z/OS
Version 2 Release 4

MVS Device Validation Support

SA38-0697-40

Note

Before using this information and the product it supports, read the information in "Notices" on page 99.

This edition applies to Version 2 Release 4 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2019-07-10

Contents

Figures

Tables

About this document

This document contains information that you need to write installation-supplied UIMs.

Who should use this document

This document is intended for system programmers who are responsible for writing installation-supplied UIMs. The user must know the hardware and software configuration characteristics of the I/O unit that needs a UIM, and should be familiar with basic MVS™ concepts, with the Input/Output Configuration Program (IOCP), and with HCD.

z/OS information

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

To find the complete z/OS® library, go to IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xi.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Knowledge Center function
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS MVS Device Validation Support, SA38-0697-40
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for z/OS Version 2 Release 4 (V2R4)

This information contains no technical changes for this release.

Summary of changes for z/OS for Version 2 Release 3 (V2R3)

This information contains no technical changes for this release.

Summary of changes for z/OS Version 2 Release 2

The following information is new, changed, or deleted in z/OS Version 2 Release 2 (V2R2).

Deleted

The following information has been removed:

- Entries for UIMs CBDUS021 and CBDUS035 for GAM/SP2 have been removed from the table in Appendix C, “IBM-supplied UIMs,” on page 79.

z/OS Version 2 Release 1 summary of changes

See the Version 2 Release 1 (V2R1) versions of the following publications for all enhancements related to z/OS V2R1:

- *z/OS Migration*
- *z/OS Planning for Installation*
- *z/OS Summary of Message and Interface Changes*
- *z/OS Introduction and Release Guide*

Chapter 1. Introduction

What is HCD?

Hardware Configuration Definition (HCD) is a z/OS component that supports you in defining both the operating system configuration and the processor hardware configuration of a system.

HCD validates the data you enter and checks it for consistency and completeness. Because HCD performs the check when the data is defined rather than when the device is accessed, inconsistencies can be corrected right away and unplanned system outages resulting from inconsistent definitions can be avoided.

The configuration data can then be used to POR/IPL or dynamically reconfigure your system. Dynamic reconfiguration management is the ability to activate a new I/O configuration during normal processing and without the need to perform a POR or IPL of the system.

For more information about HCD, see *z/OS HCD User's Guide*.

What are HCD Unit Information Modules?

The HCD unit information modules (UIMs) are a set of modules, apart from HCD, that describe the characteristics of a device, control unit, and ESCON director, supported by z/OS MVS or VM. (From now on, this book will use the term I/O unit for all types of I/O equipment, such as device, control unit, and ESCON director.) UIMs are involved in the validation of user input to HCD. They are also used at IPL or dynamic activation time to build the unit control blocks (UCBs). Only I/O units that are supported by UIMs can be configured with HCD and included in the IPL process.

IBM supplies a set of UIMs listed in Appendix C, “IBM-supplied UIMs,” on page 79. IBM-supplied UIMs are provided with HCD and with the device support code that you have installed. The UIMs provided with the device support code define the device values for z/OS MVS systems. The UIMs provided with HCD complement the device values for VM systems. Use the HCD ***Query supported hardware and installed UIMs*** function to display a list of:

- Supported I/O devices
- Supported control units
- Supported switches (ESCON directors)
- UIMs

In addition, you can use the HCD batch utility “Print a Configuration or Supported Hardware Report” to print the actual status of hardware supported in your installation:

- Processors
- Control units
- Devices

Converting UIMs Running with MVSCP

If you have installation-written UIMs that currently run with MVSCP, you need to convert those UIMs to run with HCD because the requirements for an HCD UIM differ from those for an MVSCP UIM.

The major changes are:

- Rename the UIM from CBPUCxxx to CBDUCxxx
- Recode the UIM to use the new/changed service routines and macros

- Code the UIM data table (UDT).
- Add help panels
- Install the UIM and associated UDT in SYS1.NUCLEUS or the UIM library defined in the UIM_LIBNAME statement in the HCD profile. For IPL, the UIMs and associated UDT must be installed in SYS1.NUCLEUS; for testing purposes, you can install them in the UIM library defined in the HCD profile.

Definition of I/O Units without UIM

If your configuration contains an I/O unit that is not supported by any supplied UIM and that cannot be substituted by an IBM device type, you can use the NOCHECK or DUMMY control unit and the DUMMY device. These control units and devices do not provide the full amount of HCD validation.

NOCHECK control unit
A control unit defined as NOCHECK allows any specifications; for example, any protocol can be specified or any device can be connected to it.

DUMMY control unit
A control unit defined as DUMMY can connect only DUMMY devices.

DUMMY device
A device defined as DUMMY is treated as a unit record device.

If your configuration contains an I/O unit that is not supported by any supplied UIM, and the NOCHECK or DUMMY control units or the DUMMY device cannot be used because certain validations should be performed at definition time, you need to provide an installation-written UIM. The following chapters explain how to write your own UIM.

Chapter 2. UIM Processing

Overview

This chapter describes the types of requests that a UIM processes:

- Initialization
- Validate device parameter
- Validate device feature
- Validate device number
- Validate device unit address
- Build Device Feature Tables
- End of data processing

An HCD unit information module (UIM) is a program (within the respective device support code) that contains information related to the I/O unit. This information is used when validating the I/O unit. Each UIM recognizes and processes the values coded for its I/O unit in the I/O configuration. Not all UIMs support single I/O units; a UIM may define a grouping of several related I/O units.

Request Sequence to the UIM

UIMs are requested for several HCD functions and during IPL of z/OS. During HCD initialization and processing and for IPL or dynamic activation, the following information is requested from the UIM:

- Allocation information about all generic devices
- Unit information for the devices
- Control unit information

During processing, HCD constructs an internal device record (IODV) from the information gathered from the panel prompts and calls the UIM to perform the following validation:

- Parameter checking
- Feature context checking
- Device number checking
- Unit address checking

During system IPL, dynamic activation, or HCD report generation, a device feature table (DFT) is built. The DFT contains information used to build the unit control blocks (UCBs), which are required to IPL and to produce device reports.

The following three figures illustrate the different UIM calls, which depend on the process that is taking place.

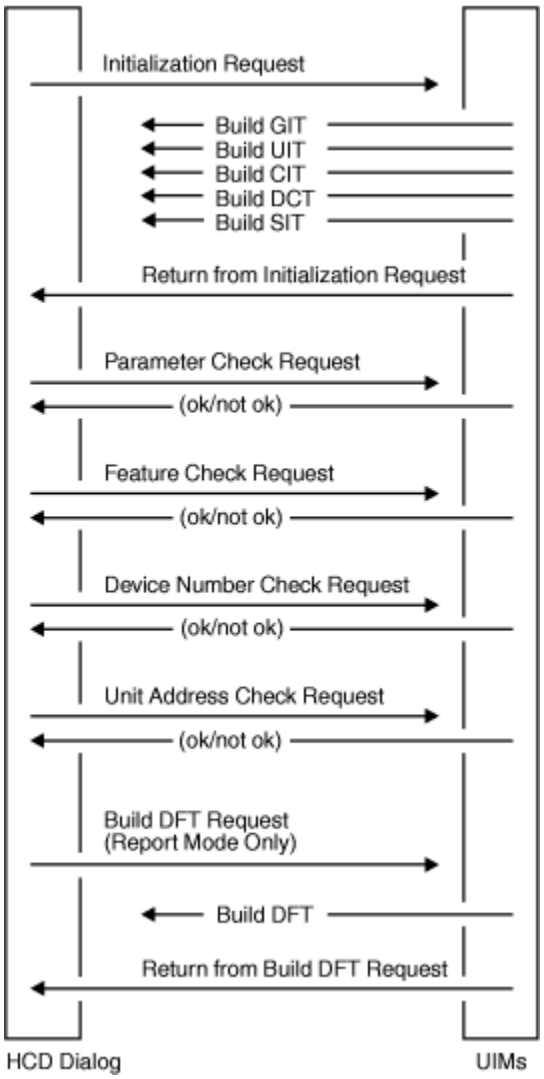

Figure 1. UIM calls between the HCD Dialog and UIMs

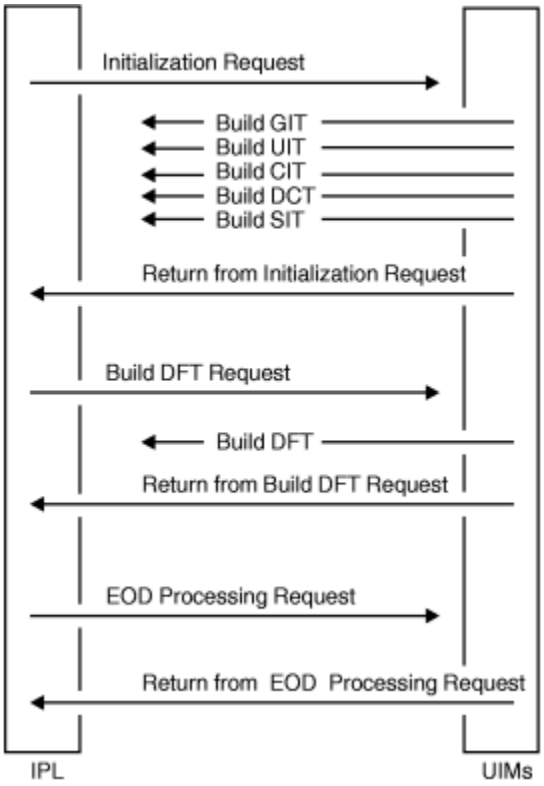

Figure 2. UIM calls between IPL and UIMs

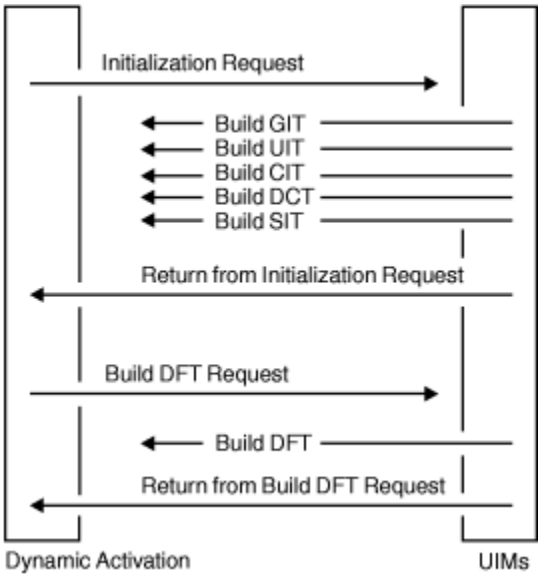

Figure 3. UIM calls between Dynamic Activation and UIMs

Summary of UIM requests

Table 1 on page 6 summarizes what type of request is called for what kind of process, including the caller of each type of request.

Table 1. Type of UIM requests

Type of request to be processed by a UIM	UIM callers				
	HCD initialization	**HCD device valid**	**HCD report function**	**IPL process**	**MVS dynamic activate**
Initialization	Yes		Yes	Yes	Yes
Validate device parameter		Yes			
Validate device feature		Yes			
Validate device number		Yes			
Validate unit address		Yes			
Build device feature tables			Yes	Yes	Yes
End of data processing				Yes	

Subsequent topics describe the structure of a UIM and the processing for each type of request.

UIM Structure

Input to the UIM is in the UIM Communication Area (UCA). The UCA contains all relevant data for interfacing with the UIM. In particular, the UCA contains the request type (UCAUIMRT). The request type tells the UIM what to do. There are several request types, and the UIM does not need to support them all. However, the UIM must be prepared to accept and tolerate any request type, even the ones that might be introduced at a later time. The initialization request is the only one that is mandatory.

The following request types are defined:

UCARINIT
Initialization request

UCARPARM
Validate device parameters

UCARFEAT
Validate device features

UCARADDR
Validate device number

UCARUADD
Validate unit address (UA) of device

UCARDTFB
Build device feature table

UCAEOD
Perform end of data (EOD) processing

Except for the initialization and EOD request call, the UCA points to an internal text record, called an I/O device text record (IODV), as shown in Figure 4 on page 7. The IODV contains all relevant information about the device to be validated or processed.

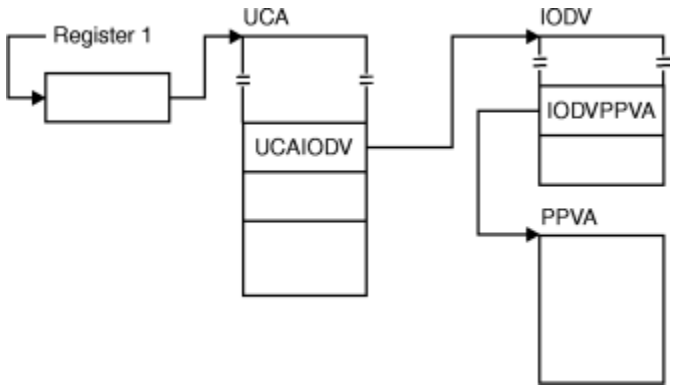

Figure 4. Accessing IODV

On entry, the UIM must follow the standard linkage conventions and save the caller's registers and establish its own savearea (because the UIM calls other UIM service routines), pointed to by register 13. Next, the UIM must push an entry on the diagnostic stack. This is done by defining a diagnostic stack entry by means of the CBDZDIAG macro and adding the entry on top of the diagnostic stack by means of the CBDIPPDS macro. Then, the UIM can examine the request code in the UCA to determine what to do.

On exit, the UIM must ensure that the correct return code is set in field UCARETC in the UCA and then remove the diagnostic entry from the stack by means of the CBDIPPDS macro.

Initialization Request

A UIM is called with a request for initialization by

- HCD initialization
- IPL processing
- Dynamic activation

As already mentioned, a UIM must be able to handle an initialization request. During this call, the UIM "registers" to HCD any control unit type, I/O device type, or ESCON director type (switch) it defines. Only I/O units that are registered to HCD are later accepted as valid.

- To register a control unit type, the UIM must set up the control unit information parameters (CIP) and then call the CIT build routine. The CIP contains descriptive information about the control unit, such as maximum and default values, as well as the list of devices that can be attached to the control unit. The entry point address of the CIT build routine is contained in the UCA. See “CIT Build Routine” on page 15 for details on how to call the CIT build routine.

  The UIM must repeatedly call the CIT build routine for each control unit type it defines.
- To register an ESCON director type, the UIM must set up the switch information parameters (SIP) and then call the SIT build routine. The entry point address of the SIT build routine is contained in the UCA. The SIP contains descriptive information about the switch, such as valid port range and attachment information. See “SIT Build Routine” on page 19 for details on how to invoke the SIT build routine.

  The UIM must repeatedly call the SIT build routine for each switch type it defines.
- Prior to registering any I/O device, the UIM must register the generic device type of the devices. This is done by setting up the generic information parameters (GIP) and then calling the GIT build routine. See “GIT Build Routine” on page 17 for details on how to call the GIT build routine. The UIM must repeatedly call the GIT build routine for each generic device type.
- To register an I/O device type, the UIM must set up the unit information parameters (UIP) and then call the UIT build routine. The UIP contains relevant information about the device. In detail, the UIP consists of multiple sections:
  - A general section containing device type/model and other data that define the physical characteristics of the device as well the list of devices that are look-alikes.
  - An MVS section containing the MVS parameters and features, default values, and so forth.

See “UIT Build Routine” on page 18 for details on how to call the UIT build routine.

The UIM must repeatedly call the UIT build routine for each device type/model it defines.

If the device is a DASD device, the UIM must also define the physical DASD characteristics of the device, such as number of cylinders and tracks. This is done by setting up the device characteristics parameters (DCP) and then calling the DCT build routine. See “DCT Build Routine” on page 18 for details on how to call the DCT build routine.

Device Parameter Validation Request

If the UIM indicates, through its unit information table (UIT), that device parameters can or must be specified, HCD calls the UIM with a request to validate all specified device parameters.

Device parameters are either common or private. Common parameters apply to all devices in the system, such as the DYNAMIC parameter (see Appendix D, “Summary of Device Information,” on page 83 for a list of common parameters). Private parameters apply only to the device they are defined to, such as the LIBRARY parameter of a 3480 or 3490 device type. When you write an HCD UIM, you can define common and private parameters for the device supported by that UIM.

If called to perform the device parameter validation for the device(s) described in the IODV, the UIM must validate the specified parameters. The UIM does not need to check if required parameters are present; this has been already ensured by HCD validation. HCD validates the following definitions:

- The parameter is supported for the device type
- Required parameters are specified
- The type of private parameter value is correct - hex, decimal, value within defined decimal range
- If a selection list is specified for a parameter value, the value is contained in the selection list.

The UIM must validate only the parameter value. The IODV contains a bitstring (IODVPARM) that indicates which parameters are specified. If the bit is on, the corresponding parameter is present. The position of the bit, representing a parameter, in the bitstring is given by the parameter identifier. See the CBDZUDT macro for the mapping of the parameter identifiers.

The values of the common parameters, which are parameters with identifiers in the range from 1 - 32, are contained in the IODV. HCD has already ensured that the parameter value is correct from the syntax point of view. For example, the SETADDR parameter must be a decimal number; in this case, HCD has already verified that the user's specification for SETADDR is decimal and can fit in the field provided in the IODV for the SETADDR parameter.

Common parameters for which a selection list was specified at "registration" time do not need to be validated. HCD has already verified that the specified value is one of the choices in the parameter selection list.

If a parameter can contain only Yes or No, it does not require any additional validation logic. HCD has already verified that either Yes or No was specified. The IODV contains a flag for each Yes/No parameter that, if set, indicates that "Yes" was specified for the parameter.

Private parameters require a slightly different handling. The values for private parameters, which are parameters with identifiers in the range from 33 - 64, are contained in the private parameter value array (PPVA) rather than the IODV. The PPVA is pointed to by IODVPPVA in the IODV. The PPVA is an array of 64 entries, one for each possible parameter. The parameter identifier can be used as an index into the PPVA. Like the common parameters, the private parameters are already verified for correct syntax according to the parameter syntax description in the UDT. The format of the parameter value stored in the PPVA depends on the type of the parameter defined in the UDT.

- If the parameter type is numeric, its value is stored in a 4-byte binary field (fullword).
- If the parameter type is hexadecimal, its value is stored in a field with a length specified in the UDT, right-justified, converted to binary and filled with leading zeros.
- If the parameter type is either alphanumeric or character, its value is stored in a field with a length specified in the UDT, left-justified, and padded with blanks.

- If the parameter type is "YESNO", its value is either PPVAYES or PPVANO, stored in a 1-byte character field.
- If the parameter type is none of these, its value is stored as is in a field with the length specified in the UDT and padded with blanks.

If the UIM detects an error, it must

- Indicate which parameter was incorrectly specified (using the parameter identifier) in the UCAPID field in the UCA.
- Issue an error message, explaining what was wrong, by means of the CBDIMSG macro.
- Set the error return code in the UCA

If the parameter validation requires additional information that is not supplied in the IODV, the UIM might call the device lookup routine to get information about:

- All devices attached to the same control unit
- All devices grouped together by means of the same PCU value
- Control unit data for a particular control unit (type/model)
- Device data for a particular device, identified by its device number.

See “Device Lookup Routine” on page 20 for more details about how to call the device lookup routine. The device lookup routine returns device information for just one device at a time. The device data is returned in the form of an IODV without the PPVA. The UIM must repeatedly call the Device Lookup Routine using the same, unmodified DEVL parameter list to pick up the data for other devices.

Note: The device lookup routine can also be called while validating the device features, the device number, or the device unit address. The routine cannot be called at DFT build time.

If the generic device type varies depending on the specification of certain parameters or features, the UIM might specify a new generic device type for the device by calling the generic update routine and passing the new generic name, a generic name that must have been previously defined as a valid generic. See “Generic Update Routine” on page 21 for more details on how to call the generic update routine. Calling the generic update routine is only allowed when the UIM indicated in the UIP at initialization time that the generic device type might change.

Device Feature Validation Request

If called to perform the device feature validation for the device(s) described in the IODV, the UIM must validate whether two specified features are mutually exclusive or the presence of one feature requires another feature to be specified. The UIM does not have to validate whether the specified feature is supported or compatible, HCD already does that validation.

Whether or not a feature is specified is indicated in the IODVFEAT bitstring, where each feature is represented by one bit. If the bit is on, the feature is specified. The order of the bits is determined by the order of the feature definitions in the UIM's associated UDT.

If the UIM detects an error, it must

- Indicate in field UCAPID in the UCA that the error occurred while checking the device features.
- Indicate in field UCAPPOS in the UCA which feature was incorrectly specified (by specifying the offset in the bitstring).
- Issue an error message, explaining what was wrong, by means of the CBDIMSG macro.
- Set the error return code in the UCA

The UIM might also set default features by setting the appropriate bit in the feature bit string (IODVFEAT). This has the same effect as if the HCD user had specified the feature.

Device Number Validation Request

If called to perform device number validation for the device(s) described in the IODV, the UIM might validate the device number for special rules (for example low order digits=zero). Note, that the UIM might

be called for a range of devices. The range value is contained in the IODV (IODVNBRD). Each device number in the range must be checked for correctness.

If the UIM detects an error, it must:

- Use the CBDIMSG macro to issue an error message that gives information about the erroneous device
- Set the error return code in the UCA.

Device Unit Address Validation Request

If called to perform the unit address validation for the device(s) described in the IODV, the UIM might validate the unit address for special rules (for example, low order digit=zero). Note, that the UIM might be called for a range of devices. The range value is contained in the IODV (IODVNBRD). Each unit address in the range must be checked for correctness. The unit address of the first device is contained in IODVUNIA.

If the UIM detects an error, it must:

- Use the CBDIMSG macro to issue an error message that gives information about the erroneous device
- Set the error return code in the UCA.

Device Feature Table Build Request

Device feature tables (DFTs) are required by:

- The IPL and dynamic activation process to build unit control blocks (UCBs) for each device contained in the configuration
- The HCD report function to generate the device report

If called to perform the DFT build for the device(s) described in the passed IODV, the UIM must set up the device feature parameters (DFP) and then call the DFT build routine. The entry point address of the DFT build routine is contained in the UCA. The DFP contains information used to construct the UCB for the device. See "DFT Build Routine" on page 16 for details on how to call the DFT build routine.

The UIM might be called with the DFT build request for a range of devices; the range value (IODVNBRD) is contained in the IODV. In this case, the UIM must call the DFT build routine repeatedly for each device in the range.

For a parallel access volume, the UIM is called only for the base device number. At that time, the UIM must also build a DFT for each alias device number.

Because a group of devices might share the same

- Device dependent segment
- Device class extension
- Device dependent extension

the DFT build routine returns the addresses of the listed areas in the UCA. These areas might then be updated during the end-of-data request.

End-of-data Request

For IPL, the UIM is called with this request only if the UCAEODAT flag is set in the UCA on return of the initialization request. This is only of interest when the device dependent segment, device dependent extension, or device class extension of the UCB needs to be updated for a group of devices sharing the same data.

In this case, the UIM must collect the necessary data while handling the DFT build requests for all devices defined for the operating system in the IODF.

The UIM must not issue any message while handling this request.

Chapter 3. Writing a UIM

Overview

Before writing a UIM, you should be familiar with *z/OS HCD User's Guide*, which explains:

- How to use the HCD facility ***Query supported hardware and installed UIMs***
- How to use the batch utility "Print Supported Hardware Report"
- How to define control units, devices, and ESCON directors in HCD.

This chapter includes the following information:

- UIM data areas
- UIM environment
- UIM recovery
- Steps to write a UIM
- Installing a UIM
- UIM service routines
- UIM macros
- UIM data tables (UDTs)
- Testing UIMs

Appendix A, "Sample of a Unit Information Module (UIM)," on page 45 and Appendix B, "Sample of a Unit Data Table (UDT)," on page 73 shows you a sample of a UIM and UDT with detailed explanation. The samples are members of SYS1.SAMPLIB(CBDSUIM) and (CBDSUDT).

UIM Data Areas

There are two control blocks, external to the UIM, that a UIM must reference:

- UIM communications area (UCA) — data area CBDZUCA
- IODEVICE internal text record (IODV) — data area CBDZITRH

The other data areas and parameters lists that a UIM uses are contained within the UIM itself.

See *z/OS MVS Data Areas* in the z/OS Internet library (www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) for mappings of these data areas.

UIM Communications Area (UCA)

The UCA contains information that HCD uses to communicate with the UIM, such as:

- The request the UIM is called with
- Error information provided by the UIM
- The entry points of the UIM service routines
- The address of the internal text record (IODV)
- The return code set by the UIM

The UCA points to:

- The CIT build routine
- The DCT build routine
- The DFT build routine

- The generic update routine
- The GIT build routine
- The SIT build routine
- The UIT build routine

IODEVICE Internal Text Record (IODV)

The IODV (IODEVICE internal text record) represents an HCD device definition. It contains the parameters and features that were specified for the device. CBDZITRH maps the IODV.

The IODV is used for all requests except initialization and end-of-data.

UIM Environment

UIMs are invoked in task mode and in problem state. A UIM must call only UIM services (UIM service routines and UIM macros), not system services, because system services are not yet available when the UIMs are called at IPL time.

Link-edit UIMs with AMODE(31) and RMODE(ANY). UIMs must not change to 24-bit addressing mode.

The same copy of a UIM is used throughout HCD processing. Thus, a UIM can store information within itself and retain this information for HCD processing.

UIMs must use the standard register save area conventions. The UIM must set register 13 to point to its own register save area before calling any UIM service routines or before issuing the CBDIPPDS or CBDIMSG macro.

Entry to an HCD UIM

Upon entry, the UIM must:

- Save the contents of the input registers.
- Set the UIM base register.
- Chain the save areas.
- Set register 13 to point to the save area contained within the UIM.
- Establish addressability to the UCA and IODV.
- Issue the CBDIPPDS macro with parameter PUSH to put an entry on the diagnostic stack.

Registers on Entry to an HCD UIM

Upon entry to a UIM, the registers are defined as follows:

Register 0
Undefined

Register 1
Pointer to a fullword containing the UCA address

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
UIM entry point address

Exit from a UIM

Upon exit, the UIM must:

- Issue the CBDIPPDS macro with parameter POP to remove the UIM's entry from the diagnostic stack.
- Restore the caller's registers.
- Return to the caller.

Registers on Exit from an UIM

Upon exit from a UIM, the registers are defined as follows:

Register 0-15
Restored

UIM Recovery

Do not code a recovery routine in any UIM. Instead, use the CBDZDIAG and CBDIPPDS macros to provide diagnostic information.

A UIM must not establish an ESTAE (extended subtask abend exit) routine to provide diagnostic information in the event that it has an abend. Rather, a UIM must:

1. Specify the diagnostic information in an HCD diagnostic stack entry, using the CBDZDIAG macro. (See "CBDZDIAG Definition Macro" on page 28.)
2. Use the CBDIPPDS macro to put (push) the entry onto the diagnostic stack on entry. (See "CBDIPPDS Executable Macro" on page 25.)
3. Use the CBDIPPDS macro to remove (pop) the entry from the diagnostic stack on exit.

Steps to Write a UIM

To write a UIM, you need to:

- Be familiar with the characteristics of your I/O unit
- Decide what validation checks are required for your I/O unit
- Specify a name for your UIM
- Create your UIM. See Appendix A, "Sample of a Unit Information Module (UIM)," on page 45 and change it according to your requirements.
- Create your UDT. See Appendix B, "Sample of a Unit Data Table (UDT)," on page 73 and change it according to your requirements.
- Write the help support.

Characteristics of your I/O Unit

Obtain from the I/O unit developer the values that describe the characteristics of the I/O unit. To describe control units in the UIM, you need values such as I/O concurrency level, channel protocol, and channel attachment capability. To describe devices in the UIM, you need values such as UCB type, generic preference value, device parameters, and device features. See the sample UIM in Appendix A, "Sample of a Unit Information Module (UIM)," on page 45 for a list of required parameters.

Naming a UIM

Installation-supplied UIMs must have member names of CBDUCxxx, where xxx is a decimal number from 001 to 256. You can use the HCD facility ***Query supported hardware and installed UIMs*** or the Supported Hardware report to find an unused number for a new UIM.

HCD loads UIMs CBDUC001-CBDUC256 and UIMs CBDUS001-CBDUS256 during its initialization.

When option VM_UIM=YES is active in the HCD profile (which is the default), HCD also loads UIMs CBDUC257-CBDUC512 and CBDUS257-CBDUS512.

When option VM_UIM=NO is active in the HCD profile, the range 257 to 512 is not loaded or used. HCD considers range 001-256 as MVS UIMs and 257-512 as VM UIMs.

Using the Sample UIM

A sample UIM is provided in SYS1.SAMPLIB(CBDSUIM). Use this sample as the basic structure for your HCD UIM. SYS1.SAMPLIB(CBDSUIM) contains:

- Overview of contents
- Sample code for DASD units with detailed comments
- The JCL to assemble and link-edit the UIM

Be sure to include the correct SYSIN and SYSLMOD data set names. Define the SYSLMOD data set as SYS1.NUCLEUS.

Note: To test the UIM, do not link it into SYS1.NUCLEUS. Instead, before testing the UIM, link it to another library and concatenate that library to the HCD load libraries. Specify that library on the UIM_LIBNAME parameter in the HCD profile statement.

UIM_LIBNAME=Name of data set containing the UIMs
All UIMs (and UDTs) are loaded from the specified data set (SYS1.NUCLEUS is the default)

UIM_LIBNAME=*
The UIMs are contained in the HCD load libraries. In this case, the data set containing the new UIM and SYS1.NUCLEUS containing the existing UIMs must be concatenated to the HCD load libraries using STEPLIB/JOBLIB statements.

Test the UIM as described in “Testing UIMs” on page 36.

Installing a UIM

UIMs must reside as separate members in SYS1.NUCLEUS or the UIM library defined in the HCD PROFILE statement. For IPL, the UIMs and UDT must reside in SYS1.NUCLEUS; for testing purposes you can install them in the UIM library defined in the HCD profile.

UIM Service Routines

The following table identifies each HCD service routine and its function used at initialization time.

Service Routine Name	Function of the Routine
CIT Build Routine	Builds control unit information tables (CITs). There is one CIT for each control unit type and model supported by a UIM. The CIT build routine resides in the HCD load library.
DFT Build Routine	Builds the device features tables (DFTs). There is one DFT for each device number in the I/O configuration. The DFT build routine resides in SYS1.NUCLEUS.
GIT Build Routine	Builds generic information tables (GITs). There is one GIT for each generic device type supported by the UIMs. The GIT build routine resides in SYS1.NUCLEUS.
DCT Build Routine	Builds device characteristics tables (DCTs). There is one DCT for each DASD device type and model supported by the UIMs. The DCT build routine resides in SYS1.NUCLEUS.

Service Routine Name	Function of the Routine
SIT Build Routine	Builds switch information tables (SITs). There is one SIT for each ESCON director type and model supported by the UIMs. The SIT build routine resides in the HCD load library.
UIT Build Routine	Builds unit information tables (UITs). There is one UIT for each device type and model supported by the UIMs. The UIT build routine resides in SYS1.NUCLEUS.

The following table identifies each HCD service routine and its function used at validation time.

Service Routine Name	Function of the Routine
Device Lookup Routine	Obtains information about other devices attached to one control unit. The device lookup routine resides in the HCD load library.
Generic Update Routine	Updates the name of a generic device type as a result of features and parameters specified for that device. The generic update routine resides in the HCD load library.

Descriptions of the service routines follow.

CIT Build Routine

To build the CITs, HCD or IPL calls the UIMs for initialization. For each control unit type that a UIM defines, it must build control unit information parameters (CIP) and call the CIT BUILD Routine. A separate CIT is built for each control unit type.

A UIM invokes the CIT build routine, in 31-bit addressing mode, by using a BALR instruction. Use the standard register save area conventions. The address of the CIT build routine is in the field UCACITP in the UCA.

CIT Build Routine Input Parameters

A UIM provides the input to the CIT build routine in the control unit information parameters (CIP). The CIP resides in the UIM and is mapped by CBDZCIP.

Registers on Entry to the CIT Build Routine

Register 0
Undefined

Register 1
Pointer to a two-word parameter list

- Word 1 - Address of the UCA
- Word 2 - Address of the CIP

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
The CIT build routine entry point address

Registers on Exit from the CIT Build Routine

Register 0-14
Restored

Register 15
Return code

Return Codes

RC 0
No errors detected, CIT built

RC 8
Errors detected, no CIT built, message written

DFT Build Routine

The IPL process, dynamic activate, or the HCD report function call the UIM with a request to build DFTs. The UIM builds the device feature parameters and calls the DFT build routine to finally build the device features tables.

A UIM must call the DFT build routine once for each DFT to be built. A DFT must be built for each device number defined to an operating system in the IODF. For a parallel access volume, a DFT must be built for the base device number and each of its alias device numbers.

A *multiple exposure device* is a device that can be *allocated* by a single device number but can be *accessed* by multiple device numbers from the system.

A *parallel access volume* can handle multiple, concurrent I/O requests to a single volume from the same system. Each base and alias unit control block (UCB) for a parallel access volume is represented by a device number in HCD. You specify the base device number for allocation. The I/O request identifies the base UCB and the system uses the base UCB or one of its alias UCBs depending on availability.

A DFT must be built for each device number defined in the I/O device internal text record (IODV).

To call the DFT build routine within a UIM, use a BALR instruction in 31-bit addressing mode. Use standard register save area conventions. The UCADFTP field in the UCA contains the address of the DFT build routine.

DFT Build Routine Input Parameters

A UIM provides the input to the DFT build routine in the device features parameters. The device features parameters reside in the UIM and are mapped by CBDZDFP.

Registers on Entry to the DFT Build Routine

Register 0
Undefined

Register 1
Pointer to a two-word parameter list

- Word 1 - Address of the UCA
- Word 2 - Address of the device features parameters

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
The DFT build routine entry point address

Registers on Exit from the DFT Build Routine

Register 0-14
Restored

Register 15
Return code

Return Codes

RC 0
No errors detected, DFT built

RC 8
Error detected, no DFT built, message written

GIT Build Routine

To build GITs, HCD or IPL calls all UIMs at initialization time. For each generic device type that a UIM defines, it must build generic information parameters (GIP) and call the GIT build routine. A GIT is built for each generic device type. The UIM must build GITs before building the UIT.

A UIM invokes the GIT build routine, in 31-bit addressing mode, by using a BALR instruction. Use the standard register save area conventions. The address of the GIT build routine is in the field UCAGITP in the UCA.

GIT Build Routine Input Parameters

A UIM provides the input to the GIT build routine in the generic information parameters (GIP). The GIP resides in the UIM and is mapped by CBDZGIP.

Registers on Entry to the GIT Build Routine

Register 0
Undefined

Register 1
Pointer to a two-word parameter list

- Word 1 - Address of the UCA
- Word 2 - Address of the GIP

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
The GIT build routine entry point address

Registers on Exit from the GIT Build Routine

Register 0-14
Restored

Register 15
Return code

Return Codes

RC 0
No errors detected, GIT built

RC 8
Error detected, no GIT built, message written

UIT Build Routine

To build the UITs, HCD or IPL calls all UIMs at initialization time. For each device type or model supported by the UIM the UIM must build unit information parameters (UIP) and call the build UIT routine.

A UIM invokes the UIT build routine, in 31-bit addressing mode, by using a BALR instruction. Use the standard register save area conventions. The address of the UIT build routine is in the field UCAUITP in the UCA.

UIT Build Routine Input Parameters

A UIM provides the input to the UIT build routine in the unit information parameters (UIP). The UIP resides in the UIM and is mapped by CBDZUIP.

Registers on Entry to the UIT Build Routine

Register 0
Undefined

Register 1
Pointer to a two-word parameter list

- Word 1 - Address of the UCA
- Word 2 - Address of the UIP

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
The UIT build routine entry point address

Registers on Exit from the UIT Build Routine

Register 0-14
Restored

Register 15
Return code

Return Codes

RC 0
No errors detected, UIT built

RC 8
Error detected, no UIT built, message written

DCT Build Routine

To build the device characteristics tables (DCT), HCD and IPL call all UIMs at initialization time. For each DASD device supported by the UIM, the UIM must build device characteristics parameters (DCP) and call the DCT build routine.

A UIM invokes the DCT build routine in 31-bit addressing mode, by using a BALR instruction. Use the standard register save area conventions. The address of the DCT build routine is in the field UCADCTP in the UCA.

DCT Build Routine Input Parameters

A UIM provides the input to the DCT build routine in the device characteristics parameters (DCP). The DCP resides in the UIM and is mapped by CBDZDCP.

Registers on Entry to the DCT Build Routine

Register 0
Undefined

Register 1
Pointer to a two-word parameter list

- Word 1 - Address of the UCA
- Word 2 - Address of the DCP

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
The DCT build routine entry point address

Registers on Exit from the DCT Build Routine

Register 0-15
Restored

Return Codes

None

SIT Build Routine

To build switch information tables (SIT), HCD and IPL call all UIMs at initialization time. A UIM that defines switches must call the SIT build routine once for each type of ESCON director that it defines. A separate SIT is built for each ESCON director type.

A UIM invokes the SIT build routine, in 31-bit addressing mode, by using a BALR instruction. Use the standard register save area conventions. The address of the SIT build routine is in the field UCASITP in the UCA.

SIT Build Routine Input Parameters

A UIM provides the input to the SIT build routine in the switch information parameters (SIP). The SIP resides in the UIM and is mapped by CBDZSIP.

Registers on Entry to the SIT Build Routine

Register 0
Undefined

Register 1
Pointer to a two-word parameter list

- Word 1 - Address of the UCA
- Word 2 - Address of the SIP

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
The SIT build routine entry point address

Registers on Exit from the SIT Build Routine

Register 0-14
Restored

Register 15
Return code

Return Codes

RC 0
No errors detected, SIT built

RC 8
Error detected, no SIT built, message written

Device Lookup Routine

The UIM calls the device lookup routine to perform the following functions:

- Return all the devices grouped together by means of the PCU number (Applicable to graphic controllers only)
- Return all the devices that are attached to the same control unit
- Return the control unit information (type and model) of the control unit identified by its control unit number
- Return the device record for a given device number

The device information is returned in the format of an IODV record — one after the other. The UIM must provide a pointer to the IODV and data area.

The device lookup routine returns data in an I/O device internal text record (IODV) format. The DEVL parameter list contains a pointer to an area large enough to hold an IODV. The device lookup routine fills that area with device information in the IODV format.

To call the device lookup routine within a UIM, use a BALR instruction in 31-bit addressing mode. Use standard register save area conventions. The UCADEVP field in the UCA contains the address of the device lookup routine.

Device Lookup Routine Input Parameters

A UIM provides the input to the device lookup routine in the DEVL parameter list. The DEVL parameter list resides in the UIM and is mapped by CBDZDEVL.

Registers on Entry to the Device Lookup Routine

Register 0
Undefined

Register 1
Pointer to a two-word parameter list

- Word 1 - Address of the UCA
- Word 2 - Address of the DEVL parameter list

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
The device lookup routine entry point address

Registers on Exit from the Device Lookup Routine

Register 0-14
Restored

Register 15
Return code

Return Codes

RC 0
Device or control unit found.

RC 4
Device or control unit not found or no more devices or control units available matching the setup criteria

RC 8
Supplied data area too short (that means the storage area for the IODV record)

RC 12
Invalid function code passed

Generic Update Routine

A UIM calls the generic update routine only when the name of a generic device type is based on certain specified features or parameters. It is used to overwrite the standard generic name that is associated with the device type.

The generic update routine can update the generic name only during parameter or feature checking. Also, the UIP must indicate that the generic update routine can update the generic name as a function of device features.

Note: The IODV specifies device features and parameters. The UCA points to the IODV. Therefore, because the UIM passes the UCA to the generic update routine, that routine has access to device features and parameters.

To call the generic update routine within a UIM, use a BALR instruction in 31-bit addressing mode. Use standard register save area conventions. The UCAUGNP field in the UCA contains the address of the generic update routine.

Registers on Entry to the Generic Update Routine

Register 0
Undefined

Register 1
Pointer to a two-word parameter list

- Word 1 - Address of the UCA
- Word 2 - Address of an 8 byte field that contains the name of the generic device type

Register 2-12
Undefined

Register 13
Address of an 18-word save area

Register 14
Return address

Register 15
The generic update routine entry point address

Registers on Exit from the Generic Update Routine

Register 0-14
Restored

Register 15
Return code

Return Codes

RC 0
No errors detected, UIT updated with the new generic name

RC 8
Error detected, no UIT updated, message written

UIM Macros

Because the UIMs are called during IPL, they cannot issue system services. Therefore, the following UIM macros are made available. UIM macros consist of executable macros and definition macros.

UIM Executable Macros

The following executable macros are used by UIMs.

Executable Macro Name	Executable Macro Function
CBDIGETM	Obtains a contiguous virtual storage area
CBDIMSG	Issues a message.
CBDIPPDS	Puts an entry on or removes an entry from the HCD diagnostic stack.
CBDISIML	Generates either a similar device list or an attachable device list.

CBDIGETM Executable Macro

The CBDIGETM macro obtains a contiguous virtual storage area, and clears it to hexadecimal zero.

The syntax of the CBDIGETM macro is as follows:

```
[label]  CBDIGETM {UNCOND|COND}
                   ,LENGTH=length|(reg)
                  [,SUBPOOL={pool_number|(reg)}]
                  [,BNDRY={DBLWD|PAGE}]
                   ,REQ=UIM
                  [,ADDRESS=address_variable]
                  [,RELATED=related value]
```

label
Specifies the label name that the system generates in the first instruction of the macro expansion.

UNCOND
Specifies an unconditional storage request. If there is not enough virtual storage, the active task terminates abnormally. UNCOND is the default.

COND
Specifies a conditional storage request. If there is not enough virtual storage, this macro provides return code 4.

LENGTH
Specifies requested virtual storage length in bytes. Length can range from 1 byte to 16 megabytes, automatically rounded up to the next multiple of 8. If you use register notation, make sure the designated register contains the length.

SUBPOOL
Specifies the subpool number from which virtual storage is allocated. Valid numbers are between 0 and 127. If you use register notation, make sure the designated register contains the subpool number. If this macro specifies a subpool, it allocates storage from that subpool. Otherwise, it allocates storage from the subpool assigned to the job step.

BNDRY
Specifies requested alignment:

DBLWD
Specifies alignment on a double word boundary.

PAGE
Specifies alignment at the start of a virtual page (a 4K boundary).

Note: The default is DBLWD.

REQ
Must be *UIM* (specifies that a UIM is issuing the macro).

ADDRESS
Specifies the variable to contain the address of acquired storage.

RELATED
Specifies an optional character string that can identify related macro calls.

Registers used by the CBDIGETM macro:

Register 1
Parameter list address

Register 14
Linkage register

Register 15
Linkage register

Make sure the UIM has addressability to the UCA when issuing the CBDIGETM macro.

Note: You must also include the CBDZGETM definition macro in the UIM. The CBDZGETM definition macro maps the GETM parameter list. The CBDIGETM executable macro builds the GETM parameter list.

Return Codes

Return Code	Reason Code	Description	Message	Abend / Reason Code
GETMOK	GETMOK	Request completed successfully	-	-
GETMWARN	GETMSNAV	Storage not available (conditional only; on unconditional request, a system abend occurs)	CBDA056I	-
GETMTERM	GETMINVF	Invalid function code	CBDA011I	00F/00110011
	GETMINVS	Invalid Subpool number	CBDA011I	00F/00110012
	GETMLEN0	Zero length provided	CBDA011I	00F/00110015

Example

```
CBDIGETM   LENGTH=BUFFLEN,ADDRESS=BUFFADDR,REQ=UIM
        .
        .
        .
BUFFADDR   DC A(0)
BUFFLEN    DC F'80'
```

This example requests a dynamic storage area with a length of 80 bytes, with the address returned in variable BUFFADDR.

CBDIMSG Executable Macro

The CBDIMSG macro issues a message that appears on a terminal or in a message log.

The syntax of the CBDIMSG macro is as follows:

```
[label]  CBDIMSG MID=message_id
                 [,VARn=(variable,{H|B|C})]
                 [,SEV=severity]
                  ,REQ=UIM
```

label
Specifies the label name that the system generates in the first instruction of the macro expansion.

MID
Specifies the message identifier. Make sure the message identifier is eight bytes long.

Note: The CBDIMSG macro can only issue messages defined in the associated UIM data table (UDT). See "UIM Data Tables (UDTs)" on page 31.

VARn
You can specify variables when defining a message. These variables cause substitutions in the message, just before it is displayed. Because many languages have different noun and verb sequences, message variables are numbered instead of sequenced. Specify a message variable by including an @n in the message text, where "n" is a number from 1 to 9.

This is a message text example: "Number of units must be @1 for the @2 device."

Each variable may be as long as 255 bytes. Specify variable type as:

H
Specifies a hexadecimal field type.

B
Specifies a fixed binary field type.

C
Specifies a character field type.

If you omit the variable type, its default is C.

SEV
Specifies the message severity. The following severities are supported:

MSGINFO
informational message. This message has no effect on HCD processing or its return code.

MSGWARN
warning message. This message has no effect on HCD processing but will cause HCD to issue a return code of 4 (unless the UIM issues a message of higher severity).

MSGERR
error message. This message will prevent HCD from building any I/O configuration members, and will cause HCD to issue a return code of 8 (unless HCD issues a message of higher severity).

MSGTERM
severe error message. This message causes HCD to end its processing and issue a return code of 16. A UIM must *never* issue a severe error message.

This parameter is optional; the default is MSGERR.

REQ
Must be **UIM** (specifies that the macro call is issued by a UIM).

Make sure the UIM can address the UCA when the UIM issues the CBDIMSG macro.

Note: You must also include the CBDZMSG definition macro in the UIM. The CBDZMSG definition macro maps the MSG parameter list. The CBDIMSG executable macro builds the MSG parameter list.

Example

To issue message CBDB805I:

```
CBDIMSG    MID=CBDB805I,VAR1=IODVUNIT,           *
     .     STMT=YES,REQ=UIM
     .
     .
CBDB805I   DC CL8'CBDB805I'
```

The message definition for CBDB805I must exist in the corresponding UDT.

The message definition in the sample UDT is:

```
         CBDZUDT MID=CBDB805I,ID=FEATURE,                              *
               TEXT='Features SHARED and SHAREDUP are mutually exclusive*
               e for device type @1.',                                 *
               HELP=CBDED05
```

In the actual message text @1 is replaced with the device unit type that is passed to the UIM through the IODV control block.

CBDIPPDS Executable Macro

The CBDIPPDS macro puts (pushes) an entry onto or removes (pops) an entry from the diagnostic stack.

The syntax of the CBDIPPDS macro is as follows:

```
 [label]   CBDIPPDS  {PUSH|POP}
                     ,DIAG=diag
                     ,REQ=UIM
                     [,RELATED=related]
```

label
Name of the label to be generated on the first instruction in the macro expansion.

PUSH
The designated diagnostic entry is to be put on the diagnostic stack. Either PUSH or POP must be specified.

POP
The designated diagnostic entry is to be removed from the diagnostic stack. Either PUSH or POP must be specified.

DIAG
Name of the diagnostic entry. This name must be specified on the label field of the CBDZDIAG macro.

REQ
Must be **UIM** (specifies that a UIM is issuing the macro).

RELATED
Specifies an optional character string that can identify related macro calls.

Make sure the UIM can address the UCA when the UIM issues the CBDIPPDS macro.

Note: A UIM must invoke the CBDZDIAG definition macro to build the diagnostic stack entry that is to be pushed onto or popped from the diagnostic stack.

Example

```
CBDIPPDS    PUSH,DIAG=DIAGDATA,REQ=UIM
     .
     .
     .
CBDIPPDS    POP,DIAG=DIAGDATA,REQ=UIM
     .
     .
     .
```

```
DIAGDATA   CBDZDIAG MODNAME=CBDUC255,        *
           CSECT=CBDUC255,COMP=SC1XL,        *
           DESC=='UIM FOR DASD 33UU, 93UU',  *
           MODCAT=UIM
```

CBDISIML Executable Macro

The CBDISIML macro generates a list, by type and model, of either similar devices or devices attachable to a specific control unit type.

- A similar device is a lookalike device, which you can define directly. For example, a 3178 device has the same characteristics as the 3270-X device, so you can define a 3178 as a 3178 *directly*.
- An attachable device is a device that you can attach to a particular control unit.

The syntax of the CBDISIML executable macro is as follows:

```
label      CBDISIML  (device[,model])
                     [,(device,[model]),(...),]
                     [TYPE={DEV|CU}]
```

Specify the device list as positional parameters.

device
Specifies the device type.

model
Specifies the device model.

TYPE
Specifies the type of device list. The default is DEV.

DEV
Indicates that the list is a similar device list.

CU
Indicates that the list is an attachable device list.

Example

```
ATT39CC     CBDISIML   (33UU,1),                    *
                       (33VV,2),                    *
                       TYPE=CU
```

This example specifies a list of devices attachable to a specific control unit.

UIM Definition Macros

The following definition macros are used by UIMs.

Definition Macro Name	Definition Macro Function
CBDZCIP	Maps the control unit information parameters (CIP) that provide input to the CIT build routine.
CBDZDCP	Maps the device characteristics parameters (DCP) that provide input to the DCT build routine (used for DASD only).
CBDZDEVL	Maps the device lookup (DEVL) parameter list.
CBDZDFP	Maps the device features parameters that provide input to the DFT build routine.
CBDZDIAG	The CBDZDIAG definition macro builds an HCD diagnostic stack entry.
CBDZGETM	Maps the getmain (GETM) parameter list built by the CBDIGETM macro.

Definition Macro Name	Definition Macro Function
CBDZGIP	Maps the generic information parameters (GIP) that provide input to the GIT build routine.
CBDZITRH	Maps the internal text record header (ITRH), the I/O device internal text record (IODV), and the private parameter value array (PPVA).
CBDZMSG	Maps the message routine parameter list, which is built by the CBDIMSG macro.
CBDZSIP	Maps the switch information parameters (SIP) that provide input to the SIT build routine.
CBDZUCA	Maps the UIM communications area (UCA).
CBDZUIP	Maps the unit information parameters (UIP).

CBDZCIP Definition Macro

The pointer CIPDVLP in CBDZCIP points to the list of attachable devices build by CBDISIML. CIPDVLC in CBDZCIP denotes the number of list elements in the attachable device list. CIPDVLC in initialized with parameter DEV.

The CBDZCIP macro maps the control unit information parameters (CIP). The CIP is the input parameter list to the CIT build routine.

The syntax of the CBDZCIP macro is as follows:

```
          CBDZCIP   [DEV=devcnt]
                 or
          CBDZCIP   TYPE=DSECT
```

DEV=devcnt
Specifies the number of entries that you want the system to generate in the attachable device list. This list identifies, by device type and model, the devices that can be attached to the control unit named in the CIP. This parameter is optional; the default is 1.

Each entry in the attachable device list consists of two fields: DEVICE TYPE and DEVICE MODEL. If a device does not have a model, its DEVICE MODEL field must be binary zero or blank.

TYPE=DSECT
Generates mapping for the attachable device list entry. Use the structure defined by the CBDZCIP definition macro to fill in each entry of the attachable device list.

Note:

1. You cannot specify a label on the CBDZCIP macro.
2. Using the CBDISIML macro is the recommended method for defining an attachable device list. See “CBDISIML Executable Macro” on page 26.

CBDZDCP Definition Macro

The CBDZDCP macro maps the device characteristics parameters of DASDs. The DCP is the input parameter list to the DCT build routine.

The syntax of the CBDZDCP macro is as follows:

```
          CBDZDCP
```

There are no input parameters on the CBDZDCP macro.

Note: You cannot specify a label on the CBDZDCP macro.

CBDZDEVL Definition Macro

The CBDZDEVL macro maps the device lookup (DEVL) parameter list.

The syntax of the CBDZDEVL macro is as follows:

```
          CBDZDEVL
```

There are no input parameters on the CBDZDEVL macro.

Note: You cannot specify a label on the CBDZDEVL macro.

CBDZDFP Definition Macro

The CBDZDFP macro maps the device features parameters. The device features parameters list is the input parameter list to the DFT build routine.

The syntax of the CBDZDFP macro is as follows:

```
          CBDZDFP [RELOC=reloc]
```

reloc
Specifies the number of entries that the system is to generate in the relocation list. The relocation list identifies fields in the device-dependent sections of the UCB (device-dependent segment, device-dependent extension, or device class extension) that point to other sections of the same UCB or another UCB. This parameter is optional; the default is 0.

Note: You cannot specify a label on the CBDZDFP macro.

A UIM may not specify more than 256 bytes of device-dependent information. The information that falls within this 256-byte limit consists of:

- UCB device-dependent segment (length, 24 bytes maximum, specified in the device features parameter field DFPDDSL)
- UCB device-dependent extension (length specified in the device features parameter field DFPDDEL)
- UCB device class extension (length specified in the device features parameter field DFPDCEL)
- Relocation list (the length of the list is computed by multiplying the number of entries in the list, which is contained in the device features parameters field DFPRELCT, by the length of a list entry, which is 12 bytes).

CBDZDIAG Definition Macro

Use the CBDZDIAG macro to build a diagnostic stack entry in which you specify certain diagnostic information. If an abnormal end (abend) occurs in the UIM, HCD's recovery routine places the diagnostic information in the system diagnostic work area (SDWA).

Use the CBDIPPDS executable macro to put entries onto, and remove entries from the diagnostic stack.

The syntax of the CBDZDIAG macro is as follows:

```
label   CBDZDIAG MODNAME=modname,
                 [MODCAT=UIM,]
                 [CSECT=csect,]
                 COMP=comp,
                 DESC=desc,
                 [VRADATA=vradata,]
                 [RELATED=('related')]
```

label
Name of the diagnostic stack entry. The labels of the fields generated in the diagnostic stack entry will start with the same four characters as **label** does. (In the event that **label** exceeds four characters, only the first four characters are used to build the labels on the generated fields.) **label** is required.

MODNAME
Load module name of the UIM. If an abend occurs, this value will be placed in SDWA field SDWAMODN. MODNAME is required.

MODCAT
Although MODCAT is an optional parameter, use it because it identifies its module as a UIM module. The module category (MODCAT) is used for trace.

CSECT
CSECT name of the UIM. If an abend occurs, this value will be placed in SDWA field SDWACSCT. This parameter is optional; the default is the assembler symbol &SYSECT; value.

COMP
Component identifier of the UIM. If an abend occurs, this value will be placed in SDWA field SDWACID. The component identifier must be five bytes long. This parameter is required.

DESC
UIM description. If an abend occurs, this value will be placed in SDWA field SDWASC. The UIM description can be a maximum of 23 bytes long. This parameter is required.

VRADATA
Name of the array that contains the addresses of data to be placed in the variable recording area (VRA) if an abend occurs. The array contains the VRA keys and data lengths, in addition to the data addresses. This parameter is optional. If it is not specified, no specific control blocks or data areas for the UIM will be placed in the VRA.

Each entry in the VRA array contains eight bytes. The format of an entry is as follows:

Offset	Length	Function
0	2	Must be set to zero in all but the last entry in the array. The last entry in the VRA array must be set to X'FFFFFFFFFFFFFFFF'. This entry denotes the end of the VRA array.
2	1	Key of VRA data, as specified in IHAVRA.
3	1	Length of VRA data.
4	4	Address of VRA data. If this field is set to zero, the ESTAE routine will skip this entry when moving data into the VRA. UIMs are permitted to dynamically update this field while the diagnostic entry is on the diagnostic stack.

related
Optional character string.

CBDZGETM Definition Macro

The CBDZGETM macro maps the parameter list built by the CBDIGETM macro.

The syntax of the CBDZGETM macro is as follows:

```
          CBDZGETM
```

There are no input parameters on the CBDZGETM macro.

Note: You cannot specify a label on the CBDZGETM macro.

CBDZGIP Definition Macro

The CBDZGIP macro maps the generic information parameters (GIP). The GIP is the input parameter list to the GIT build routine.

The syntax of the CBDZGIP macro is as follows:

```
          CBDZGIP [DENS=dens],
                  [GENDNMS=gendnms]
```

dens
Specifies the number of entries that the system generates in the density list. This list contains the densities that are supported by the generic device type. This parameter is optional; the default is 0.

gendnms
Specifies the number of entries that the system generates in the compatible generic device name list. This list contains the generic names of devices for which this generic device type can be used to satisfy allocation requests. This parameter is optional; the default is 0.

Note: You cannot specify a label on the CBDZGIP macro.

CBDZITRH Definition Macro

The CBDZITRH macro maps the internal text record header (ITRH), the I/O device internal text record (IODV), and the private parameter value array (PPVA).

The syntax of the CBDZITRH macro is as follows:

```
          CBDZITRH
```

There are no input parameters on the CBDZITRH macro.

Note: You cannot specify a label on the CBDZITRH macro.

CBDZMSG Definition Macro

The CBDZMSG macro maps the message routine (MSGR) parameter list, which is built by the CBDIMSG macro.

The syntax of the CBDZMSG macro is as follows:

```
          CBDZMSG
```

There are no input parameters on the CBDZMSG macro.

Note: You cannot specify a label on the CBDZMSG macro instruction.

CBDZSIP Definition Macro

The CBDZSIP macro maps the switch information parameters (SIP) of ESCON directors. The SIP is the input parameter list to the SIT build routine.

The syntax of the CBDZSIP macro is as follows:

```
          CBDZSIP [CUL=entrycnt],
                  [SWL=attswcnt],
                  [TYPE=DSECT]
```

CUL=entrycnt
Specifies the number of entries in the switch control unit list. This list contains one entry for each control unit that can be defined as switch control unit. This parameter is optional; the default is 1.

SWL=attswcnt
Specifies the number of attachable switches. It must be set to the number of switches to be contained in the attachable switch list. This parameter is optional; the default is 1.

TYPE=DSECT
Generates the mapping structure for the attachable switch list.

Note: You cannot specify a label on the CBDZSIP macro.

CBDZUCA Definition Macro

The CBDZUCA macro maps the UIM communications area (UCA).

The syntax of the CBDZUCA macro is as follows:

```
          CBDZUCA
```

There are no input parameters on the CBDZUCA macro.

Note: You cannot specify a label on the CBDZUCA macro instruction.

CBDZUIP Definition Macro

The CBDZUIP macro maps the unit information parameters (UIP). The UIP is the input parameter list to the UIT build routine.

The pointer UIPMSIMP points to the list of similar devices built by CBDISIML.

The syntax of the CBDZUIP macro is as follows:

```
CBDZUIP [TYPE={GEN|DSECT}]
        [,DFLT=dflt]
        [,SIM=sim]
        [,MLTS=mlts]
        [,SEL=sel]
```

TYPE
Specifies request type.

GEN
Generate the UIP structure. (GEN is the default if you omit the TYPE parameter.)

DSECT
Include the following DSECTs:

- Similar device list entry
- Parameter default list entry
- Parameter selection list entry.

Note: If you code TYPE=DSECT, you cannot use any other parameter.

DFLT
Specifies the number of entries that the system generates in the parameter default list. This list contains information about parameters that have defaults. The DFLT parameter is optional; its default is zero.

SIM
Specifies the number of entries that the system generates in the similar device list. This list identifies, by device types and models, those devices that are similar to the device named in the UIP. The SIM parameter is optional; its default is zero.

Note: IBM recommends using the CBDISIML macro to generate a device list. (See "CBDISIML Executable Macro" on page 26.)

MLTS
Specifies the number of entries to be generated in the module lists table (MLT) list. This parameter is optional; the default is 1. (The maximum number of MLTs allowed for a device is 5.)

SEL
Specifies the number of entries that the system generates in the parameter selection list. This list contains the parameters that will be in CBDZUIP parameter list. The SEL parameter is optional; its default is zero.

Note: You cannot specify a label on the CBDZUIP macro.

UIM Data Tables (UDTs)

A UIM data table (UDT) contains information that you need to externalize for national language translation, including entries for:

- Unit (device) descriptions
- Parameter descriptions

- Feature descriptions
- Messages

Write one UDT for each UIM. UIMs and UDTs are associated using the 3-digit number at the end of the names. Although a UIM can have more than one UDT (one for each supported language), each UIM has only one active UDT because the language is selected during HCD start up. Each UDT is loaded with its associated UIM. If the UDT's requested language version does not exist, the English version is loaded. However, if a UIM does not have at least an English UDT in the library when another language is selected, the I/O units represented by that UIM are considered unknown.

How to Write a UDT

To write a UDT, follow these steps:

1. Name your UDT, using CBDxCnnn format, where x is the language code and nnn is the number assigned to your UIM. (See "CBDZUDT Macro" on page 32.)

   Note: IBM provides a sample UDT in SYS1.SAMPLIB(CBDSUDT). Each IBM-supplied UDT has the CBDxSnnn format.
2. Write a CBDZUDT macro for each UDT entry. (See Figure 5 on page 36, which is a sample IBM-provided English UDT.)

   Note: Group all CBDZUDT macros that have identical parameters. For example, group all CBDZUDT macros with FEAT parameters sequentially.
3. Install the UDT in the same library that contains the UIMs.

CBDZUDT Macro

The CBDZUDT macro defines one entry in the UDT.

The syntax of the CBDZUDT macro is as follows:

```
CBDZUDT   UDT=name,UIM=name,DESC='text'[,LANG=name]
  or
CBDZUDT   UNIT='description'[,HELP=name]
  or
CBDZUDT   FEAT=feature,TEXT='description'[,HELP=name]
  or
CBDZUDT   CFEAT={feature|(feature,feature[,feature]...  .)}
  or
CBDZUDT   PARA=(parameter[,nn]),TEXT='description'
                [,PARATYPE=(type[,length])][,HELP=name]
  or
CBDZUDT   MID='CBDBnnnI',TEXT='description'[,HELP=name][,ID=parameter]
  or
CBDZUDT   DEVICE=unit[,MODEL=model],HELP=name,TYPE=type,ID=parameter
```

UDT
Specifies the eight-character name of the UDT. Follow the naming convention as described in "How to Write a UDT" on page 32.

UIM
Specifies the eight-character name of the UIM associated with this UDT.

DESC
Specifies the description of the UIM. Enter no more than 60 characters.

LANG
Specifies a one-character code for the language supported by the UDT. Currently, a UDT can have one of the following language codes:

- E (English)
- J (Japanese)

UNIT
Specifies the description of the unit (device). Enclose the description in apostrophes. Enter no more than 60 characters.

HELP
Specifies the name of the help panel associated with this feature, parameter, message, or unit (device associated with this UDT).

FEAT
Specifies the name of the feature supported by the device associated with the UDT. Enter no more than 10 characters. (Entering "*" creates a place holder for FEAT in this UDT.) The order of features must be the same as in the UIM and UIP.

TEXT
Specifies the feature, parameter, or message text. You must code this parameter if you also code FEAT, PARA, or MID. You must enclose the text in apostrophes. For FEAT and PARA, enter no more than 44 text characters. For MID, enter no more than 120 text characters.

CFEAT
Specifies the name of one or more obsolete, but compatible, features supported by the device associated with the UDT. If you specify two or more compatible features, enclose them in parentheses. Enter no more than 10 characters per feature name.

PARA
Specifies the common or private parameter, "parameter(,nn)," where "parameter" is the parameter name and "nn" is the private parameter identifier, a decimal number from 33 through 64.

Note: You may use the same private parameter identifiers in each of many UIMs, because UIMs do not share private parameters.

Valid required and optional common parameter names are:

- ADAPTER
- DYNAMIC
- LOCANY
- NUMSECT
- OFFLINE
- OWNER
- PCU
- SETADDR
- TCU

For migration, additional valid common parameter names are:

- ADDRESS
- FEATURE
- MODEL
- UNIT

Private parameter names cannot exceed eight characters.

PARATYPE
Specifies the type of private parameter. (You cannot use PARATYPE with any common parameter.) You must specify PARATYPE as follows:

```
PARATYPE=(type[,length][,LIST,count][,RANGE,first,last])
```

TYPE
"type" can be one of the following:

NUM
Parameter value must be numeric (0-9)

HEX
Parameter value must be hexadecimal (0-9,A-F)

ALPHANUM
Parameter value must be alphanumeric

ALPHANUM*
Parameter value must be alphanumeric or *

CHAR
Parameter value can contain any characters

YESNO
Parameter value must be either YES or NO

NAME
Parameter value must be a name

NAME*
Parameter value must be a name or *

length
Maximum length allowed for the value on the HCD panel.

LIST
To specify more than one value for parameter.

count
Maximum number of values.

RANGE
To specify a subrange.

first
Minimum value of range.

last
Maximum value of range.

Examples:

```
CBDZUDT   PARA=CLASS,TEXT='Printer output spooling class',
          PARATYPE=(ALPHANUM*,4,LIST,8),HELP=CBDEH13
```

```
CBDZUDT   PARA=CLASS,TEXT='Size of delayed purge queue',
          PARATYPE=(NUM,1,RANGE,1,9)
```

MID
Specifies the message identifier, “CBDBnnnI,” where “nnn” is a decimal number from 500 through 999.

Note: Because many languages have different noun and verb relationships, message variables are numbered instead of sequenced. Specify a message variable by including an @n in the message text, where “n” is a number from 1 to 9.

This is a message text example: “Number of units must be @1 for the @2 device.”

ID
Specifies the parameter associated with the message or help panel.

If ID is associated with a message, it specifies the parameter associated with that message.

If ID is associated with a help panel overwrite table (HPOT), it specifies the parameter associated with that help panel, and the CBDZUDT macro also uses the TYPE parameter to specify the type of help. For TYPE=SPECIAL, the ID parameter must specify one of the following:

DEVNUM
Specifies device number help for the device defined through DEVICE and MODEL parameters.

DEVRANGE
Specifies device range (number of devices) help for the device defined through DEVICE and MODEL parameters.

DEVTYPE
Specifies device type help.

DEVUA
Specifies unit address help.

DEVICE
Specifies the name of the device (unit) for which the default help panel name should be overwritten by the name specified in the HELP parameter. This help panel overwrite table (HPOT) allows you to provide device-specific help for a parameter. (See Chapter 4, "HCD Help Support," on page 39.)

MODEL
Specifies the model number (optional) of the unit (device).

TYPE
Specifies the type of help:

PARA
Specifies parameter help. The ID parameter specifies which parameter is associated with this help.

FEAT
Specifies feature help. The ID parameter specifies which feature number is associated with this help.

SPECIAL
Specifies special help. The ID parameter specifies what special help is associated with this help:

DEVNUM
Specifies device number help for the device defined through DEVICE and MODEL parameters.

DEVRANGE
Specifies device range (number of devices) help for the device defined through DEVICE and MODEL parameters.

DEVTYPE
Specifies device type help.

DEVUA
Specifies unit address help.

Figure 5 on page 36 shows the UDT for a channel to channel (CTC) device. Also, see the sample UDT in SYS1.SAMPLIB(CBDSUDT).

```
CBDZUDT UDT=CBDES014,UIM=CBDUS014,
      DESC='UIM for CTC Devices'
CBDZUDT
      UNIT='Channel-to-channel Adapter'
CBDZUDT
      UNIT='Multisystem Channel Communication Unit'
CBDZUDT
      UNIT='Serial Channel-to-channel Adapter'
CBDZUDT
      UNIT='Remote Channel-to-channel Unit'
CBDZUDT
      UNIT='Basic Mode ESCON Channel-to-channel Adapter'
CBDZUDT PARA=OFFLINE,
      TEXT='Device considered online or offline at IPL',
      HELP=CBDFP08
CBDZUDT PARA=DYNAMIC,
      TEXT='Device has been defined to be dynamic',
      HELP=CBDFP12
CBDZUDT PARA=LOCANY,
      TEXT='UCB can reside in 31 bit storage',
      HELP=CBDFP15
CBDZUDT FEAT=370,
      TEXT='Attached to System/370',
      HELP=CBDEB01
CBDZUDT DEVICE=3088,
      TYPE=SPECIAL,
      ID=DEVRANGE,
      HELP=CBDEB02
END
```

Figure 5. UDT Example

Testing UIMs

Test your installation-written UIMs carefully *before* you IPL your system.

Testing UIMs with HCD

To test UIMs with HCD, use the procedures detailed in *z/OS HCD User's Guide* to:

- Initialize HCD. This checks that the UIMs have been loaded by HCD.
- With the HCD function ***Query supported hardware and installed UIMs*** you can verify whether the control unit and device defined in your UIM are displayed.
- Use the HCD batch utility “Print Supported Hardware Report” to verify whether the control units and devices defined in your UIM are shown.
- With the HCD function ***Define, modify, and view configuration data*** you can check whether the control unit and device specified in your UIM can be defined in the dialog.
- Use the report facility against the IODF to detect certain errors before using the IODF during system IPL. For example, this can detect errors in a UIM's device features table (DFT) build process.

If no errors are encountered during these tests, there are probably no errors in the UIMs.

Testing UIMs During IPL

Certain errors in UIMs might cause a wait state code during IPL. Such errors can, of course, only be tested during IPL.

If an error occurs, analyze the problem according to *z/OS HCD User's Guide*, then correct it.

Note: To test the UIM, do not link it into SYS1.NUCLEUS. Instead, before testing the UIM, link it to another library and concatenate that library to the HCD load libraries. Specify that library on the UIM_LIBNAME parameter in the HCD profile statement.

UIM_LIBNAME=Name of data set containing the UIMs
All UIMs (and UDTs) are loaded from the specified data set (SYS1.NUCLEUS is the default)

UIM_LIBNAME=*
The UIMs are contained in the HCD load libraries. In this case, the data set containing the new UIM and SYS1.NUCLEUS containing the existing UIMs must be concatenated to the HCD load libraries using STEPLIB/JOBLIB statements.

Test the UIM as described in "Testing UIMs" on page 36.

Installing a UIM

UIMs must reside as separate members in SYS1.NUCLEUS or the UIM library defined in the HCD PROFILE statement. For IPL, the UIMs and UDT must reside in SYS1.NUCLEUS; for testing purposes you can install them in the UIM library defined in the HCD profile.

Chapter 4. HCD Help Support

Hardware configuration definition (HCD) provides extensive online help support, which varies according to message status, and cursor position. When you request help, HCD displays specific help information in a panel (screen). (For details about HCD online help, see *z/OS HCD User's Guide*.)

HCD help panels reside in an interactive system productivity facility (ISPF) load library partitioned data set.

Creating Help Panels

For each help panel that HCD can display, HCD requires one help member. Therefore, to create a help panel, you must create a help member. Copy the CBDZHELP macro, which contains the help generation macros, then use the help generation macros to create a help member.

Each help member contains both text and supporting code for one help panel. When someone requests help, HCD retrieves the appropriate help member, then displays the text for that help panel.

A help member consists of:

- A header, which contains control information that HCD uses but does not display.
- A reference phrase array, which contains a list of reference phrases and associated help member names. HCD does not display the reference phrase array, which associates reference phrases in the text lines with help member names.

  Each reference phrase needs its own help member. For example, you can separately place two new terms in two reference phrases, then separately define these new terms in two associated help members. Each reference phrase appears as an input area on the help panel. A user can tab the cursor to either reference phrase, then press ENTER to display the text lines that define the term in that reference phrase.
- Text lines, which HCD displays on the help panel.

The CBDZHELP macro contains the following help generation macros:

HDR
Builds the help member header

RP
Builds one entry in the reference phrase array

TXT
Builds one text line.

These help generation macros are described in “HDR Macro” on page 40, “RP Macro” on page 41, and “TXT Macro” on page 42.

Figure 6 on page 40 shows example help generation macros that would create help member CBDED15.

Figure 7 on page 40 shows an example of a message help panel.

You must assemble and link-edit each help member. For assembly, as shown in Figure 6 on page 40, you must:

1. Code COPY CBDZHELP
2. Code the help generation macros in the following sequence:
   a. HDR macro - only one
   b. RP macro(s) - one or more (optional)
   c. TXT macro(s) - one or more

```
  PRINT OFF              Suppress listing of HCD help generation macros
  COPY  CBDZHELP         Include HCD help generation macros
  PRINT ON,NOGEN         Do not list macro expansion
*
  HDR  NAME=CBDED15,TITLE='HCD help member CBDED15',WIDTH=53,       *
               DESC='X.X.X COPYRIGHT INFO',                        *
               HIGHLI=YES
*
  RP   PHRASE='MVS',HELP=CBDEDXX
*
  TXT 'Number of Devices'
  TXT ' '
  TXT 'Specify a decimal value from 1 to 8, or omit. If you do'
  TXT 'not specify a value for number of devices,<MVS>uses a'
  TXT 'default value of 2.'
  TXT ' '
  TXT 'The value you specify determines how many device numbers'
  TXT 'MVS assigns. (It always assigns a minimum of eight.) MVS'
  TXT 'first assigns the device number you specify and then uses'
  TXT 'that number as a base to calculate the additional device'
  TXT 'numbers that it assigns.'
  TXT '-end-'
  END
```

Figure 6. Example Help Generation Macros

```
         PRINT OFF
         COPY  CBDZHELP
         PRINT ON
*
 HDR NAME=CBDEG07,TITLE='Help panel for CBDEG07',WIDTH=60
*
 TXT ' '
 TXT ' CBDB027I  NUMSECT value must be in the range of 0 to @1. '
 TXT ' '
 TXT ' '
 TXT ' Explanation:                                              '
 TXT '   The number of 256-byte buffer sections in the control  '
 TXT '   unit is out of valid range.                            '
 TXT ' '
 TXT ' System Action:                                           '
 TXT '   The system waits for user response.                    '
 TXT ' '
 TXT ' User Response:                                           '
 TXT '   Correct the NUMSECT parameter.                         '
 END
```

Figure 7. Example of Message Help Panel

Note: @1 is related to the VARn variable of messages, see “CBDIMSG Executable Macro” on page 24.

HDR Macro

The HDR macro generates the header of a help member. As shown in Figure 6 on page 40, code the HDR macro so that it follows COPY CBDZHELP and precedes all RP and TXT macro calls in the help member.

The syntax of the HDR macro is as follows:

```
[label]  HDR     NAME=name[,TITLE='xxxx']
                 [,WIDTH={53|60}]
                 [,DESC='xxxxxxx']
                 [,HIGHLI={YES|NO}]
                 [,RPDLM=xy]
```

label
Specifies the label name that the system generates in the first instruction of the macro expansion.

NAME
Specifies the help member's CSECT name. The name can be up to 7 alphanumeric characters long, but its first character must be alphabetic (A-Z).

TITLE
Specifies the title for the assembler listing. You must enclose the title in apostrophes.

WIDTH
Specifies help panel's width. Valid values are 53 and 60. The default value is 60. (To guard against problems in the header, do not use width values other than either 53 or 60.)

DESC
Specifies a 1 to 255-character description, such as a copyright statement, that is to appear in the help member. HCD does not display this description. You must enclose the description in apostrophes.

HIGHLI
Specifies whether the first text line is highlighted (displayed with a different color). The default is NO.

YES
Indicates that the first text line (help title) is highlighted.

NO
Indicates that the first text line is at the same brightness as all subsequent text lines.

RPDLM
Specifies the reference phrase delimiters (starting and ending indicators) for a reference phrase. A reference phrase is a word or phrase in the help text that has additional help information associated with it. The specification must consist of exactly two characters, not enclosed in apostrophes, and not separated by commas or blanks. The default is <>. Specify the RPDLM parameter if you do not want the default indicators.

- The first character indicates the start of a reference phrase.
- The second character indicates the end of the reference phrase.

Note: These characters appear as blanks on the help panel. However, the reference phrase itself appears as an input area on the help panel, allowing the user to tab the cursor to it, then press ENTER to display the help information for that phrase.

Figure 6 on page 40 shows an example HDR macro. For this HDR macro, its NAME parameter specifies “CBDED15” as the help member's CSECT name, its TITLE parameter specifies the title as “HCD help member CBDED15,”, its WIDTH parameter specifies the width as 53 characters, its DESC parameter specifies the description as “X.X.X COPYRIGHT INFO”, and its HIGHLI parameter specifies “YES” for highlighting the first line of text. (Because this HDR macro has no RPDLM parameter, the reference phrase delimiters default to <>.)

RP Macro

The RP macro is optional. Each RP macro generates one entry in a help member's reference phrase array. Within a help member, group all RP macros together, following the HDR macro.

The syntax of the RP macro is as follows:

```
[label]  RP       PHRASE='xxxxxx'[,HELP={name|abc*  }]
```

label
Specifies the label name that the system generates in the first instruction of the macro expansion.

PHRASE
Specifies a reference phrase, which is a word or phrase that has additional help information associated with it. A reference phrase appears within a single help text line, and can be up to 32 characters long.

You must enclose the reference phrase in apostrophes. You can use the RPDLM parameter of the HDR macro to specify different reference phrase delimiters.

HELP
Specifies the name of the help member that describes the reference phrase. Each reference phrase needs its own help member.

name
The name can be up to 7 alphanumeric characters long.

abc*
You may specify a generic name to display all reference help members that have names beginning with the same specified characters together as one entry. You may specify as many as 6 common characters. For example, if you specify “abc*,” you group all help members that have names beginning with three characters “abc.” In this example, the “*” represents as many as four unique characters at the end of each help member name.

Note: If you specify a generic name to group help members, all members in that group must have the same width.

If you omit the HELP parameter, the RP macro generates a special name. When someone requests help for the phrase, its special name creates a temporary combined reference, appending all other listed reference help members.

Figure 6 on page 40 shows an example RP macro. For this RP macro, its PHRASE parameter specifies “MVS” as a reference phrase, and its parameter specifies “CBDEDXX” as the help member that describes the “MVS” reference phrase.

TXT Macro

Each TXT macro generates one line of text in a help member. Within a help member, group all TXT macros together, following any RP macro.

When someone requests help, HCD displays each line of text as you specified it through a TXT macro.

The syntax of the TXT macro is as follows:

```
[label]  TXT      'text-line'
```

label
Specifies the label name that the system generates in the first instruction of the macro expansion.

text-line
Specifies the text line. Its maximum length is WIDTH minus 1. (Specify WIDTH as a parameter in the HDR macro.) You must enclose the text line in apostrophes.

The text line can contain one or more reference phrases. You must enclose each reference phrase between the starting and ending indicators for a reference phrase. (Specify these starting and ending indicators through the RPDLM parameter in the HDR macro.) The default starting indicator is a < character, and the default ending indicator is a > character.

You do not need to duplicate reference phrases within a help panel. If a reference phrase appears more than once on a panel, you should place delimiters only around the first occurrence of that phrase.

IBM recommends that you specify “TXT” in unit record columns 2 through 4, and the text line's beginning apostrophe in column 6. This lets you enter either 53 or 60 text-line characters into a single unit record. This simplifies help panel maintenance, because text lines appear similarly in assembler source code and on the help panel.

Figure 6 on page 40 shows example TXT macros. These multiple TXT macros generate the multiple text lines of help member CBDED15.

Note: Only the first “MVS” phrase in these text lines has reference phrase delimiters because you do not need duplicate reference phrases within a help panel.

Testing Help Panels

While in help mode, you can use the HELPTEST command to display any help panel without simulating the conditions that normally cause HCD to display that help panel.

To display a help panel, enter “HELPTEST xxxxxxx,” where xxxxxxx is the name of the help panel, which was defined by the NAME parameter of the HDR macro. After displaying a help panel through HELPTEST, you can display the help panel for any of its reference phrases.

For example, if you enter “HELPTEST CBDED15” while in the help mode, &hcd displays help panel CBDED15 if help member CBDED15 includes the NAME parameter of the HDR macro as shown in Figure 6 on page 40. After displaying help panel CBDED15, you can display the “CBDEDXX” help panel for the “MVS” reference phrase. (See “RP Macro” on page 41.)

HCD UIM Help Support

HCD provides:

- UIM help panels for device parameters
- UIM data table (UDT) help pointers
- message help panels.

Parameter Help Panels

HCD provides default help panels for the following device parameters:

- ADAPTER
- DYNAMIC
- LOCANY
- SETADDR
- OFFLINE
- OWNER
- TCU

HCD displays a default help panel if the UIM does not specify a help panel name for the appropriate parameter. In other words, the UIM needs to provide its own help panel only if the default help is inappropriate.

Help Panel Overwrite Tables (HPOTs)

A help panel overwrite table (HPOT) is part of a UIM data table (UDT). (See “UIM Data Tables (UDTs)” on page 31.) An HPOT lets you change the help panel name specification for a specific parameter for a particular unit or model. Use an HPOT when help information for a parameter varies among devices that are supported by the same UIM, or help information varies from that in the default help panels.

HCD UIM Message Help

Make sure that each HCD UIM provides a message help panel for each message defined in the UDT associated with that UIM. You may use the same message help panel with more than one UIM. Make sure the each message help panel contains the following:

- Explanation
- System action
- User response.

Appendix A. Sample of a Unit Information Module (UIM)

```
******************************** TOP OF DATA ************************
*
*       The CBDSUIM member in SYS1.SAMPLIB can be used as a model
*       by customers when writing a Unit Information Module (UIM).
*       Customer-written UIMs are used to define non-IBM I/O units,
*       including devices, control units and ESCON directors,
*       in an I/O configuration. UIMs are invoked by the
*       Hardware Configuration Definition (HCD), by MVS IPL, and
*       by MVS Dynamic Activate.
*
*       For each UIM, a corresponding UDT must be developed.
*
*       Instructions:
*
*       1) Define a name for your UIM, of the format
*          CBDUCnnn, with nnn between 001 and 256.
*          Note: The sample UIM uses the number 255. If you like
*                to use another number, replace the number.
*
*       2) Copy this Sample UIM to a PDS member with the name
*          chosen for your UIM.
*
*       3) Change all strings "CBDUC255" in the UIM to the
*          chosen name.
*
*       4) Change the UIM according to your needs.
*
*       5) Separate the JCL at the end of the UIM,
*          and correct the names in the JCL.
*          Assemble and link-edit your UIM using the JCL.
*
*--------------------------------------------------------------------
*  Note:
*  If you write an UIM, you should know the hardware and software
*  configuration characteristics of the I/O unit that needs a UIM
*  and should be familiar with the basic concepts of MVS and IOCP.
*
*  Following additional documentation is required:
*
*  - z/OS MVS Device Validation Support
*  - z/OS MVS Data Areas
*  - IOCP User's Guide
*--------------------------------------------------------------------
*  Attention: The UIM must not use any MVS services, except those
*             described in the manual z/OS MVS Device Validation Support
*
*********************************************************************
         TITLE 'CBDUC255: Sample UIM'
*   * START OF SPECIFICATIONS ***************************************
*
*01* MODULE NAME = CBDUC255
*
*01* DESCRIPTIVE NAME = SAMPLE UIM
*
*********************************************************************
*   PROPRIETARY STATEMENT =
*
*    LICENSED MATERIALS - PROPERTY OF IBM
*    THIS SAMPLE IS "RESTRICTED MATERIAL OF IBM"
*    5655-068 (C) COPYRIGHT IBM CORP. 1990, 1995
*
*   END PROPRIETARY STATEMENT
*********************************************************************
*
*01* FUNCTION =
*
*      This sample UIM describes 2 sets of DASD equipment:
*
*      Control Units:             39CC-6            93CC
*
*      Devices attachable
```

```
*      to above control units:    33UU-1            93UU
*                                 33UU-2
*                                 33VV
*
*      MVS GENERIC names
*      for above devices:         33GG              93GG
*
*      The following parameters are recognized for the 33UU-1, 33UU-2
*      and 33VV DASD devices:
*         Common parameters:  OFFLINE, DYNAMIC, FEATURE
*         Private parameter:  DASDPOOL
*
*      The following features are recognized for the 33UU-1, 33UU-2
*      and 33VV DASD devices:
*         SHARED, SHAREDUP, ALTCTRL
*
*      The following parameters are recognized for the
*      and 93UU DASD device:
*         OFFLINE, DYNAMIC, FEATURE
*
*      The following features are recognized for the 93UU DASD
*      device:
*         ALTCTRL
*
*
*02*   OPERATION =
*      This unit information module defines the device dependent
*      support for the 33GG and 93GG DASD DEVICES.
*
*    o When called with the initialization call,
*
*    - CBDUC255 builds the parameter list for the Generic Information
*      Table and calls the GIT Build Routine to create the GITs
*      for the following generics:
*                 .33GG
*                 .93GG
*
*    - CBDUC255 builds the parameter list for the Unit Information
*      Table and calls the UIT Build Routine to create the UITs
*      for the following devices:
*                 .33UU-1 and its look-alike devices 33UU-2, 33VV
*                 .93UU
*
*    - CBDUC255 builds the parameter list for the Control Unit
*      Information Table and calls the CIT Build Routine to
*      create the CITs for the following control units:
*                 .39CC-6
*                 .93CC
*
*    - CBDUC255 builds the parameter list for the Device
*      Characteristics Table and calls the DCT Build Routine to
*      create the DCTs for the following devices:
*                 .33UU
*                 .93UU
*
*    o When called by the HCD validation routines with a parameter
*      check request,
*      no parameter check is performed because there are no
*      additional rules for the provided parameters than
*      those already supported by HCD.
*
*    o When called by the HCD validation routines with a feature
*      check request,
*      the features the user specified, contained in the IODV,
*      are validity checked.
*
*    o When called by the HCD validation routines with a device number
*      check request,
*      no device number check is performed as no special rules for the
*      device number are applicable for 33GG and 93GG devices.
*
*    o When called by the HCD validation routines with a unit address
*      check request, it is checked if the starting unit address of
*      a 93UU device definition is even-numbered.
*
*    o When HCD runs in Report Mode and during IPL and dynamic
*      activation, CBDUC255 is called to build the parameter list
*      for the Device Feature Table build routine for each device
*      defined in the IODF.
*      The DFTs are used to build the UCBs for the configuration.
*
*    o When, during MVS IPL, CBDUC255 is called for end-of-data
```

```
*      processing, no special action is taken as no end-of-data
*      processing is required.
*                                                                     *
*03*     RECOVERY OPERATION =
*         If an unexpected error occurs in CBDUC255, the ESTAE
*         routine CBDMSTAE established in module CBDMGHCP
*         will provide the diagnostic information.
*         No recovery is done during IPL. Any unexpected errors
*         during IPL will cause a wait state to be loaded.
*
*01* NOTES =
*
*02*   DEPENDENCIES = None
*
*02*   RESTRICTIONS = None
*
*01* MODULE TYPE = Procedure
*
*02*   PROCESSOR = ASSEMBLER-H
*
*02*   MODULE SIZE = For exact size see assembler listing
*
*02*   ATTRIBUTES =
*
*03*     LOCATION = Private
*
*03*     STATE = Problem
*
*03*     AMODE = 31
*
*03*     RMODE = Any
*
*03*     KEY = User
*
*03*     MODE = Task
*
*03*     SERIALIZATION = None
*
*03*     TYPE = Non-reusable
*
*01* ENTRY POINT = CBDUC255
*
*02*   PURPOSE = See FUNCTION
*
*02*   LINKAGE = Standard Linkage
*
*03*     CALLERS =
*         HCD Routines
*         (Functional Initialization routine,
*          Validation routines,
*          Report routine),
*         IPL Routine
*         Dynamic Activate Routine
*
*01* INPUT =
*       UCA
*       IODV (anchored off UCA),
*       for the following request types:
*            UCARADDR
*            UCARDFTB
*            UCARPARM
*            UCARFEAT
*            UCARUADD
*
*
*02*   ENTRY REGISTERS =
*        Register     0 - Undefined
*        Register     1 - Pointer to a one word parameter list,
*                         defined as follows:
*                           Word 1 - Address of the UCA
*        Registers 2-12 - Undefined
*        Register    13 - Address of an 18-word save area
*        Register    14 - Return address
*        Register    15 - Entry point address
*
*
*01* OUTPUT =
*       Causes GITs for generics supported by this UIM to be built.
*       Causes UITs for devices  supported by this UIM to be built.
*       Causes CITs for CUs      supported by this UIM to be built.
*       Causes DCTs for each defined device type to be built.
*       Causes DFTs for devices  supported by this UIM to be built.
```

```
*       Modifies the UCA.
*
*02*   EXIT REGISTERS =
*       Registers 0-15 - Restored to contents on entry
*
*02*   RETURN CODES = see UCA ( set in UCA )
*
*01* EXIT NORMAL = Returns to the caller
*
*01* EXIT ERROR = None
*
*01* EXTERNAL REFERENCES =
*
*02*   ROUTINES =
*       CIT Build Routine
*       DCT Build Routine
*       DFT Build Routine
*       GIT Build Routine
*       UIT Build Routine
*
*02*   DATA AREAS =
*       CBDZDIAG - Diagnostic Stack Entry
*
*02*   CONTROL-BLOCKS =
*       Common name  Macro Name   Usage
*       -----------  ----------   -----------
*       CIP          CBDZCIP      write
*       DCE          IECDDCE      read
*       DCP          CBDZDCP      write
*       DFP          CBDZDFP      write
*       GIP          CBDZGIP      write
*       IODV         CBDZIODV     read
*       MSGR         CBDZMSG      write (via CBDIMSG)
*       UCA          CBDZUCA      read/write
*       UCB          IEFUCBOB     read
*       UIP          CBDZUIP      write
*
*01* TABLES = None
*
*01* MACROS EXECUTABLE =
*      CBDIMSG   -  Write Message
*      CBDIPPDS  -  Push/Pop Diagnostic Stack Entry
*
*01* CHANGE ACTIVITY =
*
*    $H0= HCD   HCSH501 940501 BOEB: Sample UIM for DASD I/O
*    $H4= OSA   HCSH521 941011 BOEB: Open systems adapter support
*    $H1= OW12423 HCSH521 950504 BOEB: Quality enhancements
*                                - Documentation of function codes
*                                - Add MODCAT to CBDZDIAG statement
*
*01* SERIALIZATION = None
*
*01* MESSAGES =
*
*      CBDB805I   Features (SHARED) and (SHAREDUP) are mutually
*                 exclusive for device type dddddddd.
*
*      CBDB814I   The left-most digit of unit address nn for device
*                 type dddddddd must be even.
*
*01* ABEND CODES = None
*
*01* WAIT STATE CODES = None
*
***** END OF SPECIFICATIONS *****************************************
         EJECT
*********************************************************************
*
*              Initial House-keeping
*
*********************************************************************
         SPACE 1
CBDUC255 CSECT
CBDUC255 RMODE ANY
CBDUC255 AMODE 31
         SPACE 1
         USING *,R15
         USING UCA,UCAPTR           Define pointer to UCA
         USING ITRH,IODVPTR         Define pointer to IODV
         B     START
         SPACE 1
```

```
         DC    AL1(LENGTH-LENEBCDC) Length of EBCDIC description
LENEBCDC DC    C'CBDUC255 '         EBCDIC description of module
         DC    C'&SYSDATE';          Compile date
LENGTH   EQU   *
         SPACE 2
START    STM   R14,R12,12(R13)      Save caller's registers
         LR    R11,R15              Set base register contents
         DROP  R15                  Drop R15 as base register
         USING CBDUC255,R11         Establish addressability to
*                                   UIM csect
         ST    R13,SAVAREA+4        Establish backward linkage in
*                                   current savearea
         LA    R10,SAVAREA          Obtain savearea address
         ST    R10,8(R13)           Establish forward linkage in
*                                   caller's savearea
         LR    R13,R10              Places this UIMs own savearea
*                                   address in register 13
         L     UCAPTR,0(R1)         Establish addressability to UCA
         L     IODVPTR,UCAIODVP     Establish addressability to IODV
*
*********************************************************************
*  Pushes a new entry on the diagnostic stack.
*  The entry
*  - provides diagnostic information for abnormal termination
*  - causes a trace entry to be written into the HCD.TRACE dataset,
*    when the HCD trace is active.
*********************************************************************
         SPACE 1
         CBDIPPDS PUSH,DIAG=DIAGDATA,REQ=UIM
         SPACE 1
         LA    R0,UCARCOK           Set up good return code
         ST    R0,UCARETC           Initialize return code
*
         EJECT
*********************************************************************
*
*        Determine what function the UIM is called to perform
*-------------------------------------------------------------------
*
*  Whenever the UIM is called, the field UCAUIMRT is set by the
*  calling routine with one of the request types listed below.
*
*  Request Type:       UIM function to be performed:
*
*  UCARINIT            Initialization request
*                      as required:
*                         build GIT
*                         build UIT
*                         build DCT (only for DASD devices)
*                         build CIT
*                         build SIT (only for ESCON directors)
*  UCARDFTB            DFT build request
*  UCARADDR            Device Number check
*  UCARPARM            Parameter check
*  UCARFEAT            Feature check
*  UCARUADD            Unit Address check
*  UCAREOD             End of data processing
*
*  On each call, the UIM must analyze the Request Type and call the
*  appropriate internal routines.
*  The UIM may be called with Request Types which are not applicable
*  to this UIM, in this case the UIM must perform no operation,
*  and return to the calling routine.
*
*-------------------------------------------------------------------
*              Handle Initialization Request
*-------------------------------------------------------------------
*
         CLI   UCAUIMRT,UCARINIT    Initialization request ?
         BNE   TSDFTBLD             ..No, branch to test if called for
*                                   another request
         BAL   R14,BUILDGIT         Calls routine to build the GIT
         BAL   R14,BUILDUIT         Calls routine to build the UIT
         BAL   R14,BUILDCIT         Calls routine to build the CIT
         BAL   R14,BUILDDCT         Calls routine to build the DCT
         B     EXIT                 Branch to leave routine
         SPACE 1
*-------------------------------------------------------------------
*              Handle DFT Build Request
*-------------------------------------------------------------------
*
TSDFTBLD DS    0H
```

```
         CLI   UCAUIMRT,UCARDFTB    Test the caller's function code
*                                   to determine if the purpose of this
*                                   call is to build the DFTs
         BNE   TSADDCHK             ..No, branch to test if called for
*                                   another request                @H1C
         BAL   R14,BUILDDFT         Call routine to build DFTs
         B     EXIT                 Branch to leave routine
*
         SPACE 1
*--------------------------------------------------------------------
*              Handle Device Number Check Request
*--------------------------------------------------------------------
*
TSADDCHK DS    0H                                                   @H1A
*        CLI   UCAUIMRT,UCARADDR    Test the caller's function code
*                                   to determine if the purpose of this
*                                   call is to check the device number
*                                                                  @H1A
*        BNE   TSPRMCHK             ..No, branch to test if called for
*                                   another request                @H1A
*        BAL   R14,ADDRCHEK         Call routine to check device # @H1A
*        B     EXIT                 Branch to leave routine        @H1A
*
*        SPACE 1                                                   @H1A
*--------------------------------------------------------------------
*              Handle Parameter Check Request
*--------------------------------------------------------------------
*
TSPRMCHK DS    0H                                                   @H1A
*        CLI   UCAUIMRT,UCARPARM    Test the caller's function code
*                                   to determine if the purpose of this
*                                   call is to check the device
*                                   parameters                     @H1A
*        BNE   TSFEACHK             ..No, branch to test if called for
*                                   another request                @H1A
*        BAL   R14,PARMCHEK         Call routine to check device
*                                   parameters                     @H1A
*        B     EXIT                 Branch to leave routine        @H1A
*
*        SPACE 1                                                   @H1A
*--------------------------------------------------------------------
*              Handle Feature Check Request
*--------------------------------------------------------------------
*
TSFEACHK DS    0H
         CLI   UCAUIMRT,UCARFEAT    Test the caller's function code
*                                   to determine if the purpose of this
*                                   call is to check the device
*                                   features.
         BNE   TSUADCHK             ..No, branch to test if called for
*                                   another request
         BAL   R14,FEATCHEK         Call routine to check the features
*                                   of the passed device
         B     EXIT                 Branch to leave routine
*
         SPACE 1
*--------------------------------------------------------------------
*              Handle Unit Address Check Request
*--------------------------------------------------------------------
*
TSUADCHK DS    0H
         CLI   UCAUIMRT,UCARUADD    Test the caller's function code
*                                   to determine if the purpose of this
*                                   call is to check the device
*                                   unit address.
         BNE   TSEODPRO             ..No, branch to test if called for
*                                   another request                @H1C
         BAL   R14,UADDCHEK         Call routine to check the features
*                                   of the passed device
         B     EXIT                 Branch to leave routine
*
         SPACE 1
*--------------------------------------------------------------------
*              Handle Parameter Check Request
*--------------------------------------------------------------------
*
TSEODPRO DS    0H                                                   @H1A
*        CLI   UCAUIMRT,UCAREOD     Test the caller's function code
*                                   to determine if the purpose of this
*                                   call is end of data processing @H1A
*        BNE   TSOTHERS             ..No, branch to test if called for
*                                   another request                @H1A
```

```
*        BAL   R14,PROCEOD          Call routine for end of data
*                                   processing                    @H1A
*        B     EXIT                 Branch to leave routine       @H1A
*
*        SPACE 1                                                  @H1A
*-----------------------------------------------------------------------
*              Handle other Request(s)
*-----------------------------------------------------------------------
*
*        There are no other requests.
*        Return to the calling routine.
*
TSOTHERS DS    0H
         EJECT ,
************************************************************************
*
*              Final House-keeping
*
*-----------------------------------------------------------------------
*
*  Pops the top entry from the diagnostic stack.
*
EXIT     DS    0H
         CBDIPPDS POP,DIAG=DIAGDATA,REQ=UIM
*
*-----------------------------------------------------------------------
*  Restores caller's registers and returns
*-----------------------------------------------------------------------
*
         L     R13,4(R13)           Obtains callers savearea.
         LM    R14,R12,12(R13)      Restore caller's registers.
         BR    R14                  Return to caller.
*
         EJECT
*
************************************************************************
*
*    Procedure: BUILDGIT
*
*    Descriptive Name: Build Generic Information Parameter
*
*    Function:  Builds the Generic Information Tables (GITs)
*               for the device types supported by this UIM.
*
*    Operation: Fills in the Generic Information Parameter
*               and calls the GIT Build Routine to create
*               the GIT.
*
************************************************************************
*  The Generic Information Table GIT is used to register
*  GENERIC device names to MVS.
*  For every GENERIC name, the UIM must set the Generic Information
*  Parameters GIP, and call the HCD routine to build the GIT.
*  (The information provided with the GIP is stored in the GIT.)
*  The GIP layout is defined in macro CBDZGIP.
*
*  All GIP fields set in this sample UIM are mandatory.
*
*  The UCB TYPE values for units, configured by means of HCD, can
*  be seen in the HCD "MVS Device Report".
*
*  The generic preference value for a generic device must
*  be UNIQUE, which means no other generic device in the
*  same MVS must have the same value.
*  For preference values used by IBM units, refer to the
*  appendix of "z/OS MVS Device Validation Support"
*
*  For affinity index values dedicated to IBM units and values
*  reserved for users, refer to macro CBDZGIP.
*
*-----------------------------------------------------------------------
*  Following GIP fields are not set by this sample UIM
*
*  Name       Description                    Remarks
*  ----------|------------------------------|-----------------------
*  GIPCMPNL  |Compatible generic device name |Used by tape UIMs
*            |list                           |
*  GIPCMPNM  |Compatible generic device name |Used by tape UIMs
*            |                               |
*  GIPDENL   |Densities supported list       |Used by tape UIMs
*  GIPDENSY  |Density                        |Used by tape UIMs
************************************************************************
```

```
*********************************************************************
*
*          Builds GIT Routine.
*
*        This routine initializes the GIT build parameter list and
*        then calls the GIT Build Service Routine.
*
*********************************************************************
*
BUILDGIT DS    0H
         ST    R14,SAVWORD1         Place return address in savearea.
*********************************************************************
*
*   Builds GIT parameter list for 33GG.
*
*********************************************************************
         XC    GIP,GIP              Zero out GIT build parameter list.
         MVC   GIPID,GIPIDNM        Insert control block ID
         MVI   GIPVER,GIPVERN       Place version number in parameter
*                                   list.
         MVC   GIPNAME,GEN33GG      Place name of generic device name
*                                   in GIP.
         MVC   GIPUCBTY,GNRCTYP1    Initialize allocation UCB type
*                                   information.
         MVC   GIPGPTPR,GNRCPRT1    Initialize generic preference
*                                   table priority.
         MVC   GIPAFFIX,=AL2(GIPNOAFF) Set the affinity index to
*                                   "No affinity consideration"
         SPACE 1
         ST    UCAPTR,PARMAREA      Store address of UCA in first word
*                                   of parmarea.
         LA    R0,GIP               Get address of GIP
         ST    R0,PARMAREA+4        Store address of GIP in second word
*                                   of parmarea.
         LA    R1,PARMAREA          Load address of parameter list
*                                   in register 1
         L     R15,UCAGITP          Pick up entry point address of     *
                                    GIT Build Routine
         BALR  R14,R15              Call routine to build GITs
         SPACE 1
*********************************************************************
*
*   Builds GIT parameter list for 93GG.
*
*********************************************************************
         XC    GIP,GIP              Zero out GIT build parameter list.
         MVC   GIPID,GIPIDNM        Insert control block ID
         MVI   GIPVER,GIPVERN       Place version number in parameter
*                                   list.
         MVC   GIPNAME,GEN93GG      Place name of generic device name
*                                   in GIP.
         MVC   GIPUCBTY,GNRCTYP2    Initialize allocation UCB type
*                                   information.
         MVC   GIPGPTPR,GNRCPRT2    Initialize generic preference
*                                   table priority.
         MVC   GIPAFFIX,=AL2(GIPNOAFF) Set the affinity index to
*                                   "No affinity consideration"
         SPACE 1
         ST    UCAPTR,PARMAREA      Store address of UCA in first word
*                                   of parmarea.
         LA    R0,GIP               Get address of GIP
         ST    R0,PARMAREA+4        Store address of GIP in second word
*                                   of parmarea.
         LA    R1,PARMAREA          Load address of parameter list
*                                   in register 1
         L     R15,UCAGITP          Pick up entry point address of     *
                                    GIT Build Routine
         BALR  R14,R15              Call routine to build GITs
         SPACE 1
         L     R14,SAVWORD1         Restore mainline's return address
         BR    R14                  and return to mainline
         EJECT
*
*********************************************************************
*
*    Procedure: BUILDUIT
*
*    Descriptive Name: Build Unit Information Parameter
*
*    Function:  Builds the UITs for the device types supported
*               by this UIM.
*
```

```
*    Operation: Fills in the Unit Information Parameter
*               and calls the UIT Build Routine to create
*               the UIT
*
*********************************************************************
*  The Unit Information Table UIT is used to register
*  type/model names and parameters of device units to HCD.
*  Only unit types which are defined by any UIT can be
*  configured by HCD and can be operated by MVS.
*
*  You can view many of the UIT fields defined by the UIM, when you
*  select "List supported devices" and in device definition panels.
*
*  For each unit type with unique configuration parameters
*  the UIM must fill the Unit Information Parameters UIP
*  and call the HCD routine to build the UIT.
*  For each UIP field, there is a corresponding UIT field.
*
*  The UIP layout is defined in the macro CBDZUIP
*
*  There are 3 UIP sections:
*  1) The General section describes device characteristics which
*     are independent from the operating system, such as type, model,
*     or attachment information, such as the maximum number of CUs
*     a device can attach to.
*     This section is required.
*  2) The MVS Section describes device characteristics which are
*     relevant for MVS only, such as generic name, device parameters
*     and features.
*     This section is required if the device is to be defined
*     as a device supported by MVS.
*  3) The VM Section - which is not shown in this sample UIM -
*     describes device characteristics which are relevant to
*     VM only, such as "RDEV device class" or "RDEV device type".
*     This section is required if the device is to be defined
*     as a device supported by VM.
*
*********************************************************************
*  Following UIP fields are not set by this sample UIM
*
*  Name       Description                     Remarks
*  ----------|-------------------------------|-----------------------
*  UIPGUSER  |UIM user value for device      |Processing control field
*            |                               |At Initialization, the
*            |                               |UIM can set a value per
*            |                               |UIP. At successive calls
*            |                               |this value is passed to
*            |                               |the UIM via the field
*            |                               |UCAUSER.
*            |                               |
* UIPGDNC    |Count of device numbers to     |Used for multiple
*            |generate for each device if    |exposure devices
*            |multiple-exposure device or    |and parallel access
*            |parallel access volume         |volumes.
*            |                               |
* UIPGDNI    |Interval between device        |Used for multiple
*            |numbers when multiple device   |exposure devices
*            |numbers are generated for the  |and parallel access
*            |same device (valid only when   |volumes.
*            |the value of UIPDNC is greater |
*            |than one)                      |
* UIPGRFLG   |Replication factor flags       |Used for multiple
*            |                               |exposure devices and
*            |                               |parallel access volumes.
*            |                               |
* UIPGPFLG   |Processing flag                | HCD internal use only;
*            |                               | UIM must not set this.
*            |                               |                     @H4A
*            |                               |
* UIPGDFLG   |Default flags                  |Defines defaults for
*  UIPGFTOU  |if 1, TIMEOUT=NO is default    |parameters which are not
*  UIPGFSTA  |if 1, STADET=NO  is default    |OS specific.
*            |                               |
* UIPGATT    |Attachment information         |OS independent
*            |                               |attachment information.
*  UIPGMNCU  |max. number of CUs a device    |The # of CUs a device
*            |can be attached to             |can be attached to is
*            |if hex zero,the value enforced |restricted to this
*            |by the dialog is taken         |maximum number by HCD.
*            |                               |
* UIPGPR     |Processing flags               |HCD internal use only,
*            |                               |UIM must not set this.
```

```
*            |                                |
* UIPMCFEA   |Map of features that are        |The features defined
*            |recognized for migration        |here are tolerated, but
*            |compatibility (bits correspond |ignored during migration
*            |to sequence of compatible       |of MVSCP decks.
*            |features in UDT. - Valid only   |
*            |if IODVFFEA flag within         |
*            |UIPOPARM is set)                |
*            |                                |
* UIPMATT    |Attachment information          |MVS attachment informa-
*  UIPMNIPC  |NIPCON device type codes         |tion, set only if NIPCON
*            |                                |device.
*            |                                |
* UIPSIMFL   |Device flags                    |
*  UIPSDFLT  |Device model is default         |Indicates that, if the
*            |                                |user does not define a
*            |                                |model, the model speci-
*            |                                |fied in UIPSMODL is used.
* UIPMLTFL   |Flags                           |
*  UIPMLTOP  |MLT contains module names       |
*            |associated with a product that |
*            |provides optional support for   |
*            |this device                     |
*********************************************************************
*
*              UIT Build Routine
*
*        This routine initializes the UIT build parameter list and
*        then calls the UIT Build Service Routine.
*********************************************************************
*
BUILDUIT DS    0H
         ST    R14,SAVWORD1         Place return address in savearea.
*
*--------------------------------------------------------------------
*
*          Builds UIT parameter list for 33UU-1
*
* Note, that the CBDZUIP macro already initializes the UIP structure.
*--------------------------------------------------------------------
*      General section
*--------------------------------------------------------------------
*  Following registers the type/model name of a device to HCD
*--------------------------------------------------------------------
*
         MVC   UIPGUNIT,UNIT33UU    Place device type in UIP
         MVC   UIPGMODL,MODL1       Place the device model in UIP
*
*--------------------------------------------------------------------
*  For each Unit defined in an UIP, there must be a unit
*  description in the corresponding UDT.
*  You can view the unit description texts in HCD, by selecting the
*  "List installed UIMs" panel, and then the
*  "View Supported devices" panel.
*  There may be multiple unit descriptions in a UDT.
*  The UIPGDESI parameter is used by HCD to find the appropriate
*  unit description for a device type.
*--------------------------------------------------------------------
*
         MVI   UIPGDESI,1          Set index to the unit description
*                                  for the device concerned in the UDT
*
*--------------------------------------------------------------------
*  In the HCD "List Supported Device" panel and in the device
*  prompt panels, the devices are grouped in order
*  to facilitate the navigation among many device types.
*
*  HCD uses the UIPGGRP parameter to associate a device type to
*  a certain device group such as DASD or Tape devices.
*  Refer to macro CBDZUIP for available group values.
*--------------------------------------------------------------------
*
         MVI   UIPGGRP,UIPGDASD     Indicate that the device belongs
*                                   to the DASD group
*--------------------------------------------------------------------
*  When a device is defined in HCD, the user can specify
*  the Replication Factor (Number of Devices).
*  Following UIP fields are used to handle the Replication Factor
*
*  UIPGDDRF     Minimum Replication Factor    (required)
*
*              Specifies the minimum number of device definitions
```

```
*              to be created.
*
*  UIPGHHRF     Maximum Replication Factor    (optional)
*
*              Specifies the maximum number of device definitions
*              to be created.
*
*  UIPGDLRF     Default Replication Factor    (required)
*
*              Specifies the default value used if the number of
*              device definitions to be created was not specified.
*--------------------------------------------------------------------
*
         MVC   UIPGDDRF,=H'1'       Set default replication factor
         MVC   UIPGDLRF,=H'1'       Set minimum replication factor
*
*--------------------------------------------------------------------
*      MVS section
*--------------------------------------------------------------------
*  The following associates the device type "33UU-1" with the
*  Generic Device "33GG".
*--------------------------------------------------------------------
*
         MVC   UIPMGNNM,GEN33GG     Set name of generic device in UIP
*
*--------------------------------------------------------------------
*  The following defines the parameters and features that are
*  applicable when defining a device unit in HCD.
*
*  Generally, the parameters must be described in the associated UDT.
*  The following parameters are applicable for all device types.
*  They need not be described in the UDT.
*
*      ADDRESS   - specifies the device number
*      UNIT      - specifies the device type
*      MODEL     - specifies the device model
*
*  HCD distinguishes between required and optional parameters.
*
*--------------------------------------------------------------------
*  Required parameters
*--------------------------------------------------------------------
*
*  Parameters defined as required for a device unit must be
*  given a value when creating the device definition.
*
*  The required parameters are set in bit string UIPMRPRM.
*  The position in the bit string is given by the parameter ID.
*  The first four bytes of UIPMRPRM are reserved for common parameters,
*  Bytes 5 through 8 are used for private parameters.
*
*  The following parameters are required for all device types:
*
*      ADDRESS   - specifies the device number
*      UNIT      - specifies the device type
*--------------------------------------------------------------------
*
         OI    IODVFLG1-IODVPRMS+UIPMRPRM,IODVFADD  ADDRESS parameter
         OI    IODVFLG2-IODVPRMS+UIPMRPRM,IODVFUNI  UNIT parameter
*
*--------------------------------------------------------------------
*  Optional parameters
*--------------------------------------------------------------------
*  Parameters defined as optional for a device unit need not be
*  given a value when creating the device definition.
*
*  The optional parameters are set in bit string UIPMOPRM.
*  The position in the bit string is given by the parameter ID.
*  The first four bytes of UIPMOPRM are reserved for common parameters,
*  Bytes 5 through 8 are used for private parameters.
*--------------------------------------------------------------------
*
         OI    IODVFLG1-IODVPRMS+UIPMOPRM,IODVFMOD   MODEL parameter
         OI    IODVFLG1-IODVPRMS+UIPMOPRM,IODVFOFF   OFFLINE parameter
         OI    IODVFLG2-IODVPRMS+UIPMOPRM,IODVFDYN   DYNAMIC parameter
         OI    IODVFLG1-IODVPRMS+UIPMOPRM,IODVFFEA   FEATURE parameter
         OI    UIPMOPRM+4,DASDPPRM                  DASDPOOL parameter
*
*--------------------------------------------------------------------
*  Features
*--------------------------------------------------------------------
*  The supported features are set in bit string UIPMSFEA.
```

```
*  Features can have the values 'Yes' or 'No'. 'No' is the default
*  value, unless the corresponding feature is defined in the
*  default feature bit string UIPMDFEA; in this case, the default
*  value is set to 'Yes'.
*
*  All features must be described in the associated UDT.
*  The positions of the features in the bit string correspond to
*  to their sequence in the UDT.
*-------------------------------------------------------------------
*
         OI    UIPMSFEA,FEATACTL                    ALTCTRL feature
         OI    UIPMSFEA,FEATSHR                     SHARED  feature
         OI    UIPMSFEA,FEATSHUP                    SHAREDUP feature
*-------------------------------------------------------------------
*  Parameter default values
*-------------------------------------------------------------------
*  The UIM allows you to set defaults or initial values for
*  parameters.
*  For some common parameters, defaults are indicated by flags
*  in the UIP:
*   - UIPMDFLG specifies defaults for
*        MODEL - If UIPMFDMD is set, and the model is not specified
*                during device definition, the default
*                device model is taken from UIPGMODL.
*        OFFLINE - If UIPMFOFF is set, the device is defined
*                as offline during IPL (OFFLINE=YES).
*        DYNAMIC - UIPMFDYC indicates whether the device supports
*                dynamic reconfiguration.
*                You can set DYNAMIC=YES only if UIPMFDYC is set.
*                If UIPMFDYC is set, the device will default to be
*                dynamically reconfigurable in the HCD dialog
*                (DYNAMIC=YES).
*                Note, however, that for the migration function
*                this default does not apply; here, if DYNAMIC is not
*                specified, it is left undefined.
*
*   To provide default values for other OS specific parameters
*   (common or private), you can provide an entry in the parameter
*   default list. The entry contains the parameter ID together with
*   the default value.
*-------------------------------------------------------------------
*
         OI    UIPMDFLG,UIPMFDMD    Indicates that, if no device model
*                                   is specified, the model specified
*                                   in UIPGMODL is used as default.
         OI    UIPMFLG2,UIPMFDYC    Indicates that the device supports
*                                   dynamic reconfiguration.
*                                   The DYNAMIC parameter is
*                                   initialized to YES in the HCD
*                                   dialog.
         L     R1,UIPMDLFP          Gets address of parameter default
*                                   list.
         USING UIPDLFPL,R1          Establishes addressability to
*                                   parameter default list.
         MVC   UIPDPID,DASD_PID     Identifies parameter to which the
*                                   default applies: DASDPOOL.
         LA    R2,L'DASD_DEF        Gets length of default value
*                                   string.
         STH   R2,UIPDLEN           Stores length of default value
*                                   in parameter default list entry.
         LA    R2,DASD_DEF          Gets address of default value
*                                   string.
         ST    R2,UIPDPTR           Stores address of default value
*                                   in parameter default list entry.
         DROP  R1                   Removes addressability of parameter
*                                   default list.
*-------------------------------------------------------------------
*  Parameter selection values
*-------------------------------------------------------------------
*
*  The UIM allows you to specify the allowed values for a parameter
*  in the parameter selection list.
*
*  The values specified in the parameter selection list for a
*  parameter serve two purposes:
*  (1) They are offered via prompt in the OS/device parameter and
*      feature panel.
*  (2) The HCD validation function checks the entered value
*      against the values specified in the parameter selection list.
*      If the entered value is not contained, an error message
*      is provided, and the parameter value is rejected. This
*      technique frees the UIM from checking the valid parameter
```

```
*      values.
*
*   To provide selection values for an OS specific parameter
*   (common or private), you must provide an entry in the parameter
*   selection list. The entry contains the parameter ID together with
*   the selection values.
*-------------------------------------------------------------------
*
         L     R1,UIPMSELP          Gets address of parameter selection
*                                   list.
         USING UIPSELPL,R1          Establishes addressability to
*                                   parameter selection list.
         MVC   UIPSPID,DASD_PID     Identifies parameter to which the
*                                   selection values apply: DASDPOOL.
         LA    R2,3                 Gets the number of selection values
*                                   for the parameter.
         STH   R2,UIPSCNT           Stores the number of selection
*                                   values in the parameter selection
*                                   list entry.
         LA    R2,L'DASD_SP1        Gets length of a parameter
*                                   selection value.
         STH   R2,UIPSLEN           Stores length of selection value
*                                   in parameter selection list entry.
         LA    R2,DASD_SEL          Gets address of selection value
*                                   string.
         ST    R2,UIPSPTR           Stores address of selection values
*                                   in parameter selection list entry.
         DROP  R1                   Removes addressability of parameter
*                                   selection list.
*-------------------------------------------------------------------
*  Similar device list
*-------------------------------------------------------------------
*
*  The UIM allows you to specify for a given device
*  a list of device types which
*  are look-alikes to the device. This frees you from
*  specifying the same UIP settings and calling the
*  UIT Build routine again if only the device types and models differ.
*
*  An entry in the similar device list causes the UIT Build routine
*  to build a UIT with the same values as specified in the UIP,
*  using the device type of the similar device list entry.
*-------------------------------------------------------------------
*
         L     R1,UIPMSIMP          Gets address of similar device
*                                   list.
         USING UIPSIMDL,R1          Establishes addressability to
*                                   similar device list.
         MVC   UIPSUNIT,UNIT33UU    Moves first similar device type to
*                                   the similar device list.
         MVC   UIPSMODL,MODL2       Moves the model of the first
*                                   device type.
         LA    R1,UIPSLENG(R1)      Advance to next entry.
         MVC   UIPSUNIT,UNIT33VV    Moves second similar device type to
*                                   the similar device list.
         DROP  R1                   Removes addressability of similar
*                                   device list.
*-------------------------------------------------------------------
*   Following indicates that 4-digit device numbers for this
*   device type are supported
*-------------------------------------------------------------------
*
         OI    UIPMFLG2,UIPMFDVN    Device supports 4-digit device
*                                   numbers
*
*-------------------------------------------------------------------
*  The MLT is the list of modules representing the device code that
*  is loaded at IPL time
*-------------------------------------------------------------------
*
         LA    R2,UIPMLTNM          Initialize pointer to MLT name list
         ST    R2,UIPMMLTP          Put this value in the UIP
         MVC   UIPMLTNM,NAMEMLT     Set MLT name.
         MVC   UIPMMLTC,ONE         Set MLT count.
*
*-------------------------------------------------------------------
*  The DDT name represents a Device Definition Table that is loaded
*  at IPL time
*-------------------------------------------------------------------
*
         MVC   UIPMDDTN,NAMEDDT     Set DDT name.
*********************************************************************
```

```
*        Call UIT Build Service Routine
*********************************************************************
         ST    UCAPTR,PARMAREA      Store address of UCA in first word
*                                   of parmarea.
         LA    R0,UIP               Get address of UIP
         ST    R0,PARMAREA+4        Store address of UIP in second word
*                                   of parmarea.
         LA    R1,PARMAREA          Store address of parameter list
*                                   in register 1
         L     R15,UCAUITP          Pick up entry point address of     *
                                    UIT Build Routine
         BALR  R14,R15              Call routine to build UITs
         SPACE 1
*********************************************************************
*
*   Builds UIT parameter list for 93UU
*
*********************************************************************
*
* Since the UIP is used for another device unit, it has to
* be initialized again.
*-------------------------------------------------------------------
         XC    UIP(UIPGELN1),UIP    Zeroes out UIP list.
* Initialize the UIP header.
         MVC   UIPGID,UIPIDNM       Sets storage descriptor in header.
         OI    UIPGVER,UIPGVER1     Sets UIP version code.
         MVC   UIPGELEN,=AL2(UIPGELN1) Sets total length of UIP.
* Initialize the UIP general section.
         OI    UIPGTYP,UIPGEN       Indicates general section.
         MVC   UIPGLEN,=AL2(UIPGLN1) Sets length of general section.
* Initialize the UIP MVS section.
         OI    UIPMTYP,UIPMVS       Indicates the MVS section.
         MVC   UIPMLEN,=AL2(UIPMLN1) Sets length of MVS section.
* Initialize the parameter default area.
         LA    R2,UIPPDFLT          Loads address of parameter default
*                                   list.
         ST    R2,UIPMDLFP          Stores address of parameter default
*                                   list in UIP.
         XC    UIPPDFLT,UIPPDFLT    Zeroes out parameter default list.
* Initialize the parameter selection list.
         LA    R2,UIPPSEL           Loads address of parameter
*                                   selection list.
         ST    R2,UIPMSELP          Stores address of parameter
*                                   selection list in UIP.
* Initialize the similar device list.
         LA    R2,UIPSIMIL          Loads address of similar device
*                                   list.
         ST    R2,UIPMSIMP          Stores address of similar device
*                                   list in UIP.
         XC    UIPSIMIL,UIPSIMIL    Zeroes out similar device list.
* Initialize the MLT name list.
         LA    R2,UIPMLTNM          Initialize pointer to MLT name list
         ST    R2,UIPMMLTP          Puts this value in the UIP
         XC    UIPMLTNL,UIPMLTNL    Zeros out the MLT name list
*-------------------------------------------------------------------
* Fills in the values for device unit 93UU.
*-------------------------------------------------------------------
*********************************************************************
*      General section
*********************************************************************
         MVC   UIPGUNIT,UNIT93UU    Place device type in UIP
         XC    UIPGMODL,UIPGMODL    Clear out model field
         MVI   UIPGDESI,1           Set index to the unit description  *
                                    for the device concerned in the UDT
         MVI   UIPGGRP,UIPGDASD     Indicate to what group the device  *
                                    belongs
         MVC   UIPGDDRF,=H'1'       Set default replication factor
         MVC   UIPGDLRF,=H'1'       Set minimum replication factor
*********************************************************************
*      MVS section
*********************************************************************
         MVC   UIPMGNNM,GEN93GG     Place name of generic device in UIP
         OI    IODVFLG2-IODVPRMS+UIPMRPRM,IODVFUNI  UNIT parameter
*
         OI    IODVFLG1-IODVPRMS+UIPMRPRM,IODVFADD   ADDRESS parameter
         OI    IODVFLG1-IODVPRMS+UIPMOPRM,IODVFOFF   OFFLINE parameter
         OI    IODVFLG2-IODVPRMS+UIPMOPRM,IODVFDYN   DYNAMIC parameter
*
         OI    IODVFLG1-IODVPRMS+UIPMOPRM,IODVFFEA   FEATURE parameter
         OI    UIPMSFEA,FEATACTL                     ALTCTRL feature
*
         OI    UIPMFLG2,UIPMFDYC    Device supports dynamic
```

```
*                                  configuration
         OI    UIPMFLG2,UIPMFDVN    Device supports 4-digit device
*
*                                  numbers
*
         MVC   UIPMLTNM,NAMEMLT     Initialize MLT name.
         MVC   UIPMMLTC,ONE         Initialize MLT count.
         MVC   UIPMDDTN,NAMEDDT     Initialize DDT name.
*********************************************************************
*        Call UIT Build Service Routine
*********************************************************************
         ST    UCAPTR,PARMAREA      Store address of UCA in first word
*                                  of parmarea.
         LA    R0,UIP               Get address of UIP
         ST    R0,PARMAREA+4        Store address of UIP in second word
*                                  of parmarea.
         LA    R1,PARMAREA          Store address of parameter list
*                                  in register 1
         L     R15,UCAUITP          Pick up entry point address of     *
                                    UIT Build Routine
         BALR  R14,R15              Call routine to build UITs
         SPACE 1
         L     R14,SAVWORD1         Restore mainline's return address
         BR    R14                  and return to mainline
*
         EJECT
*
*********************************************************************
*
*    Procedure: BUILDDCT
*
*    Descriptive Name: Build Device Characteristics Parameters
*
*    Function:  Fills in the Device Characteristics Parameters
*               for the devices defined by this UIM
*
*    Operation: Fills in the Device Characteristics Parameters
*               for the devices defined by this UIM and calls the
*               DCT Build Routine to create a DCT entry
*
*********************************************************************
*
*              DCT Build Routine
*
*        This routine initializes the DCT build parameter list and
*        then calls the DCT Build Service Routine
*
*********************************************************************
*
BUILDDCT DS    0H
         ST    R14,SAVWORD1         Place return address in savearea.
*********************************************************************
*
*          Builds the device characteristics table entry for a
*          33GG device.
*
*********************************************************************
         XC    DCP,DCP              Zero out DCT build parameter list.
         MVC   DCPID,DCPIDNM        Insert control block ID
         MVI   DCPTYPE,DCP3390      Sets index into DCT
         MVC   DCPLNGTH,=AL1(DCPEND-DCPENTRY) Sets length of DCP entry.
         MVC   DCPCYL,=H'1113'      Sets physical number of cylinders
*                                  per volume.
         MVC   DCPTRK,=H'15'        Sets number of tracks per cylinder.
         MVC   DCPOVR0,=H'1428'     Sets record 0 overhead.
         MVC   DCPSECT,=AL1(224)    Sets total number of records
         MVC   DCPBPSEC,=H'272'     Sets bytes per sector.
         MVC   DCPTRKLN,=AL2(58786) Sets number of bytes per track.
         OI    DCPFLAGS,DCPMODU     Indicates track requires modulo
*                                  arithmetic.
         MVC   DCPMOD1,=H'34'       Sets modulo factor.
         MVC   DCPALT,=H'15'        Sets number of alternate tracks.
         SPACE 1
         ST    UCAPTR,PARMAREA      Store address of UCA in first word
*                                  of parmarea.
         LA    R0,DCP               Get address of DCP
         ST    R0,PARMAREA+4        Store address of DCP in second word
*                                  of parmarea.
         LA    R1,PARMAREA          Store address of parameter list
*                                  in register 1
         L     R15,UCADCTP          Pick up entry point address of     *
                                    DCT Build Routine
```

```
         BALR  R14,R15              Call routine to build DCTs
         SPACE 1
*********************************************************************
*
*          Builds the device characteristics table entry for a
*          93GG device.
*
*********************************************************************
         XC    DCP,DCP              Zero out DCT build parameter list.
         MVC   DCPID,DCPIDNM        Insert control block ID
         MVI   DCPTYPE,4            DCPTYPE of 9345 used
         MVC   DCPLNGTH,=AL1(DCPEND-DCPENTRY) Sets length of DCP entry.
         MVC   DCPCYL,=H'1440'      Sets physical number of cylinders
*                                   per volume.
         MVC   DCPTRK,=H'15'        Sets number of tracks per cylinder.
         MVC   DCPOVR0,=H'1184'     Sets record 0 overhead.
         MVC   DCPSECT,=AL1(213)    Sets total number of records
         MVC   DCPBPSEC,=H'238'     Sets bytes per sector.
         MVC   DCPTRKLN,=AL2(48280) Sets number of bytes per track.
         OI    DCPFLAGS,DCPMODU     Indicates track requires modulo
*                                   arithmetic.
         MVC   DCPMOD1,=H'35'       Sets modulo factor.
         MVC   DCPALT,=H'15'        Sets number of alternate tracks.
         SPACE 1
         ST    UCAPTR,PARMAREA      Store address of UCA in first word
*                                   of parmarea.
         LA    R0,DCP               Get address of DCP
         ST    R0,PARMAREA+4        Store address of DCP in second word
*                                   of parmarea.
         LA    R1,PARMAREA          Store address of parameter list
*                                   in register 1
         L     R15,UCADCTP          Pick up entry point address of     *
                                    DCT Build Routine
         BALR  R14,R15              Call routine to build DCTs
         SPACE 1
         L     R14,SAVWORD1         Restore mainline's return address
         BR    R14                  and return to mainline
*
         EJECT
*
*********************************************************************
*
*    Procedure: BUILDCIT
*
*    Descriptive Name: Build Control Unit Information Parameter
*
*    Function:  Fills in the Control Unit Information Parameters
*
*    Operation: Fills in the Control Unit Information Parameters
*               and calls the CIT Build Routine to create the CIT.
*
*********************************************************************
*
*  The Control Unit Information Table CIT is required by HCD
*  for the validation of a control unit definition.
*  For each CU type, a separate CIT is required.
*  Only control unit types defined by any CIT can be configured
*  with HCD.
*
*  The CIT contains parameters such as:
*      type, model, attachment information, minimum/maximum values,
*      default values
*
*  You can view many of the CIT values in the HCD "Supported
*  Control Units" panel, and in dialog panels used to define
*  control units.
*
*  The UIM must set the Control Unit Information Parameters CIP
*  and call the HCD routine to build the CIT.
*  The CIP information is stored in the CIT.
*
*  The CIP layout is defined in the macro CBDZCIP.
*
*********************************************************************
*  Following CIP fields are not set by this sample UIM
*
*  Name       Description                     Remarks
*  ----------|-------------------------------|-----------------------
*  CIPFLAG   |Flag byte                      |
*   CIPFCUD  |If 1, device and CU are        |
*            |physically the same            |
*   CIPFDMOD |If 1, this model is the default|If the user does not
```

```
*            |model for this CU              |specify a CU model,
*            |                               |HCD uses the model
*            |                               |defined in this CIP
*            |                               |as default model
*            |                               |
*  CIPPRFLG  |Processing flags               |HCD internal use only,
*            |                               |the UIM must not set
*            |                               |
*  CIPRUAN   |recommended number of          |
*            |unit addresses                 |
*            |(contains hex zero, if no value|
*            |is defined)                    |
*            |                               |
*  CIPMINDV  |minimum number of devices      |
*            |connected to the control unit  |
*            |(contains hex zero, if no value|
*            |is defined)                    |                      @H4A
*            |                               |
*  CIPCUTYP  |CU type                        |
*   CIPCUCTC |CU type = CTC                  |Set for CTC virtual CU
*            |                               |only
*   CIPCUSWI |CU type = Switch               |Set for ESCON director
*            |                               |virtual CU only
*   CIPCUOSA |CU type = OSA                  |Set for open systems
*            |                               |adapter only          @H4A
*            |                               |
*  CIPLMXNO  |Maximum number of logical      |
*            |control units supported        |
*            |(contains hex zero, if no value|
*            |is defined)                    |
*            |                               |
*  CIPUADEF  |Unit address rules for control |
*            |unit                           |
*   CIPMINUA |Min. number of unit addresses  |The user must define at
*            |                               |least this # of unit
*            |                               |addresses.
*            |                               |The user should specify
*            |                               |at least this number of
*            |                               |devices. Otherwise, at
*            |                               |build production IODF
*            |                               |time, a warning message
*            |                               |is issued.
*            |                               |
*   CIPMAXUA |Max. number of unit addresses  |The user must not define
*            |                               |more than this # of unit
*            |                               |addresses.
*   CIPMXUAR |Maximum number of unit address |
*            |ranges supported by CU         |
*            |                               |
*  CIPVALUA  |Valid unit addresses; unit     |Can only be set if
*            |addresses are mapped to bit    |CIPEXTPT is set; must be
*            |array.                         |set if CIPEXTPT is set.
*            |                               |                      @H4A
*            |                               |
*  CIPREQUA  |Unit addresses required to be  |Can only be set if
*            |defined for the control unit;  |CIPEXTPT is set; must be
*            |unit addresses are mapped to   |set if CIPEXTPT is set.
*            |bit array.                     |                      @H4A
*            |                               |
*  CIPPROUA  |Unit addresses recommended to  | Can only be set if
*            |be defined for the control unit| CIPEXTPT is set; must be
*            |                               | set if CIPEXTPT is set.
*            |                               |                      @H4A
*   CIPRUARN |Number of address ranges in the| A maximum of 8 unit
*            |array of recommended unit      | address ranges may be
*            |addresses                      | specified.           @H4A
*   CIPRUARS |Structure of recommended unit  | Array of 8           @H4A
*            |address ranges                 |                      @H4A
*    CIPRUA  |Starting unit address          |                      @H4A
*    CIPRUARF|Replication factor             | Specify the range value
*            |                               | reduced by 1.        @H4A
*            |                               |
*  CIPLPATH  |Logical path information       |
*   CIPMXPTH |max. number of logical paths   |
*            |supported by the control unit. |
*            |(contains hex zero if no max.  |
*            |is defined)                    |
*   CIPMNGRP |Minimum group attachment value | If CIPMXPTH is set to
*            |The counted number of logical  | a value greater than
*            |paths per channel path type    | zero, CIPMNGRP must
*            |is rounded to the next multiple| also be set to a value
*            |of this value before the check | greater than zero.
```

```
*           |against CIPMXPTH is performed. |
*           |                                |
*  CIPEXTPT  |Pointer to CIP extension area  | If CIPEXTPT is set the
*           |(contains hex zero, if no value| following fields must
*           |is defined)                    | be set, too:
*           |                               | CIPVALUA, CIPREQUA,
*           |                               | CIPPROUA            @H4A
*********************************************************************
*********************************************************************
*
*              CIT Build Routine
*
*        This routine initializes the CIT build parameter list and
*        then calls the CIT Build Service Routine.
*
*********************************************************************
BUILDCIT DS    0H
         ST    R14,SAVWORD1         Place return address in savearea.
         SPACE 1
*********************************************************************
*
*          Builds CIP for control unit 93CC.
*          Note: The CBDZCIP macro already initializes the
*                CIP data structure.
*
*********************************************************************
*  Following registers the control unit type '93CC' w/o model
*  number to HCD.
*  In HCD, this control unit type must be defined to identify the CU.
*-------------------------------------------------------------------
*
         MVC   CIPUNIT,CNTL93CC     Place control unit name in CIP
         XC    CIPMODL,CIPMODL      Indicate that the control unit does*
                                    not have a model number
*
*-------------------------------------------------------------------
*  In the HCD "List Supported Control Units" panel and in the
*  control unit prompt panels, the CUs are grouped
*  in order to facilitate the navigation among many CU types.
*
*  HCD uses the CIPGROUP parameter to associate a CU type to
*  a certain CU group such as DASD or tape control units.
*  Refer to macro CBDZCIP for available group values.
*
*  The following associates the control unit to the group of DASD CUs.
*-------------------------------------------------------------------
*
         MVC   CIPGROUP,=A(CIPGDASD) Get control unit group for this CU*
                                    and store into CIP
*
*-------------------------------------------------------------------
*  The UIM can define which channel protocols are supported
*  by the CU and which protocol is default.
*  Every protocol is represented by a bit, multiple protocols can
*  be defined as being supported, but only one default protocol.
*  For available protocols and defaults refer to CBDZCIP.
*
*  Here, '3.0 MB data streaming' and '4.5 MB data streaming' is set,
*  as default '3.0 MB data streaming' is defined.
*-------------------------------------------------------------------
*
         MVI   CIPSPROT,CIPSPSTR+CIPSP4MB       Set supported protocols*
                                    for this CU
         MVI   CIPDPROT,CIPDPDS     Set default protocol  =
*                                          data streaming protocol
*
*-------------------------------------------------------------------
*  CIPATTT defines to which channel path types a CU is attachable.
*  Every channel path type is represented by a bit, multiple channel
*  path types can be defined, for all possible channel path types
*  refer to field CIPATTT in macro CBDZCIP.
*
*  Following, the channel path types for BL, CNC and CVC are set.
*-------------------------------------------------------------------
*
         MVC   CIPATTT,=AL2(CIPATBL+CIPATSER+CIPATFX)                 *
                                    Sets attachment information for    *
                                    this CU.
*
*-------------------------------------------------------------------
*  CIPUADEF defines the unit address rules for the control unit.
*  - CIPMINUA specifies the minimum number of unit addresses that
```

```
*    must be assigned to a control unit when defining it.
*  - CIPMAXUA specifies the maximum number of unit addresses that
*    can be assigned to a control unit when defining it.
*  - CIPMXUAR specifies the maximum number of unit addresses ranges
*    that can be assigned to a control unit when defining it.
*
*  HCD validates these rules and rejects any definitions not
*  adhering to them.
*
*  The following statements specify that for a 93CC control unit:
*  - at least 32 unit addresses must be specified
*  - at most  64 unit addresses can be specified
*  - at most   1 unit address range can be specified
*--------------------------------------------------------------------
*
         MVC   CIPMINUA,MINUA32   Specifies that at least 32 unit      *
                                    addresses must be specified.
         MVC   CIPMAXUA,MAXUA64   Specifies that at most  64 unit      *
                                    addresses can  be specified.
         MVC   CIPMXUAR,MAXUAR1   Specifies that at most 1 unit        *
                                    address range can be specified.
*
*--------------------------------------------------------------------
*  The following defines the rules and limits for the CUADD parameter.
*  CUADD parameters apply to ESCON control units which support
*  logical addressing. IBM processors allow a maximum range
*  of logical addresses of 0..15.
*
*  The next instructions define the following for control unit 93CC:
*  - Setting CIPLFCUS  indicates, that this CU supports logical
*    addresses
*  - Setting CIPLFRS indicates, that a CUADD range is defined.
*  - The minimum value for CUADD is set to 0
*  - The maximum value for CUADD is set to 8 (the highest value
*    which can be set into CIPLMAX is 15).
*--------------------------------------------------------------------
*
         MVI   CIPLFLGS,CIPLFRS+CIPLFCUS   Sets Logical CU addressing  *
                                           flags
         MVI   CIPLMIN,0            Sets minimum value of allowed CUADD
         MVI   CIPLMAX,8            Sets maximum value of allowed CUADD
*
*--------------------------------------------------------------------
*  The default I/O concurrency level is correlated to the "SHARED"
*  parameter of an IOCP CNTLUNIT macro instruction.
*  For all available defaults refer to CBDZCIP.
*
*  Next, a default value of type 2 is specified, which means SHARED=NO,
*  multiple I/O requests are allowed.
*--------------------------------------------------------------------
*
         MVI   CIPDIOCL,CIPDIOT2    Set default I/O concurrency level
*
*--------------------------------------------------------------------
*  The following defines that HCD checks for unit address range
*  starting with X'00', if the CU is attached to an ESCON channel
*--------------------------------------------------------------------
*
         OI    CIPVALF,CIPUAES0     Indicates that unit address range
*                                   should start with X'00', if
*                                   connected to an ESCON channel path
*
*--------------------------------------------------------------------
*  HCD checks, that not more devices are attached to this CU,
*  than defined in CIPMXDEV (if the value is not zero).
*--------------------------------------------------------------------
*
         MVC   CIPMXDEV,MAX64       A maximum number of 64 devices
*                                   can be attached to the CU.
*
*--------------------------------------------------------------------
*  Initialize the attachable device list showing all devices which
*  can be attached to the control unit concerned.
*  The type-models in the list must be registered by a UIT.
*
*  With the model parameter CIPADEVM you can determine how a
*  device type is recognized in HCD.
*  A) When you specify an explicit model number, this number must
*     be defined in HCD.
*  B) When you specify a blank character string X'40', then
*     the device must be specified without model (only device type).
*  C) When you specify X'00' this will work like a "wild card"
```

```
*     character, which means any model can be specified, provided
*     it is defined by a UIT.
*
*  Following, a device list containing 1 entry is specified,
*  with the device type parameter CIPADEVT set to 93UU, and
*  the model parameter CIPADEVM set to X'40', see case B).
*--------------------------------------------------------------------
*
         L     R1,CIPDVLP           Pick up device address of dev list
         USING CIPADEVS,R1          Establish addressability
         MVC   CIPADEVT,UNIT93UU    Sets device unit.
         MVC   CIPADEVM,BLANKS      Device has no model.
         DROP  R1
         MVC   CIPDVLC,=F'1'        Sets count of devices in attachable
*                                   device list.
         SPACE 1
*
*--------------------------------------------------------------------
* Call CIT Build Service Routine
*--------------------------------------------------------------------
*
         SPACE 1
         ST    UCAPTR,PARMAREA      Store address of UCA in first word
*                                   of parmarea.
         LA    R0,CIP               Get address of CIP
         ST    R0,PARMAREA+4        Store address of CIP in second word
*                                   of parmarea.
         LA    R1,PARMAREA          Store address of parameter list
*                                   in register 1
         L     R15,UCACITP          Pick up entry point address of     *
                                    CIT Build Routine
         BALR  R14,R15              Call routine to build CITs
         SPACE 1
**********************************************************************
*
*   BUILDS CIP FOR Control unit 39CC-6.
*
**********************************************************************
*
* Since the CIP is used for another control unit, it has to
* be initialized again.
*--------------------------------------------------------------------
         XC    CIP(CIPADEV-CIP),CIP Zeroes out CIP list.
         MVC   CIPID,=CL4'CIP '     Sets storage descriptor in header.
         MVI   UIPGVER,X'01'        Sets CIP version code.
         LA    R2,CIPADEV           Address of attachable device list
         ST    R2,CIPDVLP           Store it into pointer
*--------------------------------------------------------------------
* Fills in the values for control unit 39CC-6.
*--------------------------------------------------------------------
         MVC   CIPUNIT,CNTL39CC     Place control unit name in CIP.
         MVC   CIPMODL,MODL6        Place control unit model in CIP.
         MVC   CIPGROUP,=A(CIPGDASD) Get control unit group for this CU*
                                    and store into CIP
         MVI   CIPSPROT,CIPSPSTR+CIPSP4MB      Set supported protocols *
                                    for this CU
         MVI   CIPDPROT,CIPDPDS     Set default protocol  =
*                                         data streaming protocol
         MVC   CIPATTT,=AL2(CIPATBL+CIPATSER+CIPATFX)                 *
                                     Set attachment information for    *
                                    this CU
*
         MVC   CIPMINUA,MINUA2    Specifies that at least 2  unit      *
                                     addresses must be specified.
         MVC   CIPMAXUA,MAXUA64   Specifies that at most  64 unit      *
                                     addresses can  be specified.
         MVC   CIPMXUAR,MAXUAR1   Specifies that at most 1 unit        *
                                     address range can be specified.
*
*
         MVI   CIPLFLGS,CIPLFRS+CIPLFCUS   Sets Logical CU addressing  *
                                           flags.
         MVI   CIPLMIN,0            Sets minimum value of allowed CUADD
         MVI   CIPLMAX,15           Sets maximum value of allowed CUADD
*
         MVI   CIPDIOCL,CIPDIOT2    Set default I/O concurrency level.
         OI    CIPVALF,CIPUAES0     Indicates that unit address        *
                                    should start with 00 if connected  *
                                    to an ESCON channel path.
*
*--------------------------------------------------------------------
*  Initialize attachable device list showing all devices which
```

```
*  can be attached to the control unit concerned
*
*  In following a device list containing 3 entries is specified.
*
*  Entry    Device Type      Device Model
*  1        '33UU'           '1'
*  2        '33UU'           '2'
*  3        '33VV'           X'40', no model
*---------------------------------------------------------------------
*
*---------------------------------------------------------------------
*
         L     R1,CIPDVLP           Pick up address of device list.
         USING CIPADEVS,R1          Establish addressability.
         MVC   CIPADEVT,UNIT33UU    Set device unit.
         MVC   CIPADEVM,MODL1       Set device model.
         LA    R1,CIPADEVL(R1)      Proceed to next entry.
         MVC   CIPADEVT,UNIT33UU    Set device unit.
         MVC   CIPADEVM,MODL2       Set device model.
         LA    R1,CIPADEVL(R1)      Proceed to next entry.
         MVC   CIPADEVT,UNIT33VV    Set device unit.
         MVC   CIPADEVM,BLANKS      Device has no model.
         DROP  R1
         MVC   CIPDVLC,=F'3'        Set count of devices in attachable
*                                   device list.
         SPACE 1
*---------------------------------------------------------------------
*        Call CIT Build Service Routine
*---------------------------------------------------------------------
         SPACE 1
         ST    UCAPTR,PARMAREA      Store address of UCA in first word
*                                   of parmarea.
         LA    R0,CIP               Get address of CIP
         ST    R0,PARMAREA+4        Store address of CIP in second word
*                                   of parmarea.
         LA    R1,PARMAREA          Store address of parameter list
*                                   in register 1.
         L     R15,UCACITP          Pick up entry point address of     *
                                    CIT Build Routine.
         BALR  R14,R15              Call routine to build CITs.
         SPACE 1
         L     R14,SAVWORD1         Restore mainline's return address
         BR    R14                  and return to mainline
     EJECT
*
**********************************************************************
*
*    Procedure: FEATCHEK
*
*    Descriptive Name: Feature Check Routine
*
*    Function:  Validity checks the specified features, sets
*               return code, and issues message.
*
*    Operation: Checks whether mutually exclusive features
*               are specified.
*               If an error occurred
*                 - sets error return code in UCA
*                 - sets a field in the UCA to indicate that
*                   the error occurred during feature check
*                 - sets a field in the UCA to indicate which
*                   feature is in error
*                 - Invokes macro CBDIMSG to issue an error
*                   message
*
*    Input:     Information contained in the internal text record
*               CBDZIODV
*
*    Output:    - Fields set in UCA
*               - Message  CBDB805I
*
**********************************************************************
FEATCHEK DS    0H
         ST    R14,SAVWORD1         Place return address in savearea.
         SPACE 1
         TM    IODVFEA1,FEATSHR+FEATSHUP  If mutually exclusive
*                                   features specified, writes message.
         BNO   FEATLBL1             .. No, return
         MVC   UCAPID,=H'4'         Sets ID of feature
*                                   where error occurred.
         MVC   UCAPPOS,=H'3'        Sets ID of feature where
*                                   error occurred.
```

```
         LA    R0,UCARCERR          Sets error return code
         ST    R0,UCARETC           into UCA parameter list.
*
*-------------------------------------------------------------------
*  The message issued below must be defined in the corresponding UDT
*-------------------------------------------------------------------
*
         CBDIMSG MID=CBDB805I,VAR1=IODVUNIT,STMT=YES,REQ=UIM
*                                  Issues message CBDB805I
         SPACE 1
FEATLBL1 DS    0H
         L     R14,SAVWORD1         Restore mainline's return address
         BR    R14                  and return to mainline
*
         EJECT
*
*********************************************************************
*
*    Procedure: UADDCHEK
*
*    Descriptive Name: Unit Address Check Routine
*
*    Function:  Validity checks the specified unit address, sets
*               return code, and issues message.
*
*    Operation: Checks for a 93UU device, whether the first digit
*               of the starting unit address is even.
*               If an error occurred
*                 - sets error return code in UCA
*                 - sets a field in the UCA to indicate that
*                   the error occurred during unit address check
*                 - Invokes macro CBDIMSG to issue an error
*                   message
*
*    Input:     Information contained in the internal text record
*               CBDZIODV
*
*    Output:    - UCA field: UCARETC
*               - Message CBDB814I
*
*********************************************************************
UADDCHEK DS    0H
         ST    R14,SAVWORD1         Place return address in savearea.
         SPACE 1
         CLC   IODVUNIT,UNIT93UU    Is it a 93UU device?
         BNE   UADDLBL1             .. No, return
         TM    IODVUNIA,X'10'       Check if low order bit of
*                                   the first digit is B'1'
*                                   in unit address of IODV record.
         BZ    UADDLBL1             .. No, return
         LA    R0,UCARCERR          Set error return code
         ST    R0,UCARETC           into UCA parameter list.
         CBDIMSG MID=CBDB814I,VAR1=(IODVUNIA,H),VAR2=IODVUNIT,         *
               STMT=YES,REQ=UIM
*                                   Issue message CBDB814I.
         SPACE 1
UADDLBL1 DS    0H
         L     R14,SAVWORD1         Restore mainline's return address
         BR    R14                  and return to mainline.
*
         EJECT
*
*********************************************************************
*
*    Procedure: BUILDDFT
*
*    Descriptive Name: Build Device Feature Parameter
*
*    Function:  Fills in the Device Feature Parameter
*
*    Operation: Fills in the Device Feature Parameter
*               and calls the DFT Build Routine to create
*               the DFT
*
*    Input:     Information supplied in the UIM
*               IODV record
*
*    Output:    DFP - Device Feature Parameter
*
*********************************************************************
*
*  The Device Feature Parameters DFP are used for building
```

```
*  the UCBs. See the comments in macro CBDZDFP.
*
*  For units configured with HCD, the values of major DFP
*  fields can be seen in the HCD "MVS Device Report".
*
*********************************************************************
*
*              DFT Build Routine
*
*********************************************************************
*
*
BUILDDFT DS    0H
         ST    R14,SAVWORD1         Place return address in savearea.
         XC    DFP,DFP              Zero out the DFP.
*
*-------------------------------------------------------------------
*  UCBs, beside the common parts, optionally may have following
*  sections:
*      Device Dependent Segment
*      Device Class Extension
*      Device Dependent Extension
*
*  As required for the DFP parameters, pointers and lengths
*  for the optional fields must be set.
*  Note:
*  In this Sample UIM no Device Dependent Extension is defined
*-------------------------------------------------------------------
*
         XC    DEVDPSEG,DEVDPSEG    Zero out the device dependent
*                                   segment.
         XC    DEVCESEG,DEVCESEG    Zero out the device class extension
         LA    R1,DEVCESEG
         LA    R0,L'DEVCESEG        Get the length of the device
*                                   class extension.
         ST    R0,DFPDCEL           Set length of device class
*                                   extension in DFP.
         LA    R0,DEVCESEG          Get address of device class
*                                   extension
         ST    R0,DFPDCEP           Set pointer to device class
*                                   extension in DFP.
         LA    R0,L'DEVDPSEG        Get the length of the device
*                                   dependent segment.
         ST    R0,DFPDDSL           Set length of device dependent
*                                   segment in DFP.
         LA    R0,DEVDPSEG          Get address of device dependent
*                                   segment.
         ST    R0,DFPDDSP           Set pointer to device dependent
*                                   segment in DFP.
         OI    DFPFL5,DFPDCC        Disconnect command chain device.
         OI    DFPFL5,DFPENVRD      Device returns environmental data.
         OI    DFPFL6,DFPIOT        Device supports I/O timing.
         MVC   DFPID,DFPCBID        Place control block ID in DFP.
         MVC   DFPVER,DFPVERN       Place the version number in DFP.
*
*-------------------------------------------------------------------
*  Following, the UCB TYPE value is set depending on the unit type
*  passed in the IODV record.
*  The UCB TYPE used here must match the value defined in the
*  GIT of the Generic Device to which the unit is associated.
*-------------------------------------------------------------------
*
         CLC   IODVUNIT,UNIT33UU    Is it a 33UU device?
         BE    DFTLBL0              .. Yes, device type found
         CLC   IODVUNIT,UNIT33VV    Is it a 33VV device?
         BNZ   DFTLBL1              .. No, continue check
DFTLBL0  DS    0H                   Start initializing DFT.
         MVC   DFPNAME,GEN33GG      Place generic name of device in DFP
         MVC   DFPUCBTY,GNRCTYP1    Initialize DFT UCB type.
         B     DFTLBL2              Continue to setting further
*                                   information.
DFTLBL1  DS    0H
         CLC   IODVUNIT,UNIT93UU    Is it a 93UU device?
         BNZ   DFTLBL3              .. No, do not set values
         MVC   DFPNAME,GEN93GG      Place generic name of device in DFP
         MVC   DFPUCBTY,GNRCTYP2    Initialize DFT UCB type.
DFTLBL2  DS    0H
         OI    DFPFLP1,DFPDYNPH     Indicate that dynamic pathing
*                                   feature is supported by the device.
         OI    DFPFLAG1,DFPPRES     Indicates that device is
*                                   permanently resident.
         MVI   DFPATI,ATTNINDX      Sets attention table index.
```

```
         MVI   DFPSNSCT,SENSBYT#    Initialize number of sense bytes.
         MVI   DFPDSTCT,STATETY#    Initialize count of statistics
*                                   table entries.
DFTLBL3  DS    0H
         MVI   DFPETI,ERPINDEX      Sets ERP index.
         TM    IODVFLG1,IODVFOFF    Offline parameter specified
*                                   at all ?
         BZ    DFTLBL4              .. No, skip ahead
         TM    IODVPFLG,IODVPOFF    Offline set on ?
         BZ    DFTLBL4              .. No, skip ahead
         OI    DFPFLAG1,DFPOFFLN    Offline, if specified
DFTLBL4  DS    0H
         TM    IODVFEAT,FEATACTL    Is ALTCTRL feature set?
         BZ    DFTLBL5              .. No, skip ahead
         OI    DFPFL5,DFPALTCU      Include feature in DFP.
DFTLBL5  DS    0H
         TM    IODVFEAT,FEATSHUP    Is SHAREDUP feature specified?
         BZ    DFTLBL6              .. No, skip ahead
         OI    DFPFLP1,DFPSHRUP     Set shareable in UP mode.
DFTLBL6  DS    0H
         TM    IODVFEAT,FEATSHR     Is SHARED feature specified?
         BZ    DFTLBL7              .. No, skip ahead
         OI    DFPTBYT2,DFPRR       Indicate device is shareable
*                                   between processors.
DFTLBL7  DS    0H
         SPACE 1
*********************************************************************
*  This loop manages successive calls to the DFT build routine
*********************************************************************
*
*  On a single build DFT request, the UIM can be requested
*  to build DFTs for multiple devices of the same type.
*  In IODVDNBR the starting device number is set.
*  In IODVNBRD the number of devices is set.
*
*  The value of IODVNBRD is used in the loop variable R9
*  of below DFTLOOP.
*-------------------------------------------------------------------
*
LOOPINIT DS    0H
         LH    R9,IODVNBRD          Obtain number of requested devices
         XR    R10,R10              Clear register for subsequent ICM.
         ICM   R10,3,IODVDNBR       Obtain device number.
*
DFTLOOP  DS    0H
         ST    R10,DFPDNBR          Establish device number to be sent
*                                   to DFT build routine.
         ST    UCAPTR,PARMAREA      Initialize parameter area.
         LA    R0,DFP               Get address of DFP
         ST    R0,PARMAREA+4        and store address in second word
*                                   of parmarea.
         LA    R1,PARMAREA          Store address of parameter list
*                                   in register 1.
         L     R15,UCADFTP          Obtain the entry point address from
*                                   the UCA
         BALR  R14,R15              Call routine to build DFT.
         LTR   R15,R15              Bad return code from DFT build ?
         BNZ   DONEDFTB             ..Yes, do not make any more calls
*                                   to build DFTs.
         A     R10,ONE              Increment device number.
         BCT   R9,DFTLOOP           Cycle until every device defined.
         SPACE 1
DONEDFTB DS    0H
         L     R14,SAVWORD1         Restore return address from
*                                   savearea.
         BR    R14                  Return to main procedure.
*
         EJECT
*********************************************************************
*
*  The following  word serves as savearea for register 14 when
*  internal procedures are called.
*
*********************************************************************
*
SAVWORD1 DS    F
*
*********************************************************************
*
*  The first 2 of the following words serve as this module's
*  parameter area for external calls.
*  The next 18 words serve as the module savearea.
```

```
*
*********************************************************************
*
PARMAREA DS    2F
SAVAREA  DS    18F
*
*********************************************************************
*
*  Device dependent constants
*
*********************************************************************
*
ONE      DC    F'1'                Constant one
MAX64    DC    F'64'               Constant 64
MINUA2   DC    H'2'                Constant 2
MINUA32  DC    H'32'               Constant 32
MAXUA64  DC    H'64'               Constant 64
MAXUAR1  DC    H'1'                Constant 1
*--------------------------------------------------------------------
* Definition of message ids used for validation checks.
*--------------------------------------------------------------------
CBDB805I DC    CL8'CBDB805I'       Message id.
CBDB814I DC    CL8'CBDB814I'       Message id.
*--------------------------------------------------------------------
* Definition of generic names.
*--------------------------------------------------------------------
GEN33GG  DC    CL8'33GG    '       Generic name 33GG.
GEN93GG  DC    CL8'93GG    '       Generic name 93GG.
*--------------------------------------------------------------------
* Definition of device units and models
*--------------------------------------------------------------------
UNIT33UU DC    CL8'33UU    '       Device type 33UU.
UNIT33VV DC    CL8'33VV    '       Device type 33VV.
UNIT93UU DC    CL8'93UU    '       Device type 93UU.
MODL1    DC    CL4'1   '           Model 1.
MODL2    DC    CL4'2   '           Model 2.
BLANKS   DC    CL4'    '           No Model.
*--------------------------------------------------------------------
* Definition of control unit types and models
*--------------------------------------------------------------------
CNTL39CC DC    CL8'39CC    '       Control unit type 39CC.
MODL6    DC    CL4'6   '           Model 6.
CNTL93CC DC    CL8'93CC    '       Control unit type 93CC.
*********************************************************************
* Definitions for private parameter DASDPOOL
*--------------------------------------------------------------------
DASDPPRM EQU   X'80'               Bit mask for DASDPOOL parameter
*--------------------------------------------------------------------
DASD_PID DC    AL2(33)             Parameter ID for DASDPOOL
*                                  parameter (must correspond
*                                  to specification in UDT).
*--------------------------------------------------------------------
DASD_DEF DC    CL8'DSP1    '       Default value for DASDPOOL
*                                  parameter.
*--------------------------------------------------------------------
DASD_SEL DS    0CL24               Parameter selection list values
*                                  for DASDPOOL parameter.
*                                  The values must be contiguous
*                                  using the same length.
DASD_SP1 DC    CL8'DSP1    '       Selection value for DASDPOOL
*                                  parameter.
         DC    CL8'DSP2    '       Selection value for DASDPOOL
*                                  parameter.
         DC    CL8'*       '       Selection value for DASDPOOL
*                                  parameter.
*********************************************************************
* *
*  DDT, MLT, UCB type values are required for building the
*  UCBs.
*  For units configured with HCD, the values of DDT, MLT and
*  UCB types can be seen in the HCD "MVS Device Report".
*--------------------------------------------------------------------
*
NAMEDDT  DC    CL8'IECVDDT5'       DDT name
NAMEMLT  DC    CL8'IEAMLT33'       MLT name
GNRCTYP1 DC    XL4'3010200C'       UCB type
GNRCTYP2 DC    XL4'3010200E'       UCB type
*
*--------------------------------------------------------------------
*  The generic preference value for a generic device must
*  be UNIQUE, which means no other generic device in the
*  same MVS must have the same value.
```

```
*  For preference values used by IBM units, refer to the
*  appendix of "z/OS MVS Device Validation Support"
*-------------------------------------------------------------------
*
GNRCPRT1 DC    F'99981'             Generic preference value
GNRCPRT2 DC    F'99984'             Generic preference value
*
STATETY# EQU   1                    Number of statistics table entries
SENSBYT# EQU   2                    Number of sense bytes
ERPINDEX EQU   0                    ERP index.
ATTNINDX EQU   64                   Attention table index.
*
*-------------------------------------------------------------------
*  Features set in the UIT are those features which HCD
*  recognizes for the device.
*  The UIP field for supported features UIPMSFEA is defined
*  for each UIM exclusively.
*  All Feature definitions like FEATACTL, FEATSHR must be
*  defined also with CBDZUDT statements in the corresponding UDT
*  The features in the UIM and in the UDT must be defined
*  in the same sequence.
*
*  The following bit masks are used to set the UIP field UIPMSFEA.
*-------------------------------------------------------------------
*
FEATACTL EQU   X'80'                ALTCTRL feature value.
FEATSHR  EQU   X'40'                SHARED  feature value.
FEATSHUP EQU   X'20'                SHAREDUP feature value.
*
DEVCESEG DS    CL40                 Device class extension
DEVDPSEG DS    CL16                 Device dependent segment
         LTORG *                    Define literals here
         EJECT ,
*********************************************************************
*
*  Register equates
*
*********************************************************************
*
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
IODVPTR  EQU   5                    IODV pointer.
UCAPTR   EQU   6                    UCA pointer.
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11                   Base register
R12      EQU   12
R13      EQU   13                   Save area address
R14      EQU   14
R15      EQU   15
*
*********************************************************************
*
*  This macro invocation generates a diagnostic stack entry
*
*********************************************************************
*
DIAGDATA CBDZDIAG MODNAME=CBDUC255,CSECT=CBDUC255,COMP=SC1XL,          X
               MODCAT=UIM,DESC='UIM FOR DASD 33UU, 93UU'           @H1C
         EJECT
*********************************************************************
* Storage declaration for control unit information parameters (CIP).
*********************************************************************
* The CBDZCIP macro maps the control unit information parameters
* (CIP).
*
*        It generates an initialized structure of the CIP.
*
* DEV       specifies the number of entries to be generated in
*           the attachable device list for the control unit.
*
*
* Note:
*  If the CIP is re-used for another control unit definition, it
*  has to be re-initialized as shown in this sample UIM.
*
*-------------------------------------------------------------------
```

```
*
         CBDZCIP DEV=3
         EJECT
         CBDZDCP
         EJECT
         CBDZDFP
         EJECT
         CBDZGIP
         EJECT
*********************************************************************
* Storage declaration for unit information parameters (UIP).
*********************************************************************
* The CBDZUIP macro maps the unit information parameters (UIP).
*
* TYPE=GEN  generates an initialized structure of the UIP.
*
* MLTS      specifies the number of entries to be generated in
*           the module lists table (MLT). The specified number
*           must be between 1 and 5; the default is 1.
*
* DFLT      specifies the number of entries that are generated
*           in the parameter default list. This list contains
*           information about default values of parameters.
*           The default value is initially shown for the parameter
*           in the HCD dialog when defining the corresponding device
*           for the operating system.
*
* SEL       specifies the number of entries that are generated
*           in the parameter selection list. If a parameter
*           selection list is specified, HCD provides a prompt
*           for the corresponding parameter showing the values of
*           the parameter selection list. The parameter selection
*           list is also used by HCD to check for the possible
*           values of a parameter.
*
* SIM       specifies the number of entries that are generated
*           in the similar device list. This list identifies,
*           by device types and models, those devices which have
*           the same characteristics as the device named in the UIP.
*
* Note:
*  If like in this UIM, the UIP is cleared before it is re-used
*  for the next UIT to be build, then the fields initialized by this
*  macro, must be refreshed by program.
*
*--------------------------------------------------------------------
*
         CBDZUIP TYPE=GEN,MLTS=1,DFLT=1,SEL=1,SIM=2
         EJECT
*********************************************************************
* Storage declaration for message service routine parameter list
* (MSGR).
*********************************************************************
         CBDZMSG ,
         EJECT
*********************************************************************
* Mapping of the Control unit Information Parameter list (CIP).
*********************************************************************
* The CBDZCIP macro maps the control unit information parameters
* (CIP).
*
* TYPE=DSECT provides mappings for attachable device list.
*--------------------------------------------------------------------
*
         CBDZCIP TYPE=DSECT
         EJECT
         CBDZITRH ,
         EJECT
         CBDZUCA ,
         EJECT
*********************************************************************
* Mapping of the Unit Information Parameters (UIP).
*********************************************************************
* The CBDZUIP macro maps the unit information parameters (UIP).
*
* TYPE=DSECT provides mappings for
*                                  the parameter default list,
*                                  the parameter selection list.
*--------------------------------------------------------------------
*
         CBDZUIP TYPE=DSECT
         END
```

```
//*********************************************************************
//JJJJJJ    JOB ,NOTIFY=UUUUUU,MSGLEVEL=(1,1),MSGCLASS=H
//ASMH    EXEC PGM=IEV90,REGION=1024K,PARM='LINECNT=55,DECK'
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DSN=XXXXXXXXXXXXXXX(CBDUC255),DISP=SHR
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SYSPUNCH  DD DSN=&OBJ(CBDUC255),DISP=(,PASS,DELETE),UNIT=SYSDA,;
//             SPACE=(TRK,(1,5,5))
//SYSGO     DD DUMMY
//SYSLIB    DD DSN=SYS1.AMODGEN,
//             DISP=SHR
//SYSLIN    DD UNIT=SYSDA,SPACE=(TRK,(30,10)),DISP=(NEW,PASS),
//             DSN=&POBJ;
//LKED    EXEC PGM=IEWL,PARM='AMOD=31,LET,LIST,NCAL,RMOD=ANY,XREF',
//             COND=(0,NE)
//SYSPRINT  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=121,BLKSIZE=1210)
//SYSLIN    DD DDNAME=SYSIN
//SYSLMOD   DD DSN=YYYYYYYYYYYYYYY(CBDUC255),DISP=SHR
//SYSUT1    DD DISP=(NEW,DELETE),SPACE=(CYL,(5,2)),UNIT=SYSDA
//OBJ       DD DSN=&OBJ,DISP=(OLD,DELETE);
//SYSIN     DD *
  INCLUDE OBJ(CBDUC255)
  NAME CBDUC255(R)
/*
```

Appendix B. Sample of a Unit Data Table (UDT)

```
******************************** TOP OF DATA ************************
*
*       The CBDSUDT member in SYS1.SAMPLIB can be used as a model
*       by customers when writing a Unit Data Table (UDT).
*
*       Instructions:
*
*       1) Define a name for your UDT which follows the format
*          CBDECxxx, with xxx between 001 and 256.
*          xxx must match the number used in the name CBDESxxx
*          used for the corresponding UIM.
*          Note: The sample UDT uses the number 255. If you like
*                to use another number, replace the number.
*
*       2) Copy this Sample UDT to a PDS member with the name
*          chosen for your UDT.
*
*       3) Change all strings "CBDEC255" in the UDT to the
*          chosen name.
*
*       4) Change the UDT according to your needs.
*
*       5) Separate the JCL at the end of the UDT,
*          and correct the names in the JCL.
*          Assemble and link-edit your UDT using the JCL.
*
*********************************************************************
*
***** START OF SPECIFICATIONS ****
*
*01* MODULE NAME = CBDEC255
*
*01* DESCRIPTIVE NAME = English Version of the Unit Data Table
*                       for UIM CBDSUIM
*
*********************************************************************
*                                                                   *
*  PROPRIETARY STATEMENT=                                           *
*   LICENSED MATERIALS - PROPERTY OF IBM                            *
*   THIS MODULE IS "RESTRICTED MATERIALS OF IBM"                    *
*   5655-068                                                        *
*   (C) COPYRIGHT IBM CORPORATION 1989, 1995                        *
*                                                                   *
*  END PROPRIETARY STATEMENT                                        *
*                                                                   *
*********************************************************************
*
*01* STATUS = HCSH501
*
*01* FUNCTION =
*      CBDEC255 defines the UIM data (English texts) for the sample
*      UIM CBDES255.
*
*02*   OPERATION = N/A
*
*03*     RECOVERY OPERATION = N/A
*
*01* NOTES =
*
*02*   DEPENDENCIES = None
*
*02*   RESTRICTIONS = None
*
*02*   REGISTER CONVENTIONS = N/A
*
*02*   PATCH LABEL = None
*
*01* MODULE TYPE = Procedure
*
*02*   PROCESSOR = ASSEMBLER-H
*
*02*   MODULE SIZE = For exact size see assembler listing
*
*02*   ATTRIBUTES =
```

```
*
*03*     LOCATION = Private
*
*03*     STATE = N/A
*
*03*     AMODE = 31
*
*03*     RMODE = Any
*
*03*     KEY = 8
*
*03*     MODE = N/A
*
*03*     SERIALIZATION = N/A
*
*03*     TYPE = Non Executable
*
*01* ENTRY POINT = CBDEC255
*
*02*   PURPOSE = See FUNCTION
*
*02*   LINKAGE = N/A
*
*03*     CALLERS = N/A
*
*01* INPUT = N/A
*
*02*   ENTRY REGISTERS = N/A
*
*01* OUTPUT = N/A
*
*02*   EXIT REGISTERS = N/A
*
*02*   RETURN CODES = N/A
*
*01* EXIT NORMAL = N/A
*
*01* EXIT ERROR = N/A
*
*01* EXTERNAL REFERENCES =
*
*02*   ROUTINES = N/A
*
*02*   DATA AREAS = N/A
*
*02*   CONTROL-BLOCKS = N/A
*
*01* TABLES = N/A
*
*01* MACROS EXECUTABLE = N/A
*
*01* SERIALIZATION = None
*
*01* MESSAGES = None
*
*01* ABEND CODES = None
*
*01* WAIT STATE CODES = None
*
*01* CHANGE ACTIVITY =
*
*    $H0= HCD   HCSH501 940515 BOEB: Sample UDT for DASD I/O
*
***** END OF SPECIFICATIONS *****************************************
         EJECT
*********************************************************************
*
* Use only CBDZUDT macros to generate the UDT.
*
* The following sequence is required:
*
*   1. UDT header definitions
*   2. Unit definitions
*   3. Parameter definitions
*      (Common and private parameters)
*   4. Feature definitions
*      The features must be defined together
*      and in the same sequence as the corresponding
*      bits in the field UIPMSFEA, set by the UIM.
*   5. Message definitions
*   6. Help definitions
*      UIM specific dialogue field helps.
```

```
*      (Not shown in this sample UDT)
*
*--------------------------------------------------------------------
*
*********************************************************************
*  HCD header definition
*********************************************************************
*  The text defined here is shown in "List Installed UIMs"
*--------------------------------------------------------------------
*
         CBDZUDT UDT=CBDEC255,UIM=CBDUC255,                        *
               DESC='UIM FOR 33UU, 93UU'
*
*********************************************************************
*  Unit Definition
*********************************************************************
*  The text defined here is shown in the HCD "View supported
*  device" sub-panel of "List Installed UIMs".
*--------------------------------------------------------------------
*
         CBDZUDT                                                   *
               UNIT='Direct Access Storage Device'
*
*********************************************************************
*  Device parameters
*********************************************************************
*  All OS specific parameters declared in an UIM must be specified
*  also in the corresponding UDT.
*
*  In HCD, a parameter for a device is recognized only if
*  (1) it is described in the UDT that belongs to the UIM defining
*      the device,
*  (2) it is specified as either required or optional parameter in
*      the UIP parameter list for the device.
*
*--------------------------------------------------------------------*
*  Common Parameters                                                 *
*--------------------------------------------------------------------*
*
*  A common parameter is defined by HCD. It has a unique ID in
*  the range between 1 to 32. The parameters that are defined
*  as common are shown in the IODV and UDT data maps.
*
*  Since common parameters are defined within HCD, the allowed
*  values for the parameters are also known to HCD. HCD performs
*  verification checks on the entered values.
*
*  The TEXT keyword allows to provide a description of the parameter.
*  This description is shown on the HCD panel where the OS related
*  parameters of the device are specified.
*
*  The HELP keyword allows to specify a load module which contains
*  help information for the parameter. This help information is
*  displayed when help is requested for the parameter.
*--------------------------------------------------------------------*
*
         CBDZUDT PARA=OFFLINE,                                      *
               TEXT='Device considered online or offline at IPL',   *
               HELP=CBDFP08
         CBDZUDT PARA=DYNAMIC,                                      *
               TEXT='Device supports dynamic configuration',        *
               HELP=CBDED37
*
*--------------------------------------------------------------------*
*  Private Parameters                                                *
*--------------------------------------------------------------------*
*
*  A private parameter is only known to the UIM to which the
*  UDT belongs to.
*  For a private parameter, an id in the range of 33 through 64 must
*  be specified, together with the parameter name (PARA keyword).
*  The ID of a private parameter need not be unique among the set
*  of installed UIMs. Instead of the parameter name, the ID is
*  stored in the HCD device definition record, and it is used to
*  map the stored parameter to the parameter name with the help
*  of the UDT.
*
*  The TEXT keyword allows to provide a description of the parameter.
*  This description is shown on the HCD panel where the OS related
*  parameters of the device are specified.
*
*  The PARATYPE keyword specifies the type of the parameter value.
```

```
*  The type can be one of the following
*
*       NUM       the parameter value is numeric (digits in 0..9)
*       HEX       the parameter value is hexadecimal (digits in 0..F)
*       ALPHANUM  the parameter value is alphanumeric
*       ALPHANUM* the parameter value is alphanumeric or '*'
*       CHAR      the parameter value can contain any character
*       YESNO     the parameter value is 'YES' or 'NO'
*
*  The type specification is used by HCD to perform a syntax check
*  for the parameter value.
*  Additionally, a maximum length can be specified for the
*  parameter value which is also verified by HCD.
*
*  The HELP keyword allows to specify a load module containing
*  help information for the parameter. This help information is
*  displayed when help is requested for the parameter.
*--------------------------------------------------------------------*
*
         CBDZUDT PARA=(DASDPOOL,33),                                  *
               TEXT='DASD pool device belongs to',                    *
               PARATYPE=(ALPHANUM*,4),                                *
               HELP=CBDED37
*
**********************************************************************
*  Device features                                                   *
**********************************************************************
*  All features declared in a UIM must be specified also in the
*  corresponding UDT. The device features are private to the UIM.
*  They are stored with an ID which is the sequence of the feature
*  description occurrence in the UDT. The UDT is required to map
*  this ID to the feature name and description.
*
*  In HCD a feature for a device is recognized only if
*  (1) it is described in the UDT which belongs to the UIM defining
*      the device,
*  (2) it is specified as a supported feature in the UIP parameter
*      list for the device.
*
*  The TEXT keyword allows to provide a description of the feature.
*  This description is shown on the HCD panel where the OS related
*  parameters and features of a device are specified.
*
*  The HELP keyword allows to specify a load module which contains
*  help information for the feature. This help information is
*  displayed when help for the feature is requested.
*--------------------------------------------------------------------*
*
         CBDZUDT FEAT=ALTCTRL,                                        *
               TEXT='Separate physical control unit path',            *
               HELP=CBDED01
         CBDZUDT FEAT=SHARED,                                         *
               TEXT='Device shared with other systems',               *
               HELP=CBDED02
         CBDZUDT FEAT=SHAREDUP,                                       *
               TEXT='Shared when system physically partitioned',      *
               HELP=CBDED03
*
**********************************************************************
*  Compatible features                                               *
**********************************************************************
*  Compatible feature are features which are accepted by the
*  HCD deck migration function but are no longer stored in the
*  device record in HCD.
*--------------------------------------------------------------------*
*
*        CBDZUDT CFEAT=2-CHANSW
*
**********************************************************************
*  Messages                                                          *
**********************************************************************
*  All messages used in the associated UIM have to be
*  defined in the UDT.
*
*  MID keyword     specifies the message identifier. The value must
*                  be "CBDBnnnI" where nnn is a decimal number in the
*                  range of 800 through 999.
*
*  ID keyword      specifies the parameter associated with the message.
*
*  TEXT keyword    specifies the message text. This text is displayed
*                  when the UIM issues a message.
```

```
*                  @n (where n is a decimal number between 1 and 9)
*                  denotes a variable which is passed by the
*                  UIM when issuing the message.
*
*  HELP keyword    specifies a load module which contains the
*                  message explanation. The text of the load module
*                  is displayed when requesting message help in HCD.
*
*                  The help modules referred to in this sample UDT
*                  belong to IBM UDTs.
*--------------------------------------------------------------------*
*
         CBDZUDT MID=CBDB805I,ID=FEATURE,                             *
               TEXT='Features SHARED and SHAREDUP are mutually exclusiv*
               e for device type @1.',                                 *
               HELP=CBDED05
         CBDZUDT MID=CBDB814I,                                        *
               TEXT='The leftmost digit of unit address @1 for device t*
               ype @2 must be even.',                                  *
               HELP=CBDED11
         END

//*******************************************************************
//JJJJJJ    JOB ,NOTIFY=UUUUUU,MSGLEVEL=(1,1),MSGCLASS=H
//ASMH    EXEC PGM=IEV90,REGION=1024K,PARM='LINECNT=55,DECK'
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DSN=XXXXXXXXXXXXXX(CBDEC255),DISP=SHR
//SYSUT1    DD UNIT=SYSDA,SPACE=(CYL,(5,5)),DISP=(NEW,DELETE)
//SYSPUNCH  DD DSN=&OBJ;(CBDEC255),DISP=(,PASS,DELETE),UNIT=SYSDA,;
//             SPACE=(TRK,(1,5,5))
//SYSGO     DD DUMMY
//SYSLIB    DD DSN=SYS1.AMODGEN,
//             DISP=SHR
//SYSLIN    DD UNIT=SYSDA,SPACE=(TRK,(30,10)),DISP=(NEW,PASS),
//             DSN=&POBJ;
//LKED    EXEC PGM=IEWL,PARM='AMOD=31,LET,LIST,NCAL,RMOD=ANY,XREF',
//             COND=(0,NE)
//SYSPRINT  DD SYSOUT=*,DCB=(RECFM=FB,LRECL=121,BLKSIZE=1210)
//SYSLIN    DD DDNAME=SYSIN
//SYSLMOD   DD DSN=YYYYYYYYYYYYYY(CBDEC255),DISP=SHR
//SYSUT1    DD DISP=(NEW,DELETE),SPACE=(CYL,(5,2)),UNIT=SYSDA
//OBJ       DD DSN=&OBJ;,DISP=(OLD,DELETE);
//SYSIN     DD *
  INCLUDE OBJ(CBDEC255)
  NAME CBDEC255(R)
/*
```

Appendix C. IBM-supplied UIMs

HCD UIMs supplied by IBM are part of the product that supports the associated device. For example, the UIM supporting, 3380s and 3390s is part of DFSMSdfp. Therefore, your installation has access to UIMs only for the products it uses. Some device types are defined as another device type. You can use the HCD query and print facility to determine if MVS supports a particular device.

The following ***partial*** list of the IBM-supplied HCD UIMs shows the product that contains the UIM and the devices the UIM defines.

Table 2. IBM-supplied HCD UIMs

Devices	HCD UIM name	Product
1030 **1050** **1050X** **115A**	CBDUS024	MVS
1287 **1288**	CBDUS032	MICR/OCR
1403	CBDUS012	DFSMSdfp
2501 **2540**	CBDUS012	DFSMSdfp
2741 **2740C** **2740X**	CBDUS024	MVS
2741C **2741P**	CBDUS025	MVS
3172	CBDUS057	MVS
3174	CBDUS027	MVS
3203 **3203-5** **3211**	CBDUS012	DFSMSdfp
3270	CBDUS004	MVS
3274	CBDUS027	MVS
3277 **3278** **3279**	CBDUS004	MVS
3284 **3286**	CBDUS031	MVS

Table 2. IBM-supplied HCD UIMs (continued)

Devices	HCD UIM name	Product
3380 **3390**	CBDUS002	DFSMSdfp
3420 **3430** **3480** **3490**	CBDUS005	DFSMSdfp
3505 **3525**	CBDUS012	DFSMSdfp
3540	CBDUS032	MICR/OCR
3704 **3705**	CBDUS023	MVS
3791L	CBDUS027	MVS
3800	CBDUS011	DFSMSdfp
3812	CBDUS031	
3820	CBDUS022	
3838	CBDUS034	VPSS/XA
3886 **3890** **3895**	CBDUS032	MICR/OCR
3995	CBDUS053	OAM
3995-151 **3995-153**	CBDUS002	DFSMSdfp
3997	CBDUS006	MVS
4245 **4248**	CBDUS012	DFSMSdfp
7770	CBDUS023	MVS
83B3	CBDUS025	MVS
BSC1 **BSC2** **BSC3**	CBDUS026	MVS
CTC **SCTC** **BCTC**	CBDUS014	MVS
DUMMY	CBDUS050	MVS
RS6K	CBDUS056	MVS

Table 2. IBM-supplied HCD UIMs (continued)

Devices	HCD UIM name	Product
SWCH	CBDUS051	MVS
TWX **WTTA**	CBDUS025	MVS

In addition to the UIMs listed in Table 2 on page 79, there are UIMs shipped with HCD that complement device values for VM systems. The names of these UIMs range from CBDUS257 to CBDUS512.

Appendix D. Summary of Device Information

For the most current device information, see the specific device publication, use the ***Query supported hardware and installed UIMs*** option on the HCD primary task selection panel, or the Print Supported Hardware support.

Table 3 on page 83 displays an IBM-provided list showing the device order that z/OS uses when it attempts to satisfy a request for a device from an esoteric device group. The order of the IBM-defined list ensures that z/OS always tries to allocate the fastest possible available device.

For each of your UIMs, you may add the generic name and generic preference value to this default list by inserting the device anywhere in the list. While you may add to the list, you cannot change the order of the IBM-defined list this way.

The following specifications are valid for all devices and, therefore, are not repeated in the table:

- Under the Features/Parameter column:
  - OFFLINE (applies to all devices)
  - DYNAMIC (applies to all devices with dynamic device support)
- Under the Control Units column:
  - NOCHECK (applies to all control units)

The following common parameters apply to all devices in a systems:

- ADAPTER
- FEATURE
- NUMSECT
- OFFLINE
- PCU
- SETADDR
- TCU
- DYNAMIC
- OWNER
- LOCANY

Table 3. Device Type Information

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
Direct Access Devices (DASDs)						
9345	9345	9340 9341 9343 9343-1	Yes	Yes	LOCANY ALTCTRL SHARED SHAREDUP	270
3390	3390	3990 3990-2 3990-3 3990-6	Yes	Yes	LOCANY ALTCTRL SHARED SHAREDUP	280

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
3995-151	3390	3995 3995-151	Yes	Yes	LOCANY ALTCTRL SHARED SHAREDUP	280
3995-153	3390	3995 3995-153	Yes	Yes	LOCANY ALTCTRL SHARED SHAREDUP	280
3380 3380-CJ2	3380	3380-CJ2 3880-13 3880-2 3880-23 3880-3 3990 3990-1 3990-2 3990-3 3990-6	Yes	Yes	LOCANY ALTCTRL SHARED SHAREDUP	290
Magnetic Tape Devices						
3490	3490	3490	Yes	Yes	LOCANY LIBRARY AUTOSWITCH ALTCTRL SHARABLE COMPACT	1000
3480		3480	Yes	Yes	LOCANY LIBRARY AUTOSWITCH ALTCTRL SHARABLE COMPACT	1100
3420-4 3420-6 3420-8	3400-5	3803	Yes	Yes	LOCANY ALTCTRL SHARABLE 9-TRACK OPT1600	1200
3420-3 3420-5 3420-7	3400-3 3400-4 3400-5	3803	Yes	Yes	LOCANY ALTCTRL DATACONV DUALDENS SHARABLE 7-TRACK 9-TRACK	1210 1230 1240
3422	3400-6	3422	Yes	Yes	LOCANY ALTCTRL SHARABLE 9-TRACK OPT1600	1220
3424	3400-6	3424	Yes	Yes	LOCANY SHARABLE	1220

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
3430	3400-6	3430	Yes	Yes	LOCANY 9-TRACK OPT1600	1220
3423	3423	3423	Yes	Yes	LOCANY	1380
3590-1	3590-1	3590	Yes	Yes	LOCANY	950
Printers						
AFP1-0		AFP1 3800-3 3800-6 3800-8 3820 3825 3827 3828 3829 3831 3835 3900 3935	Yes	Yes	BURSTER	1750
3825		AFP1 3825	Yes	Yes	BURSTER	1750
3827		AFP1 3827	Yes	Yes	BURSTER	1750
3828		AFP1 3828	Yes	Yes	BURSTER	1750
3829		AFP1 3829	Yes	Yes	BURSTER	1750
3831		AFP1 3831	Yes	Yes	BURSTER	1750
3835		AFP1 3835	Yes	Yes	BURSTER	1750
3900		AFP1 3900	Yes	Yes	BURSTER	1750
3935	AFP1	AFP1 3935	Yes	Yes	BURSTER	1750
3800-1	3800	3800-1	Yes	Yes	BURSTER CGS1 CGS2	1780
3800-3	3800	3800-3	Yes	Yes	BURSTER	1780
3800-6	3800	3800-6	Yes	Yes	BURSTER CGS1 CGS2	1780
3800-8	3800	3800-8	Yes	Yes	BURSTER	1780
3820	3820	3820	Yes	Yes		1800
4248	4248	4248	Yes	Yes		1850

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
4245	4245	4245 6120	Yes	Yes		1890
3211	3211	3811	Yes	Yes		1900
3203-4 3203-5	3203	3203	Yes	Yes		2000
1403-N1 1403-2	1403	2821	Yes	Yes	UNVCHSET	2100
1403-7	1403	2821	Yes	Yes		2100
Unit Record Devices						
2501-B1 2501-B2	2501	2501	Yes	Yes	CARDIMAGE	2300
3505	3505	3505	Yes	Yes	CARDIMAGE	2400
3525	3525	3505	Yes	Yes	CARDIMAGE TWOLINE MULTILINE	2500
2540R-1		2821	Yes	Yes	CARDIMAGE	2800
2540P-1	2540-2	2821	Yes	Yes	CARDIMAGE	2900
Graphic (Display) Devices						
HFGD 5081 6091	HFGD	5088-1 5088-2 6098	Yes	Yes		3260
2250-3	2250-3	2840-2 3258 5088-1 5088-2	No	Yes	PCU NUMSECT ALKYB2250 PRGMKYBD	3500
3251	2250-3	3258 5088-1 5088-2 6098	No	Yes	PCU NUMSECT ALKYB2250 PRGMKYBD	3500
3277-1	3277-1	3272 3274 3791L 6120 7171	Yes	Yes	LOCANY ASCACHAR ASCBCHAR DOCHAR FRCHAR GRCHAR KACHAR UKCHAR AUDALRM MAGCDRD NUMLOCK PTREAD SELPEN ASKY3277 DEKY3277 EBKY3277 OCKY3277 KB70KEY KB78KEY KB81KEY	3700

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
3278-1	3277-1	3174 3272 3274 3791L 6120 7171	Yes	Yes	LOCANY ASCACHAR ASCBCHAR DOCHAR FRCHAR GRCHAR KACHAR UKCHAR AUDALRM MAGCDRD NUMLOCK PTREAD SELPEN ASKY3277 DEKY3277 EBKY3277 OCKY3277 KB70KEY KB78KEY KB81KEY	3700
3178 3179 3180-1 3191 3192 3192-F 3193 3194 3270-X 3290	3277-2	3174 3272 3274 3791L 6120	Yes	Yes	LOCANY ASCACHAR ASCBCHAR DOCHAR FRCHAR GRCHAR KACHAR UKCHAR AUDALRM MAGCDRD NUMLOCK PTREAD SELPEN ASKY3277 DEKY3277 EBKY3277 OCKY3277 KB70KEY KB78KEY KB81KEY	3800
3277-2	3277-2	3272 3274 3791L 6120 7171	Yes	Yes	LOCANY ASCACHAR ASCBCHAR DOCHAR FRCHAR GRCHAR KACHAR UKCHAR AUDALRM MAGCDRD NUMLOCK PTREAD SELPEN ASKY3277 DEKY3277 EBKY3277 OCKY3277 KB70KEY KB78KEY KB81KEY	3800

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
3278-2 3278-2A 3278-3 3278-4 3278-5 3279-S2B 3279-S3G 3279-2A 3279-2C 3279-2X 3279-3A 3279-3X	3277-2	3174 3272 3274 3791L 6120 7171	Yes	Yes	LOCANY ASCACHAR ASCBCHAR DOCHAR FRCHAR GRCHAR KACHAR UKCHAR AUDALRM MAGCDRD NUMLOCK PTREAD SELPEN ASKY3277 DEKY3277 EBKY3277 OCKY3277 KB70KEY KB78KEY KB81KEY	3800
3279-2B 3279-3B	3277-2	3174 3272 3274 3791L 5088-1 5088-2 6098 6120 7171	Yes	Yes	LOCANY ASCACHAR ASCBCHAR DOCHAR FRCHAR GRCHAR KACHAR UKCHAR AUDALRM MAGCDRD NUMLOCK PTREAD SELPEN ASKY3277 DEKY3277 EBKY3277 OCKY3277 KB70KEY KB78KEY KB81KEY	3800
3471 3472 3481 3482 3483		3174 3274	Yes	Yes	LOCANY ASCACHAR ASCBCHAR DOCHAR FRCHAR GRCHAR KACHAR UKCHAR AUDALRM MAGCDRD NUMLOCK PTREAD SELPEN ASKY3277 DEKY3277 EBKY3277 OCKY3277 KB70KEY KB78KEY KB81KEY	3800
Terminal Printers						

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
3284-1	3284-1	3272 3274 3791L	No	No	DOCHAR FRCHAR GRCHAR KACHAR UKCHAR PTREAD	4100
3284-2	3284-2	3272 3274 3791L	No	No	DOCHAR FRCHAR GRCHAR KACHAR UKCHAR PTREAD	4200
3286-1 3287-1C	3286-1	3174 3272 3274 3791L 6120	No	No	DOCHAR FRCHAR GRCHAR KACHAR UKCHAR PTREAD	4300
3262-13 3262-3	3286-2	3174 3272 3274 3791L 4248 6120	No	No	DOCHAR FRCHAR GRCHAR KACHAR UKCHAR PTREAD	4400
3268-2 3286-2 3287-1 3287-2 3287-2C 3289-1 3289-2 3812 4224 4250 5210	3286-2	3174 3272 3274 3791L 6120	No	No	DOCHAR FRCHAR GRCHAR KACHAR UKCHAR PTREAD	4400
3288	3286-2	3272 3274 3791L	No	No	DOCHAR FRCHAR GRCHAR KACHAR UKCHAR PTREAD	4400
Optical or Magnetic Character Readers						
3890	3890	3890	No	No		4800
3886	3886	3886	No	No		4900
1287	1287	1287	No	No		5000
1288	1288	1288	No	No		5100
3895	3895	3895	No	No		5400
Diskette Devices						
3540	3540	3540	No	No		5600
Array Processor						
3838	3838	3838	No	No		5900

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
Telecommunication Devices and Controllers						
1030	AAA1	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOPOLL	6100
1050	AAA2	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOPOLL AUTOANSR AUTOCALL	6200
115A	AAA5	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR	6500
83B3	AAA6	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR	6600
TWX	AAA7	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOANSR AUTOCALL	6700
2740 3767-1 3767-2	AAA8	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOPOLL AUTOANSR AUTOCALL CHECKING INTERRUPT OIU SCONTROL XCONTROL	6800
BSC1	AAA9	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU DUALCODE DUALCOMM	6900
BSC2	AAAA	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU AUTOANSR AUTOCALL DUALCODE DUALCOMM	7000

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
BSC3	AAAB	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU AUTOPOLL DUALCODE DUALCOMM	7100
7770-3	AAAC	7770-3	Yes	Yes	OWNER	7200
WTTA	AAAD	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR	7300
2741P	AAAE	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOANSR	7400
3745	3745	3745	Yes	Yes	ADAPTER OWNER	7450
2741C	AAAF	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOANSR	7500
3725		3725 3745	Yes	Yes	ADAPTER OWNER	7550
3720	3720	3720	Yes	Yes	ADAPTER OWNER	7575
3705	3705	3705	Yes	Yes	ADAPTER OWNER	7600
3174	3174	3174 3704 3705 3720 3725 3745 3791L 6120	Yes	Yes	OWNER	7650
3274	3274	3272 3274 3704 3705 3720 3725 3745	Yes	Yes	OWNER	7675
3272	3791L	3272 3274 3791L	Yes	Yes	OWNER	7700

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
3791L	3791L	3174 3272 3274 3704 3705 3720 3725 3745 3791L 7171	Yes	Yes	OWNER	7700
7171	3791L	7171	Yes	Yes	OWNER	7700
3704	3704	3704	Yes	Yes	ADAPTER OWNER	7800
Channel-to-Channel Control Units and Intersystem Connections						
SCTC		SCTC	Yes	Yes	LOCANY	8300
BCTC	BCTC	SCTC	Yes	Yes	LOCANY	8350
OSA	OSA	OSA	Yes	Yes		8360
OSAD	OSAD	OSA	Yes	Yes		8361
RS6K	RS6K	RS6K	Yes	Yes		8389
3172	3172	3172	Yes	Yes		8398
CTC	CTC	CTC SCTC 3088 3737	Yes	Yes	LOCANY 370	8400
3088	CTC	3088 8232	Yes	Yes	LOCANY 370	8400
3737	CTC	3737	Yes	Yes	LOCANY 370	8400
8232	CTC	8232	Yes	Yes	LOCANY 370	8400
Optical Library Data Server						
3995	3995	3995	Yes	Yes		8600
Telecommunciaton Devices						
1050X	AAAG	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOANSR AUTOCALL	9700
2740X	AAAH	2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOANSR AUTOCALL CHECKING	9800

Table 3. Device Type Information (continued)

Device Type	Generic Name	Control Units	Dynamic Support	4-Digit Support	Features/ Parameters	Preference Value
2740C		2701 3704 3705 3720 3725 3745	No	No	ADAPTER TCU SETADDR AUTOANSR AUTOCALL CHECKING	9900
Dynamic Switches						
SWCH	SWCH	SWCH 9032 9033	Yes	Yes		10500
9032	SWCH	9032	Yes	Yes		10500
9033	SWCH	9033	Yes	Yes		10500
Miscellaneous Devices						
DUMMY	DUMMY	DUMMY	Yes	Yes	LOCANY	99991
3848-1	3848	3848-1	No	No		99999

Appendix E. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact z/OS web page (www.ibm.com/systems/z/os/zos/webqs.html) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or 3.1 or 3.1.1. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number 3.1) are mutually exclusive alternatives. If you hear the lines 3.1 USERID and 3.1 SYSTEMID, your syntax can include either USERID or SYSTEMID, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you

hear the lines `3* , 3 HOST, 3 STATE`, you know that you can include `HOST, STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST,` and `2 STATE`, you know that you must include `HOST, STATE,` or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

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
Site Counsel
2455 South Road

Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming Interface Information

This book is intended to help the customer to write installation-supplied unit information modules (UIMs) for the hardware configuration definition (HCD). This information documents intended programming interfaces that allow the customer to write programs to obtain services of z/OS.

Trademarks

Glossary

This glossary defines technical terms and abbreviations used in z/OS MVS documentation. If you do not find the term you are looking for, refer to the index of the appropriate manual.

action message retention facility (AMRF)
A facility that, when active, retains all action messages except those specified by the installation in the MPFLSTxx member in effect.

action message sequence number
A decimal number assigned to action messages.

Advanced Program-to-Program Communications (APPC)
A set of inter-program communication services that support cooperative transaction processing in a SNA network.

allocate
To assign a resource for use in performing a specific task.

AMRF
action message retention facility

APPC
Advanced Program-to-Program Communications

automated operations
Automated procedures to replace or simplify actions of operators in both systems and network operations.

AVR
Automatic volume recognition.

CART
Command and response token.

CNGRPxx
The Parmlib member that defines console groups for the system or sysplex.

command and response token (CART)
A parameter on WTO, WTOR, MGCRE, and certain TSO/E commands and REXX execs that allows you to link commands and their associated message responses.

command prefix facility (CPF)
An MVS facility that allows you to define and control subsystem and other command prefixes for use in a sysplex.

console
That part of a computer used for communication between the operator or user and the computer.

console group
In MVS, a group of consoles defined in CNGRPxx, each of whose members can serve as a console to display synchronous messages or provide auto-activation facilities for the system console.

CONSOLxx
The Parmlib member used to define message handling, command processing, and MCS, HMCS and SMCS consoles.

control unit
Synonymous with device control unit.

conversational
Pertaining to a program or a system that carries on a dialog with a terminal user, alternately accepting input and then responding to the input quickly enough for the user to maintain a train of thought.

CPF
Command prefix facility.

DASD
Direct access storage device.

data definition name
The name of a data definition (DD) statement, which corresponds to a data control block that contains the same name. Abbreviated as ***ddname***.

data definition (DD) statement
A job control statement that describes a data set associated with a particular job step.

data set label
(1) A collection of information that describes the attributes of a data set and is normally stored on the same volume as the data set. (2) A general term for data set control blocks and tape data set labels.

deallocate
To release a resource that is assigned to a specific task.

device control unit
A hardware device that controls the reading, writing, or displaying of data at one or more input/output devices or terminals.

device number
The unique number assigned to an external device.

device type
The general name for a kind of device; for example, 3330.

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

display console
In MVS, an MCS, HMCS or SMCS console whose input/output function you can control.

DOM
An MVS macro that removes outstanding WTORs or action messages that have been queued to a console

end-of-tape-marker
A marker on a magnetic tape used to indicate the end of the permissible recording area, for example, a photo-reflective strip, a transparent section of tape, or a particular bit pattern.

entry area
In MVS, the part of a console screen where operators can enter commands or command responses.

extended MCS console (EMCS)
In MVS, a console other than an MCS or SMCS console from which operators or programs can issue MVS commands and receive messages. An extended MCS console is defined through an OPERPARM segment.

full-capability console
An MCS, HMCS or SMCS console that can receive messages and send commands. See ***message-stream console*** and ***status-display console***.

hardcopy log
In systems with multiple console support or a graphic console, a permanent record of system activity. See SYSLOG or OPERLOG.

hardware
Physical equipment, as opposed to the computer program or method of use; for example, mechanical, magnetic, electrical, or electronic devices. Contrast with ***software***.

hardware configuration dialog
In MVS, a panel program that is part of the hardware configuration definition. The program allows an installation to define devices for MVS system configurations.

HCD
Hardware configuration definition.

HMCS
HMC multiple console support.

initial program load (IPL)
The initialization procedure that causes an operating system to begin operation.

instruction line
In MVS, the part of the console screen that contains messages about console control and input errors.

internal reader
A facility that transfers jobs to the job entry subsystem (JES2 or JES3).

IPL
Initial program load.

JES2 multi-access spool configuration
A multiple MVS system environment that consists of two or more JES2 processors sharing the same job queue and spool

keyword
A part of a command operand or Parmlib statement that consists of a specific character string (such as NAME= on the CONSOLE statement of CONSOLxx).

line number
A number associated with a line in a console screen display.

MAS
Multi-access spool.

master authority console
In a system or sysplex, a console defined with AUTH(MASTER)

MCS
Multiple console support.

MCS console
A non-SNA device defined to MVS that is locally attached to an MVS system and is used to enter commands and receive messages.

message flooding automation
An automation that reacts to the message flooding situation.

message processing facility (MPF)
A facility used to control message retention, suppression, and presentation.

message queue
A queue of messages that are waiting to be processed or waiting to be sent to a terminal.

message-stream console
An MCS console which receives messages but from which an operator cannot enter commands. See ***full-capability console*** and ***status-display console***.

message text
The part of a message consisting of the actual information that is routed to a user at a terminal or to a program.

message window
The area of the console screen where messages appear.

MMS
In MVS, the MVS message service.

MPF
Message processing facility.

MPFLSTxx
The Parmlib member that controls the message processing facility for the system.

multiple console support (MCS)
The operator interface in an MVS system.

multi-access spool (MAS)
A complex of multiple processors running MVS/JES2 that share a common JES2 spool and JES2 checkpoint data set.

multisystem console support
Multiple console support for more than one system in a sysplex. Multisystem console support allows consoles on different systems in the sysplex to communicate with each other (send messages and receive commands)

MVS message service (MMS)
An MVS component that allows an installation to display messages translated into other languages on a console or terminal.

NIP
Nucleus initialization program.

nonstandard labels
Labels that do not conform to American National Standard or IBM System/370 standard label conventions.

nucleus initialization program (NIP)
The stage of MVS that initializes the control program; it allows the operator to request last minute changes to certain options specified during initialization.

offline
Pertaining to equipment or devices not under control of the processor.

online
Pertaining to equipment or devices under control of the processor.

operations log (OPERLOG)
In MVS, the operations log is a central record of communications and system problems for each system in a sysplex.

OPERLOG
The operations log.

OPERPARM
In MVS, a segment that contains information about console attributes for extended MCS consoles running on TSO/E.

out-of-line display area
For status-display and full-capability MCS and SMCS consoles, areas of the screen set aside for formatted, multi-line display of status information written in response to certain MVS and subsystem commands.

PFK
Program function key.

PFK capability
On a display console, indicates that program function keys are supported and were specified at system generation.

PFKTABxx
The Parmlib member that controls the PFK table settings for MCS consoles in a system.

printer
A device that writes output data from a system on paper or other media.

program function key (PFK)
A key on the keyboard of a display device that passes a signal to a program to call for a particular program operation.

program status word (PSW)
A doubleword in main storage used to control the order in which instructions are executed, and to hold and indicate the status of the computing system in relation to a particular program.

PSW
Program status word.

remote operations
Operation of remote sites from a host system.

roll mode
The MCS, HMCS and SMCS console display mode that allows messages to roll off the screen when a specified time interval elapses.

roll-deletable mode
The console display mode that allows messages to roll off the screen when a specified time interval elapses. Action messages remain at the top of the screen where operators can delete them.

routing
The assignment of the communications path by which a message will reach its destination.

routing code
A code assigned to an operator message and used to route the message to the proper console.

shared DASD option
An option that enables independently operating computing systems to jointly use common data residing on shared direct access storage devices.

SMCS
SNA Multiple Console Support consoles are consoles that use SecureWay Communications Server to provide communication between operators and MVS as opposed to MCS consoles, which do direct I/O to the device.

software
(1) All or part of the programs, procedures, rules, and associated documentation of a data processing system. (2) Contrast with hardware. A set of programs, procedures, and, possibly, associated documentation concerned with the operation of a data processing system. For example, compilers, library routines, manuals, circuit diagrams. Contrast with ***hardware***.

status-display console
An MCS console that can receive displays of system status but from which an operator cannot enter commands. See ***full-capability console*** and ***message-stream console***.

subsystem-allocatable console
A console managed by a subsystem like JES3 or NetView® used to communicate with an MVS system.

synchronous messages
WTO or WTOR messages issued by an MVS system during certain recovery situations.

SYSLOG
The system log data set.

system log (SYSLOG)
In MVS, the system log data set that includes all entries made by the WTL (write-to-log) macro as well as the hardcopy log. SYSLOG is maintained by JES in JES SPOOL space.

sysplex
A multiple-MVS system environment that allows MCS, HMCS, SMCS consoles or extended MCS consoles to receive messages and send commands across systems.

system console
In MVS, a console attached to the processor controller used to initialize an MVS system.

terminal
A device, usually equipped with a keyboard and some kind of display, capable of sending and receiving information over a link.

terminal user
In systems with time-sharing, anyone who is eligible to log on.

virtual telecommunications access method (VTAM®)
A set of programs that maintain control of the communication between terminals and application programs running under DOS/VS, OS/VS1, and OS/VS2 operating systems.

volume
(1) That portion of a single unit of storage which is accessible to a single read/write mechanism, for example, a drum, a disk pack, or part of a disk storage module. (2) A recording medium that is mounted and demounted as a unit, for example, a reel of magnetic tape, a disk pack, a data cell.

volume serial number
A number in a volume label that is assigned when a volume is prepared for use in the system.

volume table of contents (VTOC)
A table on a direct access volume that describes each data set on the volume.

VTAM
Virtual telecommunications access method.

VTOC
Volume table of contents.

wait state
Synonymous with waiting time.

waiting time
(1) The condition of a task that depends on one or more events in order to enter the ready condition. (2) The condition of a processing unit when all operations are suspended.

warning line
The part of the console screen that alerts the operator to conditions requiring possible action.

wrap mode
The console display mode that allows a separator line between old and new messages to move down a full screen as new messages are added. When the screen is filled and a new message is added, the separator line overlays the oldest message and the newest message appears immediately before the line.

write-to-log (WTL) message
A message sent to SYSLOG or the hardcopy log.

write-to-operator (WTO) message
A message sent to an operator console informing the operator of errors and system conditions that may need correcting.

write-to-operator-with-reply (WTOR) message
A message sent to an operator console informing the operator of errors and system conditions that may need correcting. The operator must enter a response.

WTL message
Write-to-log message

WTO message
Write-to-operator message

WTOR message
Write-to-operator-with-reply message.

Index

I

K

M

N

P

S

T

U

Z

SA38-0697-40