=user-exit-name,*option1*,*option2*,...

Where:

exit-point-name
: The exit point name

user-exit-name
: The user exit name to be called at this exit point

*option***n**
: 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.

VTASK PF Key Profile

VTASK uses this member to define the command to be displayed for PF keys. The default *membername* is defined in DKNVOPTS for the variable &PFKYMEM. The default shipped for CPCS-I is VTSKPFKS. Additional members may be defined (see Appendix A of the *CPCS-I Terminal Operations Guide*, "Supervisor Commands for VTASK," for use of the PFKEYLOAD command).

The records in this member have the following format:

Position	Description
1–2	Constant PF
3–4	Numbers 01–24
5	Blank
6–65	Command to be presented
66–71	Blank
72–80	User sequence number

Application Profiles

The following application profiles are discussed in detail:

- DEFT
- DFTP
- HCDM
- MRGE
- OLMS
- RMIT

DEFT Profile Record Formats

DEFT profiles are user-created, partitioned data-set members that are used to control selected internal functions of the DEFT input program. Each profile member contains 13 records. All records must be present even if they are not used.

Important!

Flags in the profile are not checked for validity. Error messages are given for invalid flag values. If the flags are values other than Y or N, then the default values are assigned. See individual flags for the default values.

Record 01: DFTI uses this record to control specific functions and assign values in the creation of strings.

Position	Description
1–4	Constant PF01. Identifies the record.
5–6	Not used – blank fill

7–38 Description of the type or source of the data being processed. Maximum of 32 characters. This description is printed on the DEFT completion report.

39–46 Bank control file key for the processing bank (BANK000)

47 MOLRI edit flag; default=N.

- **Y** DFTI passes input-item records to the host user-pocket-select routine before writing them to the MDS.
- **N** DFTI writes input-item records directly to the MDS and bypasses the host user-edit routine.

48 Electronic only flag; default=N.

- **Y** Indicates that there are no paper documents to match the electronic input data for CDMP processing
- **N** Indicates that there are paper documents to match the electronic data.

49 Reserved

50–51 Not used – blank fill

52–54 Not used – blank fill

55–62 Member name of sort-pattern definition for DFTI to use for processing input data (SPTYP*xxx*)

63–70 Name of sort-pattern definition file (default name is DKNSPDEF)

71 Item-sequence-number option flag; default=Y.

- **Y** DFTI assigns new item-sequence numbers to items processed.
- **N** DFTI writes the items to the MDS with the item numbers as received.

72 String type. This value is used as the string type for the data captured if the string type was not received from the input screen (manual start) or the header record of the input file or the string type is not in the 05 record of the profile as part of an assigned string name.

73–80 Reserved

Record 03: This record is used to define optional user exits that DFTI calls and additional control information.

Position	Description
1–4	Constant PF03. Identifies the record.
5–12	Program name of DFTI optional user-exit 1
13–20	Program name of DFTI optional user-exit 2
21–52	Reserved

Positions 53 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 53 - 59 is N.

53	Bypass sub-total voucher
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 the automatic assignment of string names.
22–80	Reserved

Record 07: Records 07 and 08 are used to control creation of a tracer record and provide the data for fields 1 through 7 of the record, if created.

Position	Description
1–4	Constant PF07. Identifies the record.
5	Generate tracer flag; default=N. **Y** DFTI generates a tracer document, including any data specified for fields 1 through 7. **N** A tracer document is not generated.
6–15	Reserved
16–30	Data for tracer-record field 1
31–45	Data for tracer-record field 2
46–60	Data for tracer-record field 3
61–80	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; default=N.
	Y DFTI generates a block document, including any data specified for fields 1 through 7.
	N A block document is not generated.
6–15	Reserved
16–30	Data for block record field 1
31–45	Data for block record field 2
46–60	Data for block record field 3
61–80	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; default=N.
	Y DFTI generates a batch document, including any data specified for fields 1 through 7.
	N A batch document is not generated.
6–15	Reserved
16–30	Data for batch-record field 1
31–45	Data for batch-record field 2
46–60	Data for batch-record field 3
61–80	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: DFTI 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 30: DFTI does not use this record, but you can use it to pass data to the user-exit programs.

Position	Description
1–4	Constant PF30. Identifies the record.
5–58	Miscellaneous flags DPF30_MSF01 through DPF30_MSF54 are for your convenience or future requirements. A user-exit program can reference them, if desired.
59–80	Available

Record 32: Records 32 and 34 give you the capability of turning on ZE flags in the string header record 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 flag byte as received.

Positions 5–12, 16–21, 26–38, and 42–49 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	ZE-BALANCED
6	ZE-DISTRIBUTED
7	ZE-MERGED
8	ZE-KILL-STATUS
9	ZE-KILL-LISTED
10	ZE-USER-RELSD
11	ZE-TRANSF-INPUT
12	ZE-DB-CR-ORDER
13	ZE-FILLER
14	ZE-ADDL-FLAGS
15	ZE-MASS-FLAGS
16	ZE-IMAGE-ENTRY
17	ZE-KEY-ENTRY-DONE
18	ZE-POWER-ENCODE-DONE
19	ZE-IA-REPAIR-DONE
20	ZE-MERGE-FOR-BALANCE
21	ZE-IA-BALANCE-DONE
22	ZE-TYPE-INPUT-FLAG
23	ZE-SIG-VERIFY-FLAG

24	ZE-PRIORITY-CODE
25	ZE-STRING-TYPE-CODE
26	ZE-OLRR-FLAG
27	ZE-REJECT-DIST-FLAG
28	ZE-KILL-PROGRESS-STATUS
29	ZE-CIMS-HSRR-FLAG
30	ZE-SOURCE-LOOKUP
31	ZE-MICR-PROCESSED
32	ZE-REGION-DISPLAY
33	ZE-KILL-SUBSET
34	ZE-HCDM-FLAG
35	ZE-ELEC-FLAG
36	ZE-CDMP-FLAG
37	ZE-CDMR-FLAG
38	ZE-OLMS-FLAG
39	ZE-FILLER
40	ZE-FILLER
41	ZE-FILLER
42–49	ZE-FUTURE-FLAG-9
50–57	ZE-FUTURE-FLAG-10
58–65	ZE-FUTURE-FLAG-11
66–80	Reserved

Record 34

Position	Description
1–4	Constant PF34. Identifies the record.
5–12	ZE-DIST-STATUS – bit representation. Each byte represents 1 bit; 0 and 1 are the only valid characters.
13–15	ZE-BANK-NUMBER (processing bank)
16–23	ZE-ENDPOINT-ID
24	ZE-TYPE-OF-WORK
25–27	ZE-NEXT-SUBSTRING
28	ZE-KILL-OPTION
29	ZE-DIST
30–33	ZE-COMPL-KB-NUM
34	ZE-DISTR
35–80	Reserved

DFTP Profile Records

This section describes the organization of the profile records contained in DKNPDFTP, a sample profile member. This member is loaded to the CPCS-I application profile data set by the DKNGAPPL job.

This profile lets the user define optional user exits for the DFTP task. Two record types are currently in this profile:

Control Record

Record Type	Position	Length	Description
Exit 01 Record	1	13	The constant “USER_EXIT_01=”
	14	8	The name of user exit 01
	22	59	Comment area
Comment Records	1	1	Must contain an asterisk
	2	79	Comment area

Record Supported: User Exit 01
Code a record starting in position 1 that reads:

```
USER_EXIT_01=xxxxxxxx
```

Where:

xxxxxxxx Names user exit 01. *xxxxxxxx* must start in position 14.

User Exit 01 is called at the end of DFTP processing. It is passed statistics on the various types of records that DFTP found on the DEFT input control file, and it can optionally react by displaying a message. For information on coding this user exit, see *CPCS-I Programming Reference*.

If DFTP cannot find a USER_EXIT_01 card in DKNPDFTP, or if it cannot find DKNPDFTP in the DKNAPPL application profile data set, it makes no User Exit 01 calls.

HCDM Profile Records

This section describes the organization of the profile records contained in the DKNAPPL data set. You can use a text editor to load the profile records.

DKNAPPL is a partitioned data set that contains members consisting of 80-byte records. Member names are in the format HCM*xxxxx*, where *xxxxx* is a 5-character field that identifies the profile member when calling DKNAPPL. The data set is specified in the CPCS-I startup JCL.

Control Record

Position	Length	Description
1	1	C–Control-record identifier

Position	Length	Description
2	1	Reconciliation-file indicator **1** Create reconciliation file **0 or Blank** No reconciliation file The reconciliation-file indicator and the report indicator cannot both be 0 or blank.
3	1	Reserved
4	17	Reconciliation-file name **Blank or asterisks (*)** Use name of process file. (First string name if multiple strings are concatenated for use as the process file.) `eeeepaabbccddtsss` Name of reconciliation file **'*'** The corresponding character of the first process-file string name that is to be used to build the reconciliation-file name.
21	1	Reserved
22	1	Report indicator **1** Create report **0 or Blank** No report The reconciliation-file indicator and the report indicator cannot both be 0 or blank.
23	1	Item-sequence-number (ISN) match indicator **1** Match on item-sequence number **0 or Blank** No item-sequence-number match
24	1	Codeline data match indicator **1** Match on all fields, digit error threshold is zero; overrides the F record. **0 or Blank** For field match criteria, see “Field Record” on page 3-10. **Note:** It is more efficient to set the codeline data match indicator to 0 or blank and use the F record to specify a match on only those fields defined to the MDS.
25	1	Reserved
26	1	Reserved
27	1	Control-document match indicator **1** Match control documents; only control document types 6, 5, 4, 3, 2, and 1 are considered. **0 or Blank** Do not include control documents in match. **Note:** Tracer documents are never matched.
28	1	Reserved
29	5	Number of items in the master file You do *not* enter a value in this field; DKNHCMI0 calculates this value, based on the MDS string directory item-count field for all master file strings.

Position	Length	Description
34	5	Reserved
39	1	Process-file sort indicator **1** Sort process file **0 or Blank** No process-file sort
40	1	Master-file sort indicator **1** Sort master file **0 or Blank** No master-file sort
41	16	Sort-field sequence **0** Item-sequence-number sequential number (bytes 5 through 12 of item-sequence number) **S** Item-sequence number with sorter number (bytes 3 through 12 of item-sequence number) **1** Amount (DIAMT) **2** Transaction code (DIPCTL) **3** Account number (DIONUS) **4** Optional field 4 (DIOPT1) **5** Sort code (DIABA) **6** Return item (DIRET) **7** Reference/serial number (DIAUX) **8 through F** As defined (DIFLD09 through DIFLD15) Sort fields can be in any sequence, however, you can specify a field only once. For example, if you entered 31, the sorter uses the amount and the account number to sort documents.
57	1	Force CDMP-style matching **1** Force CDMP-style matching **0 or Blank** Do not force CDMP-style matching
58	1	Missing items in place **1** Missing items placed after previously matched Master file item **0 or Blank** Missing items placed at the end of batch
59	1	Electronic-only items in place **1** Missing items placed after previously matched Master file item **0 or Blank** Missing items placed at the end of batch
60	6	Reserved

Field Record

Position	Length	Description
1	1	F–Field-record identifier

Position	Length	Description
2	3	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. **Blank** Do not match on field 01. **000 through 256** Digit-error threshold; match on this field. Valid digit-error threshold values for fields 1 through 7 are 000 through 028. Values for fields 9 through 15 are 000 through 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.
5	3	Field 02 match indicator
8	3	Field 03 match indicator
11	3	Field 04 match indicator
14	3	Field 05 match indicator
17	3	Field 06 match indicator
20	3	Field 07 match indicator
23	3	Field 09 match indicator
26	3	Field 10 match indicator
29	3	Field 11 match indicator
32	3	Field 12 match indicator
35	3	Field 13 match indicator
38	3	Field 14 match indicator
41	3	Field 15 match indicator
44	37	Reserved

The mass-data-set record definitions that the user created sets the field definitions. For more information, see “MDX Macro Examples” on page 2-48.

User Options Record

Position	Length	Description
1	1	U–User-options record identifier
2	75	User area for user-exit HCMX002
76	4	Reserved

OLMS Profile Records

OLMS profiles are defined exactly the same as DFTI profile records. For this information, see “DFTP Profile Records” on page 3-8.

RMIT Profile Records

This section describes the organization of the profile records contained in the DKNPRMIT, a sample profile member. This member is loaded to the CPCS-I application profile data set by the DKNGAPPL job.

This profile lets the user define a switch for the RMIT task. Three record types are currently in this profile.

Figure 3-1. RMIT Task Profile

Record Type	Position	Length	Description
Control Records	1	48	C-control identifier
	49	3	Control setting
Comment Records	1	1	Must contain an asterisk
	2	79	Comment area
End Read Card	1	9	Must contain a literal LAST_CARD
	10	71	Comment area

This profile contains values needed by RMIT for processing. If RMIT cannot find the member DKNPRMIT in DKNAPPL, or if the member contains only comments or is completely empty, RMIT uses all default values.

Record Supported: **Control Record**

Code a record starting in position 1 that reads:

```
SET_LISTED_FLAG_WHEN_RMITING_ELECT_DEFT_STRINGS=xxx
```

Where:

xxx Is YES (RMIT will SET LISTED FLAG when it finishes), or NO (RMIT will not SET LISTED FLAG); this is for the electronic endpoints processed. NO is the default.

xxx must start in position 49.

Record Supported: **END Read Record**

Code a record starting in position 1 that reads:

```
LAST_CARD
```

This card instructs RMIT to immediately close the profile, ignoring all further cards. It can be used to keep RMIT from needlessly reading through comment cards at the end of its profile.

Chapter 4. User Programming Requirements

Overview

This chapter provides general use programming interface and associated guidance information.

This chapter describes the programming you must complete to create a system that meets the document-processing requirements of your financial institution. The topics discussed include MICR and OLRR stacker-select routines, extract programming, and sort program build utility (DKNCSBU). Detailed record descriptions are included for the bank control file (DKNBCF), endpoint name-and-address file (DKNAB), and sort-pattern definition records. The chapter concludes with a section on security options and RACF security installation procedures. These options let you control the tasks that operators can perform and define any restrictions.

Adding an Application Task

To add a function to CPCS-I, you must create application programs or tasks and inform CPCS-I of the load module name, its CPCS-I characteristics, and its resource needs. You can use similar existing CPCS-I tasks as models when you write new CPCS-I user tasks. When you complete the application program, use the CPCS-I program DKNBLDL to define this task to CPCS-I. Normally, the CPCS-I programmer maintains DKNBLDL.

The CPCS-I executive task DKNATASK controls all attaching and detaching of application tasks. Before DKNATASK performs an ATTACH, it compares the new task's resource needs and characteristics, as specified in DKNBLDL, to the current CPCS-I environment and resources available. This comparison ensures that the task runs correctly. DKNATASK obtains the resources that it needs, such as main storage and spool data sets.

When attached, the application task receives the address of its application program task control block (APTCB) and the address of its start parameters from DKNATASK. There is one APTCB for each application task; the APTCB is an integral control block for application task performance in CPCS-I. The address of the APTCB serves as the interface to many of the services that CPCS-I provides. DKNATASK moves the start parameters into the APTCB.

Important!

The application program control blocks (APCBs) in the BLDL table must be in alphabetic sequence by load module name. If they are not in alphabetic sequence, some tasks will not start.

DKNATASK searches the DKNBLDL table serially for a load module name. When DKNATASK finds the name, the search ends. If DKNATASK reaches a name that is beyond the requested module, the search ends and DKNATASK assumes that the module is not part of the CPCS-I system.

After you add a new task, you must stop and restart CPCS-I to make the task eligible for running.

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 alphabetic sequence by program module. Details for the APCB operand follow:

Name	Operation	Operands
SYMBOL	APCB	[NAME=*task name*] [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*}] [MOLRIEX=**blank**|*molri_user_exit_name*}] [MSUPV={**0**|**1**}] [OKONDMP={**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**}] [TIMECK={**0**|**1**}] [USREXIT=*optional user exit name*] [USRPARM={**X**}] [WORK=*amount of work storage required*]

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 name field.

Note: You do not need to use DKN as the first 3 characters of the name.

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

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

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.

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

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-I output class (A through Z, 0 through 9) of any printed output produced by the task. The CLASS parameter of the MDEF macro determines the default value. For more information about the MDEF macro, see "MDEF Parameters" on page 2-30. Using this operand and the printer control capabilities of the DKNFORM task enables you to direct output to specific printers. For example, you can direct the remittance and DCV 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 ddname is assigned as the CPCS-I printer class.

Note: If you make this last digit the same as the SYSOUT class of the CPCS-I JES printer, you can locate the print buffer data set more easily. See "Dynamically Allocating Data Sets" on page 2-22 for details.

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

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

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 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.

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

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-I 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.

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

0 This is not an end-cycle processing task (default value).
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 50.

EXTSK={Y|N}
Specifies whether the associated task is a user executive task. User executive tasks can be manually or automatically started.

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

HIVOL={0|1}
Specifies whether the task requires a large spool data set for printed output.

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).

1 This task requires a large spool data set.

Note: 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.

HLOG={0|1}
Specifies whether DKNLOADR posts the host logon process.

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

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

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 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 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 000.

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

0 The INSC supervisor terminal does not need to start this task (default value).

1 The INSC supervisor terminal must start this task.

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

N This module cannot receive electronic mail (default value). CPCS-I sets the internal program indicator to X'00'

Y This module can receive electronic mail. CPCS-I sets the internal program indicator to X'01'

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

Note: DKNDCVS is an example of a CPCS-I task that requires MAX=1 in its APCB macro in DKNBLDL, because DKNDCVS uses a dynamically allocated temporary 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 1. For more information about the storage manager, see the *CPCS-I Programming Guide*.

MOLRIEX=blank|*molri_user_exit_name***}**
Specifies which CPCS-I applications use the MOLRI user-exit routine. Any application that requires the MOLRI interface can code its own MOLRIEX parameter. The value of MOLRIEX in the BLDL table overrides the value of the bank control file option of MOLRIEX. Multiple applications can specify different user exits because of this BLDL table capability. The default value is blank, which indicates that there is no user exit.

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

0 The MICR supervisor terminal does not need to start this task (default value).

1 The MICR supervisor terminal must start this task.

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

0 This task *cannot* run while the DKNDUMP task is running (default value).

1 This task *can* run while the DKNDUMP task is running. Do **not** specify `OKONDMP=1` for any task that updates the MDS; this can cause the output from the DKNDUMP program not to be valid, creating a "no backup" CPCS-I environment. The standard CPCS-I DKNBLDL program has the correct specifications for this operand.

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. 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. The default value is 0.

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

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

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-I tasks fall into this category.

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'). Tasks such as DKNOLMS, which is typically grouped at the *individual* level (in other words, one WRB for each UoW), may process more than one UoW using only one DKNOLMS execution.

Specifying SMMULT=1 enables Enhanced System Manager 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 Enhanced System Manager.

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

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

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

1 This task *can* be tracked but *cannot* be started by Enhanced 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.

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

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

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

1 This task requires special testing (for example, DKNCOMP, DKNDUMP, and DKNECYC).

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

0 The SYST supervisor terminal does not need to start this task (default value).

1 The SYST system supervisor terminal must start this task.

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

0 This task is available for automatic logoff if it is inactive for a period of time that exceeds the period specified in the MDEF macro (default value).

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 MDEF macro.

USREXIT=*optional user exit name*
Specifies to CPCS-I 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.

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.

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.

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

WORK=*amount of work storage required*

Specifies the amount of user work storage that the task requires (applies to assembler programs only, usually re-entrant). For MDS records larger than 50 bytes, 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. You should specify the correct number of strings for INSTRG and OUTSTRG.

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. The DEMO task can run concurrently with DKNDUMP.

User Executive Tasks

A CPCS-I user executive task is similar to any other CPCS-I fixed executive task, such as DKNATASK, except for the following differences:

- Although CPCS-I defines the external interfaces, the user is responsible for the coding and maintenance of the task.
- The user must define this task in the BLDL table with the `EXTSK=Y` and `MAX=1` parameters.
- Although the CPCS-I master task (DKNMTASK) starts the CPCS-I fixed executive tasks, DKNATASK starts the user executive tasks.
- Abnormal ending of the user executive task does not cause CPCS-I to end abnormally.

- You can automatically start the user executive tasks by specifying `AUTO=1` in the APCB macro of the BLDL table. You can stop and reactivate user executive tasks, unlike fixed executive tasks, while CPCS-I is still active.

The following things distinguish a user executive task from a CPCS-I application task:

- Application tasks can be completely written in COBOL, but at least part of the user executive task must be written in assembler language. (See "Responsibilities of User Executive Tasks" below.)
- Although you can define several copies of application tasks in the BLDL table with the `MAX=nn` operand, you can define only one copy of the user executive task.
- The CPCS-I operator can type a 4-letter task identification to start an application task that starts, stops, and communicates with a user executive task. For example, if ELRLLOG is a user executive task, defined with EXTSK=Y, and ELRHLOG is the application task that communicates with ELRLLOG, the CPCS-I operator can type `HLOG INIT` to start ELRLLOG, if ELRHLOG is written to start ELRLLOG on the `HLOG INIT` command.

  A user executive task may also be started by a CPCS-I operator by typing its own 4-letter task identification. For example, if DKNEXT1 of a user executive task is not active, a CPCS-I operator may type `EXT1` to start or re-start it. Once active the user executive task may process commands, if it is written to scan for communication buffers with its destination code.

  These commands, in the form of communication buffers, may be sent by another application task or by a CPCS-I operator. For example, the operator may type `EXT1 CMD1` to send a communication buffer to the user executive task DKNEXT1 that contains the CMD1 command.

  Note: CPCS-I always checks to see if the user executive task is active before sending a communication buffer to it. If it is not active it will be re-activated, but the command will be sent in the TCBSPARM field of the APTCB instead of a separate communication buffer.

See the *CPCS-I Programming Reference* for more information about writing user executive tasks.

Responsibilities of User Executive Tasks

The user executive tasks have the following responsibilities:

- After receiving control from CPCS-I, the task must search through the parameter list extension to find its extension block, communication buffer code, and communications ECB address. If it cannot find its name in one of the parameter list extension fields, the user executive task should end.

  Note: The address of the first parameter list extension is in the field ETSKEXTA. XLSKLEN defines the length of the parameter list extension. The task should look for its task name in the field XSKNAM. XL8'FFFFFFFFFFFFFFFF' indicates the end of the parameter list extension.
- The task receives communication from other CPCS-I tasks by issuing a WAIT in the ECB field XTSKECBC.
- The task owner must define the 56-byte communication buffer so that other CPCS-I tasks can communicate with the task.
- The user executive task must process and free, on a timely basis, all communications buffers addressed to it. Otherwise, CPCS-I might run out of space in its communication buffer pool.

A sample user executive task, DKNEXT1, may be found in CPCSI.V01R01.SDKNSAM2. This program demonstrates the proper techniques for initializing user executive tasks, creating a communication buffer translate table, and receiving and processing inbound communication buffers. It also contains extensive comments on the environment that user executive tasks run in. You may use DKNEXT1 as a guide to creating your own user executive tasks.

Communication with Application Tasks

Application tasks should use the following procedure to communicate with the user executive tasks:

- Call DKNGETXI to determine and save the communications buffer code. For further details, see the DKNGETXI module description in the *CPCS-I Programming Guide*.
- Set up the communications buffer as defined by *user executive task* documentation.
- Issue a call to DKNGETB2 to send the communications buffer to the user executive task.
- Process all the error codes correctly. If the BLDL table does not define the user executive tasks, errors will occur.

DKNCSBU—Sort Program Build Utility

The DKNCSBU task enables a workstation-based application to load the sorter version of the sort program from CPCS-I and run it on a workstation. The sort program includes the initialization data (IREC), the user stacker-select routine, and the user table.

Task Initiation

To begin, DKNCSBU should have an APCB entry added to DKNBLDL in the following format:

```
APCB NAME=DKNCSBU,AUTO=1
```

The interface rules that the CSBU invoker has to follow for task initiation are:

1. Invoker has to post DKNATASK to attach DKNCSBU 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 that is coming from the caller with the sort program request information
   - Third fullword—address of the DKNSBUB control block
2. It is the user's responsibility to develop an Advanced-Program-to-Program Communication (APPC) program that interfaces with CSBU and the Sort Control Facility (SCF).

   The calling program is also responsible for preventing CPCS-I from shutting down if an abend occurs.

User Data Preparation for Sort Programs

This section outlines the preparation 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 000 and 999 that the CPCS-I installation assigns). Bank number 000 is defined as the system default bank number. The system default bank number should contain all default options defined by the financial institution. Bank number 000 is normally defined as the default bank number.
- Name and address information that the report tasks use.
- Special kill-list instructions printed on each kill list.
- Customized CPCS-I option data.

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.

Detailed Bank Control Input for DKNBCFLD

The input data set to the DKNBCFLD program (ddname BCFCARDS) is device-independent, and it must consist of 80-character, logical records. The system default processing financial institution requires 44 input records. The bank default and any other processing financial institution requires at least two input records (Record 001 and Record 002). This establishes the bank number, name, and address information. The following fields are common to each record.

Position	Description
1	Constant literal – 0
2–4	Three-digit bank number, 000 through 999 (000 for the CPCS-I system default bank)
6	Constant literal – 0
5–7	Record-type sequence number, 001 through 0998
8	*Blank*

Records 001–010 and 020–099 are descriptions of the data requirements for all processing institutions. Records 011–019 are input records for the system default bank.

Record 001

Position	Description
9–34	Bank name-and-address, first line; 26 positions
35–38	*Blanks*
39–64	Bank name-and-address, second line; 26 positions
65	*Blank*
66–73	Sort code, 8 positions
74–80	*Blanks*

Record 002

Position	Description
9–34	Bank name-and-address, third line; 26 positions
35–38	*Blanks*
39–64	Bank name-and-address, line 4; 26 positions
65–80	*Blanks*

Record 003

Position	Description
9–52	Bank-constant information, line 1; 44 positions
53–80	*Blanks*

Record 004

Position	Description
9–52	Bank-constant information, line 2; 44 positions
53–80	*Blanks*

Record 005

Position	Description
9–52	Bank-constant information, line 3; 44 positions
53–80	*Blanks*

Record 006

Position	Description
9–52	Bank-constant information, line 4; 44 positions
53–80	*Blanks*

Record 007

Position	Description
9–52	Bank-constant information, line 5; 44 positions
53	*Blank*
54–61	User field 1, 8 positions
62	*Blank*
63–70	User field 1, 8 positions
71	*Blank*
72	User flag 1, 1 position
73	User flag 2, 1 position
74	User flag 3, 1 position
75	User flag 4, 1 position
76–80	*Blanks*

Record 008

Position	Description
9	*Blank*
10	Kill-pocket tracers flag **Y** Yes **N** No **Blank** No
11	*Blank*
12	Block sequence flag **B** Place block slips before documents. **A** Place block slips after documents. **Blank** Use MDEF setting.
13	Batch sequence flag **B** Place batch slips before documents. **A** Place batch slips after documents. **Blank** Use MDEF setting.
14–15	Number of MICR exit 3 buffer codeline slots. The value must be numeric and greater than 16. The MICR exit 3 receives a default of 16 data matching buffer slots.
16–32	*Blanks*
33	Microfilm option **B** Both (front and back) **F** Front of check **Blank** Both (front and back)
34	*Blank*
35	PC dash-transmission option **Y** Yes **N** No

Blank No

36 Account dash-transmission option

Y Yes
N No
Blank No

37 Auxiliary dash-transmission option

Y Yes
N No
Blank No

38 *Blank*

39 Reserved

40 *Blank*

41 Divider spray option

A Place dividers after transit bundles.
B Place dividers before transit bundles.
Blank Use default bank definition

42 *Blank*

43 Final merge flag

0 Final merge is optional.
1 Final merge is not allowed.
2 Final merge is required.
Blank Final merge is optional.

44–46 *Blanks*

47 Reserved

48 *Blank*

49 CDMP option field.

P Pull resync bundles that do not match during CDMP.
R Do not pull unmatched bundles during CDMP.
Blank Do not pull unmatched bundles during CDMP.

50–59 *Blank*

60 99-M-string processing flag. This flag specifies which strings to extract and also which strings to delete or keep. Valid values are:

0 Transfer 00-M-strings for subset 000. This is the default.

1 Transfer 00-M-strings for all subsets

2 Transfer 99-M-strings for subset 000 and keep the 00-M-strings that correspond to the transferred 99-M-string.

3 Transfer 99-M-strings for all subsets and keep the 00-M-strings that correspond to the transferred 99-M-string.

4 Transfer the 99-M-strings for subset 000 and delete the 00-M-strings that correspond to the transferred 99-M-string.

5 Transfer 99-M-strings for all subsets and delete the 00-M-strings that correspond to the transferred 99-M-string.

61–68 In-work, quick-kill endpoint ID 1. Items that sort to a pocket for this endpoint do not receive dividers that separate bundles and do not produce kill lists. They are considered on-us, already killed items.

69–76 In-work, quick-kill endpoint ID 2. Items that sort to a pocket for this endpoint do not receive dividers that separate bundles and do not produce kill lists. They are considered on-us, already killed items.

Usage Notes:

1. Endpoint values in positions 63 through 70 are considered quick-kill and can be transferred by DKNMCRE.
2. Endpoints in positions 71 through 80 are considered quick-kill, but require a string-listed flag on all output D-strings before DKNMCRE processing.
3. Quick-kill endpoints are identified by entering the significant digits first, followed by Xs to indicate *don't care* digits.

   For example, an endpoint `8XXXXXXX` represents all endpoints considered on-us, quick-killed that begin with 8 in the range `80000000` through `89999999`. An endpoint of `900000XXX` represents endpoints considered on-us, quick-killed in the range `90000000` through `90000999`.

77–80 *Blanks*

Record 009

Position	Description
9–18	Tracer-document name field
19–28	Divider-document name field
29–38	Block-document name field
39–48	Batch-document name field
49–58	Sub-batch-document name field
59–68	Assist-document name field
69–78	STV control-document name field

Record 010

Position	Description
9–18	HSRR sub-tracer document name
19–28	CDMR sub-tracer document name
29	*Blank*
30	ICRE totals displayed **0** No totals printed **1** Print totals on ICRE DIAG Report **Blank** No totals printed
31–80	*Blanks*

Note: The following input records 011–019 are for the system default bank.

Record 011

Position	Description
9–16	MICR exit 1 routine name
17	*Blank*
18–25	MICR exit 4 routine name
26	*Blank*
27–34	MICR exit 5 and CSBU exit 1 routine name

35	*Blank*
36–43	MICR exit 6 and CSBU exit 2 routine name
44–80	*Blanks*

Record 012

Position	Description
9–16	VTASK exit 1 (scroll buffers) routine name
17	*Blank*
18–25	VTASK exit 2 (scroll initialization) routine name
26	*Blank*
27–34	VTASK exit 3 (PCB release 1) routine name
35	*Blank*
36–43	VTASK exit 4 (PCB release 2) routine name
44	*Blank*
45–52	VTASK exit 5 (terminal logon)routine name
53–80	*Blanks*

Record 013

Position	Description
9–16	SECR exit routine name
17	*Blank*
18–25	CYCL exit routine name
26	*Blank*
27–34	MTASK exit 1 routine name
35	*Blank*
36–43	MTASK exit 2 routine name
44–80	*Blanks*

Record 014

Position	Description
9	MCRE transfers rehandle D-strings
	Y Yes
	N No
	Blank No
10	MCRE transfers subpass I-strings
	Y Yes
	N No
	Blank No
11	MCRE transfers subpass M-strings
	Y Yes
	N No
	Blank No
12–80	*Blanks*

Record 015 – 019

Reserved

Record 020

Usage Notes:

These notes also apply to records 21 through 28. For additional information, see Figure 4-1 on page 4-25 and Figure 4-2 on page 4-25.

1. If dashes occur on the physical document, they should appear in the identifier.
2. 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
09	Tracer MICR-field location
10–25	Tracer MICR-field identification for the primary field
26–41	Additional tracer MICR-field identification
42	*Blank*
43	Alternate tracer MICR-field location
44–59	Alternate tracer MICR-field identification
60–75	Additional alternate tracer MICR-field identification
76–80	*Blanks*

Record 021

Position	Description
09	HSRR sub-tracer MICR-field location
10–25	HSRR sub-tracer MICR-field identification
26–41	Additional HSRR sub-tracer MICR-field identification
42	*Blank*
43	Alternate HSRR sub-tracer MICR-field location
44–59	Alternate HSRR sub-tracer MICR-field identification
60–75	Additional alternate HSRR sub-MICR-field identification
76–80	*Blanks*

Note:

If you do not want to assign a HSRR sub-tracer number when creating a system default bank, place an * in position 11 and spaces in positions 12-27. All other input is ignored.

Record 022

Position	Description
09	CDMR sub-tracer MICR-field location
10–25	CDMR sub-tracer MICR-field identification
26–41	Additional CDMR sub-tracer MICR-field identification
42	*Blank*
43	Alternate CMDR sub-tracer MICR-field location
44–59	Alternate CMDR sub-tracer MICR-field identification
60–75	Additional alternate CMDR sub-tracer MICR-field identification
76–80	*Blanks*

Note: If you do not want to assign a CDMR sub-tracer number when creating a system default bank, place an * in position 11 and spaces in positions 12-27. All other input is ignored.

Record 023

Position	Description
09	Divider MICR-field location
10–25	Divider MICR-field identification
26–41	Additional divider MICR-field identification
42	*Blank*
43	Alternate divider MICR-field location
44–59	Alternate divider MICR-field identification
60–75	Additional alternate divider MICR-field identification
76	*Blank*
77	Divider serial-number field location
78–80	*Blanks*

Record 024

Position	Description
09	Block MICR-field location
10–25	Block MICR-field identification
26–41	Additional block MICR-field identification
42	*Blank*
43	Alternate block MICR-field location
44–59	Alternate block MICR-field identification
60–75	Additional alternate block MICR-field identification
76	*Blank*
77	Block serial field
78–80	*Blanks*

Record 025

Position	Description
09	Batch MICR-field location
10–25	Batch MICR-field identification
26–41	Alternate batch MICR-field location identification
42	*Blank*
43	Alternate batch MICR-field location
44–59	Alternate batch MICR-field identification
60–75	Additional alternate batch MICR-field location identification
76	*Blank*
77	Batch serial field
78–80	*Blanks*

Record 026

Position	Description
09	Sub-batch MICR-field location
10–25	Sub-batch MICR-field identification
26–41	Additional sub-batch MICR-field identification
42	*Blank*
43	Alternate sub-batch MICR-field location
44–59	Alternate sub-batch MICR-field identification
60–75	Additional alternate sub-batch MICR-field identification
76–80	*Blanks*

Note: If you do not want to assign a sub-batch number when creating a system default bank, place an * in position 11 and spaces in positions 12-27. All other input is ignored.

Record 027

Position	Description
09	Assist-document MICR-field location
10–25	Assist-document MICR-field identification
26–41	Additional assist-document MICR-field identification
42	*Blank*
43	Alternate assist-document MICR-field location
44–59	Alternate assist-document MICR-field identification
60–75	Additional alternate assist-document MICR-field identification
76–80	*Blanks*

Note: If you do not want to assign an assist-document number when creating a system default bank, place an * in position 11 and spaces in positions 12-27. All other input is ignored.

Record 028

Position	Description
09	STV control MICR-field location
10–25	STV control MICR-field identification
26–41	Additional STV control MICR-field identification
42	*Blank*
43	Alternate STV control MICR-field location
44–59	Alternate STV control MICR-field identification
60–75	Additional alternate STV control MICR-field identification
76–80	*Blanks*

Note: If you do not want to assign a STV control number when creating a system default bank, place an * in position 11 and spaces in positions 12-27. All other input is ignored.

Records 029 – 039

Reserved

Record 040

Position	Description
9–16	MICR exit 2 routine name
17	*Blank*
18–25	MICR exit 3 routine name
26	*Blank*
27–34	MOLRI exit routine name
35–80	*Blanks*

Records 041 – 042

Cards open

Position	Description
9–80	*Blanks*

Record 043

Position	Description
9–16	Reserved
17	*Blank*
18–25	Reserved
26	*Blank*
27–34	CDIF exit routine name

35	*Blank*
36–43	DFTI exit 1 routine name
44	*Blank*
45–52	DFTI exit 2 routine name
53	*Blank*
54–61	DFTO exit routine name
62–80	*Blanks*

Records 044 – 045

Cards open

Position	Description
9–80	*Blanks*

Record 046

Position	Description
9–16	MDIS exit routine name
17	*Blank*
18–25	FSCN exit routine name
26	*Blank*
27–34	IDCR and ODCR routine names
35	*Blank*
36–43	MRGE exit
44–80	*Blanks*

Records 047

Cards open

Position	Description
9–80	*Blanks*

Record 048

Position	Description
9–16	HCDM exit 1 routine name
17	*Blank*
18–25	HCDM exit 2 routine name
26	*Blank*
27–34	HCDM exit 3 routine name
35	*Blank*
36–43	HCDM exit 4 routine name
44–80	*Blanks*

Records 049

Cards open

Position	Description
9–80	*Blanks*

Record 050

Position	Description
9–16	PRUP exit routine name
17	*Blank*
18–25	PR130 exit routine name
26	*Blank*
27–34	PR145 exit routine name

35	*Blank*
36–43	PRAR exit routine name
44–80	*Blanks*

Records 051

Cards open

Position	Description
9–80	*Blanks*

Records 052

Position	Description
9–16	RCVY exit
17–80	*Blanks*

Records 053

Cards open

Position	Description
9–80	*Blanks*

Records 054–099

Cards open

Position	Description
9–80	*Blanks*

Identifying Control Document Fields

The MICR task and DKNMOLRI (using the DKNTYPER subroutine) both identify control documents 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. Figure 4-1 describes the document field characteristics.

Figure 4-1. Control-Document Field Identifiers

Document Field	CPCS-I Identifier	Maximum Length
Amount	1	12 digits
Transaction code	2	2 digits
In-work (account number)	3	8 digits
Sort code	5	6 digits

Specific fields of control documents are arbitrarily defined and cannot be used for control-document identifiers. Figure 4-2 describes the control-document field definitions.

Figure 4-2. Control-Document Field Definitions

Document Type	Field ID				
	7	5	3	2	1
Tracer			TGID (see note)		
Divider				Kill-bundle identifier	
Block					Control amount
DCV					Control amount
Sub-batch					Control amount
STV					Control amount

Note: Pre-encode the tracer group ID (TGID) in field 3. It consists of a 7-digit number, xxxxyyy, 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). The value for *yyy* must be unique.

The TGID field on the tracer documents cannot contain a dash between the tracer-group number and the slip number.

Changing the Default Bank

The default bank number (normally 001) can be changed using the following steps:

1. Change the value of BCF-DEFAULT-BANK in the DKNCRBCF copybook. The new value must be numeric in the range from 001 through 999.
2. Compile the following modules:
   - DKNBCFLD
   - DKNBCFIO
   - DKNDFTOX
   - DKNMCRE.

   Compile any other user-modified programs that use the BCF-DEFAULT-BANK value.
3. Run a job to re-create the DKNBCF data set. You can use part of the GENBCFAB job stream for this task.
4. Change CPCS-I to use the new DKNBCF data set.

After you complete this procedure, any programs that use the default bank number returned by DKNBCFIO will receive the new bank number.

Adding Banks to the Bank Control File

After you create the bank control file, you can use the following steps to add additional bank records:

1. Create a partitioned data-set member with the same format as the CPCSI.V01R01.SDKNSAM2 member BCFENU.
2. Use the bank control file records to change this member and create your own bank definitions for specific processing requirements.

   Multiple banks can be defined in one partitioned data-set member. For each bank you create, change the bank number in columns 3 through 5 of each record for that bank.
3. Run a job to update the DKNBCF data set.

You can modify the GENBCF job stream as follows:

a. Delete step 1 of the job and change the disposition of the DKNBCF DD statement to DISP=SHR.

b. Change the CARD DD statement to point to the new member and run step 2 again.

This allows the DKNBCFLD program to perform all necessary edits before it creates a bank record in your bank control file.

After the load program has completed successfully, your bank control file will include the new bank data.

Important!

If the options for a bank are being changed, the member BANK*nnn* must be deleted from the bank control file before you run DKNBCFLD with DISP=SHR on the DD statement.

Endpoint Name-and-Address Record Formats

Creating an endpoint name and address record requires one to five 80-byte records, depending on the amount of information to be stored. 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. For a description of on-us kill identification, see Note 3 on page 4-36.

Record 001

Position	Description
1	Record code – 1
2–9	Endpoint number. It must be the same on all five 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) **I** Electronic image.
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)
69–80	*Blank*

Record 002

Position	Description
1	Record code – 2
2–9	The endpoint number. It must be the same on all five 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*

Record 003

Position	Description
1	Record code – 3
2–9	The endpoint number. It must be the same on all five records.
10	*Blank*
11–60	First line of special instruction to or from the user site; 50 positions
61	*Blank*
62	AB kill print indicator: **Y** Print. **N** Do not print.
63	*Blank*
64	DCV summary print indicator: **P** Print. **N** Do not print DCV summary. **F** Print to file. **B** Print and write to file (both).

65	*Blank*
66	DCV reconciliation option: **Y** Include DCVs in reconciliation. **N** Do not include DCVs in reconciliation.
67	*Blank*
68	Type of DCV **C** Credit **D** Debit **M** Mixed
69	Outwork missing/free in detail reports indicator: **Y** Print. **N** Do not print.
70	Inwork internal detail reports indicator **Y** Print. **N** Do not print.
71–80	*Blank*

Record 004

Position	Description
1	Record code – 4
2–9	The endpoint number. It must be the same on all five records.
10	*Blank*
11–60	Second line of special instruction to or from the user site; 50 positions
61	*Blank*
62–69	The ddname used to dynamically allocate the DEFT output file
70–80	*Blank*

Record 005

Position	Description
1	Record code – 5
2–9	The endpoint number. It must be the same on all five 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*

Sort-Pattern-Definition Record Formats

The sort-pattern-definition file contains sort information that MICR and Enhanced System Manager retrieve 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. The P record is used as a delimiter between sorter runs. All records between P records are used for the sorter run defined by the previous P record. An R record, multiple K records, and a B record can follow each P record. 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.

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 re-synchronization, subset-creation, and CDMP 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 sort-pattern definition 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.

Standard 3890 Sort-Pattern Definition Example

```
P1000000S001S1          021     0     20100            20
RP      RP001
R            0101
M              01
K011250000202125000570312500010041250001105125000130600000125098888888 8
K10888888881188888888
P2070000S001S2          001222220     10100             0
RP      RP001
K011250002802125001820312500188041250071305125007550612500756071250078 0
K0812500785093250711710125100 07
P2080000S001S3          001222220     10100             0
RP      RP001
K010000000102000000020300000003040000000405000000050600000006070000000 7
K080000000809000000091000000010110000001 1
```

Expanded Format 3890/XP Series Sort-Pattern Definition Example

```
P1000000S002S1          011     0     10100XF          10
RP      RP002                           XP
R                   01
M                   01
K018888888802888888880388888888040099999905125000020612500057071250001 0
K08125000110912500013100000012 5
P2110000S002S2          001222220     10100             0
RP      RP002
K010000000102000000020300000003040000000405000000050600000006070000000 7
K080000000809000000091000000010110000001 1
```

B — Bank Description Record

This record designates the processing financial institution.

Position	Length	Description
1	1	'B'=record identifier
2	3	Reserved
5	3	Processing bank
		000–999 Site financial institution
8	65	Reserved
73	8	Sequence number (for user maintenance)

E — Document Capture-Type Record

This record designates the type of document to be captured for a prime pass.

Position	Length	Description
1	1	'E'=record identifier
2	3	Reserved

Position	Length	Description
5	3	Type of document being captured **000** Unqualified **001** Pre-qualified (default) **002** Signature-card data **003** Returns **004** Cycle/exceptions **005** Statement sort **006** OCR kill **007** OCR non-kill **008** Wholesale kill **009** Drafts (credit card voucher) **010** Inwork DCV **011** Outwork DCV **012** Inclearing
8	65	Reserved
73	8	Sequence number

H — Document Processor Hardware Options Record

This record designates OCR or MICR options for a sort pass, including power encoder options.

Position	Length	Description
1	1	'H'=record identifier
2	3	Reserved
5	1	OCR read system 1. (Valid only for 3891/XP and 3892/XP Document Processors).* **Blank** Off (default) **Y** On **N** Off
6	1	OCR read system 2 (Valid only for 3891/XP and 3892/XP Document Processors).* **Blank** Off (default) **Y** On **N** Off
7	1	OCR read system 3. Must be off if either read system 1 or read system 2 is on. (Valid on all 3890/XP Series Document Processors.) **Blank** Off (default) **Y** On **N** Off
8	2	Reserved
10	1	MICR read system 1 **Blank** Off (default) **Y** On **N** Off

* For additional information on this option, see the *3890/XP Series Programming Guide*.

Position	Length	Description
11	1	MICR read system 2 (high-read, only on 3891/XP and 3892/XP Document Processors.) **Blank** Off (default) **Y** On **N** Off
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	Power encode path 2 **Blank** On (default) **Y** On **N** Off
18	1	Reserved
19	4	Reserved
23	1	Reserved
24	7	Fields to encode Digits 1 through 7 are valid. **1** Amount field **2** Transaction code field **3** Account-number field **4** Extended control field **5** Sort code 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)

* For additional information on this option, see the *3890/XP Series Programming Guide*.

I — Sorter Image-Capture Options Record

This record identifies the image-capture specifications for each document.

(**Note:** High Performance Transaction System Application Library Services Release 3 supports image capture for the 3890/XP and 3892/XP Document Processors.)

Position	Length	Description
1	1	'I'=record identifier
2	3	Reserved
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/off-line to host processor
		Blank Online (default) **ONLINE** Imaging processor online to host processor (CIMS) **OFFLINE** Imaging processor off-line 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, which is the rightmost byte of the P/C field. The value is packed decimal, so that the 4-character value is loaded into 2 bytes.
53	22	Reserved	
75	8	Sequence number	

J — Reject-Pocket-Number Record

This record designates reject pockets or unencoded pockets for a sort pass.

Position	Length	Description
1	1	'J'=record 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 consolidated reject 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 changes the string name.

K — Kill-Pocket Description Record

This record designates kill pockets for a sort pass.

Position	Length	Description
1	1	'K'=record identifier
2	2	CPCS-I pocket number of kill pocket
4	8	Endpoint ID (numeric) for kill pocket
12		See usage notes
22		See usage notes
32		See usage notes
42		See usage notes
52		See usage notes
62		See usage notes
72	1	Reserved
73	8	Sequence number

Usage Notes:

1. Each fine-sort group endpoint is defined to CPCS-I as a single endpoint (90000*nnn*) for splitting purposes. These items are directed to an on-us kill pocket during prime or rehandle passes.
2. 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.)
3. 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. After you mark it, it is released the next time you run DKNMCRE.

For additional information about using quick-kill endpoints, see page 4-19.

M — Mixed-String Combination-Key Description Record

This record enables mixing of rehandle strings for different sort types. 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'=record 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

This record designates various options for a sort pass.

Important!

If you want divider re-synchronization for a particular pass, you must include this record following the pass descriptor record (P) for the previous and current passes.

The divider re-synchronization will not occur when you run divider re-synchronization with mixed subsequent passes (indicators on and off). The first tracer does not have an indicator on during the prime pass. The divider re-synchronization will occur if you include this record following the subsequent pass descriptor record (P).

Position	Length	Description
1	1	'O'=record identifier
2	3	Reserved
5	1	Divider re-synchronization indicator **Blank** No options (default) **D** Divider data set created on prime pass. Divider re-synchronization on subsequent passes. For additional information, refer to the following position 9.

Position	Length	Description
6	1	Prime pass type **blank** Conventional prime pass **S** Subset prime pass **E** Enhanced prime pass
7	1	Sorter initialization at entry breaks option. This option is valid when prime pass type = E; This option is ignored on simulated captures. It is required for the divider spray between entries when dividers precede transit bundles. This option eliminates small kill bundles at the beginning of the entries. It involves longer pauses between entries with the result of the sorter initialization taking place at entry breaks. **blank** No sorter initialization at entry breaks **Y** Sorter initialization at entry breaks
8	1	Reserved
9	4	Rehandle pocket divider merge count. The number of items between divider merge slips. The value can be 0 through 9999. Used only if the divider re-synchronization indicator equals D.
13	1	Tracer-in-kill-pocket option (this option is valid for enhanced and subset prime passes) **Y** Place a tracer in each kill pocket ahead of each subset on subset prime passes. Place a tracer in each kill pocket ahead of each entry on enhanced prime passes. **N** No tracers in kill pockets (default).
14	1	Reserved
15	1	Enhanced prime/subset pause option **blank** No pause between entries/subsets **N** No pause between entries/subsets **Y** Pause between subsets/subsets. (The MICR status screen shows the statistics corresponding to the last entry/subset at each entry/subset break. This allows you to continue, suspend, or end the capture at this point.)
16	8	MICR user exit 2 routine name.
24	8	MICR user exit 3 routine name. If this is blank, CPCS-I uses the name in the BCF.
32	2	MIPI exit work-field number **Blank** None **09–15** MDS field number to use for MIPI exit work field **Important!** This field must not be defined in the MDS and must be defined to the sorter.

Position	Length	Description
34	1	CDMP processing options **Blank** Default to the BCF setting. **R** Remove paper bundles that do not match electronic data. **P** Process paper bundles that do not match electronic data.
35	1	Extended codeline data match (ECDM) option (valid only on subsequent, CDMP and CDMR passes). The values are: **Y** Enable ECDM **N** Disable ECDM **Blank** Disable ECDM (default value)
36	1	Divider spray option **A** Place dividers after transit bundles **B** Place dividers before transit bundles **Blank** Default to bank control file setting
37	2	Reserved
39	8	MDIS user-exit routine name
47	1	MICR user-exit interface type (this is a temporary option, implemented to facilitate migration to the new MICR user-exit interface). **O** Old MICR user-exit interface Anything other than O (default), new MICR user-exit interface.
48	25	Reserved
73	8	Sequence number

P — Pass-Description Record Standard Format

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'=record identifier
2	7	Source pass/pocket history 'PAABBCC'. For example, PAABBCC=2110000 means 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) **4** Mixed debit/credit (non-POD) Prime pass only; defaults to zero

Position	Length	Description
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.
56	1	Endorsing option
		Blank No endorsing **1** Endorsing position 1 **2** Endorsing position 2 **3** Endorsing position 3
57	10	Available
68	2	Codeline data match, prime-pass option.
		BlankBlank Do not perform CDMP (default). **MP** Do perform CDMP.
70	3	Available
73	8	Sequence number (for user maintenance)

P — Pass-Description Record Expanded Format

This record defines the type of sort pass initialization you can perform.

Position	Length	Description
1	1	'P'=record 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.

Position	Length	Description
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) Endorse position **Blank** No item number; 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*blankblank*
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.
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/endorse position (HSRR pass) **Blank** No item number; not valid on prime pass **1** Position 1 on HSRR **2** Position 2 on HSRR **3** Position 3 on HSRR **4** Position 4 on HSRR (3890/XP only) **5** Position 5 on HSRR (3890/XP only) **6** Position 6 on HSRR (3890/XP only)

Position	Length	Description
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 endorsing 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	2	Codeline data match, prime-pass option **BlankBlank** Do not perform CDMP (default). **MP** Do perform CDMP.
70	3	Available
73	8	Sequence number (for user maintenance)

R — Rehandle-Pocket-Description Record

This record designates rehandle pockets for a sort pass.

Position	Length	Description
1	1	'R'=record identifier
2	2	CPCS-I pocket 1 tracer count*
4	2	CPCS-I pocket 2 tracer count*
6	2	CPCS-I pocket 3 tracer count*
⋮		
70	2	CPCS-I pocket 35 tracer count*
72	1	Reserved

* For kill pockets, the tracer counts must be blank. If you select the tracer-in-kill-pocket option, CPCS-I directs one tracer document to each kill pocket.

Position	Length	Description
73	8	Sequence number

* For kill pockets, the tracer counts must be blank. If you select the tracer-in-kill-pocket option, CPCS-I directs one tracer document to each kill pocket.

Important!

If this is a prime-pass R record, the sum of:

```
(pkt 1 tracer cnt + ... pkt 35 tracer cnt) + 9 + nbr of kill pkts
```

must not be more than the MAXRP parameter in the MDEF 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 MDEF, perform the MICR task generation, and perform a cold start.

BMSG — BEGIN Message-Description Record Expanded Format

This optional record enables operator messages to display on the BEGIN screen after sort-pattern definition editing is complete.

Position	Length	Description
1	4	'BMSG'=record 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 Standard and Expanded Format

You can use this optional record to define the size of a field in the record coming from the document processor.

Warning: Values and definitions specified by usage of FLDnn cards override any options specified in the CPCS-I RDR options for this sort type.

Position	Length	Description
1	5	'FLDnn'=record 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.
14	2	Reserved
16	2	Number of digits—the maximum size of the field. Fields 2 through 7 only; maximum is 28. Field 1 maximum is 16.
18	10	Reserved

Position	Length	Description
28	2	Field digit-error threshold. Values 0 through 15 (fields 1 through 7 and 9 through 15). The default value is what is specified (or defaulted) in the CPCSRDR definition.
30	1	Reserved
31	1	Codeline data match indicator (valid fields 1 through 7 and 9 through 15) **Y** Active **N** Not 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 Series Stacker Control Instructions Reference*.)
36	1	Symbol error correction indicator for field (valid fields 1 through 7 only) **Blank** Not active (default) **N** Not active **Y** Active
37	2	Reserved
39	1	Extended field type definition (valid for fields 8 through 15). The values are: **C** Character field (each byte is 1 character) **D** Digit field (each byte is 2 digits) **Blank** Digit field (default value).
40	3	Reserved
43	12	Descriptive field name (optional)
55	4	Reserved
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 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*blank*'=record identifier
4	5	Reserved
9	8	File name (required)
17	4	Reserved
21	8	Optional date stamp (MM/DD/YY). This is the only acceptable format for date stamp information.
29	2	Reserved
31	5	Optional time stamp (HH.MM)
36	37	Reserved
73	8	Sequence number (for user maintenance)

RP — Run-Profile-Description Record 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*blank*'=record identifier
4	5	Unused
9	8	Run profile name (optional)
17	4	Unused
21	8	Date stamp—YY/MM/DD (optional). This field allows your SCI program to check the date stamp on the DKNTAL label for specification of the correct run profile. See the *CPCS-I Programming Guide* for additional information.
29	2	Unused
31	8	Time stamp—HH.MM.SS (optional). This field allows your SCI program to check the time stamp on the DKNTAL label for specification of the correct run profile. See the *CPCS-I Programming Guide* for additional information.
39	2	Unused
41	5	Document processor model **Blank** Any document processor **XP***blankblankblank* Any XP processor **3892***blank* Any 3892/XP **PE***blankblankblank* Any power encoder **3890***blank* Any 3890 **3890A** 3890 Model A **3890B** 3890 Model B **3890X** 3890 Model X **IMAGE** Any document processor with image capacity
46	28	Unused

Position	Length	Description
74	8	Sequence number (for user maintenance)

TY — Pass-Type Description Record 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*blank*'=record 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 4-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.
7	26	Unused
73	8	Sequence number (for user maintenance)

Generating DCV Power-Encode Strings

DCV summary information is reported as a part of Kill processing or is run separately using DCVS. These online tasks call the DKNDCVX module, which summarizes the selected endpoints and produces the summary report.

You can also use DKNDCVX to produce a special power-encode subset D-string based on current summary requests (strings, endpoints, and endpoint tables). This D-string contains a record of each DCV found among the endpoints being reported.

A single subset D-string is created, based on the information requested on the DCV summary report. The subset D-string contains all information for the endpoints associated with the report request. DKNTGSS assigns a unique subset tracer number to each subset D-string.

You must do the following to generate the DCV power-encode D-string:

- In the DKNAB file, designate the outgoing kill-pocket endpoint for DCV power encoding by changing the AB-DCVSUM-PRINT-IN field, (AB-REC3, column 64) to one of the following values:

  B Directly summarize the endpoint and produce a power-encode string.

  F Bypass the summary from Kill and summarize, using DCVS to produce a power-encode string.

  P Summarize the endpoint directly and do not produce a power-encode string.

  N Bypass the summary from Kill and summarize, using DCVS without producing a power-encode string.

- To selectively generate strings with specific endpoints grouped together, create special endpoint tables in the DKNEP data set.

The power-encode D-string does not produce kill bundles; therefore, DKNMCRE cannot delete the string. DKNECYC deletes the strings if the DCVX task sets the user-release flag on.

Important!

If *any* of the endpoints summarized are flagged as DCV power-encode endpoints, a string is generated for whatever is summarized.

If you want to have the DCV power-encode D-string contain DCVs for only one endpoint, you must summarize that endpoint separately. If you want to process DCVs for a select group, you must create selective power-encode endpoint tables for this purpose.

If multiple endpoints are summarized, the power-encode D-string is assigned a mixed endpoint (`99000001`). Otherwise, the single endpoint value is assigned to the string.

Document-Based Electronic Funds Transfer (DEFT) Processing

To load document-based electronic funds transfer (DEFT) data into the filing system, you can use a batch job that loads the electronic DEFT data (which accompanies the documents) into an MVS data set. This data must be transferred from a variety of sources and converted to a standard format. You must provide a program to transfer the data and perform this load.

Append the filing system utility program (DKNDFTF) as a second step in the batch job. This program enters the MVS data-set names, together with the DEFT user-profile name, into the filing system control data set.

If you have specified to do so, Enhanced System Manager periodically starts the DKNDFTP program. This program interrogates the control file and, if it finds new or unprocessed DEFT file entries, starts the DEFT input program (DKNDFTI) for each file entry. A record of these DKNDFTI starts can be optionally maintained and passed to a user exit for statistics-gathering purposes.

A sample JCL member resides in CPCSI.V01R01.SDKNSAM1 using the name DKNGDEFT. You can tailor the JCL to perform this task.

Important!

It is the responsibility of the user to have backup and recovery methods for the MVS data set used as input to DEFT processing.

Data-Entry Filing System

The institution that receives the data must provide a program that places DEFT data into a sequential MVS data set for use as input to the CPCS-I DEFT input system. This program is run as part of the batch job that creates the DEFT input file.

The layout of the MVS data set should correspond with the definition of the MDS and contain all codeline data from the original document in addition to any user-entered data and flags.

DKNDFTF runs as a step in the batch job created for the DEFT external preparation task. DKNDFTF creates a control record on the DEFT VSAM control file for each input data set (ddnames DEFTDS00 through DEFTDS99) created in the job. It enters the data-set names into the filing system control data set with the user DEFT profile name.

DKNDFTP is periodically started by Enhanced System Manager. DKNDFTP interrogates the control file; and, if it finds new or unprocessed DEFT file entries, initiates the DEFT input program (DKNDFTI) for each file entry. A record of these DKNDFTI starts can be optionally maintained and passed to a user exit for statistics-gathering purposes.

The EXEC statement for DKNDFTF includes a PARM operand that is set equal to one of the following values:

CREATE Create a record in the DKNDEFTD VSAM data set to describe an MVS DEFT data set. This is the default value.

DELETE Delete a record in the DKNDEFTD VSAM data set. PARM2 for this function is not used. The records to be deleted are identified by specifying the names of the DEFT data sets contained by the records.

You need to use the PARM= value only when you delete records.

The ddname of the DEFT input file must be one of the reserved ddnames DEFTDS00 through DEFTDS99. The job must also include a DD statement with the ddname DEFTPROF with the fully qualified data-set name and member name that defines the DEFT profile used for this data. Include a job step to run program DKNDFTF, which creates a control record on the data-entry filing system control file.

A sample JCL member resides in CPCSI.V01R01.SDKNSAM1 using the member name DKNGDEFT.

Programming Requirements

You should consider the following when you develop your DEFT file-creation program:

- Each DEFT file must contain a header record.

  This record defines the cycle or cycle date, bank number, or DKNDFTI report options. The header also identifies whether the electronic data is compressed or uncompressed. The full record layout is provided in the CPCS-I COBOL copybook DKNCRDFT and in assembler copy member DKNDFTHT.

  DKNDFTI also uses the header record to determine when a new MDS string should be created. A single DEFT file can produce multiple MDS strings based on the number of header records it contains. These strings can vary in their cycle and bank numbers; however, they remain under the control of the same sort type and DEFT profile member.

  The DEFT trailer record is optional. If present, it contains debit and credit total amounts and counts. DKNDFTI uses these values as control totals on the completion report. The record format for trailer records is also defined in DKNCRDFT and DKNDFTHT.

- The format of the electronic detail data must correspond to the MDS layout.

  The data format can be in compressed or uncompressed format. The file header record defines the format of the data.

- Your program can include items in the DEFT file that have no corresponding paper documentation, for example, cash posting records.

  These records are written to the MDS with other items. Flagging these items as *electronic-only* avoids their being included in the DEFT reconciliation report as *missing* items. You can do this during the DEFT file creation process by setting an indicator in byte 1 of Flag 4 in the MDS record.

Data Capture

The CPCS-I program DKNDFTI (transaction DFTI) reads the user-created sequential MVS data sets and adds them as strings on the MDS. DKNDFTI assigns tracer-group numbers for the strings it is generating, maintains and assigns sub-set numbers for strings, and, unless specifically requested not to by the user, assigns item-sequence numbers for items processed.

DKNDFTP starts DFTI automatically when it finds new or unprocessed entries in the control file.

DKNDFTO—DEFT Electronic Output

DEFT output processing captures prepared DEFT paper items from 3890/XP Series document processors. The detailed information (ZC images) is kept in an MVS data set for transmission to a financial institution.

Use the online task DFTO to create the MVS data set. This task verifies the user input (endpoint or cycle) and then scans the kill-bundle file for any electronic D-string's kill bundles that are flagged as not yet having been processed by DFTO. If the task finds any electronic kill bundles, it loads the kill-bundle information to a temporary file and passes the file to the DKNDFTOX module.

The following are required to generate a DEFT-output electronic transmission file:

- Flag the outgoing kill-pocket endpoint as electronic within the DKNAB data set. Turn on the AB-KILL-TYPE field. Change the field value, located in AB-REC1 column 67, to I.
- Give the outgoing kill-pocket endpoint an 8-character DD name, within its DKNAB member, that matches a subsequent DD in the DSAT table. A sample entry appears in the DSAT table as `DD=DFTO0001`.

  Update the AB-OUT-DDNAME-IN field. This field is in AB-REC4, columns 62 to 69. Set up a DSAT entry for each endpoint designated for transmission.
- Re-generate the DKNAB data set.
- If a user exit is required, create one modeled on the sample DEFTOEXT module supplied in CPCSI.V01R01.SDKNSAM2. Name the exit in the Bank Control File entry for the appropriate bank(s). Modify the BCF card 043, columns 54 through 61.
- To prevent the next transmission attempt from failing, delete the MVS file after the transmission. You can do this in either the transmission JCL or the batch job. You do not have to bring down CPCS-I to delete the file.

If DFTO is automatically or manually started with a string as input, use the following format for the 8-character DD name within DKNAB that matches a DD in the DSAT table:

```
DD = DFTbbbxx

bbb = three digit bank number
xx = CR for credit or DB for debit
```

Important!

DKNRMIT does not turn on the ZB-LISTED flag for any D-strings flagged as electronic. This is to prevent DKNMCRE from processing the strings before transmission. DKNDFTOX, which is the processing module that DKNDFTO calls, turns on the ZB-LISTED flags.

DKNMRGB: Setting Merge Options

This section describes the options provided by CPCS-I for:

- Handling free and missing codelines
- Setting the maximum number of entries in the R-string buffer and piggyback buffer
- Determining which control documents are treated as paper documents or as non-paper documents.

Codeline Controls

You can control how DKNMRGB handles missing codelines.

missing codeline a codeline that does not have a matching item sequence number/tracking number on the R-string.

free codeline a non-piggyback codeline on the R-string that does not have a matching item sequence number/tracking number on the I-string.

By default, DKNMRGB places all **free** codelines at the end of the string, and leaves all **missing** codelines in place; it reports the missing codelines by sending a message to the CPCS supervisor. You can set the following switch to have missing codelines deleted, instead.

```
FILLER                        PIC  X(01)     VALUE  'N'.
88  SW-DELETE-MISSING-CORRECTION             VALUE  'Y'.
88  SW-ONLY-REPORT-MISSING                   VALUE  'N'.
```

Buffer Controls

You can set the maximum number of entries in the R-string buffer. The default is 5000 codelines (up to 4999 entries in the R-string).

See the tagname `MRGEB-MAX-CODELINES` in DKNMRGB.

You can set the maximum number of entries in the piggyback buffer. The default is 999 codelines (up to 889 piggybacked codelines, including the piggyback control number).

See the tagname `MRGEB-MAX-PIGGYBACKS` in DKNMRGB.

Note: You should set the limit based on the actual number of codelines expected in the R-string. DKNMRGB uses an internal COBOL table to buffer the entire R-string; the maximum size of this table is 16 megabytes. If the number of codelines to be buffered would cause the table to exceed 16 megabytes, you must use one of the other merge options (options 1 through 9).

Control-Document Controls

You can specify which control documents are to be treated as paper documents and which are to be treated as non-paper documents. The default is CPCS documents. Change this according to your needs.

See the tagname `WS-USER-CNTL-DOCS` in DKNMRGB.

DKNSCAT Debugging Controls

The debugging switches below are located in module DKNSCA2. They control whether SCAT builds a 00-R-string when the DI-30-DOCH bit is turned on for a true piggyback.

Note: These switches are provided only for debugging; in a production environment, they should be set to the default settings only.

```
77  U-USER-OPTIONS-01       PIC  X(01)      VALUE 'S'.
    88  U-SCAT024W-STOP-SCAT                VALUE 'S'.
    88  U-SCAT024W-DO-NOTHING               VALUE 'N'.

77  U-USER-OPTIONS-02       PIC  X(01)      VALUE 'N'.
    88  U-SCAT024W-DONT-TOUCH-DI3           VALUE 'N'.
    88  U-SCAT024W-RESET-DI30DOCH           VALUE 'R'.
```

Results: If U-USER-OPTIONS-01 switch is set to STOP-SCAT, DKNSCAT ends with an error.

If U-USER-OPTIONS-01 switch is set to DO-NOTHING, DKNSCAT will produce unpredictable results.

If U-USER-OPTIONS-01 switch is set to DO-NOTHING, and U-USER-OPTIONS-02 switch is set to RESET-DI30DOCH, DKNSCAT will produce a scatted R-string.

The default setting is STOP-SCAT and DONT-TOUCH-DI30DOCH.

CPCS-I 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 3698 (X'E72') bytes of program storage of the document processor. CPCS-I macros generate this SSP along with the user's SCIs. The SSP performs the following functions:

- Initializing and setting CPCS-I-reserved switches and areas within the document processor
- Processing subsets and I-strings
- Controlling the updating of item number and microfilm number
- Identifying document types
- Selecting and processing CPCS-I control documents
- Allocating to pockets (tracer spraying)
- Issuing SCI pause instructions for tracer-verification errors and re-synchronization of codeline data matching
- Routing all batch/DCV, 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.

You code SSPs with SCIs that reside in sorter-program storage. They consist of a PROLOG 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 4-3 on page 4-53 and Figure 4-5 on page 4-55.

Standard Stacker-Select Prolog

A standard-sort SSP consists of two parts: the ENTR macro (which is 80 bytes long) and PROLOG. The required 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 Series Stacker Control Instructions 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 re-entrant into the CPCS-I 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.

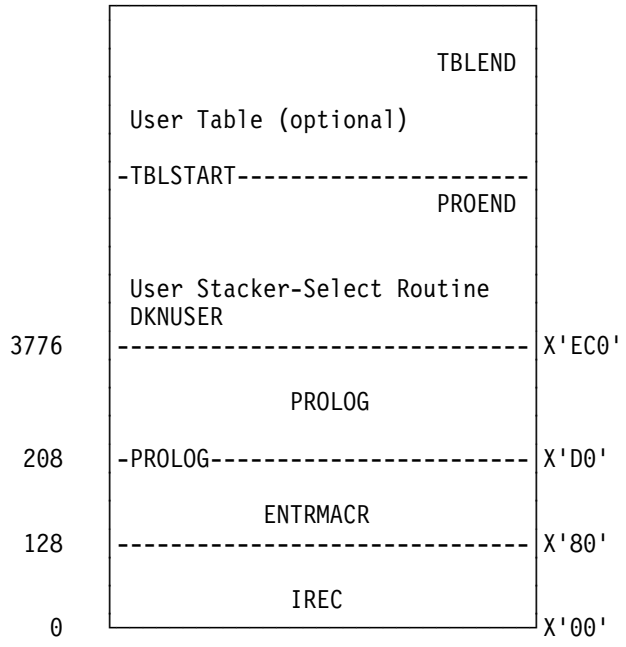


Figure 4-3. CPCS-I Sorter-Program Storage Map—Standard Sort

When the CPCS-I standard-sort SSP completes processing both non-control documents and CPCS-I block and batch/DCV 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 PROLOGX macro is the first statement coded in expanded sort primary stacker selection routines. PROLOGX uses 2- and 4-byte SCIs based on a type indicator, a program size specification, a 32-byte prefix, and a table address list.

You must use the following convention for SCI routines:

```
NAME     PROLOG
         THI     2,X'80'             IS THIS A CPCS-I 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 4-4. Sample Format for SCI Routines

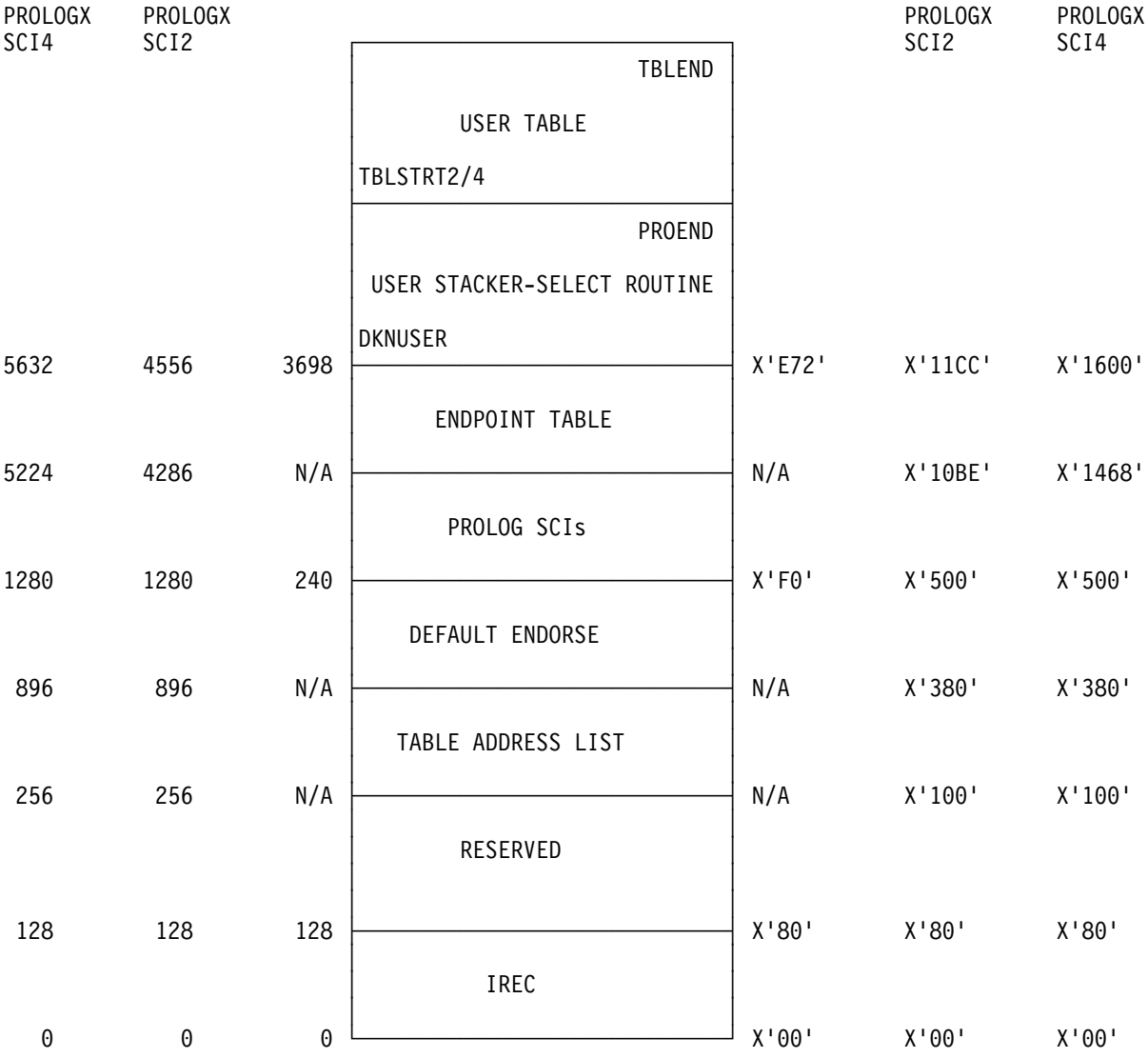


Figure 4-5. CPCS-I Sorter-Program Storage Map—Expanded Sort

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 sort-pattern definition. 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-I use. (For a description of header and status bytes and save-area locations, see “Document Processor Header-Byte, Status-Byte, Save-Area, and Counter” on page 4-57.) 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, make 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.
- 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.
- If you are using divider documents, the divider-serial field in the bank-control file determines which field is used as the reference/serial number field.

  PROLOG uses bytes 0 and 1 of the document data header to validate this field. All other fields have their validity bits set on, indicating that they are valid, regardless of any digit or length errors in those fields.

Document Processor Header-Byte, Status-Byte, Save-Area, and Counter

You must set the document processor's header and status bytes, defined in Figure 4-6, where indicated. For more information about header and status bytes, see the *3890/XP Series Stacker Control Instructions Reference*.

Figure 4-6 (Page 1 of 2). CPCS-I Document Processor Header Byte, Status Byte, and Counter

Byte	Bit	Setting
Header		
2	0	
		0 User document **1** CPCS-I control document
	1–2	If CPCS-I 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** In-work **11** In-work 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.
9		
	0	1=Field 4 valid
	1	1=Field 6 valid
	2	1=Sub-batch document (byte 2, bits 0 through 2=111) 1=Subtotal voucher document (byte 2, bit 0=1) 1=CDMR subset tracer document (byte 2, bits 0 through 2=101)
	3	1=AST pointer document (byte 2, bit 0=1) 1=HSRR subset tracer document (byte 2, bits 0 through 2=101)
Status		
	0–2	Reserved for CPCS-I

Figure 4-6 (Page 2 of 2). CPCS-I Document Processor Header Byte, Status Byte, and Counter

Byte	Bit	Setting
	3–7	Available to the user
Counters		
	1–2	Reserved for CPCS-I
	3–16	Available to the user

The CPCS-I PROLOG uses the following save areas in the document processor:

0–487 Available to the user
488 CDMP DCV-expected flag
489 Available to user
490–493 Divider count for divider spraying
494–497 Pocket index for divider spraying
498 Merge before main-in-progress flag
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-I SCI Macros and User SCI Coding

There are a number of macros within CPCS-I 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, PROLOGX, TBLSTRT2, and TBLSTRT4, work with the 3890/XP Series document processors only. The expanded format sort-pattern definition records are required to initialize these sorts (see “Sort-Pattern-Definition Record Formats” on page 4-30). These macros provide SCI routines that process documents at the document processor. CPCS-I 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-I loading of the SCI routines. For additional information on SCI routines, see Figure 4-8 on page 4-64.

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

The PROLOG macro must be the first statement of the user's SCI program. It generates the CPCS-I 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

The PROEND macro must be the last statement of the user's SCI program. It delimits the SCI program so that CPCS-I 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

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

The TBLEND macro must be the last statement of the user's SCI table. It delimits the SCI table so that CPCS-I 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-I 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.

PROLOGX Macro

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 4-5 on page 4-55.)

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-I 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** | *nnnn*K | *nn*M}] [DIVDOC={**YES | NO**}] [ASSMTYP={**blank** | **S** | **M**}]

TYPE={SCI2 | SCI4}
This is an optional parameter that indicates SCI type. The default is SCI4.

PGMSIZE={0 | *nnnn*K | *nn*M}]
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-I ignores this parameter. If present,

PGMSIZE must be the same in the 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 indicates whether the user-sort routine (DKNUSER) receives control to process divider documents. YES specifies that the divider document must be passed to the user's SCI routine. The default is NO.

If you decide to have the user-sort routine process divider documents (DIVDOC=YES), the PROLOGX macro does the following:

- Validates the divider document
- Selects a pocket based on the pocket count for merge-before-main divider documents, using the select-pocket-by-counter (SPC) instruction
- Sets the divider document bits on in the user portion of the record header (header-byte 2, bits 0 through 2 are set to 100).

The following rules apply to the user-sort routine when you specify DIVDOC=YES:

- The user-sort routine must not change the pocket selection for the merge-before-main divider documents as the PROLOGX macro makes the pocket selection. Instead, the user-sort routine obtains the pocket selection from header-byte 6.
- The user-sort routine must give an SPC instruction for valid regular dividers, and must give an FM instruction for invalid regular dividers.
- You must add the following code immediately after the PROLOGX statement:

```
           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
```

- The user-sort routine should use the DKNRTN command to return control to CPCS-I. This is standard procedure.

ASSMTYP={blank | S | M}
This is an optional parameter that can expand PROLOGX as a DSECT and include labels used to set gates. The blank option (the default) and the S option both generate PROLOG SCI; the M option expands PROLOGX as a DSECT and includes labels used to set gates.

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 4-7 is an example of a 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   (In-work pocket)
                   21AAAAAAAA,   M/P=2/1
                        .
                        .
                        .
                   6105311004,   M/P=6/1   (Kill pocket)
                   66AAAAAAAA,   M/P=6/6   (Last table entry)
```

Figure 4-7. Example of DKNEPTBL Table

TBLSTRT2

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-I 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

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-I 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* | *nnnn*K | *nn*M [,PGMSIZE={**0** | *nnn*K | *nn*M}]

TBLLOAD=*nnnnn* | *nnnn***K |** *nn***M**
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-I SCI macros all generate the following 32-byte prefix. The CPCS-I 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)
                                                (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
```

Figure 4-8. Sample Expanded-Format SCI Routine and Table

```
*****************************************************************

          TITLE   'TBLSTRT4 SAMPLE '
          AMODE    31
          RMODE    ANY
MT4SAMP1  TBLSTRT4 TBLLOAD=128K,PGMSIZE=256K
KILLTB    CSTBL    12,0                  CPCS-I POCKET 1
          CSTBL    13,1                  CPCS-I POCKET 2
          CSTBL    14,2                  CPCS-I POCKET 3
          CSTBL    15,3,4,5              CPCS-I POCKET 4
          CSTBL    16,6,7,8,9            CPCS-I POCKET 5
          CSTBL    11,T                  CPCS-I POCKET REJECT
LASTLABL  DC       CL30'128K TBLSTRT4 PASS 1- MT4SAMP1'
          TBLEND
```

Figure 4-8. Sample Expanded-Format SCI Routine and Table

Examples of SCI Routines Using Tables

Figure 4-9 and Figure 4-10 are examples of SCI routines.

Figure 4-9. Main SCI Program

Location		Source Statement
	MAIN	PROLOG
400		*
		*
		(SCIs)
		*
		CST (0,2),KILLTB
		*
		*
600	KILLTB	EQU MAIN+X'530'
		PROEND

Figure 4-10. 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-I tasks OLRR, ADJ and DFTI 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
            :
            :
MRDBDSCT    DSECT
            COPY  MRDBDSCT
            ORG   MRDIMG
            COPY  DIDSCT
            ORG   ,
```

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-I 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 4-11 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 4-11. 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

2. Field DIFLAG2

   Equate
   DIMTCR On if this is a control document
   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)

3. Field DITYPEI

   If flag DIMTCR is off, you cannot set any flags in this field, because this field represents the control document type. If flag DIMTCR is on, you can set any flags in this field.

4. Field DITYPEU

   This byte is available to the user.

5. Field DI3FLG3

   This field includes the document type and validity indicators. Host edit routines must set field validity indicators and also validity indicators for batch and block control documents and select them to the reject pocket.

Setting Pocket Numbers

Your host codeline edit routine should not use the CPCS-I pocket number. CPCS-I automatically takes the pocket code placed in buffer status byte 4 (MRDBS4), converts it, and places the result in the CPCS-I pocket number byte (DIPKT field).

If you identify a pocket as a reject pocket in your host codeline edit routine definition, CPCS-I 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 4-12 shows examples of this converted pocket number for the document processor.

Figure 4-12. Converted Pocket Number for the Document Processor

Buffer Status Byte (4)	Document Processor Pocket (Module/Pocket = MP)	Pocket # in CPCS-I for Good Pockets		Pocket # in CPCS-I for Reject Pockets	
		SDE	Report	SDE	Report
X'11'	Reject	NA		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 user parameter area
3–12 Unpredictable
13 Standard save area register.
14 Return register
15 Host codeline edit routine entry point.

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 stacker-select, user-edit routine.

Field Name	Field Size	Content Value	Modified by	Field Content Description
MUPAID	CL04	MUPA	CPCS-I	MUPA identification
MUPAENT	XL02		User	CPCS-I entry tracer-group number (binary)
MUPACYC	CL02	1 through 9 A through L	User	CPCS-I entry cycle number (Left justified, alphanumeric)
MUPATYPE	XL01		User	CPCS-I 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.
MUPASTEP	Fullword		DKNMOLRI	Contains address of stacker-select table. (Binary zeros if no table is defined.)
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 4-73).
Additional user address fields reserved for user interfaces				
MUPAUSF1	Fullword	Address	User	MUPA address field reserved for user (set initially to binary zeros)
MUPAUSF2	Fullword	Address	User	MUPA address field reserved for user (set initially to binary zeros)
MUPAUSF3	Fullword	Address	User	MUPA address field reserved for user (set initially to binary zeros)
Flag bytes for MUPAFLG1, MUPAFLG2, and MUPAFLG3				
MUPAFLG1	XL01			MUPA flag byte 1
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.

Field Name	Field Size	Content Value	Modified by	Field Content Description
MUPADSPK	Equate	B'001000000'	DKNOLRR (or other user interfaces)	OLRR indicator: Do not let the user-edit routine set the pocket. This indicator is set to signal that the user stacker-select edit-routine pocket decision must not be changed from the original pocket decision made by MICR capture. MUPA label MUPASPKT contains the original pocket decision if the interface routines decide to reset the pocket field in the CPCS-I DI image.
MUPAFLG2	XL01			MUPA flag byte 2
MUPAPASA	Equate	B'00000001'	User	Indicated by the interface routine of the caller to pass to user-edit routines CPCS-I CTL images that include TRACERS and DIVIDERS. If this field is not set, TRACERS and DIVIDER control-document-type images are not sent to the user-edit routines.
MUPAFLG3	XL01			MUPA flag byte 3 – Reserved
DKNMOLRI interface program function byte				
MUPAFUNC	XL01		User	DKNMOLRI interface program function indicator
MUPAOPEN	Equate	B'10000000'	User	Indicates that the interface program requests a user-edit sort routine LOAD function. MOLRI uses the MUPASSN and MUPASTN fields to get the user pocket-select routine and table names when the CPCS-I MDS string header is not passed. If specified, the user-exit routine is called to perform the DKNMOLRI interface load function of the user stacker-select routines.
MUPACLOS	Equate	B'00001000'	User	Indicates that the interface program requested a shutdown function that signals DKNMOLRI to free storage acquired and make the same communication to the DKNTYPER control document determination routine. If specified, the user-exit routine is called to optionally perform the DKNMOLRI interface delete function of the user stacker-select routines.
MUPAUTYP	CL08		User	Indicates the type of requested interface processing. If this field contains blanks or low values, DKNMOLRI sets this field to the value of `USEREDIT`. USEREDIT assumes the functions of the OLRR and DEFT input interfaces.
MUPAAPPL	CL08		User	Indicates the interface application name (for example, DKNOLRR) If this is blank, DKNMOLRI extracts the name from the APCBNAME field, using the application task control block (APTCB) passed by the interface routine.

Field Name	Field Size	Content Value	Modified by	Field Content Description
MUPAXWA	XL04		CPCS-I	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 BCF 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 online reject re-entry operation. When allocated, this area initializes to X'00'.
MUPASSN	CL08		DKNMOLRI	User-interface selected primary stacker-select edit routine name. This field can be supplied by either the user interface, the name specified for the MOLRIEXT parameter, or in a CPCS-I defined string header record passed to DKNMOLRI during the MUPAOPEN initialization function. If it is passed in the CPCS-I string-header record, DKNMOLRI adds a suffix R to the routine's module name.
MUPASTN	CL08		DKNMOLRI	User-interface optional stacker-select table name. This field is supplied by the user interface, the user-exit routine (specified by the MOLRIEXT parameter), or during the MUPAOPEN initialization function.
MUPASSEP	Fullword		DKNMOLRI	Contains the entry-point address of the specified user stacker-select edit routine in MUPASSN. This field can be set by a specified user-exit routine (MOLRIEXT parameter) in the CPCS-I BLDL entry for DKNOLRR, or by a globally specified user-exit routine by bank with the CPCS-I bank control file. For information on using the bank-control file, see "Bank Control File Record Formats" on page 4-15.
MUPASPTR	Fullword		DKNMREAD	Contains the address of the sorter document-buffer area that MREAD uses to communicate with the MREAD user-exit routine
MUPASPKT	XL01		DKNMOLRI	Contains the pocket-code value before modification by the stacker-select, user-edit routine
MUPABANK: Contains an 8-byte field for alphanumeric bank information				
	CL04		DKNMOLRI	Reserved for future use
MUPABNKN	CL03	000 – 255	User	Contains a 3-byte field for alphanumeric data to indicate the bank number. If this value is not specified, the user's default bank number is used. See "Bank Control File Record Formats" on page 4-15.
MUPADOPT	XL01			Contains the dash option from DKNBCF. See "Bank Control File Record Formats" on page 4-15.

Field Name	Field Size	Content Value	Modified by	Field Content Description
USEREDIT function fields				
MUPAIMGA	Fullword	Address	DKNMOLRI	Fullword address containing the pointer to the user's interface CPCS-I MDS DI image Contains this address pointer even if the MRDBUF DSECT is passed with the DI image located at label MRDIMG
MUPAUEXA	Fullword	Address	DKNMOLRI	A 4-byte field containing the entry-point address of the user exit (if specified) that DKNMOLRI loads into storage
MUPAMLWA	Fullword	Address	DKNMOLRI	Contains the fullword address of DKNMOLRI's work areas **Note:** This field must contain binary zeros on the first call to DKNMOLRI. During subsequent calls, this field must not be modified.
MUPATYPA	Fullword	Address	DKNMOLRI	Contains the fullword entry-point address of the control document determination routine DKNTYPER. This field is set by DKNMOLRI at initialization time.
MUPATYPW	Fullword	Address	DKNTYPER	Contains fullword address of DKNTYPER's work areas. Reserved for use by the DKNTYPER routine. This field must not be modified by user interfaces.
MUPAMSGA	Fullword	Address	CPCS-I	A fullword address containing the pointer to DKNMOLRI message buffer (60-byte area) when a non-zero return code returns in register 15. The first 2-byte section of this message area contains the actual message length in binary.
	4F	Address	CPCS-I	4F - Reserved for future use
MUPALEN	XL(152)	Equate	CPCS-I	Length of user parameter area

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
30 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-I 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-I 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-I program library.

DKNPLST 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 (83 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
- Sort code (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)
- Sort code (field 5)
- Account number (field 3)
- Transaction code (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
- Sort code (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-I, 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-I 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.

Extract Programming

After CPCS-I captures and processes input documents, the next step is to extract the codeline data records, usually by M-string, from CPCS-I. 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 4-79.

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-I log tapes to retrieve the M-strings. (See “Data-Set Recovery Procedures” on page 1-29.) 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 MDEF macro (see “MDEF Parameters” on page 2-30), the extract program can flag extracted M-strings as being user-processed.

User Application Requirements

CPCS-I 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-I Programming Guide*.
- You can change application tasks to refer to the re-definable 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 (minus 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.

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 MDIS user-exit routines.

Note: ImagePlus High Performance Transaction System Application Library Services 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 DKNCRZD. For assembler modules, see the DI3BYTE2 field in the copybook DIDSCT.

DKNCRZD contains the uncompressed format COBOL format codeline record layout. DIDSCT contains the compressed assembler format.

Adjusted M-strings can contain the following types of High Performance Transaction System Balancing adjustments:

INSERT This adjustment is indicated by the following record pair:

1. Insertion control record.

Field	Flag	Flag Description
DIFLAG2	DIMTCR	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 sequence number as the associated insertion control record, with the exception of the first digit, which is always 1.

Field	Flag	Flag Description
DI3BYTE3	DI3INS	Insert

DELETE This adjustment is indicated by a deletion control record. The record contains the deleted record detail data and the following flags:

Field	Flag	Flag Description
DIFLAG2	DIMTCR	Control record
DITYPEI	DIBALDCH	Deletion or change
DI3BYTE2	DI3DEL	Delete

CHANGE This adjustment is indicated by the following record pair:

1. A change control record contains the original detail data and the following flags:

Field	Flag	Flag Description
DIFLAG2	DIMTCR	Control record
DITYPEI	DIBALDCH	Deletion or change
DI3BYTE2	DI3CHG	Change

2. A change detail record contains the new data. This record has the same 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 This adjustment is a combination of a DELETE adjustment and an INSERT adjustment. A DELETE adjustment corresponds to the moved-from item. An INSERT adjustment corresponds to the moved-to item. All records for the move have the same sequence number, except the detail records that have 1 as the first digit of the sequence number. The records are identified by the flags shown, except the DI3BYTE2 field that contains the following flag and flag description:

Field	Flag	Flag Description
DI3BYTE2	DI3MVD	Move

Power-Encoding Records

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 sequence number is the same as the associated insert control record or move control record.

Field	Flag	Flag Description
DIFLAG2	DIMTCR	Control record
DITYPEI	DIBAPE	Power encoding
DI3BYTE3	DI3PE	Power encode

2. Power-encoding detail record. This record contains the inserted data or final moved data (if the moved item data is changed before or after the move). The sequence number is the same as the associated power-encoding control record, except that the first digit is 1.

Field	Flag	Flag Description
DI3BYTE3	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 you are using extract applications, a double posting can occur if you do not bypass power-encoding records (identified by 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 are not distributed, except individual delete records (not part of a move).

- Insert detail adjustment records are not distributed.
- Change and power-encoding detail records are distributed.

Power-Encode Subsequent Passes

When the DKNMIPI user-exit routine 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-Encode 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. The deleted items, removed during balancing, are listed as missing items.

To eliminate these missing item conditions, you can activate a DKNMDIS user-exit routine to stop distribution of the delete control records. This also eliminates codeline data matching failures caused by groups of contiguous deletes removed during balancing. The IBM-supplied sample DKNMDIS user-exit routine (DKNMDIX2) includes part of the code needed to do this.

To aid in sub-pass reconciliation, the HCDM task lists adjusted records with the message HPTS ADJUSTED ITEM in the missing and free items report. The PLST task shows adjusted records flagged as B (HPTS balanced) in the entry master list.

To aid in resolving post-kill out-of-balance situations the KILL and RMIT tasks show adjusted records flagged as B (HPTS balanced) in the remittance reports.

For additional information on report layouts, see Appendix E in the *CPCS-I Terminal Operations Guide*.

Notes:

If the HCDM task is not run after each sub-pass:

1. High Performance Transaction System adjusted records may not be identified in the remittance reports.
2. High Performance Transaction System adjusted items may not be properly encoded on the second and third power-encode sub-passes.

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 data in fields 1 through 8 for codeline data matching. It also contains the new data in the work field for power-encoding and is flagged for use in codeline data matching.

   When the DKNMIPI user-exit routine is active, CPCS-I 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 “use 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

This section describes the options provided by CPCS-I for data security and resource protection. CPCS-I lets you control, through IDs and passwords, which operators can log on to the CPCS-I system. You can also define the tasks that the operator is allowed to start.

Three security options are available using the SCRTY parameter in the MDEF macro:

SCRTY=NONE
Indicates that you do not use the data security option. If you omit the parameter, the default is none.

SCRTY=RACF
Lets you provide RACF control of CPCS-I resources.

SCRTY=USER
Lets you provide access security through a supplied user-exit for control other than RACF or the operator logon table.

Resource Access Control Facility Option

The MDEF generation parameter, SCRTY=RACF, indicates that the IBM Resource Access Control Facility (RACF) security subsystem is the security option for CPCS-I. 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, DKNSGON and DKNSGOF control access to CPCS-I through IDs, passwords, and other parameters defined in RACF for the respective CPCS-I 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-I 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-I 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-I Programming Guide* and the *CPCS-I Programming Reference*.

RACF Security Installation Procedure

You must complete the following tasks when using RACF facilities for security with CPCS-I.

1. Define a RACF class and class group to protect your CPCS-I resources.
2. Connect your CPCS resources to the RACF class and/or class group structure.
3. Define a RACF group for your CPCS-I user IDs.
4. Code the proper MDEF parameters in order to activate your RACF security procedure (see “MDEF Parameters” on page 2-30).

The following procedures illustrate, in greater detail, how to implement RACF security within CPCS-I.

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 RACFSET JCL in CPCSI.V01R01.SDKNSAM1 shows an example of how to define a resource class and resource class group.
   - The default RACF class is $CPCSI. To change it, substitute your choice in the JCL.
   - To use a group of resources, code a group in the JCL. CPCS uses $GCPCSI 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-I class. Every time you modify the RACF CPCS-I 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 RACFCPCS JCL in CPCSI.V01R01.SDKNSAM1. 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 $CPCSI and connect groups of resources to the CLASS GROUP ($GCPCSI). For an example, see “Establishing Your RACF Structure” on page 4-85.

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-I 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. If the value of SCRGP is 0, TSO users defined to any RACF group can log on to CPCS-I. The users may then access any resources that are not protected or are 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.

4. The following MDEF parameters may be coded in order to activate your RACF security procedure for CPCS-I.

   - Code `SCRTY=RACF` in the MDEF macro generation.
   - Specify the RACF class on the MDEF `SCRCL=` parameter for the resources that you protected. The default class name for CPCS is $CPCSI. If you used a different class name in the RACFSET and RACFCPCS JCL, change the parameter to reflect the proper class name.
   - The MDEF macro has a default parameter, CHGPSWD=0, that lets you maintain a password within the rules established in the definition to RACF for CPCS-I. If you do not want an operator to be able to change the password, code CHGPSWD=1 in the MDEF macro.

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.

The following is a sample group structure:

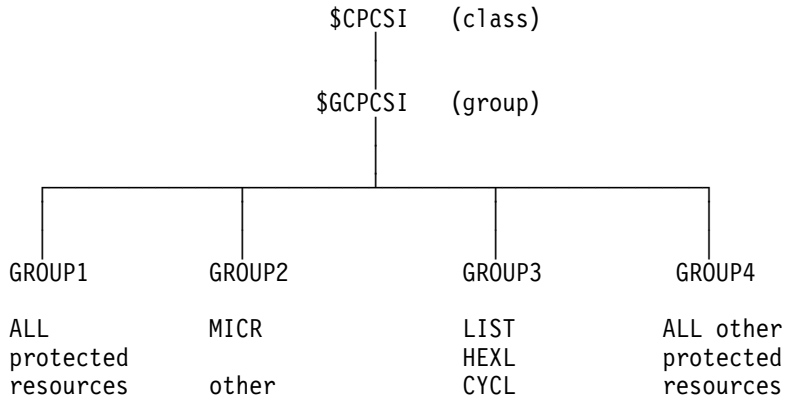


Figure 4-13. Sample RACF Class/Group to User ID Structure

The groups would be joined to resource class group $GCPCSI.

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:

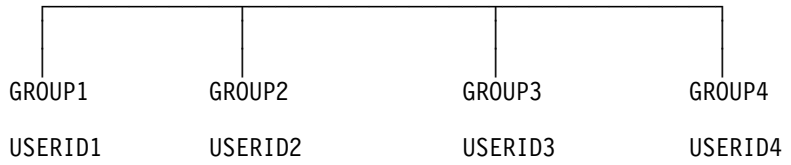


Figure 4-14. Sample RACF Class/Group to User ID Structure

This example lets USERID1 access all resources. USERID2 can access only resources in GROUP2. Any resource not defined to the class $CPCSI or class group $GCPCSI is not protected. Any user ID given access to resources can also access undefined resources.

You must understand this resource class structure. It is important to the security of the CPCS-I 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:

- *Resource Access Control Facility Security Administrator's Guide*
- *Resource Access Control Facility Command Language Reference*
- *System Programming Library: Resource Access Control Facility*.

CPCS-I Commands That You Can Protect with RACF

After you create the RACF class CPCS-I ($CPCSI) using the RACFSET JCL member in CPCSI.V01R01.SDKNSAM1, use the RACFCPCS JCL member in CPCSI.V01R01.SDKNSAM1 to protect all supported RACF resource names in CPCS-I and its supported features (such as Enhanced System Manager). Use the following guidelines to make your decision.

- RACF cannot protect the following commands:

  SGOF
  SGON

- RACF cannot protect the following commands individually; however, they are controlled through the SUPV command:

  ANODE
  ASGF
  CANCEL
  FNODE
  HALTMICR
  PDUMP
  STRTMICR

User-Supplied Security Option

CPCS-I includes an option in the MDEF macro that lets you use a product other than RACF to control which operators can log on CPCS-I.

To establish your non-RACF security installation you must change the SCRTY parameter in the MDEF macro to SCRTY=USER. This causes the following events to occur:

- CPCS-I places the code X'03' into the PARMLIST field, SCRTYFLG, indicating user security.
- CPCS-I loads the user security exit routine 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 security exit routine:

  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-I issues a write to the operator (WTO) error and sets SCRTYFLG=SCRTNONE (no security) and writes a message to the SCROLL data set.

- CPCS-I calls the user security exit system initialization through the DKNSECR module, which, in turn, calls the user security exit routine.
- CPCS-I analyses the return code from the exit and issues error messages. If a non-zero return code is encountered, the MDEF macro generation issues a WTO error, sets SCRTYFLG=SCRTNONE, and writes a message to the SCROLL data set.

You must supply the user security exit routine.

When you log on CPCS-I, the DKNSGON 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 the user security exit routine.
3. Uses the return code to route control for further signon processing.

DKNSECR provides support in a CPCS-I environment for a user-specified security system and for a single control point in CPCS-I 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 exit routine.
3. DKNSECR passes the return code from the user security exit routine back to the calling routine (DKNSGON).

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.

Bibliography

The publications in this bibliography contain information related to CPCS-I.

ACF/VTAM Publications

The following publications are related to the ACF/VTAM Version 3 Release 4 product:

IBM ACF/VTAM Programming, SC31-6436

IBM Planning and Reference for NetView, NCP, and VTAM, SC31-6124

IBM VTAM Programming for LU 6.2, SC31-6437.

Document Processor Support Publications

The following publications are related to document processor support:

IBM 3890/XP Series Document Processor General Information, GA34-2012

IBM 3890/XP Series Programming Guide, GC31-2662

IBM 3890/XP Series SPXServ Reference, GC31-2704

IBM 3890/XP MVS Support and 3890/XP VSE Support Program Reference, SC31-2654

High Performance Transaction System Publications (Version 1)

The following publications are related to the IBM ImagePlus High Performance Transaction System (Version 1):

IBM ImagePlus High Performance Transaction System General Information Manual, GC31-2706

IBM ImagePlus High Performance Transaction System Application Library Services Programming Reference, SC31-2794

IBM ImagePlus High Performance Transaction System Installation Guide, SC31-3943

High Performance Transaction System Publications (Version 2)

The following publications are related to the IBM ImagePlus High Performance Transaction System (Version 2):

IBM ImagePlus High Performance Transaction System Planning Guide, GC31-4005

IBM ImagePlus High Performance Transaction System Installation Guide, GC31-4006

IBM ImagePlus High Performance Transaction System Application Library Services Operations Guide, SC31-4010

IBM ImagePlus High Performance Transaction System Application Library Services Programming Guide, SC31-4011

IBM ImagePlus High Performance Transaction System Application Library Services Programming Reference, SC31-4012

MVS Publications (Version 5)

The following publications are related to MVS:

IBM MVS/ESA Programming: Extended Addressability, GC28-1468

IBM MVS/ESA Installation Exits, SC28-1459

IBM MVS/ESA Initialization and Tuning Reference, SC28-1452

IBM MVS/ESA JCL User's Guide, GC28-1473

IBM MVS/ESA System Messages, Volume 1 (ABA-ASA), GC28-1480

IBM MVS/ESA System Messages, Volume 2 (ASB-EWX), GC28-1481

IBM MVS/ESA System Messages, Volume 3 (GDE-IEB), GC28-1482

IBM MVS/ESA System Codes, GC28-1486

IBM MVS/DFP Version 3 Release 3: Utilities, SC26-4559

To help you find other MVS library references for various release levels, check:

MVS/ESA Library Guide for MVS/ESA System Product Version 4, GC28-1601

MVS/ESA System Product Library Guide with JES2, GC28-1423

RACF Publications

The following publications are related to RACF:

IBM Resource Access Control Facility Command Language Reference, SC28-0773

IBM Resource Access Control Facility Security Administrator's Guide, SC28-1340

IBM System Programming Library: Resource Access Control Facility, SC28-1343.

IBM Resource Access Control Facility Master Index, GC28-1035

CPCS Enhanced System Manager Publication

The following publication is related to Enhanced System Manager:

IBM Check Processing Control System: Enhanced System Manager User's Guide, SC31-4002

Glossary

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

A

ABA. American Bankers Association.

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

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

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

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

ALS. Application Library Services.

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

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

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

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

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

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

APTCB. Application program task control block.

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

AST. Assist document.

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

B

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

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

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

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

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

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

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

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

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

BCF. Bank Control File.

BDAM. Basic direct access method.

BGC. Bank Giro Credit.

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

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

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

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

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

C

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

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

CDM. Codeline Data Matching.

CDMP. Codeline Data Matching Prime.

CDMR. Codeline Data Matching Rejects.

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

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

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

cheque. UK spelling of “check.”

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

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

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

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

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

codeline data record. See *data record.*

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

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

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

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

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

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

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

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

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

control slip. See *control document.*

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

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

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

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

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

cross record. See *XREC.*

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

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

D

DASD. Direct access storage device.

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

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

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

DCV. Docket Control Voucher.

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

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

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

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

DEFT. Document-based Electronic Funds Transfer.

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

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

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

DFD. Data Flow Diagram.

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

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

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

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

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

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

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

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

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

D-string. Distributed string.

E

ECDM. Extended codeline data matching.

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

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

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

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

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

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

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

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

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

EPC. Extended process control field.

ERP. Enhanced reject processing.

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

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

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

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

F

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

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

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

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

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

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

FSG. Fine sort group.

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

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

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

G

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

H

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

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

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

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

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

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

HSRR. High-speed reject re-entry.

I

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

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

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

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

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

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

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

inscribe. Synonym for *encode.*

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

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

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

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

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

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

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

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

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

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

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

I-string. Input string.

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

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

J

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

JGC. Joint Giro Credit.

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

JCL. Job Control Language.

JES. Job entry subsystem.

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

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

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

K

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

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

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

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

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

L

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

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

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

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

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

LU. Logical unit.

LU 6.2. Logical unit 6.2 protocol.

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

M

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

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

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

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

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

MDS. Mass data set.

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

MICR. Magnetic ink character recognition.

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

MIS. Management Information System.

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

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

M-string. Merged string.

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

O

OCR. Optical character recognition.

OLMS. Online manual split.

OLRR. Online reject re-entry.

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

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

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

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

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

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

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

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

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

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

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

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

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

* Trademark of IBM

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

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

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

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

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

P

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

pass-to-pass control. A process that maintains the total amount and item control of a group of documents on subsequent passes, when control has been established on the previous pass.

path. The path of a functional unit of work is the ordered list of tasks processed for the associated CPCS-I string. See also *flow code* and *flow control.*

pending status queue. A first-in-first-out System Manager queue through which CPCS-I applications interface to the System Manager, in sequence, to perform UOW creations, deletions, inquiries, and updates.

piggyback item. An item that was missing from its assigned pocket in a sorter and sorted “free” to an unidentified pocket, as when one document attaches itself to or overlaps another during processing.

pocket 1-1. See *system reject pocket.*

PRAD. Propagation of Adjustments.

presenting bank. A UK term for the bank sending documents and DCVs and requesting funds for the DCVs.

prime pass. The first pass of an entry on a document processor.

printing after the fact. See *deferred printing.*

process control field. Used in the US by the payor bank to know which process applies to each item. In the UK this field is called *transaction code* and is used to identify document types.

proof. Receives checks that come from tellers, mail and night depository, and internal departments of the financial institution. Proof balances transactions and inscribes or encodes the monetary amount in MICR.

proof of deposit. The act of totalling items at the deposit level and ensuring that the total of the credits equals the total of the debits.

propagation of adjustments. The process of ensuring that adjustments made in Balancing and elsewhere are carried forward to kill/remittance and other system output processes.

R

RACF. Resource Access Control Facility.

RBA. Relative block address.

reconcile. To find and correct the cause of a difference between two sets of totals.

reconciliation. See *balancing.*

reconditioning. The process of straightening folded items, inverting upside-down items, flipping reversed items, and removing any residual staples or rubber bands.

reference. A UK term for a field encoded on credit documents, corresponding to the 6-digit Serial field on debits. The Reference field may be up to 18 digits in length and (if printed in OCR) may contain alphanumeric characters.

rehandle pocket. A document processor pocket that receives items for multiple endpoints. Items directed to rehandle pockets are processed again on a later pass.

reject. A document that cannot be read in its entirety by a document processor or that fails certain editing checks. This document is normally directed to a special pocket called a reject pocket.

reject string (R-string). Strings that are created by the online reject re-entry task. Each R-string represents checks that have been re-entered online. R-strings are input to the DKNMRGE task.

relationship. Shows the parent/child hierarchy of units of work.

relative block address (RBA). In CPCS-I, the calculated location of a specific record.

remit. A UK term; to send items to another financial institution.

remittance file. A UK term for an MVS data set that is created by Remittance (Kill) processing. It is essentially an electronic version of the remittance list and may be used to support DEFT input processing at the receiving institution.

remittance list. A UK term for a CPCS-I Kill List that is produced to support negotiation and settlement of a batch of documents prefixed by a DCV. It is used for conventional interchange between clearing operations.

repass. See *rehandle pocket.*

rerun. A group of items that are sorted into a pocket on one pass and later brought into a document processor for more sorting.

Resource Access Control Facility (RACF). An MVS security subsystem that determines the validity of each operator's ID password and that controls operator access to application tasks and transactions.

responding bank. A UK term for the bank making payment on documents/DCVs received from the presenting bank.

restart. An initiation of the CPCS-I system after a system failure. A restart is generally used to start the system (after an abnormal end of a task) to cause the executive routines to re-establish the system to the status that existed before the failures.

restart buffer. An area where records are stored in an IBM 389x/XP Series document processor during online operations until they are sent to the host. The buffer is accessed during automatic restart.

resynch document. A control document used in DEFT processing to match DEFT data to the documents currently being processed on Prime and also used to separate and identify kill bundles on output.

return item. A check that is not honored by the maker's financial institution and that is returned to the depositor's financial institution.

routing/transit number field. An encoded check field that represents the financial institution on which the check is drawn. In the UK, this is referred to as the *Sort Code*.

R-string. Reject string.

S

SCI. Stacker Control Instruction.

scroll. The ability to use the DKNSCRL application to page through or look at the scroll data set. This data set includes supervisor terminal messages and DKNATASK log messages.

SDE. String directory entry.

separator. See *divider slip.*

sequence number. A number, assigned to a document, that uniquely identifies its position in a group of incoming or outgoing work.

serial field. A UK term for the 6-digit field, (equivalent to the check number in the US), which is normally the serial number of a check. On credits, the same field is called a Reference and may be up to 18 digits in length.

settlement. The act of bringing sets of related figures from two financial institutions into agreement. Adjustments are made to offset the differences.

simulated sorter. A CPCS-I facility that allows a user to run MICR, using an input file without a physical sorter.

slip. A slip is a control document used to prefix bundles for control purposes.

SMOF. System Manager Online Functions.

SNA. Systems Network Architecture.

sort code. A UK term for the field (equivalent to the routing transit field in the US) which identifies the bank and branch to which a debit or credit item belongs. It is in the format *BB-bbbb*, where *BB* identifies the bank, and *bbbb* identifies the branch within that bank. It may be printed in MICR (on checks and some credits) or in OCR (on some credits). If printed in MICR, the two parts of the field are separated by a dash (SS4).

sorter station (also document-processor station). A work station consisting of a document processor and a terminal for operator communications. Synonym for document-processor station.

sort pattern. A table used by the sort routine to determine the pocket to which a check is to be directed.

sort-pattern definition file. A collection of records that contains control information that MICR in CPCS-I uses to set up and control document sorting; it also contains data about endpoints.

sort routine. A time-dependent routine that does all processing required to direct a document to a specific document processor pocket.

sort program. A routine that performs all processing required to select a document to a pocket.

spool data set. A data set used to store printed output lines. Each spool (Simultaneous Peripheral Operations On-Line) data set is written by a CPCS-I application task and is read by the CPCS-I output writer as it is being printed.

SSB. String status block.

SSM. String segment map.

Stacker Control Instruction (SCI). SCI is the name of a language used to write programs to control the sorting of documents on a 389x document processor.

statistics. The processing of unit-of-work (UOW) data through a statistical program such as the ImagePlus Application Library Services (MIS) system. This term can also refer to the processing of unit-of-work data through a user-written statistical program.

string. The data records representing a group of items, for example, an I-string, a D-string, or an M-string. See related definitions for details.

string segment map (SSM). One of three types of segment maps in CPCS-I. Each string in the system is associated with a string segment map. Each bit in a map represents a segment of direct access storage.

string status block (SSB). This CPCS-I control block is maintained by the MDS programs for every open string.

STV. Subtotal voucher.

subcomponent. Functional subset of a component where subsetting is appropriate based on data use, logic flow, or other factors relating to modules.

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

subsequent pass. A pass on which previously sorted items are resorted for further distribution.

subset. A defined portion of an entry, indicated by one or more tracer groups.

subset processing. Processing a portion of an entry beyond the document-entry step before the whole entry is run through the document processor.

subset string. A predefined group of data records that represents a portion of the physical items in an entry. A subset string can contain multiple tracer groups.

substitution. See *misread.*

subtotal voucher (STV). An optional UK document that can be inserted into a batch of documents to mark the point at which a cumulative subtotal is printed on the accompanying remittance list.

supervisor. (1) An MVS term used to refer to the system nucleus in internal storage. (2) A person responsible for operation of a financial institution area.

supervisory terminal. A special terminal or operating mode used in CPCS-I.

System Manager. A subsystem of CPCS-I that directs and controls the operations.

System Manager Online Functions (SMOF). A set of application-level tasks that monitor and modify the queues and databases of System Manager.

system reject pocket. The first physical pocket on the document processor. It is used by CPCS-I to hold machine and user-selected rejects.

System Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration of, networks.

T

tab key. A keyboard function key. The tab key causes the cursor to position to the next colon on the screen or to the top of the screen.

task. A CPCS-I application or function. User-written tasks must be in the CPCS-I BLDL list.

task list. The ordered list of tasks to be performed for a unit of work. It is determined by selecting the flow code for a given flow control record.

task status. A representation of what will happen, what is happening, or what happened during processing of this unit of work. Can be pending, ready, or complete. See related definitions for details.

total system. A system in which the computer is used for all phases of an operation.

tracer. A check-processing document used to provide pass-to-pass control.

tracer group. A grouping of documents between sets of tracers for control purposes. If subset processing is

in operation, this tracer group normally becomes a unit of work that can be processed independently of other units of work within that entry.

tracer ID. The tracer group and slip numbers corresponding to a tracer slip.

transaction code. A UK term for the 2-digit field that identifies debit, credit and control document types (similar to the Process Control Field in the US). A blank transaction code is a valid identifier for a check.

transit. The sorting of checks to external destinations. See also *out-clearing* and *outwork.*

U

unit of work (UOW). A logical entity that the System Manager uses to track a piece of work through CPCS-I. It can be informational or functional. See also *functional unit of work.*

UOW. Unit of work.

UOW status. This status represents the state of a unit of work and its associated string. Can be pending, ready, or complete.

V

Virtual Storage Access Method (VSAM). An access method for indexed or sequential processing of fixed or variable-length records on direct access storage devices.

Virtual Telecommunications Access Method (VTAM). A set of programs that control the communication between terminals and application programs.

VSAM. See *Virtual Storage Access Method.*

VTAM. See *Virtual Telecommunications Access Method.*

W

warm start. An initiation of the CPCS-I system, causing the contents of the MDS and the control data sets to be retained. A warm start is generally used for restarting CPCS-I after a normal ending.

WD (wrongly delivered). A UK term for items (debits or credits, not DCVs) that have been dispatched to the wrong bank. They are returned rather than redirected.

XREC. The dynamic control block that maps the string data at various points in the system. It cross-records or maps the string as it is in the data base, or as it is in the data space.

work. Any document or group of documents that CPCS-I processes.

work flow. An ordered list of tasks for a specific CPCS-I string. Each CPCS-I string must have a work flow.

Z

zero-balancing. The procedure that ensures that generated totals for a group of items plus any documented errors minus the control total equals zero.

Numerics

3890/XP Document Processor. A document processor in the 3890/XP Series of document processors that can read and sort documents at a rate of up to 2400 documents per minute.

3890/XP Series document processors. A series of high-speed document processors that can read and sort up to 1000, 1700, or 2400 documents per minute. These document processors include the IBM 3890/XP Document Processor, the IBM 3891/XP Document Processor, and the IBM 3892/XP Document Processor.

3891/XP Document Processor. A document processor in the 3890/XP Series of document processors that can read and sort documents at a rate of up to 1700 documents per minute.

3892/XP Document Processor. A document processor in the 3890/XP Series of document processors that can read and sort documents at a rate of up to 1000 documents per minute.

3892/XP Power Encoder Feature. An optional device that can be attached to the 3892/XP Document Processor to encode the MICR codeline field on a document.

99 M-string. See *balanced M-string.*

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Communicating Your Comments to IBM

Check Processing Control System
International MVS/ESA™
Customization Guide
Release 1

Publication No. SC31-2943-03

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
International MVS/ESA™
Customization Guide
Release 1

Publication No. SC31-2943-03

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 _____

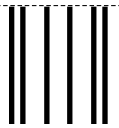
Company or Organization _____

Phone No. _____

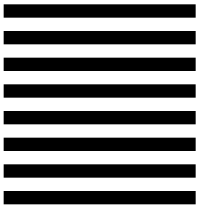
Readers' Comments — We'd Like to Hear from You
SC31-2943-03

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

SC31-2943-03

Cut or Fold
Along Line

Program Number: 5799-FKT

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC31-2943-03