IBM

Large
Systems
Technical
Support

Technical
Bulletin

Extending JES2 Using Table Pairs

M. E. Swallow
Edited by: S. W. Wood

National Technical Support
Washington Systems Center

GG66-0282-00
February 1988

Washington Systems Center
Gaithersburg, Maryland
Technical Bulletin

EXTENDING JES2 USING TABLE PAIRS

Mark E. Swallow
Edited by Scott W. Wood

GG66-0282-00
February, 1988

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis **without any warranty either expressed or implied**. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Publications are not stocked at the address given below; requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to: IBM Washington Systems Center, JES2 Support, 18100 Frederick Pike, ISG/Building 183 Room 2T74, Gaithersburg, MD 20879

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

Abstract

This presentation introduces the concepts of table pairs and describes the uses of table pairs in the JES2 component of MVS/System Product (MVS/SP). The aim is to help you understand the considerations necessary to migrate JES2 source modifications and exit code to tables.

IBM introduced table pairs in the JES2 component of MVS/SP JES2 1.3.3. They provide a means to alter JES2 processing and achieve tailoring of the JES2 component of an MVS/SP JES2 system. Table pairs are not meant to replace the JES2 exit facility. They are intended to work either with or without the JES2 exit facility, depending upon the needs of your installation.

As with exit points and source modifications, only experienced system programmers with a thorough knowledge of system programming, JES2 programming conventions, and JES2 design and code should attempt to use this material. If you attempt to write exit routines, install new exit points, or implement the table pair function described in this bulletin without this special knowledge and experience you run the risk of seriously degrading the performance of your system or causing complete system failure.

Documents key to understanding this material include *MVS/XA SPL: JES2 Initialization and Tuning*, form SC23-0065, *MVS/XA SPL: JES2 Modifications and Macros*, form LC23-0069, *MVS/XA JES2 Logic*, form LY24-6008, and JES2 component source code. In the latter, listings of modules HASPTABS and HASPSTAB contain helpful examples of JES2 tables.

We developed the document and coding example using MVS/SP JES2 2.1.5. Although the editor and reviewers have attempted to update this to the 2.2.0 level, we may have unintentionally missed a few details which we therefore must leave to the reader to uncover. Also, changes or enhancements to JES2 anytime can change coding details and some of the more specific examples herein. As always, consult the manuals which correspond to the version, release, and level of the JES2 component of MVS/SP you are running.

Mark Swallow developed and presented this material for GUIDE 65 session SY-7141, July, 1986, in Chicago. Harry Familetti presented it at SHARE 67 sessions O382 and O383, August, 1986, in Atlanta.

THIS PAGE INTENTIONALLY LEFT BLANK

Preface

How to Read This Book

Unless you have a high tolerance for pain and have falling-out-of-the-chair insurance, we don't recommend you try to read this straight through. Instead, we recommend you:

- look at the table of contents, and then
- read the overview of the document contained in "Introduction".

Then, especially if you're new to modifying JES2, read the chapter called "What Are JES2 Table Pairs?".

After you've sat on that for awhile and had at least a second cup, pick one of the table pair examples in "Examples of Table Pairs" and skim that to get a general idea of the techniques you will need to master.

If after all this you still feel you need to add function in your installation, then go back and **study** the chapter which describes how to add table pairs for the function you need to develop. Be sure you have ready access to JES2 source code libraries, a copy of *SPL: JES2 Modifications and Macros*, and lots of patience.

Thanks

Thanks to the following people for their comments, encouragement, close reading of the text, and testing. None are responsible for any mistakes left herein. (If you find one, tell us using the Reader Comment Form, please.)

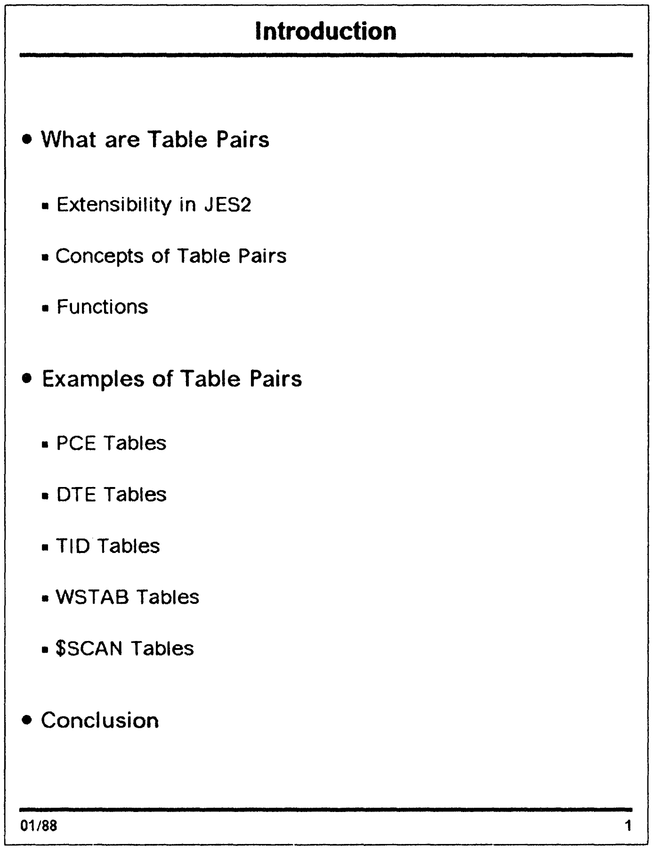
- Steve Anania
- Bernie Becker
- Bill Coltin
- Harry Familetti
- John Kinn

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

Introduction

Overview of Presentation



Introduction

- What are Table Pairs
  - Extensibility in JES2
  - Concepts of Table Pairs
  - Functions
- Examples of Table Pairs
  - PCE Tables
  - DTE Tables
  - TID Tables
  - WSTAB Tables
  - $SCAN Tables
- Conclusion

01/88 1

This presentation will cover three general topics:

1. Definition of table pairs

2. Examples

3. Conclusions

First, we will discuss what table pairs are. We will look at extensibility in JES2 by comparing exits versus table pairs and give positive and negative aspects of both. Next we will discuss the concepts of table pairs and their potential functions.

Second, we will show examples of table pairs that currently exist in JES2:

- Processor Control Element (PCE) tables. These tables are used to create IBM and installation-defined processors in JES2. We will show an example in a security processor.

- Daughter Task Element (DTE) tables. These tables are used to create IBM and installation-defined subtask elements. We will discuss how communication between installation-defined processors and installation-defined subtasks takes place. We will show an example in a security processor.

- Trace Id (TID) tables. These tables are used to create IBM and installation-defined trace records and format them. We will show an example of an installation-defined trace identifier using the installation-defined security PCE and DTE.

- Work Selection (WSTAB) tables. These tables are used to create IBM and installation-defined work selection criteria for use with the WS= operand on printers, punches, offload devices, etc. We will show an example of installation-defined work selection criteria.

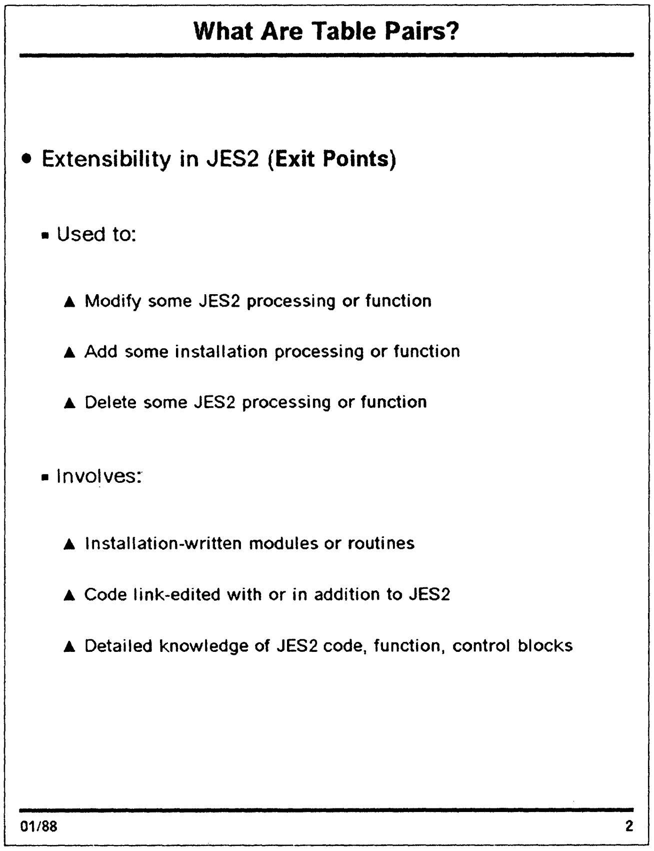
- $SCAN tables. These tables are used to create IBM and installation-defined initialization statements and commands. The $SCAN facility is the most complex and extensible example of table pairs in JES2. We will show an example of an installation-defined initialization statement and an installation-defined command.

As we present each of the above tables we will discuss the key control blocks and fields.

Third, we will present some general conclusions.

What Are JES2 Table Pairs?

JES2 Table Pairs Versus JES2 Exits



What Are Table Pairs?

- Extensibility in JES2 (**Exit Points**)
  - Used to:
    - ▲ Modify some JES2 processing or function
    - ▲ Add some installation processing or function
    - ▲ Delete some JES2 processing or function
  - Involves:
    - ▲ Installation-written modules or routines
    - ▲ Code link-edited with or in addition to JES2
    - ▲ Detailed knowledge of JES2 code, function, control blocks

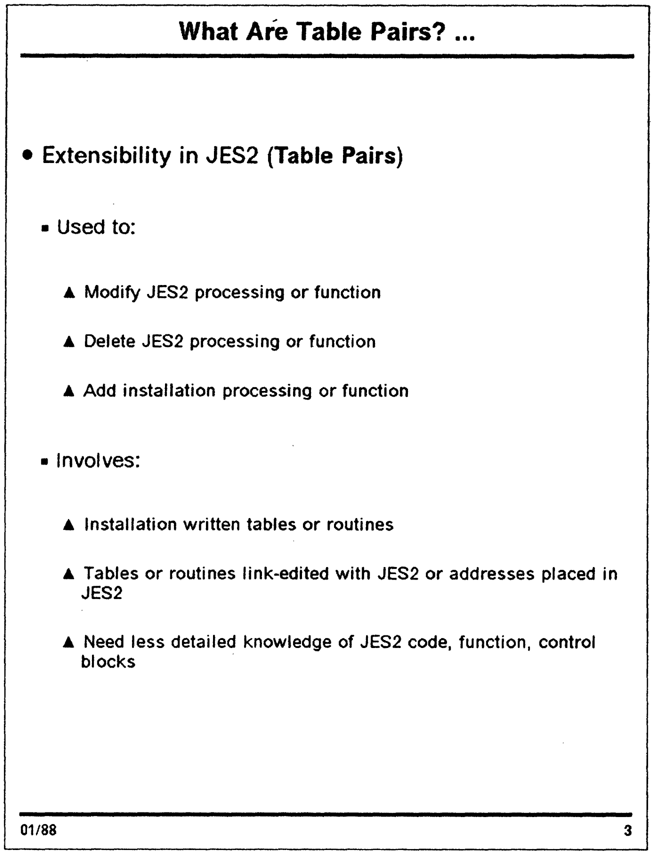
01/88 2

Exit points were introduced into JES2 in the MVS/SP JES2 1.3.0 product. When you code exit points you may modify some JES2 processing or function, add some installation processing or function, or delete some JES2 processing or function. Notice that we use the word 'some' here. This is due to the variation in function that exits points provide. The location, the services available, and the environment where the exit is called all affect what you are capable of achieving at a particular exit point. Therefore, there may be an exit point where you are capable of modifying JES2 processing but where you are not capable of deleting JES2 function or adding installation function.

In order to use the exit facility, you must write exit modules to contain your exit routines. Your modules can be link-edited with JES2 (in certain instances) or they may be independent of JES2.[1] In general, coding exits requires detailed knowledge of JES2, its coding conventions and idiosyn-

[1] Best general practice is to keep exits separate from JES2 modules and then use the LOAD initialization statement to tell JES2 about them.

crasies, its functions, capabilities, and drawbacks, and its control blocks both in content as well as structure. Thus, using an exit point in JES2 can be a daunting endeavor for the uninitiated.



What Are Table Pairs? ...

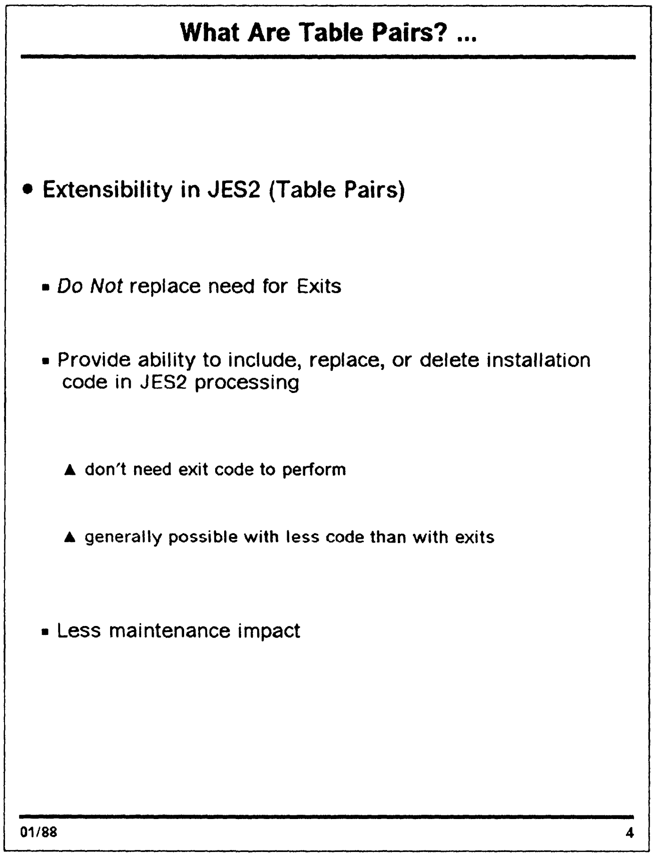
- Extensibility in JES2 (**Table Pairs**)
  - Used to:
    - ▲ Modify JES2 processing or function
    - ▲ Delete JES2 processing or function
    - ▲ Add installation processing or function
  - Involves:
    - ▲ Installation written tables or routines
    - ▲ Tables or routines link-edited with JES2 or addresses placed in JES2
    - ▲ Need less detailed knowledge of JES2 code, function, control blocks

01/88 3

Table pairs were introduced into JES2 in the MVS/SP JES2 1.3.3 product as a complement to exit points. When you code table pairs, you can modify JES2 processing or function, delete JES2 processing or function, or add installation processing or function. Notice that unlike exit points, you can modify, delete, or add function without restriction. Note, however, we don't recommend deleting JES2 function unless you understand the implications. We will discuss the implications of deleting JES2 function under each of the examples in the section "Examples of Table Pairs" on page 17.

In order to use table pairs, you must create installation table pairs and perhaps also supporting routines, then either link-edit them with JES2 modules or define the table addresses to JES2. Depending on what table you are going to add, modify or delete, this generally takes less detailed knowledge of JES2 code, function and control block structure and content than using JES2 exits would take. If you wish to add an initialization statement to JES2, this would probably require nothing more than a table entry to define the statement and to specify where to place the input. If you require more specialized processing than that supplied by JES2, then you can create supporting routines. In all of JES2's initialization and command tables, only about one-sixth require supporting pre-scan or post-scan supporting routines. Therefore it is not likely that you will require such a supporting routine. If, however, you wish to add a processor (PCE), this requires code and expertise and adds a level of complexity.

Generally, table pairs provide a structured mechanism to change JES2 processing. This makes changes somewhat less complex than what is required when you use an exit.



What Are Table Pairs? ...

- Extensibility in JES2 (Table Pairs)
  - *Do Not* replace need for Exits
  - Provide ability to include, replace, or delete installation code in JES2 processing
    - ▲ don't need exit code to perform
    - ▲ generally possible with less code than with exits
  - Less maintenance impact

01/88 4

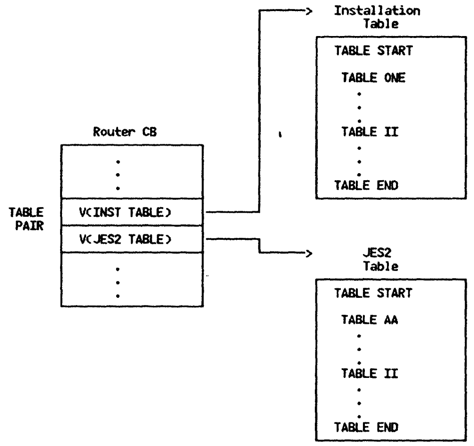
It is important to emphasize that table pairs do not replace the need for exit points. Table pairs and exit points can meet their respective requirements independently or together. However, using table pairs imposes fewer constraints and less complexity than using exit points since you can add, delete, or modify JES2 processing. Furthermore, since less code is usually needed, there is less of a maintenance impact.

Concepts

Next, we discuss concepts of table pairs in order to introduce key points used in implementing the table pair functions of JES2.

What Are Table Pairs? ...

- Concepts of Table Pairs (Description)



01/88 5

Table pairs in JES2 start with a router control block that contains a pair of addresses. This pair of addresses is known as a 'table pair'. The first address in the pair of addresses points to an installation-defined table and the second address points to a JES2-defined table. Each table is defined by a TABLE START and by a TABLE END. The table information is contained within the TABLE START and TABLE END delimiters.

In the example above, the installation table contains two table elements. The first is a table entry that describes the element 'ONE' and the second table entry describes the element 'II'. The IBM-supplied table in the example also contains two table elements of information. The first is a table entry that describes the element 'AA' and the element 'II'.

JES2 uses these tables when it is processing the items 'ONE', 'II', and 'AA'.

1. First JES2 isolates the item to process (e.g., 'ONE', 'II', or 'AA') in the input source data.

2. Next it goes to the router control block to find the table pair to use to process the isolated item.

3. Then it attempts to find the installation table. If the first table pair pointer is zero, then JES2 will search the JES2 table only. If the table pair pointer is non-zero, then this value is assumed to be the address of the installation table. In this way the installation table, if it exists, is *always* searched *prior* to the JES2 table. The installation table is optional and will not exist unless you create it.

4. If the item to process is located in the installation table, then processing continues using the installation table entry. If the item is not found in the installation table, then JES2 will search the JES2 table. If the item to process is located in the JES2 table, then processing continues

using the JES2 table entry. If the item is not found in the JES2 table then an error message is issued.

Therefore, using the table arranged as described in the figure above, the three input items 'ONE', 'II', and 'AA' will be processed. Let's assume they are encountered in that order.

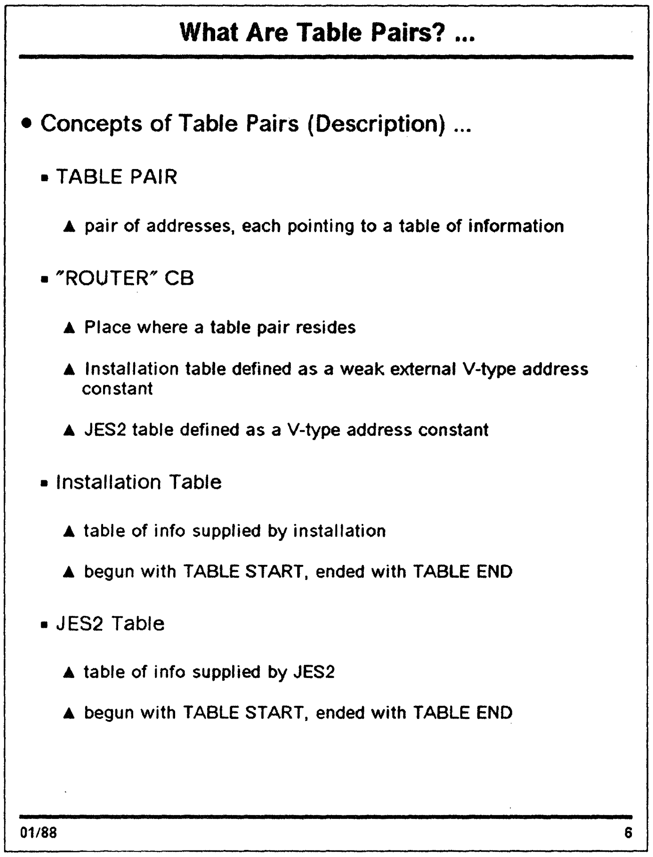
First, the input item 'ONE' is processed. The item 'ONE' is located and isolated in the input stream. Next, the address of the installation table is found from the table pair in the router control block. The installation table is searched by examining each table element for a match for the input 'ONE'. In this example the first table element matches the input. This table element will be used by JES2 to process the input 'ONE'. Notice that the JES2 table does not include a table element that describes 'ONE'. Therefore, the installation has added some processing or function to JES2 without modifying any JES2 code.

Next the input item 'II' will be processed. The item 'II' is located and isolated from the data handed to JES2. Next the address of the installation table is found from the table pair in the router control block. The installation table is searched by examining each table element for a match for the input 'II'. In this example, the table element that matches the input is found later in the installation table. This table element will be used by JES2 to process the input 'II'. Notice that the JES2 table includes a table element that describes 'II'. Since a match was found in the installation table, JES2 never searched the JES2 table. Thus the installation has replaced or modified some processing or function without modifying any JES2 code.

Finally, the input item 'AA' is processed. The item 'AA' is located and isolated from the data handed to JES2. Next the address of the installation table is found from the table pair in the router control block. The installation table is searched by examining each table element for a match for the input 'AA'. In this example, there is no element that matches the input of 'AA' in the installation table. Therefore, processing continues by searching the JES2 table for an element that matches 'AA'. When this table element is found in the JES2 table, the input 'AA' will be processed by this table element.

Deleting a table element is done by deleting the table that contains the element. For example, if you wished to delete the processing for the input 'AA', you would zero the second table pair address that pointed to the JES2 table that contained the element for 'AA'. If there were elements in the JES2 table that you would not want deleted along with the element for 'AA', you would have to copy those elements to an installation table.[2] It is not recommended that you delete JES2 tables. However, the ability to do so is not inhibited since there may be times when such function is required.

[2] An alternative might be to provide a null installation table element for the item to be 'deleted'.



What Are Table Pairs? ...

- Concepts of Table Pairs (Description) ...
  - TABLE PAIR
    - ▲ pair of addresses, each pointing to a table of information
  - "ROUTER" CB
    - ▲ Place where a table pair resides
    - ▲ Installation table defined as a weak external V-type address constant
    - ▲ JES2 table defined as a V-type address constant
  - Installation Table
    - ▲ table of info supplied by installation
    - ▲ begun with TABLE START, ended with TABLE END
  - JES2 Table
    - ▲ table of info supplied by JES2
    - ▲ begun with TABLE START, ended with TABLE END

01/88 6

In summary:

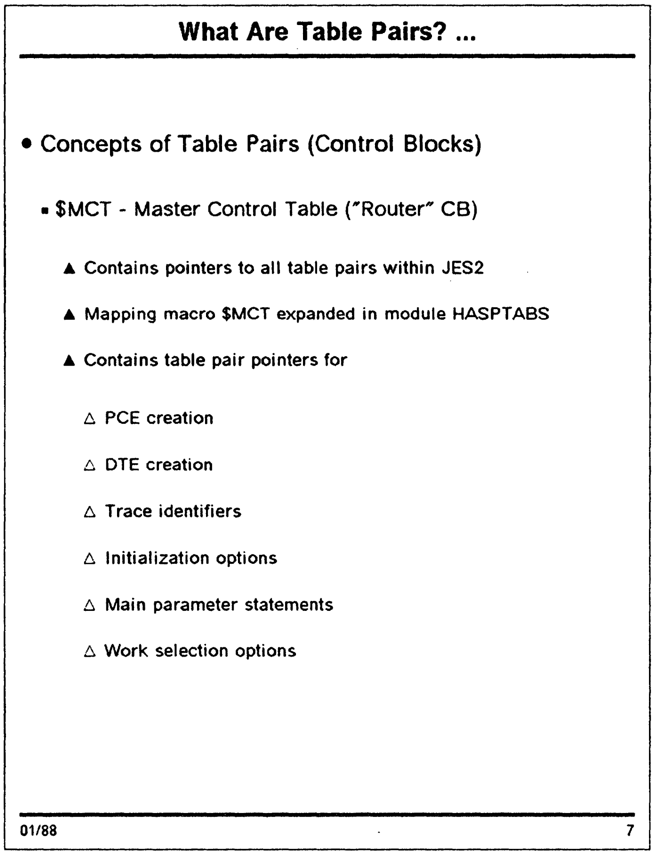
- A table pair consists of a pair of addresses where the first address is the address of an installation table of information and the second address is the address of the JES2 table of information. One or the other of these fields may be zero, but not both. If the installation table pointer is zero then no installation table exists and JES2 will use the JES2 table to attempt to process the input. If the JES2 table pointer is zero then the input must be found in the installation table or else the input is marked as invalid.

- The router control block contains one or more table pair addresses. The installation table fields of the table pair are defined as weak external V-type address constants. Therefore, installation tables may be link-edited with JES2 to have the linkage editor resolve the installation table addresses. If the installation table is not link-edited with JES2 then you must fill in the address of its table into the first of the correct table pairs. We will provide more information on the specific tables and what must be done later (in "Examples of Table Pairs" on page 17). The JES2 table entries are defined as V-type address constants and the JES2 table addresses are placed into the table pairs by the linkage editor.

- An installation table consists of a table of information defined by the installation. The table begins with a TABLE START and concludes with a TABLE END.

- The JES2 table is supplied by IBM with the JES2 product. The table begins with a TABLE START and concludes with a TABLE END. Each function that uses the table pair capability has its own table pair.

Master Control Table



What Are Table Pairs? ...

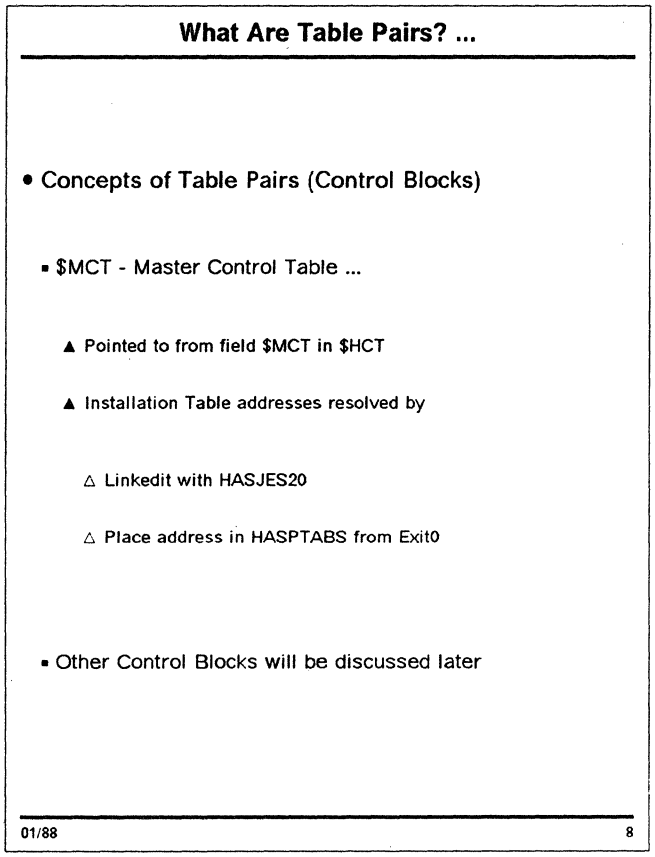
- Concepts of Table Pairs (Control Blocks)
  - $MCT - Master Control Table ("Router" CB)
    - ▲ Contains pointers to all table pairs within JES2
    - ▲ Mapping macro $MCT expanded in module HASPTABS
    - ▲ Contains table pair pointers for
      - △ PCE creation
      - △ DTE creation
      - △ Trace identifiers
      - △ Initialization options
      - △ Main parameter statements
      - △ Work selection options

01/88 7

The router control block referred to above is called the Master Control Table ($MCT). This control block contains all of the table pairs in JES2. The mapping macro for this control block is called the $MCT and is expanded in the JES2 module HASPTABS.

The $MCT contains the table pair pointers for:

- Processor creation (PCE's)
- Subtask creation (DTE's)
- Trace identifiers
- Initialization options (e.g., COLD, NOREQ, WARM, etc.)
- Main parameter statements (e.g., CKPTDEF, SPOOLDEF, etc.)
- Work selection options



What Are Table Pairs? ...

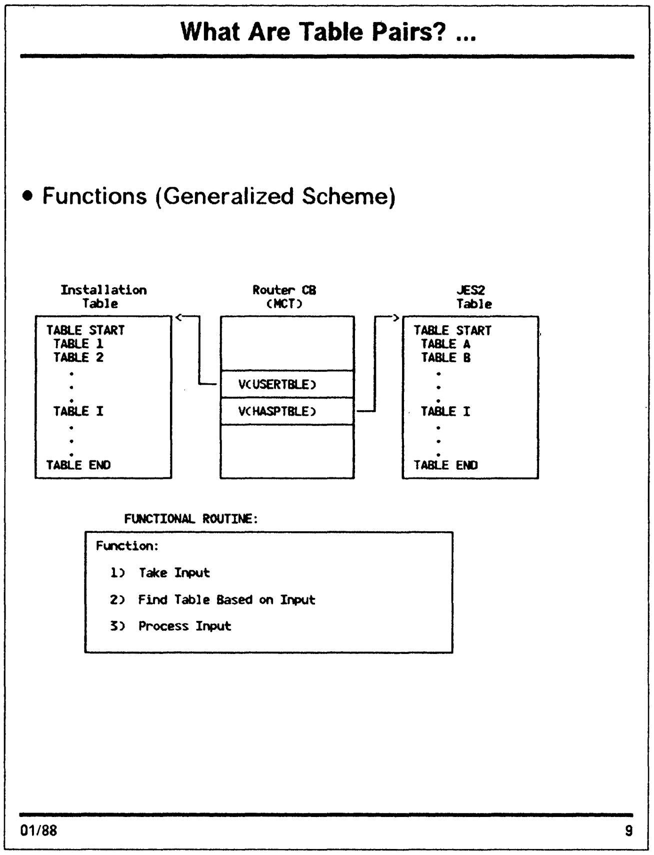
- Concepts of Table Pairs (Control Blocks)
  - $MCT - Master Control Table ...
    - ▲ Pointed to from field $MCT in $HCT
    - ▲ Installation Table addresses resolved by
      - △ Linkedit with HASJES20
      - △ Place address in HASPTABS from Exit0
  - Other Control Blocks will be discussed later

01/88 8

The Master Control Table ($MCT) is pointed to from the $HCT field $MCT. Addresses of the installation tables can be resolved by either link-editing the installation table with JES2 (using the appropriate name, as will be described later) or by placing the address of the installation table into the $MCT through an exit. Exit 0 (initialization) can be used for this purpose.

Some of the other key control blocks for table pairs will be described in the section "Examples of Table Pairs" on page 17.

Functional Routines



What Are Table Pairs? ...

- Functions (Generalized Scheme)

FUNCTIONAL ROUTINE:

Function:

1) Take Input

2) Find Table Based on Input

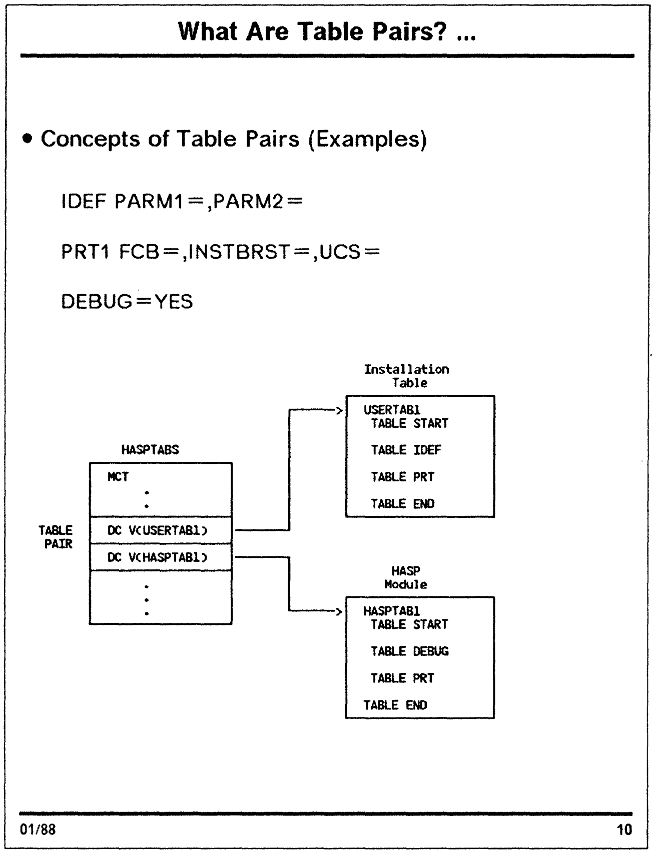
3) Process Input

01/88 9

A functional routine uses the table pair as a means to process input. The general flow is to process some input by first isolating the item to process. Then the routine finds the corresponding table element (either installation or JES2) which defines the input item. Then it processes the input using the table element.

General Example

Next, we explain examples of the processing done by a functional routine.



What Are Table Pairs? ...

- Concepts of Table Pairs (Examples)

IDEF PARM1=,PARM2=

PRT1 FCB=,INSTBRST=,UCS=

DEBUG=YES

Installation Table

USERTAB1
TABLE START
TABLE IDEF
TABLE PRT
TABLE END

HASPTABS

MCT

TABLE PAIR

DC V(USERTAB1)

DC V(HASPTAB1)

HASP Module

HASPTAB1
TABLE START
TABLE DEBUG
TABLE PRT
TABLE END

01/88 10

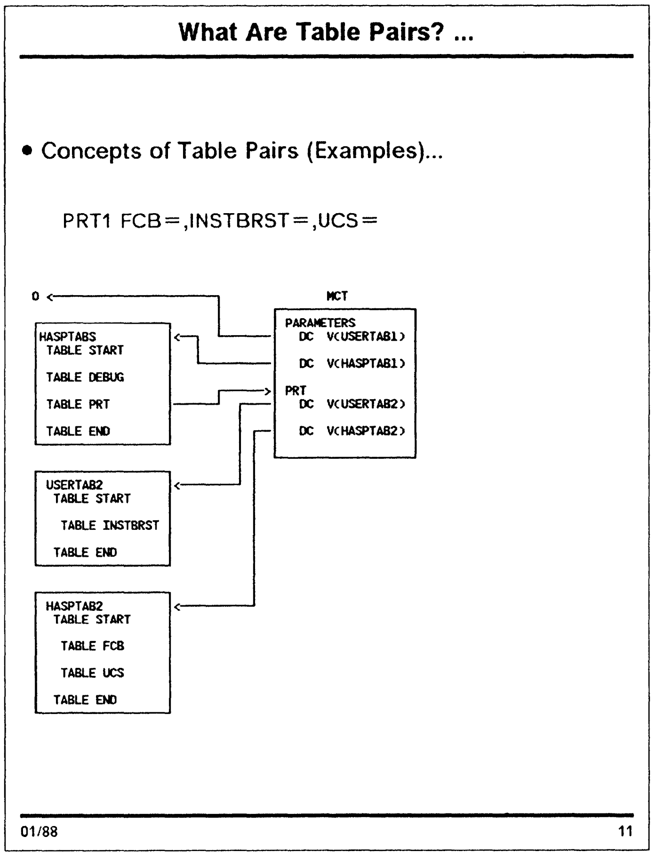
In the example shown in the figure above, there are three initialization statements. A functional routine will accept the input of 'IDEF PARM1 = ,PARM2 = ' and process it using the table structure shown.

Located in the JES2 module HASPTABS is the Master Control Table that contains the main parameter statement table pair. The first entry is the pointer to the installation parameter statement table USERTAB1. If you want the linkage editor to resolve the table address, you would have to name the table USERTAB1 and link-edit the table with JES2. If you did not want to link-edit with JES2, you would have to place the address of the table into this table pair entry.

The functional routine would first isolate the input IDEF from the input passed it. Next it would find the table pair in the MCT and search the installation table for the element that matched IDEF. Once it found the matching element, it would use this table element to process the statement 'IDEF PARM1 = ,PARM2' and not search the JES2 table. In this way, you have added an initialization statement and the functional routine has not been modified.

To process the input 'PRT1 FCB = ,INSTBRST = ,UCS = ', the functional routine would once again isolate the first keyword in the input 'PRT1' and, using the table pair, search the installation table for the element that matched the input. Once it found the table element that matched PRT1, it would use this table element to process 'PRT1 FCB = ,INSTBRST = ,UCS = ' and not search the JES2 table. Note that the functional routine did not search the JES2 table and did not find the JES2 table element for PRT. In this way, you replaced an initialization definition without modifying the functional routine.

Finally, to process the input 'DEBUG = YES', the functional routine would isolate the keyword 'DEBUG', find the table pair, and search the installation table. When no matching table element was found in the installation table, the functional routine would obtain the address of the JES2 table from the table pair and search it. Once found, the table element for 'DEBUG' in the JES2 table would be used to process the initialization statement.



What Are Table Pairs? ...

- Concepts of Table Pairs (Examples)...

PRT1 FCB=,INSTBRST=,UCS=

0 <

MCT

PARAMETERS
DC V(USERTAB1)
DC V(HASPTAB1)
PRT
DC V(USERTAB2)
DC V(HASPTAB2)

HASPTABS
TABLE START
TABLE DEBUG
TABLE PRT
TABLE END

USERTAB2
TABLE START
TABLE INSTBRST
TABLE END

HASPTAB2
TABLE START
TABLE FCB
TABLE UCS
TABLE END

01/88 11

There can be multiple levels of tables to define parameters to JES2. In the example above, the functional routine will search two levels of tables to process the input 'PRT1 FCB=,INSTBRST=,UCS='. The functional routine will:

1. isolate the first keyword 'PRT1'.

2. obtain the table element to process the keyword. The functional routine will get the address of the installation table. Since this address is zero, there is no installation table. Therefore, it will search the JES2 table until it finds the table element that matches 'PRT1'.

3. process the 'PRT1' statement using this table element. The table element for 'PRT1' tells the functional routine that to process the rest of this statement it must use another level of tables. These tables are pointed to from the table pair at PRT in the MCT.

4. isolate the next keyword 'FCB'.

5. obtain the table element to process the keyword. The functional routine will get the address of the installation table from the first entry in the PRT table pair. Since 'FCB' is not in that table, the function routine will search the JES2 table. The table element for 'FCB' is found in the JES2 table.

6. process 'FCB' using this table element.

7. isolate the next keyword 'INSTBRST'.

8. obtain the table element to process the keyword. The functional routine will get the address of the installation table from the first entry in the PRT table pair. It will find the table element that matches 'INSTBRST' in the installation table.

9. process 'INSTBRST' using this table element.

10. isolate the next keyword 'UCS'.
11. obtain the table element to process the keyword. The element will be found in the JES2 table.
12. process 'UCS' using this table element.

The functional routine will then determine that there is no more input and indicate completion.

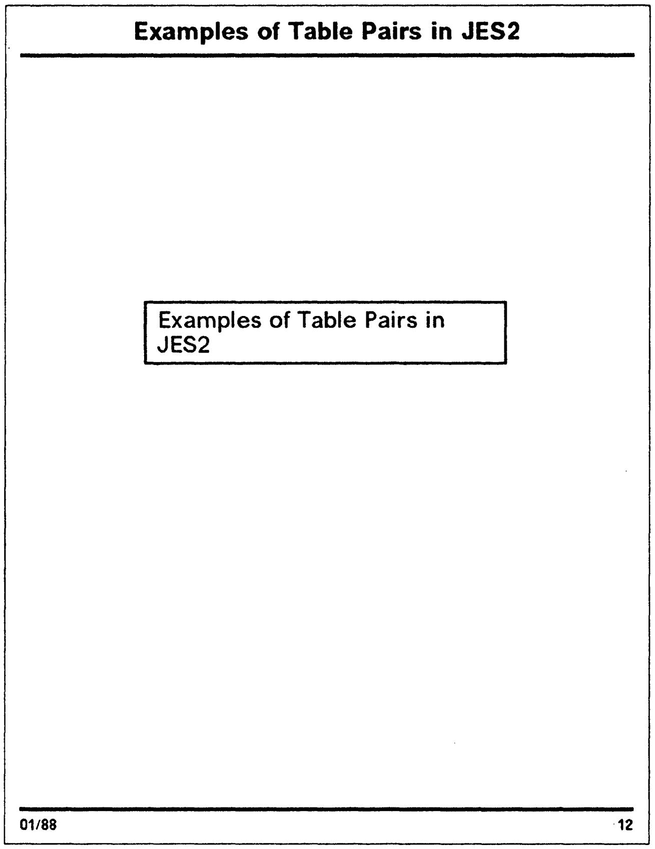
Summary

To summarize:

1. Table pairs are a pair of addresses where the first address is a pointer to an installation table of information and the second address is a pointer to the JES2 table of information.
2. The installation table is searched before the JES2 table to find a matching table element for some input.
3. A functional routine is one that makes use of the table pairs to process some input.

THIS PAGE INTENTIONALLY LEFT BLANK

Examples of Table Pairs



Examples of Table Pairs in JES2

Examples of Table Pairs in JES2

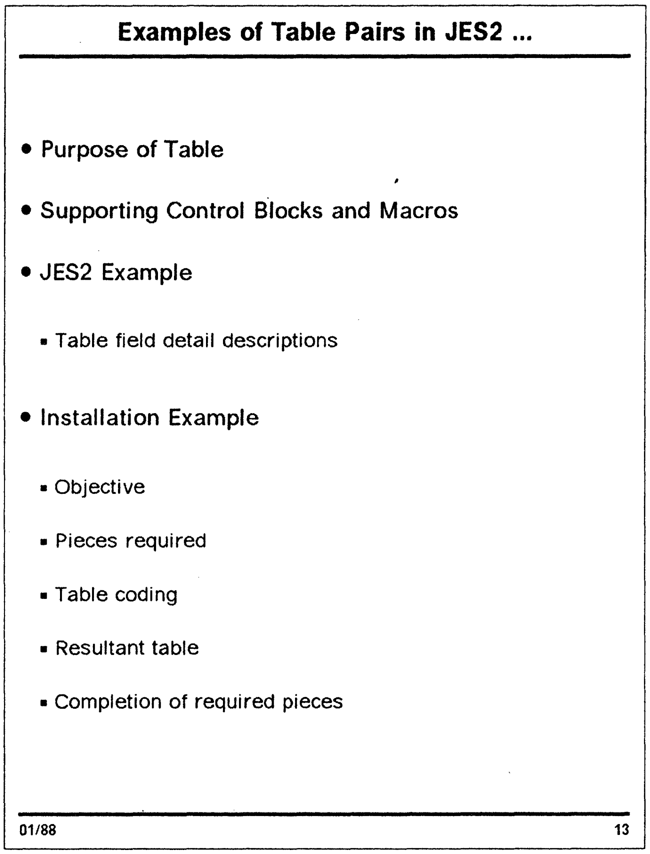
01/88 12

There are five ways JES2 uses table pairs. JES2 uses table pairs for:

1. PCE Tables
2. DTE Tables
3. TID Tables
4. WS Tables
5. $SCAN Tables

This chapter shows how JES2 makes use of table pairs. Some of the functions are more complex than others, but all make use of table pairs and therefore allow you to add, modify, or delete JES2 functions or processing without directly modifying JES2 source code.

How These Examples Are Shown



Examples of Table Pairs in JES2 ...

- Purpose of Table
- Supporting Control Blocks and Macros
- JES2 Example
  - Table field detail descriptions
- Installation Example
  - Objective
  - Pieces required
  - Table coding
  - Resultant table
  - Completion of required pieces

01/88 13

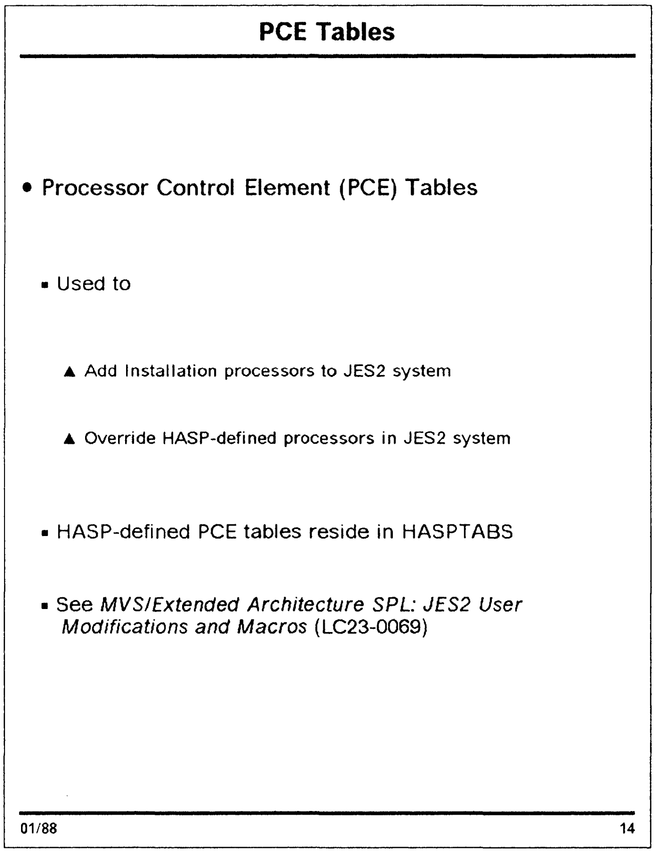
As we present each example we provide the following information:

- The purpose of the table or function. What is it that you can add, modify or delete? This will not be detailed but will point you to other books that can be read to gather greater detail.
- Describe some of the supporting control blocks and macros. These are typically key control blocks and macros that you would have to understand and make use of to fulfill the appropriate function.
- Step through a JES2 example. Describe what the table contains using a JES2 table element as an example. This will describe the table element field values.
- Step through the creation of an installation table and table element. This involves:
  - describing the objective the table is trying to satisfy;
  - identifying what pieces besides the installation table are required, if any;
  - coding the table element;
  - describing what the final table looks like; and,
  - describing what other required pieces look like.

"Appendix A. Table Pairs Coding Example" on page 147 contains coded examples of the specific installation sample we are creating in this bulletin. The examples there are inter-related to show how the tables can be used together. This is not required. That is, it is not necessary to code a PCE table (create your own processor) *and* code your DTE table (create your own subtask). In fact, it may make no sense to design interrelated tables for your particular use of JES2 table pairs. The examples are contrived to show what can be done, not necessarily what should be done.

PCE Tables

What Is a PCE Table?



PCE Tables

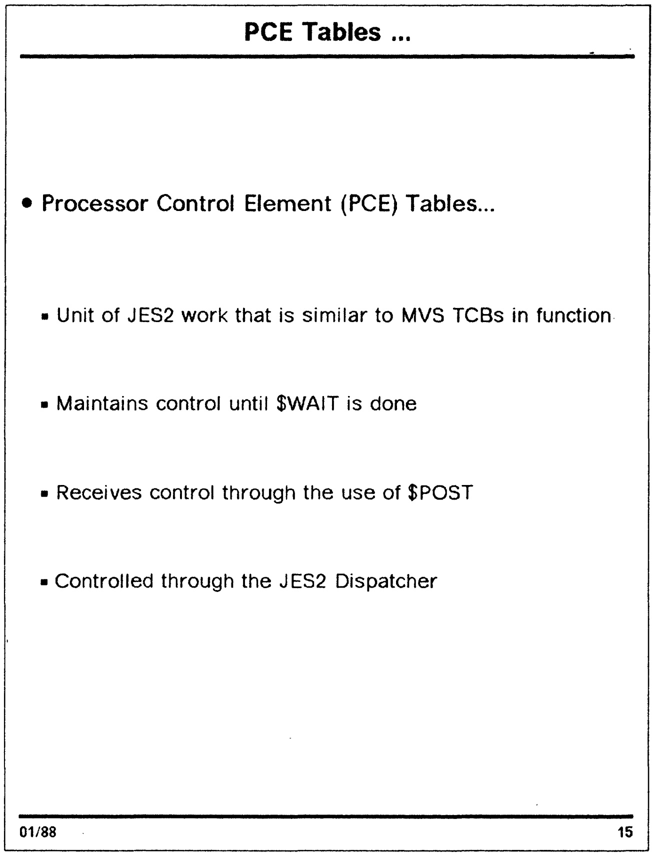
- Processor Control Element (PCE) Tables
  - Used to
    - ▲ Add Installation processors to JES2 system
    - ▲ Override HASP-defined processors in JES2 system
  - HASP-defined PCE tables reside in HASPTABS
  - See *MVS/Extended Architecture SPL: JES2 User Modifications and Macros* (LC23-0069)

01/88 14

The Processor Control Elements (PCE) tables are used to add installation-defined processors (PCEs)[3] to a JES2 system or to override JES2-defined processors. Notice that deleting a JES2 processor is not on our agenda. This is because we recommend you do not delete any JES2 processors.

The JES2-defined PCE tables reside in the JES2 module HASPTABS. Some of the following information can be found in *SPL: JES2 User Modifications and Macros.*

[3] The term 'PCE' refers either to the JES2 unit of work (processor) or to the control block which represents the processor. Where the distinction is important we have tried to add terms like 'processor' or 'control block' to the term 'PCE' where it occurs.

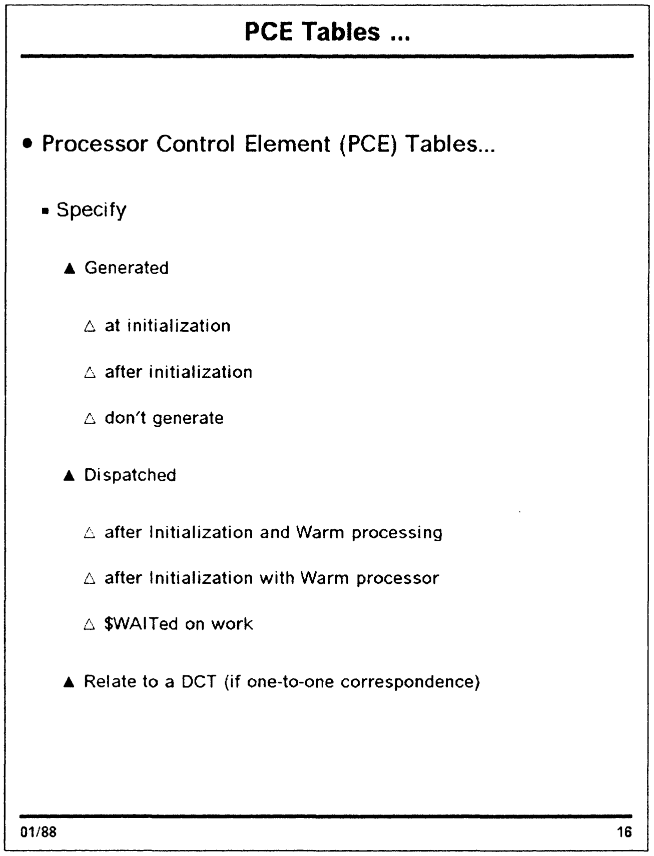


PCE Tables ...

- Processor Control Element (PCE) Tables...
  - Unit of JES2 work that is similar to MVS TCBs in function
  - Maintains control until $WAIT is done
  - Receives control through the use of $POST
  - Controlled through the JES2 Dispatcher

01/88 15

JES2 Processor Control Elements (PCEs) represent units of JES2 work. In this way they are similar to MVS Task Control Blocks (TCBs). The JES2 dispatcher gives control to a PCE. Unlike TCBs, this JES2 unit of work will not be preempted by the JES2 dispatcher. No other PCE will gain control until and unless this PCE directly relinquishes control. This is done when the JES2 process issues a $WAIT. When a $WAIT is done, control is given to the JES2 dispatcher, which saves the registers in the PCE control block that represents the JES2 processor and then dispatches another JES2 processor. A JES2 processor is ineligible for dispatching until it is $POSTed.



PCE Tables ...

- Processor Control Element (PCE) Tables...
  - ▪ Specify
    - ▲ Generated
      - △ at initialization
      - △ after initialization
      - △ don't generate
    - ▲ Dispatched
      - △ after Initialization and Warm processing
      - △ after Initialization with Warm processor
      - △ $WAITed on work
    - ▲ Relate to a DCT (if one-to-one correspondence)

01/88 16

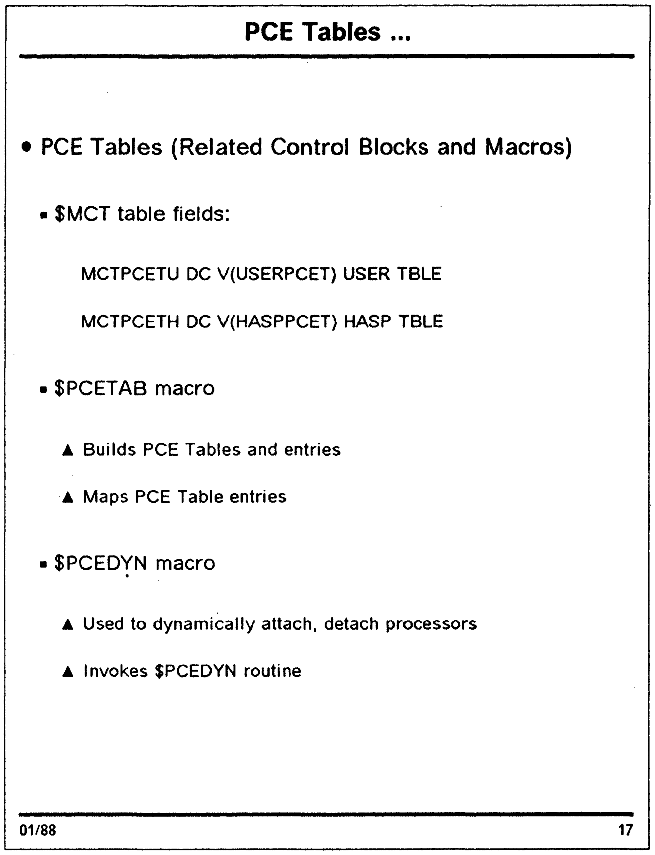
The PCE table, among other things, describes when the processor should be generated, when it should be dispatched, and whether it is related to a device.

PCEs may be generated during JES2 initialization or after initialization. Therefore, you could specify that a processor be created and be present for the life of JES2 or that it be created only upon installation demand (i.e., after initialization). This provides a way to save storage or other resources. You can also specify that a processor should never be created. This would be useful for documentation purposes. JES2 has such a table element to document the initialization PCE. This PCE is needed prior to the ability to create a processor through the PCE functional routine.

You may also specify when the processor should be given control. If you want a processor to be given control concurrent with the HASPWARM processor for final initialization processing, this can be specified. If the installation processor doesn't need to take control until initialization has completed but concurrent with the other JES2 processors, this can be specified. You can also indicate that a processor is only to get control when it is $POSTed for work (i.e., it is some sort of service processor).

Processors can also be associated with a device. This is done by pointing to a particular DCT table from the PCE table. This is a one-to-one correspondence. That is, one PCE is associated with one device.

PCE Control Blocks and Macros



PCE Tables ...

- PCE Tables (Related Control Blocks and Macros)
  - $MCT table fields:

    ```
    MCTPCETU DC V(USERPCET) USER TBLE

    MCTPCETH DC V(HASPPCET) HASP TBLE
    ```

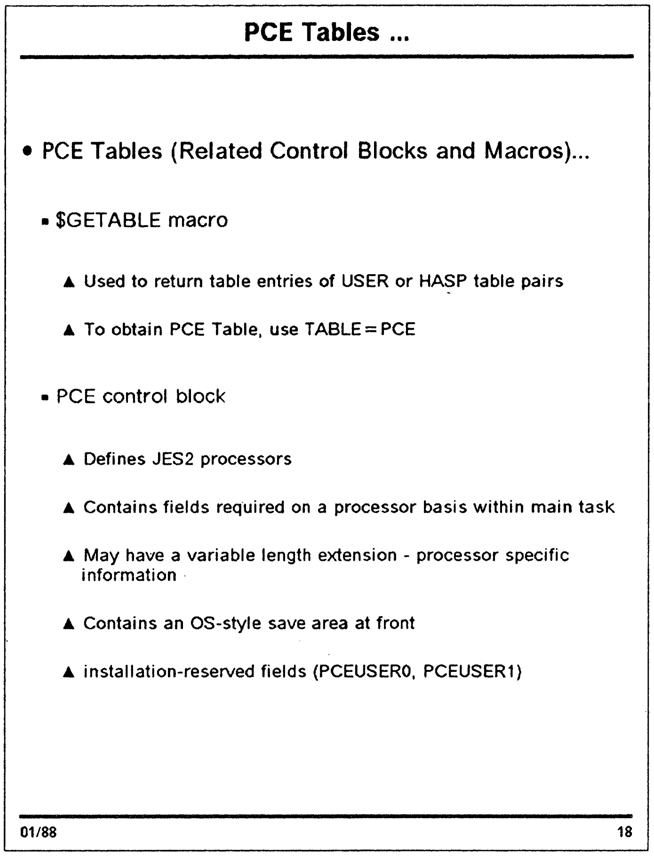
  - $PCETAB macro
    - ▲ Builds PCE Tables and entries
    - ▲ Maps PCE Table entries
  - $PCEDYN macro
    - ▲ Used to dynamically attach, detach processors
    - ▲ Invokes $PCEDYN routine

01/88 17

The table pair used to point to the PCE tables is located in the $MCT. The field MCTPCETU will contain the address of the installation table, if such a table exists. If you want to link-edit your table with JES2 you must name the table USERPCET and link-edit it with HASJES20. The JES2-defined PCE table is pointed to from the $MCT field MCTPCETH and is named HASPPCET.

To aid in creating PCE tables, JES2 supplies a macro named $PCETAB. This macro builds both the JES2 and installation tables and table elements. This macro also contains the mapping macro for the PCE table and element. We will describe this macro and its operands more thoroughly later on ("A JES2 PCE Table" on page 29).

JES2 provides a mechanism to dynamically attach and detach processors via the $PCEDYN service. This service is invoked by the $PCEDYN macro. The service routine makes use of the PCE tables for the attaching and detaching of the processor (PCE).



PCE Tables ...

- PCE Tables (Related Control Blocks and Macros)...
  - $GETABLE macro
    - ▲ Used to return table entries of USER or HASP table pairs
    - ▲ To obtain PCE Table, use TABLE=PCE
  - PCE control block
    - ▲ Defines JES2 processors
    - ▲ Contains fields required on a processor basis within main task
    - ▲ May have a variable length extension - processor specific information
    - ▲ Contains an OS-style save area at front
    - ▲ installation-reserved fields (PCEUSER0, PCEUSER1)

01/88 18

The $GETABLE macro invokes the $GETABLE service routine that is located in the module HASPTABS. This service obtains a table element from the user or JES2 table. To obtain a PCE table, you would code TABLE=PCE operand. This macro will return the table element of the specified ID or, if LOOP is specified, return the next table element after the specified identifier.

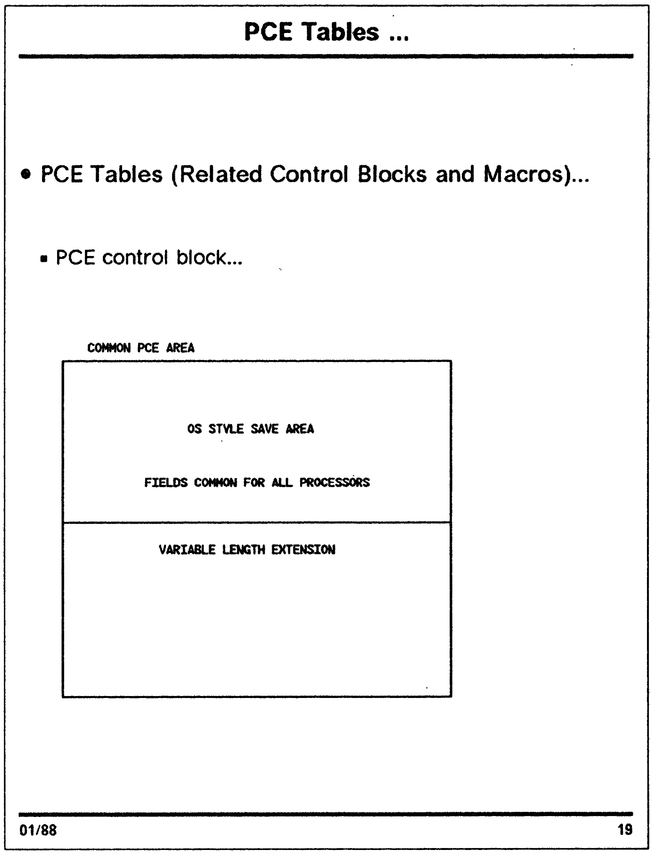
The major control block for adding or modifying a processor is the PCE (Processor Control Element). PCEs represent and define JES2 processors. This control block contains fields that are required on a processor basis within the JES2 main task.[4] The PCE is composed of a common section (all JES2 processor PCEs contain this common section) and an optional variable length section that is unique between processor types and contain processor specific information. The various processor types in JES2 include:

- Input
- JCL Conversion
- Execution
- Output
- Print
- Purge

The PCE common section includes an OS-style save area at the front. This is pointed to by R13 in the JES2 main task (i.e., it points to the PCE with the OS-style save area in the front) which MVS services use as the available save area. In addition, two installation-reserved fields are con-

[4] See the topic 'JES2 Structure' in *MVS/XA JES2 Logic* if the term 'main task' is unfamiliar to you in this context.

tained in the common section. These two words are the PCEUSER0 and PCEUSER1 fields in the PCE control block.



PCE Tables ...

- PCE Tables (Related Control Blocks and Macros)...
  - PCE control block...

COMMON PCE AREA

OS STYLE SAVE AREA

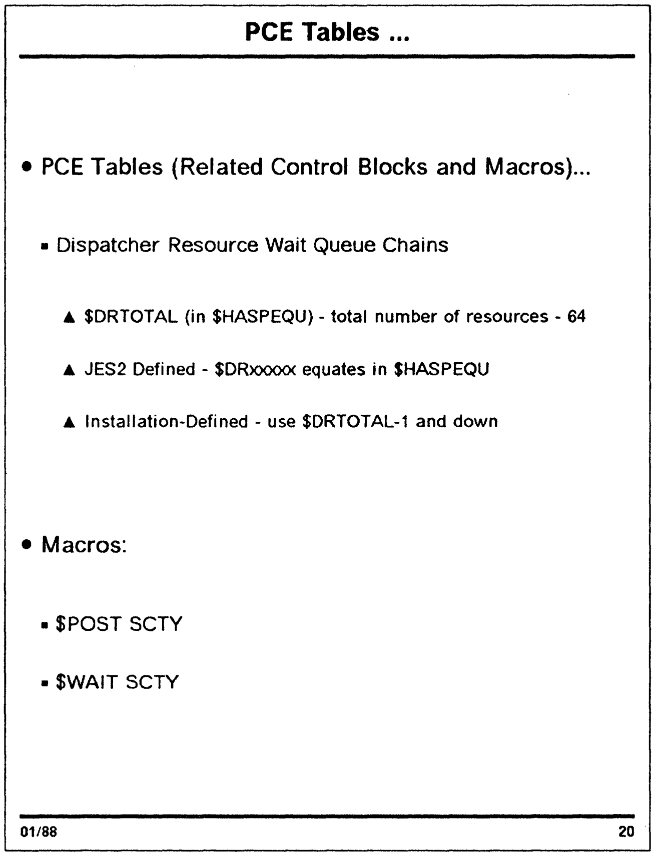
FIELDS COMMON FOR ALL PROCESSORS

VARIABLE LENGTH EXTENSION

01/88 19

The figure above illustrates what the PCE contains. The common area contains the OS-style save area at the front, followed by those fields that are common for all types of processors.

The variable length extension area is an optional extension to the common area that contains PCE-type specific information. Thus, the PCE extension for the reader (Input) PCE would be the same as other reader PCEs but different from the printer PCE extension area. The size of this extension area is specified on the PCE table.



PCE Tables ...

- PCE Tables (Related Control Blocks and Macros)...
  - Dispatcher Resource Wait Queue Chains
    - ▲ $DRTOTAL (in $HASPEQU) - total number of resources - 64
    - ▲ JES2 Defined - $DRxxxxx equates in $HASPEQU
    - ▲ Installation-Defined - use $DRTOTAL-1 and down
- Macros:
  - $POST SCTY
  - $WAIT SCTY

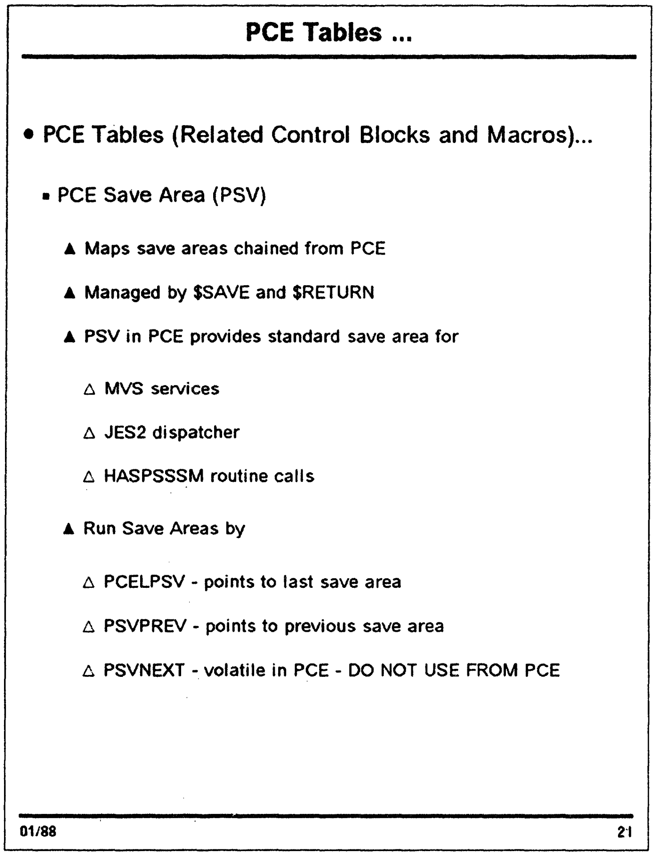
01/88 20

JES2 processors, unlike MVS tasks, maintain control of JES2 processing until they issue a $WAIT macro. When the $WAIT macro is issued, the JES2 dispatcher receives control and places the PCE on a queue for the requested resource. In JES2 the total number of resource queues is defined in $HASPEQU via an equate named $DRTOTAL. $DRTOTAL is defined for 64 resource queue chains. When the processor issues a $WAIT macro with a one- to five-character resource name, the macro and dispatcher place the processor on that $DRxxxxx queue, where $DRxxxxx is one of up to 64 resource names defined via an EQU. JES2-defined resources start at 0 and increase. You have the ability to define installation resource queues starting at 63 and decreasing.

Therefore, if a processor issued a '$WAIT SCTY', the dispatcher would place the processor on the wait queue defined as $DRSCTY.[5] This processor would remain on this queue until a $POST SCTY was done. When the $POST is done, the processors on the $DRSCTY wait queue are put on the JES2 ready queue where they will be dispatched by the JES2 dispatcher.

Additional information can be found on the $WAIT and $POST macros in *SPL: JES2 User Modifications and Macros.* Also, reference source module $HASPEQU for the IBM-defined resource queues.

[5] ('SCTY' is an installation-defined resource, as in the sample code in "Appendix A. Table Pairs Coding Example" on page 147.)



PCE Tables ...

- PCE Tables (Related Control Blocks and Macros)...
  - PCE Save Area (PSV)
    - ▲ Maps save areas chained from PCE
    - ▲ Managed by $SAVE and $RETURN
    - ▲ PSV in PCE provides standard save area for
      - △ MVS services
      - △ JES2 dispatcher
      - △ HASPSSSM routine calls
    - ▲ Run Save Areas by
      - △ PCELPSV - points to last save area
      - △ PSVPREV - points to previous save area
      - △ PSVNEXT - volatile in PCE - DO NOT USE FROM PCE

01/88 21

All save areas in the JES2 main task are chained from a PCE. The PCE contains the PSV (PCE Save Area) that maps save areas chained from the PCE as well as the save area in the PCE itself. Save areas chained off the PCE are managed by JES2 $SAVE and $RETURN services. The PCE save area is used for MVS service calls, by the JES2 dispatcher to save current register contents when the processor is $WAITed, and by calls to HASPSSSM service routines.

In order to run the JES2-style save areas you must run the save areas backwards. Thus, use field PCELPSV which points to the last (most recent) save area chained from the PCE and use PSVPREV in that save area to point to the previous save area. Do not use PSVNEXT from the PCE since this is a volatile field which may be overlaid by MVS services, the JES2 dispatcher, or HASPSSSM even with JES2-style save areas chained from the PCE.

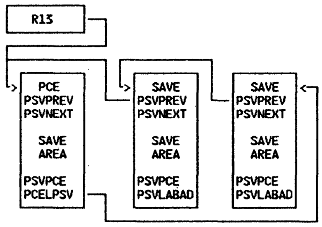
JES2 save areas are nearly identical to standard OS save areas in format, but not in the way they are used and accessed. So:

- Register 13 does not point to an available save area in the JES2 main task. One can do a STM into R13 (the PCE) but the correct approach would be to do a $SAVE to obtain a JES2 save area and save the registers in the JES2 main task environment.
- You cannot use register 13 to follow the chain of save areas from the JES2 main task, since R13 (the PCE) is kept as an available save area for calls to MVS services, not JES2 routines.
- The save area format is different in that there are extra words on the end of JES2 save areas that we use to point to the PCE (PSVPCE) and the $SAVE identifier at the location where the $SAVE was issued (PSVLABAD).

PCE Tables ...

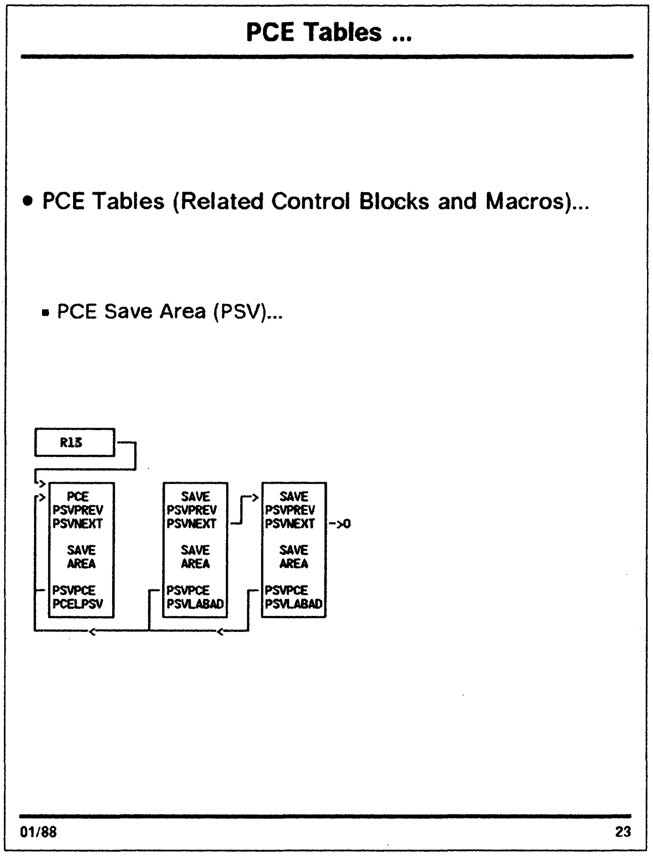
- PCE Tables (Related Control Blocks and Macros)...

  - PCE Save Area (PSV)...



01/88 22

The figure above illustrates the chaining used for JES2 save areas. The PCE field PCELPSV will point to the last (most recent) JES2 save area and by using PSVPREV, the save areas can be chained back to the PCE. The save area in the PCE is thus available for use by other services that require OS-style save areas.



PCE Tables ...

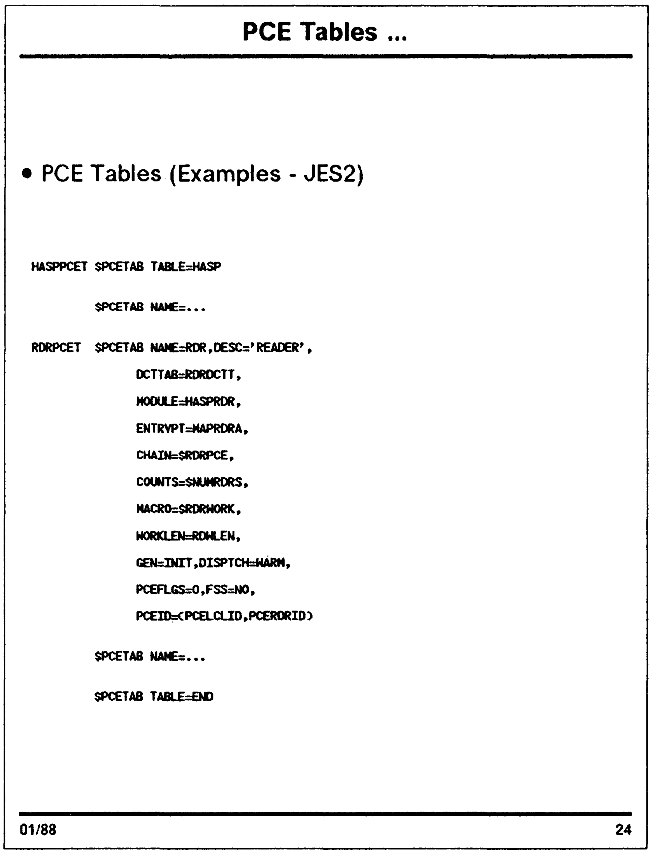
- PCE Tables (Related Control Blocks and Macros)...
  - PCE Save Area (PSV)...

R13
PCE
PSVPREV
PSVNEXT
SAVE
AREA
PSVPCE
PCELPSV
SAVE
PSVPREV
PSVNEXT
SAVE
AREA
PSVPCE
PSVLABAD
SAVE
PSVPREV
PSVNEXT
->0
SAVE
AREA
PSVPCE
PSVLABAD

01/88 23

You can use the PCE field PSVPCE from any JES2 save area to obtain the PCE address. In addition, while running the JES2 save areas, the PSVNEXT field is valid. Do not, however, use this field from the PCE, since it may not be valid.

A JES2 PCE Table



PCE Tables ...

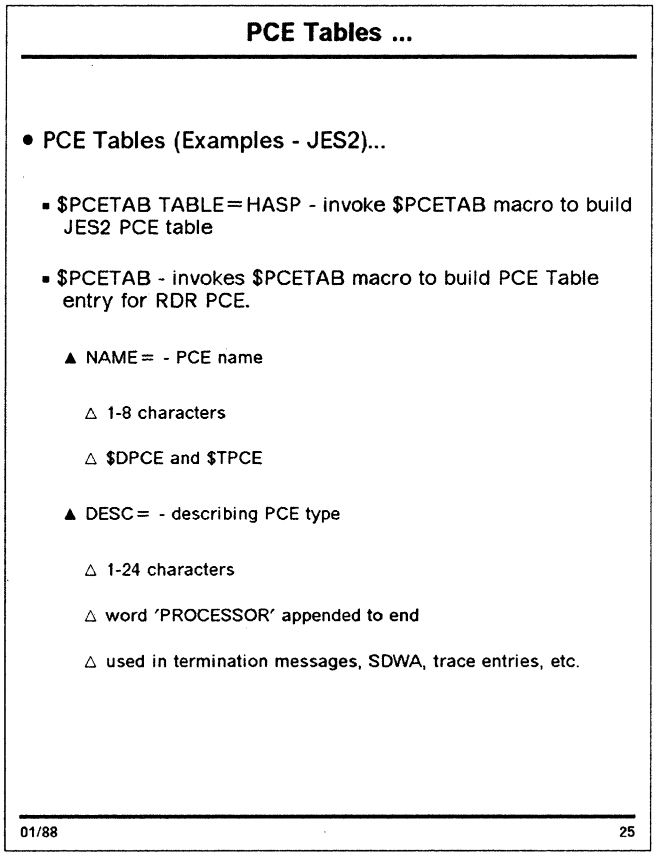
- PCE Tables (Examples - JES2)

```
HASPPCET $PCETAB TABLE=HASP

         $PCETAB NAME=...

RDRPCET  $PCETAB NAME=RDR,DESC='READER',
               DCTTAB=RDRDCTT,
               MODULE=HASPRDR,
               ENTRYPT=MAPRDRA,
               CHAIN=$RDRPCE,
               COUNTS=$NUMRDRS,
               MACRO=$RDRWORK,
               WORKLEN=RDWLEN,
               GEN=INIT,DISPTCH=WARM,
               PCEFLGS=0,FSS=NO,
               PCEID=(PCELCLID,PCERDRID)

         $PCETAB NAME=...

         $PCETAB TABLE=END
```

01/88 24

The figure above illustrates what the JES2 PCE table looks like. The table element shown represents all the information that JES2 needs to generally define a JES2 reader processor. This is the table element that is passed to the $PCEDYN service to create the reader PCE. Notice that the name of the PCE table is HASPPCET, the same as that specified in the V-type address constant in the $MCT.

Now we will describe each operand on the $PCETAB macro and how each should be specified.



PCE Tables ...

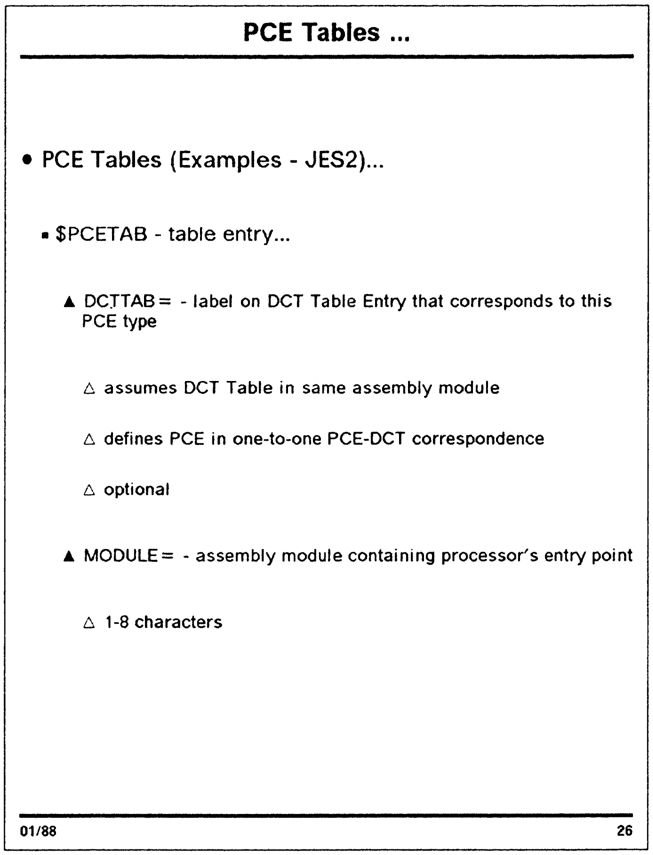
- PCE Tables (Examples - JES2)...
  - $PCETAB TABLE=HASP - invoke $PCETAB macro to build JES2 PCE table
  - $PCETAB - invokes $PCETAB macro to build PCE Table entry for RDR PCE.
    - ▲ NAME= - PCE name
      - △ 1-8 characters
      - △ $DPCE and $TPCE
    - ▲ DESC= - describing PCE type
      - △ 1-24 characters
      - △ word 'PROCESSOR' appended to end
      - △ used in termination messages, SDWA, trace entries, etc.

01/88 25

The JES2 table definitions are started by specifying 'TABLE=HASP'. This indicates to JES2 that this table is a JES2 table. You would specify 'TABLE=USER' to indicate that the table is an installation-coded table. Specifying whether it is a JES2 or installation table determines default values for the ENTRYPT and CHAIN $PCETAB operands. We will discuss these operands later. Specifying TABLE=HASP or TABLE=USER is the means JES2 provides to indicate the start of the table (TABLE=START) as discussed in "Concepts" on page 6.

When $PCETAB is specified with operands other than TABLE=, the macro generates a table element. In the above example, the table element that is generated will be for the Reader PCE.

The NAME operand specifies a one- to eight-character name. The command processor for $DPCE (display PCE) and $TPCE (set PCE) commands uses this name. The DESC operand specifies a one- to 24-character description of the PCE type. You can assume that the word 'PROCESSOR' will be appended to the characters specified. Termination messages, the SDWA, $TRACE entries, and other places throughout JES2 use the term.



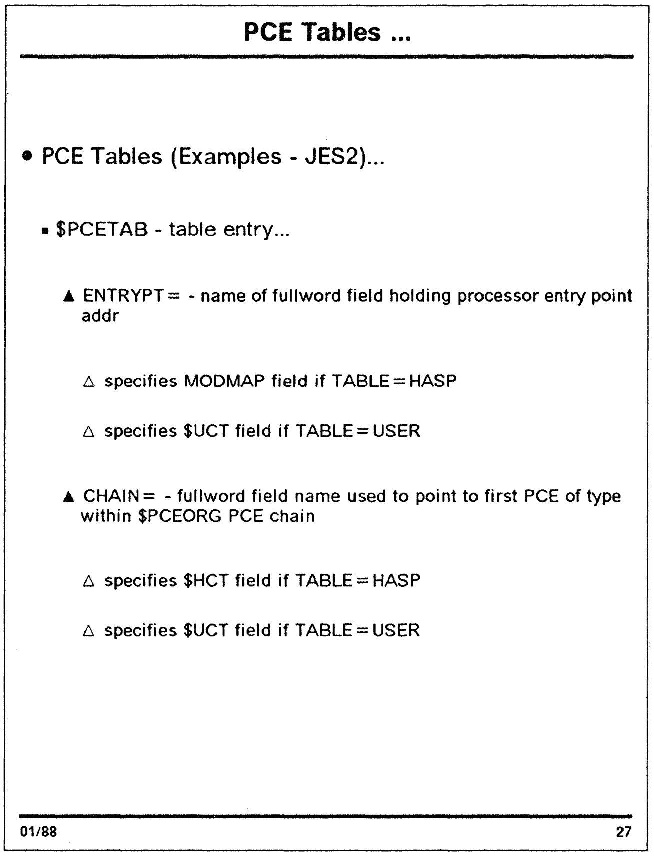
PCE Tables ...

- PCE Tables (Examples - JES2)...
  - $PCETAB - table entry...
    - ▲ DCTTAB= - label on DCT Table Entry that corresponds to this PCE type
      - △ assumes DCT Table in same assembly module
      - △ defines PCE in one-to-one PCE-DCT correspondence
      - △ optional
    - ▲ MODULE= - assembly module containing processor's entry point
      - △ 1-8 characters

01/88 26

The DCTTAB operand is used only if the processor being defined is a processor that will control a device. In the example for the RDR processor, this processor will be controlling a reader device. In order to match the PCE with the device, the DCTTAB operand is coded by pointing it to the $DCTTAB macro call that defines the device. DCTTABs are also included in the HASPTABS module but are not at this time installation-extensible. The PCE may only specify one DCT in this way, so therefore, the PCE can only correspond with one DCT.

The MODULE operand specifies the name of the assembly module containing the processor's entry point. This name is a one- to eight-character name. In the example, the module that contains the RDR processor's entry point is the HASPRDR module. This operand is only used for documentation. JES2 code does not use this field.



PCE Tables ...

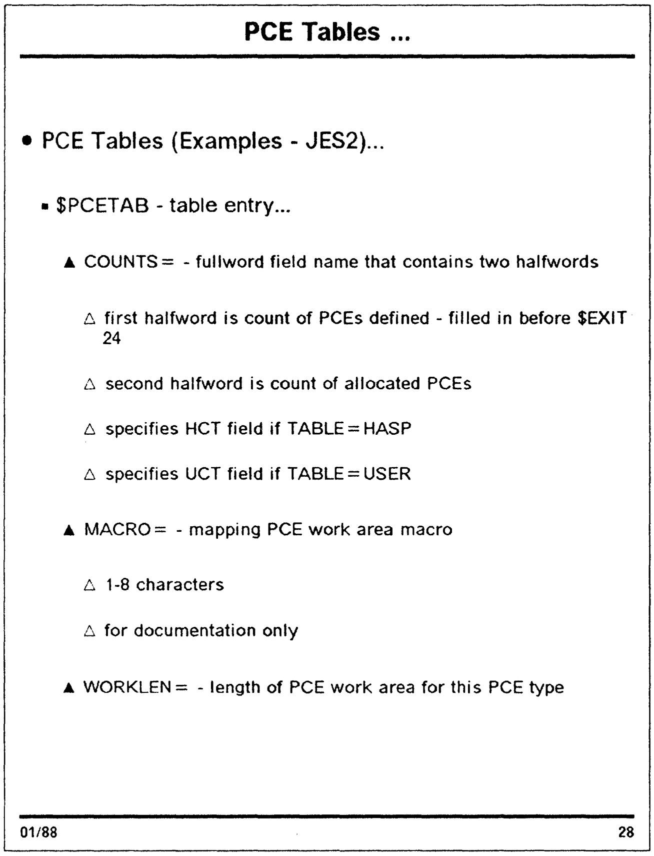
- PCE Tables (Examples - JES2)...
  - $PCETAB - table entry...
    - ▲ ENTRYPT= - name of fullword field holding processor entry point addr
      - △ specifies MODMAP field if TABLE=HASP
      - △ specifies $UCT field if TABLE=USER
    - ▲ CHAIN= - fullword field name used to point to first PCE of type within $PCEORG PCE chain
      - △ specifies $HCT field if TABLE=HASP
      - △ specifies $UCT field if TABLE=USER

01/88 27

The ENTRYPT operand tells JES2 where the entry point address is for this processor. This operand must be set to a fullword field for the entry point address. In the example, ENTRYPT=MAPRDRA; the fullword field was MAPRDRA. Since the table that contains the reader element is a JES2 table, this field is defaulted to be in $MODMAP. If you specify this field in an installation table it is defaulted to be in the $UCT. If you wish to override this default you would specify 'ENTRYPT=(name,$MODMAP)'. The field must be in either $MODMAP or the $UCT. (The $UCT, or User Control Table, is a control block obtained by the installation and chained from the $HCT from field $UCT in the $HCT.)[6]

The CHAIN operand is also a fullword field that tells JES2 where to chain the initial PCE of this type. All PCEs can be run by starting at $PCEORG in the $HCT. You use the specified CHAIN field to run PCEs of this type. In the example, CHAIN=$RDRPCE is the field that points to the first reader processor. PCEPREV and PCENEXT are fields used to chain to the next PCE. Since the table that contains the reader table element is a JES2 table, this field is defaulted to be in $HCT. If you specify this field, it is defaulted to be in the $UCT. If you wish to override this default you would specify 'CHAIN=(field,HCT)'. The field must be in either the $HCT or the $UCT.

[6] An example of how to define and chain a $UCT is in "Appendix A. Table Pairs Coding Example" on page 147.



PCE Tables ...

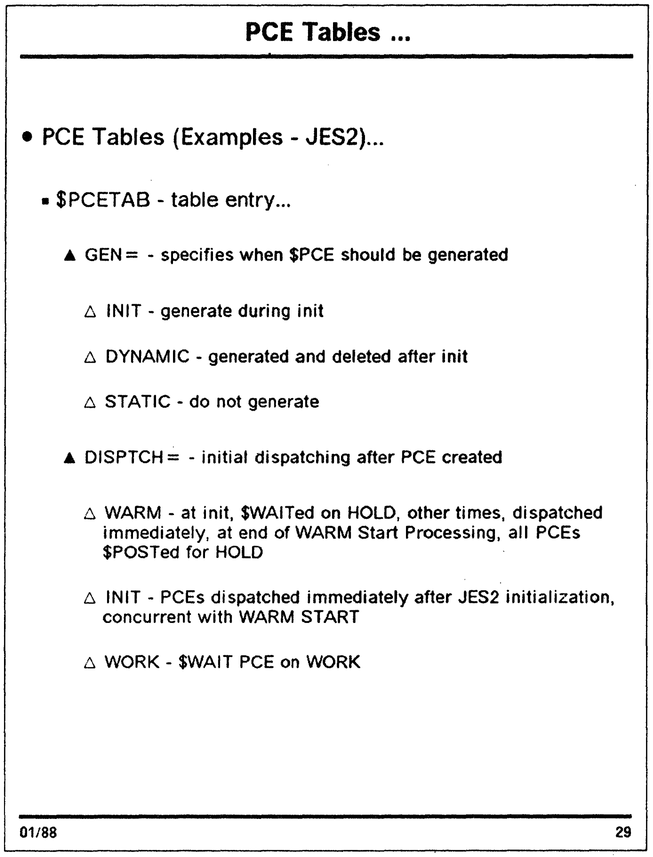
- PCE Tables (Examples - JES2)...
  - $PCETAB - table entry...
    - ▲ COUNTS= - fullword field name that contains two halfwords
      - △ first halfword is count of PCEs defined - filled in before $EXIT 24
      - △ second halfword is count of allocated PCEs
      - △ specifies HCT field if TABLE=HASP
      - △ specifies UCT field if TABLE=USER
    - ▲ MACRO= - mapping PCE work area macro
      - △ 1-8 characters
      - △ for documentation only
    - ▲ WORKLEN= - length of PCE work area for this PCE type

01/88 28

The COUNTS operand tells JES2 how many processors of this type should be created. This field points to a two halfword field where the first halfword specifies how many processors to create and the second halfword is where the $PCEDYN service saves how many are operative at the moment. In the example, COUNTS=$NUMRDRS indicates that at label $NUMRDRS in the $HCT is located the two halfwords. The default location for the field is in the $HCT if the table is a JES2 table and the $UCT if the table is an installation table. This can be overridden by specifying 'COUNTS=(field,HCT)' if the field that you want to use is in the $HCT.

The MACRO operand only documents what macro maps the PCE variable extension area. This mapping macro name can be from one to eight characters in length. In the example, MACRO=$RDRWORK, $RDRWORK is the name of the mapping macro for the reader variable extension area.

The WORKLEN operand tells JES2 how long the variable extension area is for this processor type. $PCEDYN uses this to $GETMAIN the $PCE and its extension in contiguous storage. In the example, WORKLEN=RDWLEN; RDWLEN is the equate for the length of the variable extension area for the reader processor.



PCE Tables ...

- PCE Tables (Examples - JES2)...
  - ▪ $PCETAB - table entry...
    - ▲ GEN= - specifies when $PCE should be generated
      - △ INIT - generate during init
      - △ DYNAMIC - generated and deleted after init
      - △ STATIC - do not generate
    - ▲ DISPTCH= - initial dispatching after PCE created
      - △ WARM - at init, $WAITed on HOLD, other times, dispatched immediately, at end of WARM Start Processing, all PCEs $POSTed for HOLD
      - △ INIT - PCEs dispatched immediately after JES2 initialization, concurrent with WARM START
      - △ WORK - $WAIT PCE on WORK

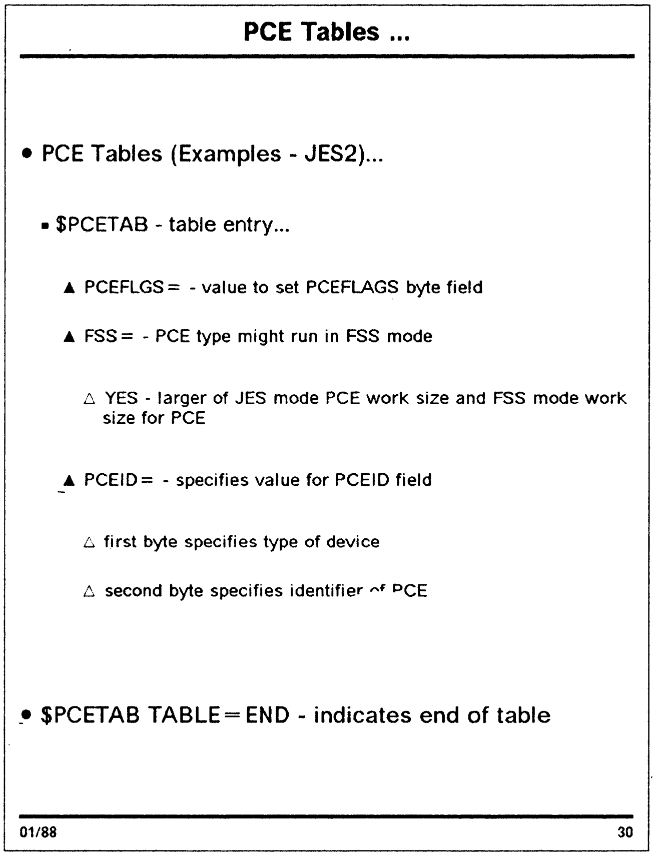
01/88 29

The GEN operand tells JES2 when this PCE should be created. There are three values that can be specified; INIT, DYNAMIC, and STATIC.

- GEN=INIT indicates to JES2 that this processor should be generated during initialization, along with most of the JES2 processors.
- GEN=DYNAMIC indicates to JES2 that this processor should not be automatically generated, but should be created through a specific $PCEDYN service call. Such processors may also be dynamically deleted.
- GEN=STATIC tells JES2 that this table is for documentation only. The Initialization PCE of JES2 is documented like this.

The DISPTCH operand tells JES2 when the processor should be dispatched after it is created. There are three values that can be specified. These values are:

- DISPTCH=WARM causes two actions to take place dependent on when the PCE is created.
  1. If the processor is created during initialization, then the processor is $WAITed on HOLD. After WARM start processing is completed the processor will be $POSTed for HOLD and the processor will be given control.
  2. If the processor is created after WARM processing, the processor is immediately dispatched.
- DISPTCH=INIT causes JES2 to give the processor control immediately after initialization, concurrent with WARM processing. For those processors attached after initialization, the processors will be dispatched immediately.
- DISPTCH=WORK causes JES2 to $WAIT the processor on WORK. Thus, the processor will not be dispatched until it is $POSTed for WORK.



PCE Tables ...

- PCE Tables (Examples - JES2)...
  - $PCETAB - table entry...
    - ▲ PCEFLGS= - value to set PCEFLAGS byte field
    - ▲ FSS= - PCE type might run in FSS mode
      - △ YES - larger of JES mode PCE work size and FSS mode work size for PCE
    - ▲ PCEID= - specifies value for PCEID field
      - △ first byte specifies type of device
      - △ second byte specifies identifier of PCE
- $PCETAB TABLE=END - indicates end of table

01/88 30

The operand PCEFLGS primes the PCE field PCEFLAGS when the PCE is created by $PCEDYN. The valid values that can be specified for this operand are:

- PCETRACE - processor is eligible for tracing
- PCEDSPXP - processor permanently exempt from non-dispatchability
- PCEDSPXT - processor temporarily exempt from non-dispatchability
- PCENWIOP - implicit $WAITs in I/O processing should be prohibited

The operand FSS indicates that the device associated with this PCE is a Functional Subsystem (FSS) device. If FSS=YES is specified, a larger base PCE is obtained.

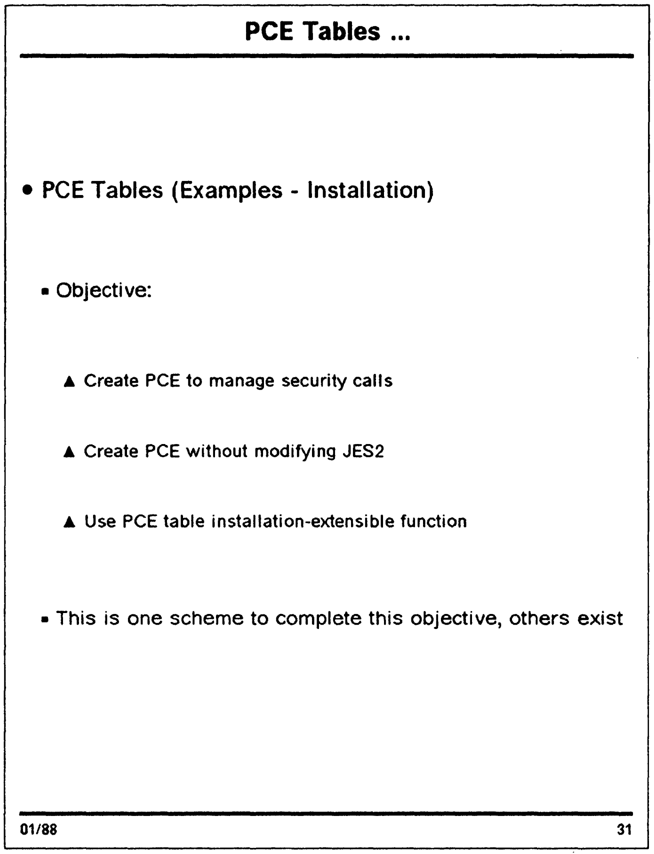
The PCEID operand specifies what values should be placed in the PCEID field in the PCE. This identifier field sets the type and identifier of the processor in the PCE. The first byte of the PCEID field specifies the type of processor. The JES2 types are:

- Non-Device processor (x′00′)
- Local Special PCE (x′01′)
- Remote Special PCE (x′02′)
- Network Special PCE - indicates NJE or XFR JT/JR/ST/JR (x′04′)
- Internal Special PCE (x′08′)
- Print Special PCE (x′80′)
- Punch Special PCE (x′40′)
- XFR Special PCE (x′20′)

The second byte of the PCE identifier field specifies the identifier of the processor. If only one value is specified for the PCEID (e.g., PCEID = value) then the specified value is placed as the identifier of the PCE. If you wish to specify your own PCE identifier you should start at 255 and work your way down. JES2 starts at 1 and increases. There are currently 30 PCE identifiers. These are defined in the $PCE macro. In the example, PCEID = (PCELCLID,PCERDRID) indicates that the PCE is a Local special PCE and its identifier is that of an Input processor.

When TABLE = END is encountered, the table is closed. This indicates the end of the JES2 PCE table. All JES2-defined PCEs are defined within this single table.

An Installation PCE Table



PCE Tables ...

- PCE Tables (Examples - Installation)
  - Objective:
    - ▲ Create PCE to manage security calls
    - ▲ Create PCE without modifying JES2
    - ▲ Use PCE table installation-extensible function
  - This is one scheme to complete this objective, others exist

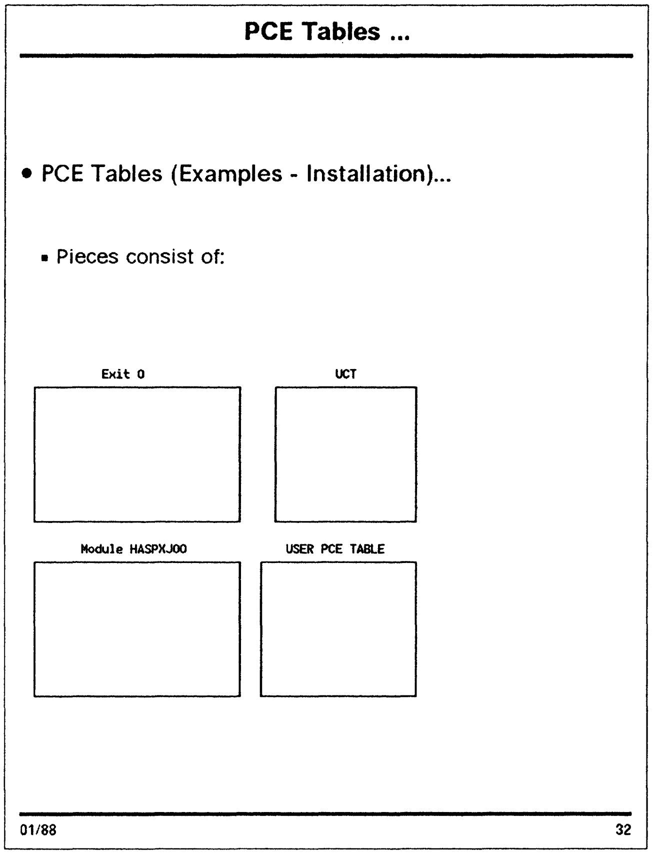
01/88 31

In order to show how you would specify an installation PCE table, the remaining description of the PCE tables will step through creating an installation-defined security PCE. This security PCE, as it is implemented here, is not required to fulfill the security objective. This example is purely for illustration.

Objective

The objective is to create a PCE to manage security calls. You wish to achieve this without modifying JES2 and to use the PCE tables as the means to define this PCE to JES2.

Required Pieces



PCE Tables ...

- PCE Tables (Examples - Installation)...
  - Pieces consist of:

Exit 0

UCT

Module HASPXJ00

USER PCE TABLE

01/88 32

To achieve the objective, you will need to code four pieces. These pieces are:

1. Exit 0

   As discussed in "Concepts" on page 6, there are two ways to link the installation table with JES2:

   a. The first of these is to link-edit the installation PCE table with the HASJES20 load module. This requires that the name of the installation table be USERPCET.

   b. If you do not wish to link-edit the installation PCE table with HASJES20 or do not wish to name the installation table USERPCET, then you must fill in the address of the installation PCE table into the $MCT at field MCTPCETU. This is the second method. This method requires that you fill in the address before invoking the $PCEDYN service routine to create the processor. Depending on when you want the processor generated, you may fill in the address early in initialization or after JES2 is up and running.

   In this example, you will fill in the address of the PCE table early in initialization, specifically in exit 0. Therefore, an exit 0 is required to load the module (if not already loaded) and resolve the address of the table.

2. UCT

   As has been indicated when examining the JES2 PCE table, there are certain operands that assume $UCT fields in the installation PCE table entries. Of course you may override these assumptions, but the objective of this effort was to use the tables and not modify JES2. Therefore, a $UCT must be created that will hold certain values.

3. Module HASPXJ00

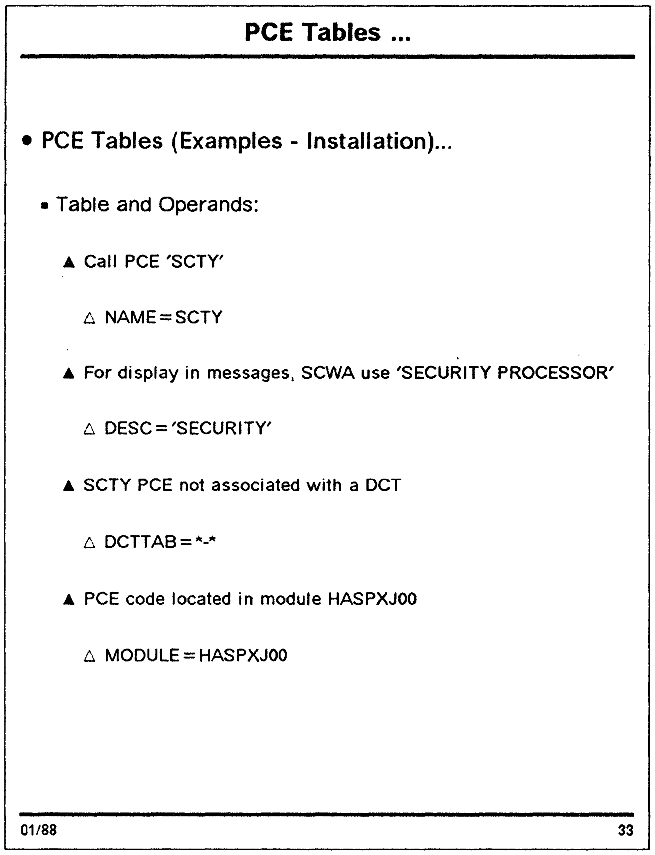
Since you are coding a new JES2 processor, you must write the code that is the processor. In this example, the code will reside in the module HASPXJ00. This name was chosen because it is one of the reserved-for-installation-use names that JES2 has set up in the $MODMAP control block. In this way, you can link-edit this module with HASJES20 and have its address in $MODMAP and not have to do the load of this exit from exit 0.

For this example, HASPXJ00 illustrates this function. However, the rest of the sample code will assume that this module is not in the HASJES20 load module and must be loaded by exit 0.

4. User PCE Table

   You will have to code a PCE table that includes an element for the processor. We will describe this installation table element in a step-wise fashion below.

Coding the Installation PCE Table



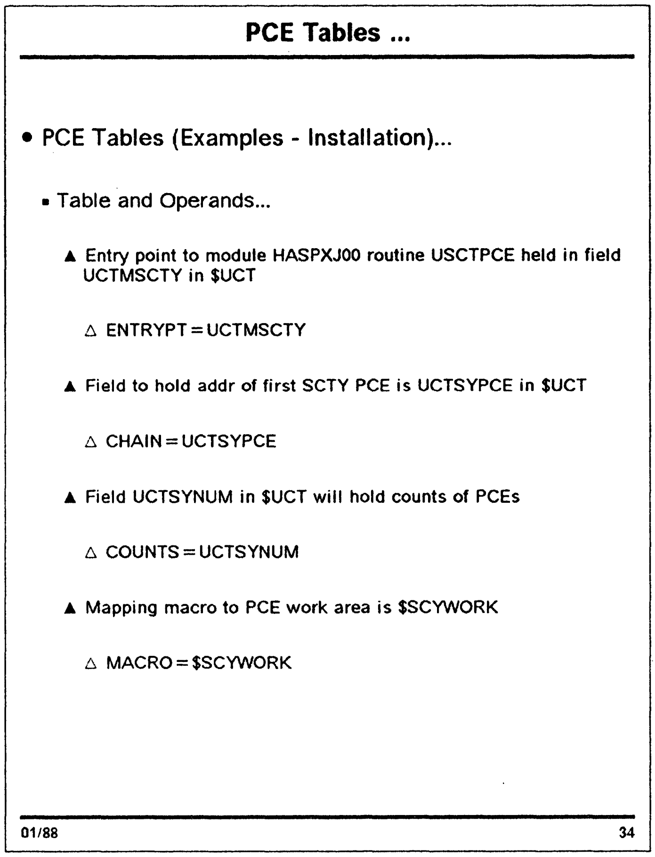
PCE Tables ...

- PCE Tables (Examples - Installation)...
  - ▪ Table and Operands:
    - ▲ Call PCE 'SCTY'
      - △ NAME=SCTY
    - ▲ For display in messages, SCWA use 'SECURITY PROCESSOR'
      - △ DESC='SECURITY'
    - ▲ SCTY PCE not associated with a DCT
      - △ DCTTAB=*-*
    - ▲ PCE code located in module HASPXJ00
      - △ MODULE=HASPXJ00

01/88 33

In the figure above, you wanted to create a security PCE which would be called SCTY. Therefore, you specified 'NAME=SCTY'. For display, the description to be issued was 'SECURITY PROCESSOR'. Therefore you specified 'DESC=SECURITY'. Remember that the word 'PROCESSOR' is appended to the end of the value specified on the DESC operand.

Since the security processor was not to be associated with a device, there was no DCT table to be specified so 'DCTTAB= *-*' was coded. MODULE=HASPXJ00 was coded since the name of the module to contain the processor code was HASPXJ00.



PCE Tables ...

- PCE Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Entry point to module HASPXJ00 routine USCTPCE held in field UCTMSCTY in $UCT
      - △ ENTRYPT=UCTMSCTY
    - ▲ Field to hold addr of first SCTY PCE is UCTSYPCE in $UCT
      - △ CHAIN=UCTSYPCE
    - ▲ Field UCTSYNUM in $UCT will hold counts of PCEs
      - △ COUNTS=UCTSYNUM
    - ▲ Mapping macro to PCE work area is $SCYWORK
      - △ MACRO=$SCYWORK

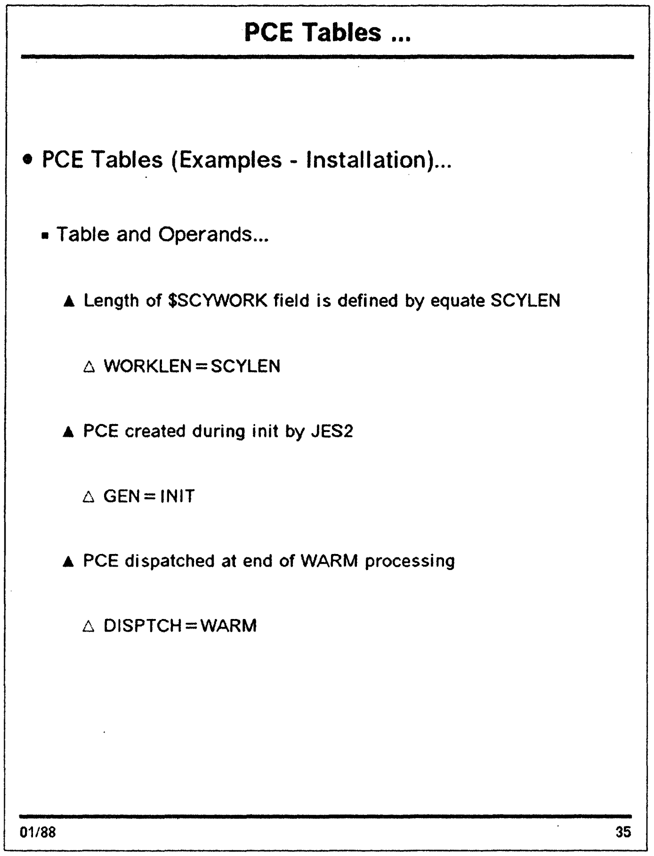
01/88 34

The field to hold the entry point address is in the $UCT. The name of the field is UCTMSCTY. It will hold the address of the routine USCTPCE. Therefore, we code 'ENTRYPT=UCTMSCTY' on the $PCETAB.

The $UCT field to hold the pointer to the first security PCE is UCTSYPCE. Thus, we code 'CHAIN=UCTSYPCE' to tell JES2 the name of the chain field. Since the table element is in the installation table, the field will default to being in the $UCT.

The COUNTS operand specifies where the $PCEDYN service routine is to find out how many PCEs of this type it may create and to keep track of how many it has created. This field defaults to being in the $UCT for the installation table, therefore the $UCT field that will hold the counts is UCTSYNUM. Thus, we specify 'COUNTS=UCTSYNUM'.

Since this processor will need fields that are unique to the security type of processor, it will need its own variable extension area. The macro that we use to map this extension area is $SCYWORK. This is documented in the $PCETAB by specifying 'MACRO=$SCYWORK'.



PCE Tables ...

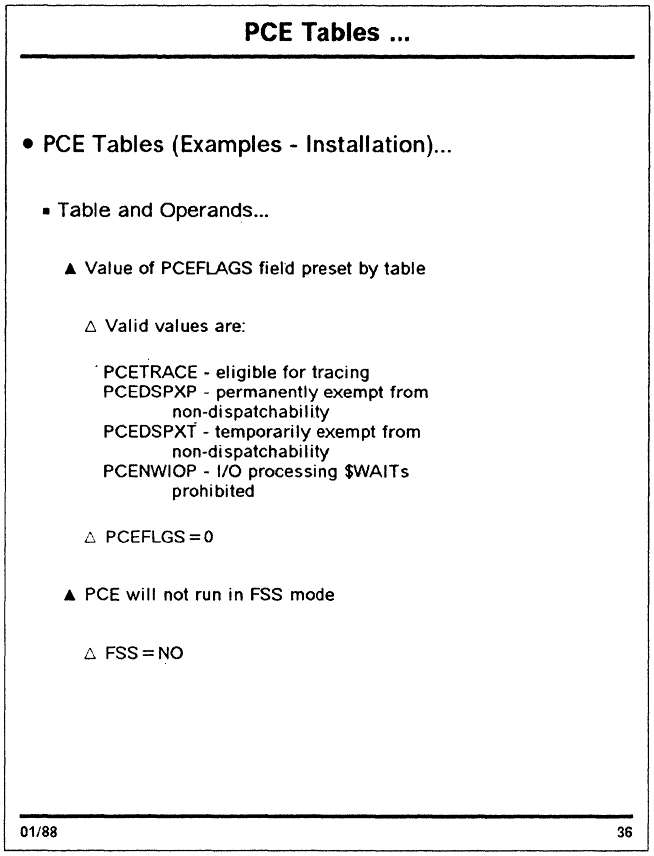
- PCE Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Length of $SCYWORK field is defined by equate SCYLEN
      - △ WORKLEN=SCYLEN
    - ▲ PCE created during init by JES2
      - △ GEN=INIT
    - ▲ PCE dispatched at end of WARM processing
      - △ DISPTCH=WARM

01/88 35

The length of the variable extension area of the security PCE is defined via the equate SCYLEN in the $SCYWORK macro. This is the value that we specify in the table: WORKLEN=SCYLEN.

You have also decided that the processor should be generated during initialization when the other JES2 processors are generated. Therefore, we specify 'GEN=INIT' on the $PCETAB macro to create the security processor.

The processor should receive control after WARM processing. This assumes that the security processor will not be needed during WARM processing. To specify this, you will code 'DISPTCH=WARM' on the macro call.



PCE Tables ...

- PCE Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Value of PCEFLAGS field preset by table
      - △ Valid values are:

        PCETRACE - eligible for tracing
        PCEDSPXP - permanently exempt from non-dispatchability
        PCEDSPXT - temporarily exempt from non-dispatchability
        PCENWIOP - I/O processing $WAITs prohibited

      - △ PCEFLGS=0
    - ▲ PCE will not run in FSS mode
      - △ FSS=NO

01/88 36

The PCEFLGS operand specifies what initial value the PCE PCEFLAGS field should contain after it is created by $PCEDYN. If the initial state of the processor should be that it:

- should be traced, then PCETRACE should be specified.
- should be marked as permanently exempt from non-dispatchability, then PCEDSPXP should be specified. There are currently five JES2 processors that are marked as permanently exempt. These are:

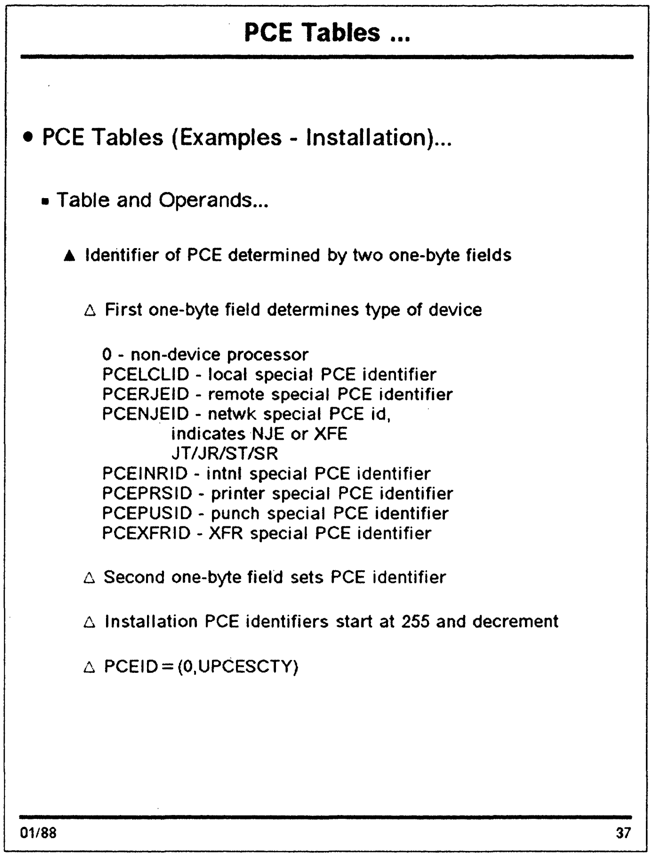
  1. Asynchronous I/O Processor - this processor handles asynchronous I/O requests
  2. Communications Processor - processes operator commands
  3. Line Manager Processor - processes line related processing
  4. STIMER/TIMER Processor - processes asynchronous timer requests
  5. Checkpoint Processor - manages the checkpoint data sets

  If the installation processor is one that should never be marked non-dispatchable, then you should set this value.

- should be marked as temporarily exempt from non-dispatchability, then PCEDSPXT should be specified. This value would be specified if some processing must be completed by this processor that would fail if the processor was marked non-dispatchable.
- cannot wait in the case of an I/O error, then you should specify PCENWIOP.

In this example, the security processor has no special requirements, therefore, we code 'PCEFLAGS=0'.

Since the security processor will not run in FSS mode, 'FSS=NO' will also be specified.



PCE Tables ...

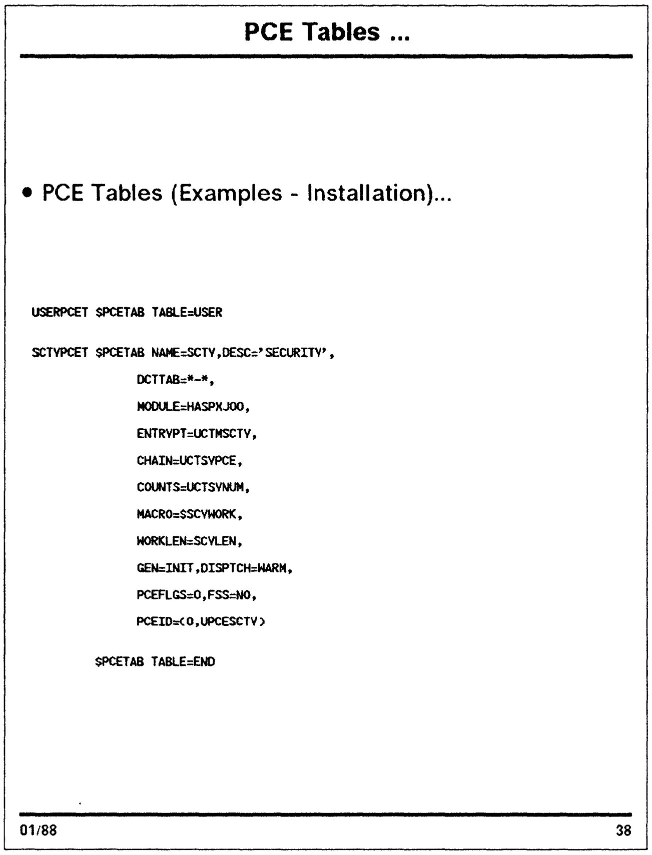
- PCE Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Identifier of PCE determined by two one-byte fields
      - △ First one-byte field determines type of device

        0 - non-device processor
        PCELCLID - local special PCE identifier
        PCERJEID - remote special PCE identifier
        PCENJEID - netwk special PCE id,
        indicates NJE or XFE
        JT/JR/ST/SR
        PCEINRID - intnl special PCE identifier
        PCEPRSID - printer special PCE identifier
        PCEPUSID - punch special PCE identifier
        PCEXFRID - XFR special PCE identifier
      - △ Second one-byte field sets PCE identifier
      - △ Installation PCE identifiers start at 255 and decrement
      - △ PCEID=(0,UPCESCTY)

01/88 37

The PCEID operand specifies the type and identifier of the processor. The type for the security processor is zero, since the processor is a non-device processor. The identifier of the processor is 255. This is because installation-specified identifiers should start at 255 and decrease since JES2 processors start at 1 and work their way up. We code an equate in the $UCT named UPCESCTY and set it to 255. Therefore, we specify the operand as 'PCEID=(0,UPCESCTY)'.

Resulting PCE Table



PCE Tables ...

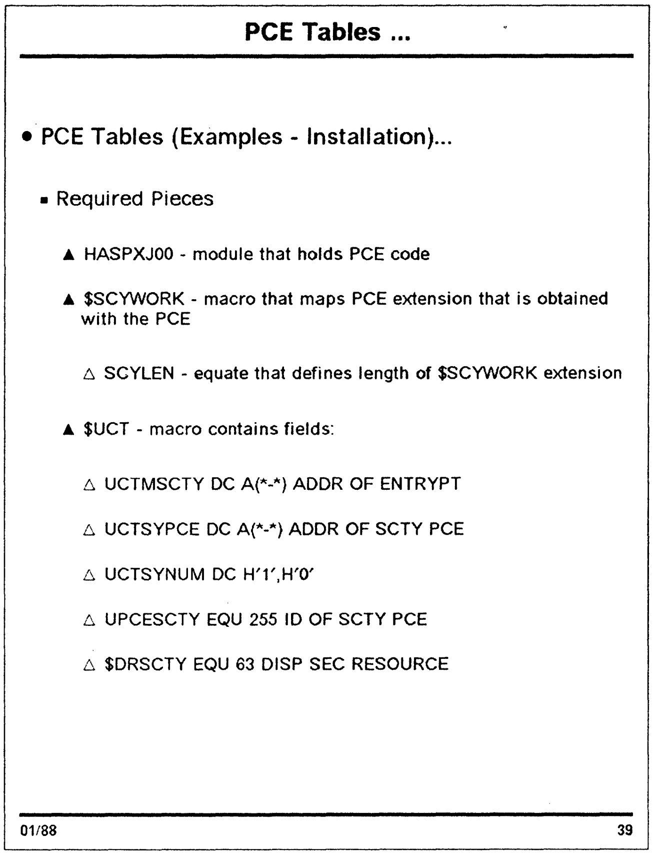
- PCE Tables (Examples - Installation)...

```
USERPCET $PCETAB TABLE=USER

SCTYPCET $PCETAB NAME=SCTY,DESC='SECURITY',
               DCTTAB=*-*,
               MODULE=HASPXJ00,
               ENTRYPT=UCTMSCTY,
               CHAIN=UCTSYPCE,
               COUNTS=UCTSYNUM,
               MACRO=$SCYWORK,
               WORKLEN=SCYLEN,
               GEN=INIT,DISPTCH=WARM,
               PCEFLGS=0,FSS=NO,
               PCEID=(0,UPCESCTY)

         $PCETAB TABLE=END
```

01/88 38

The figure above shows the resulting installation PCE table to add a security processor to JES2. The table is begun with a TABLE = USER to tell JES2 that this is an installation table. The name of the processor will be SCTY and we will call it 'SECURITY PROCESSOR'. The code for the processor will reside in the HASPXJ00 module with the entry point address contained at the field UCTMSCTY in the $UCT. The first security processor is chained from the UCTSYPCE field in the $UCT. The count of how many security processors that can be generated and the count of how many have already been created will reside at the two halfword field UCTSYNUM in the $UCT.

Security processors will require their own variable extension area that is mapped by macro $SCYWORK and is SCYLEN in length. The processors should be generated during JES2 initialization but not dispatched until after WARM processing. No default PCE characteristics need be set (i.e., PCEFLGS is equal to zero) and the PCE is not an FSS supported processor. The processor's identifier is 255, as set by the equate UPCESCTY. The table is delineated by the TABLE = END operand.



PCE Tables ...

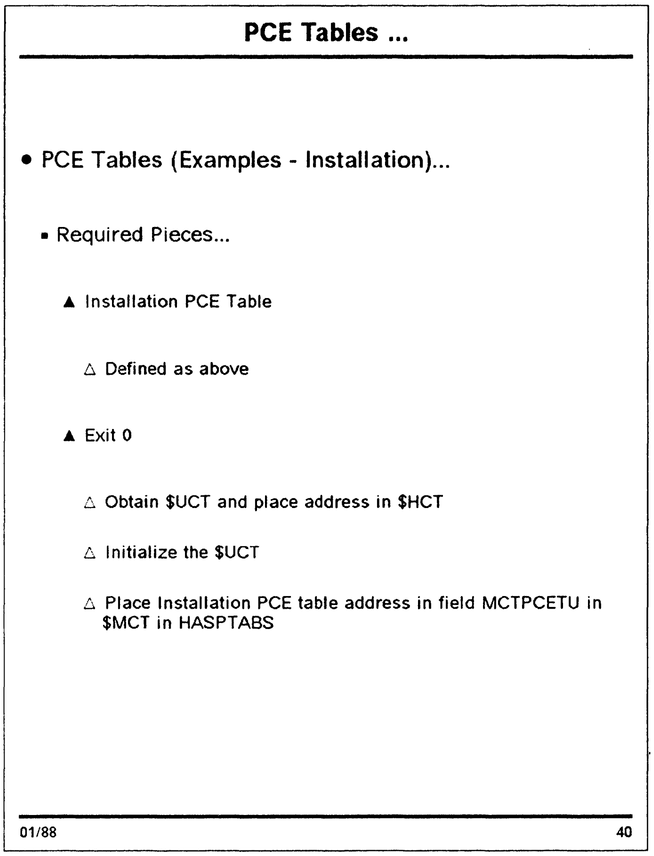
- PCE Tables (Examples - Installation)...
  - ■ Required Pieces
    - ▲ HASPXJ00 - module that holds PCE code
    - ▲ $SCYWORK - macro that maps PCE extension that is obtained with the PCE
      - △ SCYLEN - equate that defines length of $SCYWORK extension
    - ▲ $UCT - macro contains fields:
      - △ UCTMSCTY DC A(*-*) ADDR OF ENTRYPT
      - △ UCTSYPCE DC A(*-*) ADDR OF SCTY PCE
      - △ UCTSYNUM DC H'1',H'0'
      - △ UPCESCTY EQU 255 ID OF SCTY PCE
      - △ $DRSCTY EQU 63 DISP SEC RESOURCE

01/88 39

Now that you have completed the installation PCE table, the other required pieces and fields may be defined. You will have to write a HASPXJ00 module that holds the PCE code. A macro must be created called $SCYWORK that will map the PCE extension. A field named SCYLEN is required within the macro to define the length of the extension area needed.

In the installation-defined $UCT, several fields must be coded. The address of the entry point for the HASPXJ00 module for the installation PCE is held in the UCTMSCTY field. The address of the first security PCE is chained from the UCTSYPCE field. The UCTSYNUM field is a two halfword field where the first field defines the number of security PCEs that are to be created and the second indicates to $PCEDYN how many have been created.

Finally, two equates must be defined. The first is the identifier of the installation security PCE (set at 255) and a dispatching security resource. The $DRSCTY equate tells the PCE that some work is ready for it to process. The installation PCE will '$WAIT SCTY' (which will result in the PCE being put on the resource queue of 63) for work. When there is work for it to do, it is $POSTed for SCTY (i.e., $DRSCTY = 63) and put on the ready queue.



PCE Tables ...

- PCE Tables (Examples - Installation)...
  - Required Pieces...
    - ▲ Installation PCE Table
      - △ Defined as above
    - ▲ Exit 0
      - △ Obtain $UCT and place address in $HCT
      - △ Initialize the $UCT
      - △ Place Installation PCE table address in field MCTPCETU in $MCT in HASPTABS

01/88 40

The last two pieces that are required are the installation PCE table, as coded above, and the code for exit 0. The exit 0 code is required to do three things.

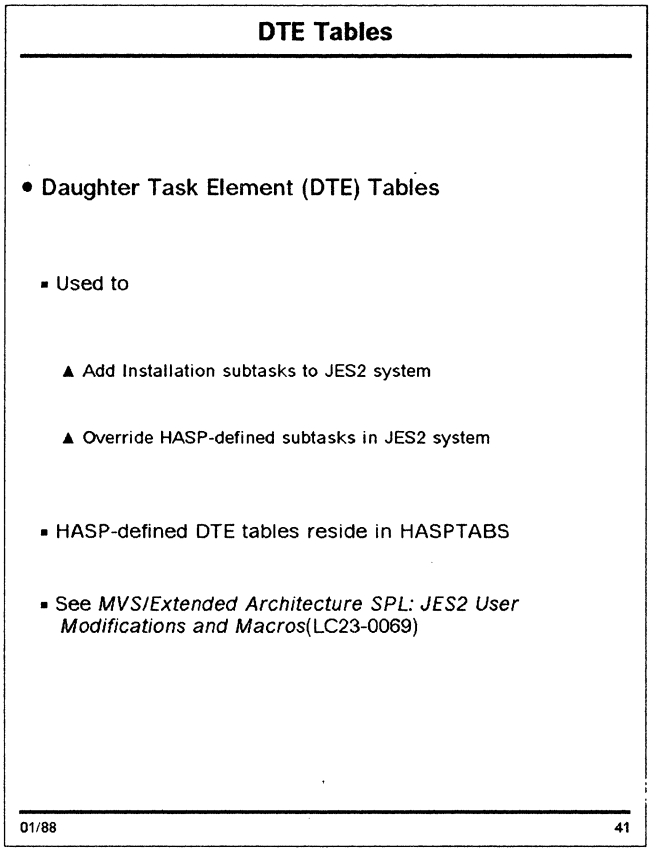
1. It must obtain the $UCT and place the $UCTs address in the $HCT.
2. It must initialize the $UCT fields. The fields that must be initialized include, at least, the UCTMSCTY, UCTSYPCE, and the first halfword of UCTSYNUM.
3. Finally, it must place the installation PCE table address in the MCTPCETU field in the $MCT in module HASPTABS.

The code that is contained in "Appendix A. Table Pairs Coding Example" on page 147 is the code as an installation would be required to code it. This code includes:

- Exit 0 that obtains the $UCT, places the $UCT address in the $HCT, initializes the $UCT, and places the installation PCE table address in the $MCT.
- The HASPXJ00 module contains, among other items that we will describe shortly, the PCE code and the installation PCE table.
- The macros for the $UCT and $SCYWORK. In addition, a $USERCBS macro extends the $MODULE macro so that you can use installation-created macros without modifying the $MODULE macro.

DTE Tables

What Is a DTE Table?



DTE Tables

- Daughter Task Element (DTE) Tables
  - Used to
    - ▲ Add Installation subtasks to JES2 system
    - ▲ Override HASP-defined subtasks in JES2 system
  - HASP-defined DTE tables reside in HASPTABS
  - See *MVS/Extended Architecture SPL: JES2 User Modifications and Macros*(LC23-0069)

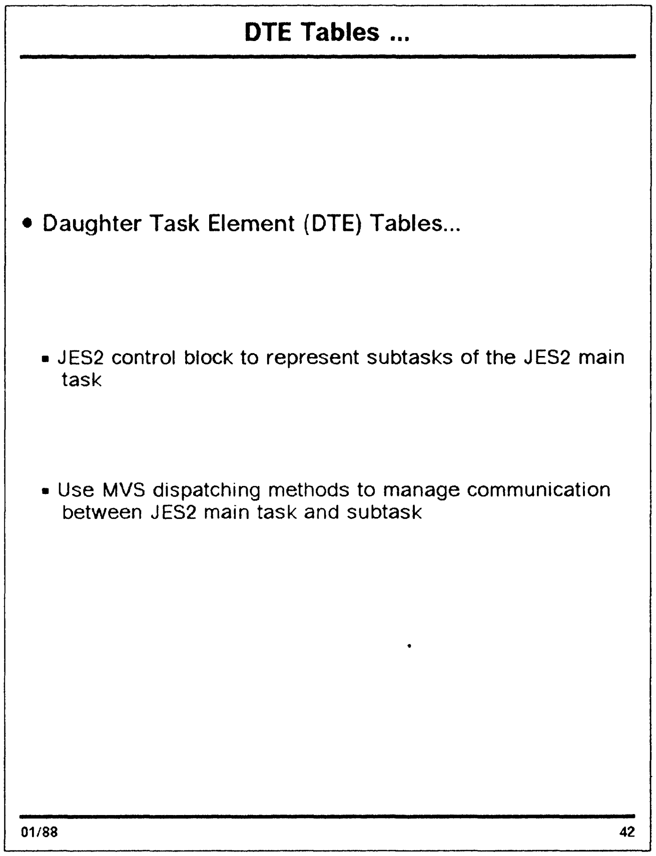
01/88 41

Daughter Task Elements (DTEs) are JES2 control blocks that represent subtasks in JES2. As PCEs represent JES2 processors in the main task environment, DTEs represent JES2 subtasks in the subtask environment.[7] Subtasks are used in JES2 to do work that may require MVS WAITs. MVS WAITs are not tolerated in the JES2 main task, so therefore, subtasks are obtained by JES2 to do this type of work.

As PCEs are tabular via the $PCETABs, the DTEs are tabular via the $DTETABs. This provides you the capability to add installation-defined subtasks and to override JES2-defined subtasks. It is not recommended that you delete JES2-defined subtasks.

The tables that define the JES2 subtasks reside in the module HASPTABS. Some of the following information on DTEs can be found in *SPL: JES2 User Modifications and Macros.*

[7] The term 'DTE' refers either to a JES2 subtask or to the control block which represents the subtask. Where the distinction is important we have tried to add terms like 'subtask' or 'control block' to the term 'DTE' where it occurs.



DTE Tables ...

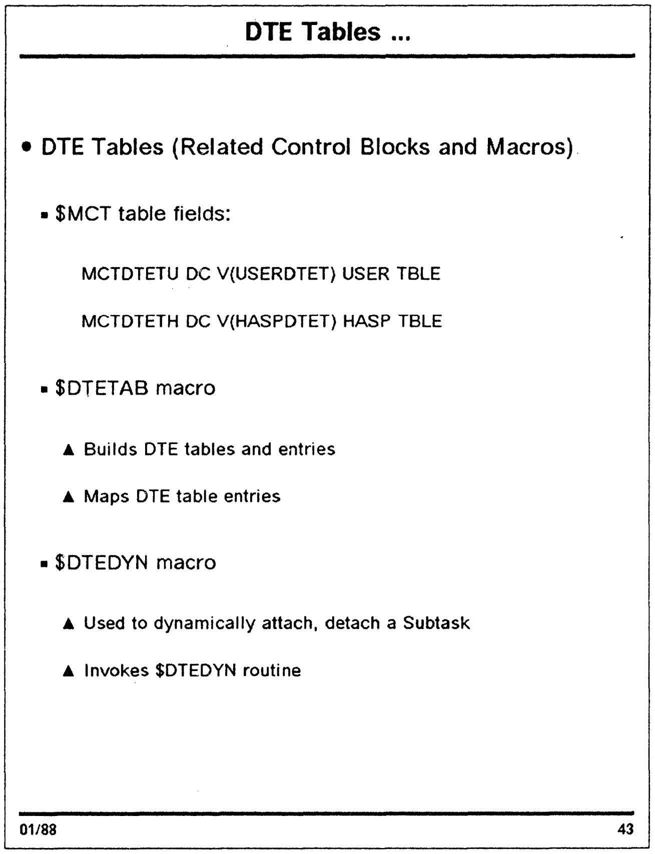
- Daughter Task Element (DTE) Tables...
  - JES2 control block to represent subtasks of the JES2 main task
  - Use MVS dispatching methods to manage communication between JES2 main task and subtask

01/88 42

The DTE is the control block that represents the subtask. This control block is available to the Main Task (a PCE processor) and the subtask. Thus, this control block assists communication between the two environments.

In order to serialize the communications between the main task and the subtask, normal MVS dispatching methods should be followed. This involves the use of $WAITs and MVS POSTs from the main task and MVS WAITs and POSTs from the subtask. Never issue an MVS WAIT from the JES2 main task and never issue a JES2 $WAIT from a JES2 subtask.

DTE Control Blocks and Macros



DTE Tables ...

- DTE Tables (Related Control Blocks and Macros)
  - $MCT table fields:

    ```
    MCTDTETU DC V(USERDTET) USER TBLE
    MCTDTETH DC V(HASPDTET) HASP TBLE
    ```

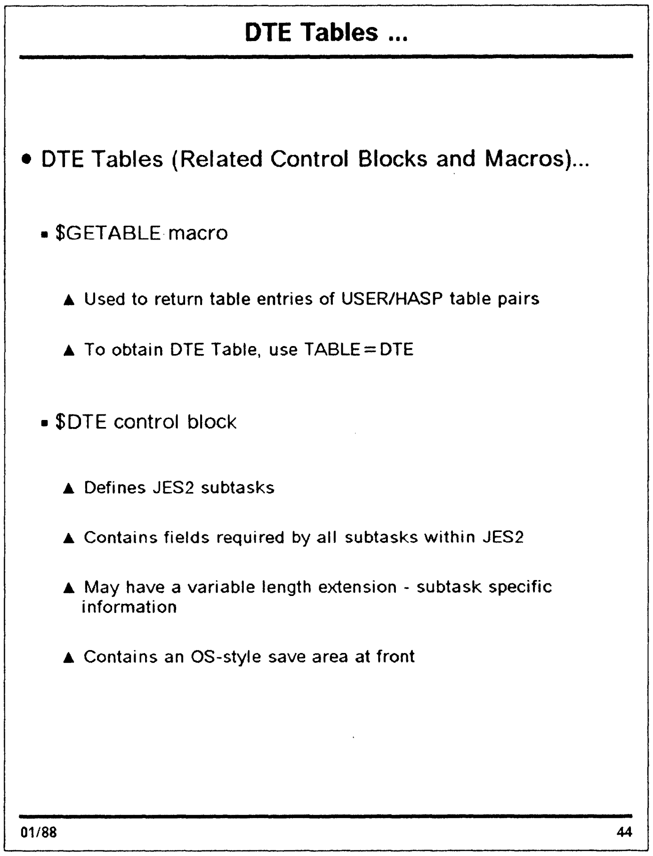
  - $DTETAB macro
    - ▲ Builds DTE tables and entries
    - ▲ Maps DTE table entries
  - $DTEDYN macro
    - ▲ Used to dynamically attach, detach a Subtask
    - ▲ Invokes $DTEDYN routine

01/88 43

The table pair which points to the DTE tables is located in the $MCT. The field MCTDTETU will contain the address of the installation table, if such a table exists. If you want to link-edit your table with JES2 you must name the table USERDTET and link-edit it with HASJES20. The JES2-defined DTE table is pointed to from the $MCT field MCTDTETH and is named HASPDTET.

To aid in creating the DTE tables, JES2 supplies a macro named $DTETAB. This macro builds both the JES2 and installation tables and table elements. This macro also contains the mapping macro for the DTE table and element. We will describe this macro and its operands more thoroughly below.

JES2 provides a mechanism to dynamically attach and detach subtasks via the $DTEDYN service. This service is invoked with the use of the $DTEDYN macro. The service routine makes use of the DTE tables for attaching and detaching of subtasks (DTEs).



DTE Tables ...

- DTE Tables (Related Control Blocks and Macros)...
  - $GETABLE macro
    - ▲ Used to return table entries of USER/HASP table pairs
    - ▲ To obtain DTE Table, use TABLE=DTE
  - $DTE control block
    - ▲ Defines JES2 subtasks
    - ▲ Contains fields required by all subtasks within JES2
    - ▲ May have a variable length extension - subtask specific information
    - ▲ Contains an OS-style save area at front

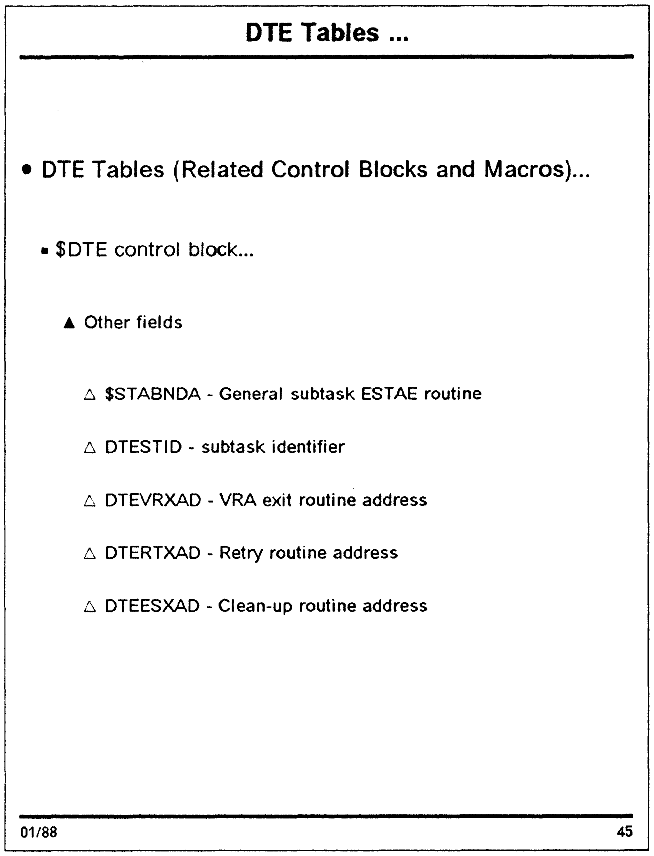
01/88 44

The $GETABLE macro invokes the $GETABLE service routine that is located in the module HASPTABS. This service obtains a table element from the user or JES2 table. To obtain a DTE table, you would code the TABLE=DTE operand. This macro will return the table element of the specified ID or, if LOOP is specified, it will return the next table element after the specified ID.

The major control block for adding and modifying of subtasks is the DTE (Daughter Task Element). DTEs represent and define JES2 subtasks. This control block contains fields that are required on a subtask basis within the JES2 subtasks. The DTE is composed of a common section (all JES2 subtask DTEs contain this common section) and an optional variable length section that is unique between subtask types and contain subtask-specific information. The subtask names are:

- HASPIMAG
- HOSALLOC
- HOSPOOL
- HASPACCT
- HASPVTAM
- HASPWTO
- HOSCNVT
- HASPOFF
- HASPCKAP (added in 2.2.0)

The common section includes an OS-style save area at the front. This is pointed to by R13 in the JES2 subtask (i.e., it points to the DTE with the OS-style save area in the front) which is an available save area.



DTE Tables ...

- DTE Tables (Related Control Blocks and Macros)...
  - $DTE control block...
    - ▲ Other fields
      - △ $STABNDA - General subtask ESTAE routine
      - △ DTESTID - subtask identifier
      - △ DTEVRXAD - VRA exit routine address
      - △ DTERTXAD - Retry routine address
      - △ DTEESXAD - Clean-up routine address

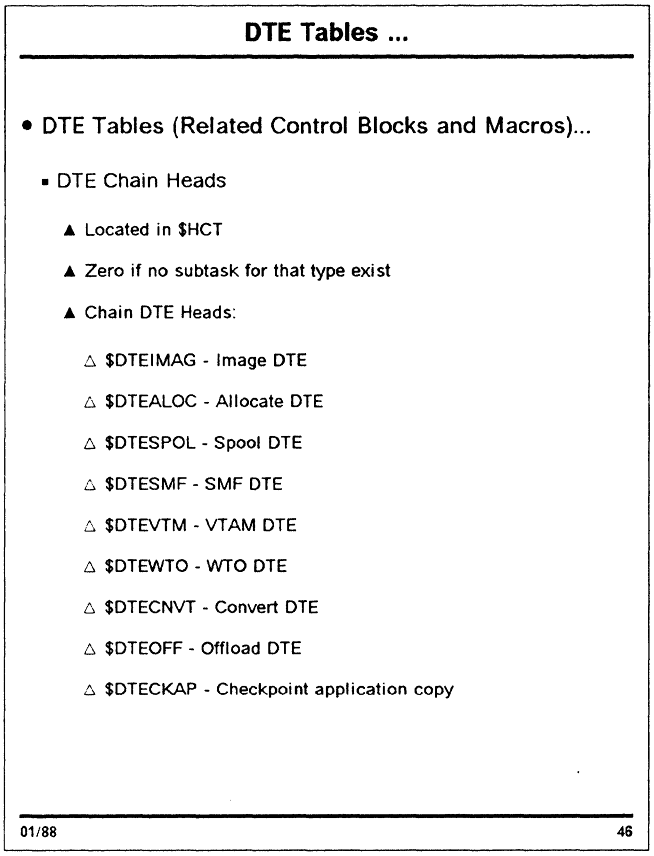
01/88 45

There are four fields that are used for subtask recovery. These fields are:

- $STABNDA - this is a field in the $HCT that contains the address of the general subtask recovery routine. If you code an ESTAE (highly recommended) you should use this routine as the recovery routine. This general recovery routine will take three "exit" calls, depending upon whether the following three fields are non-zero.
- DTEVRXAD - this is a field in the DTE that contains the address of a VRA "exit" routine. This routine will receive control from the JES2 general subtask recovery routine to complete the variable recording area (VRA) in the SDWA. In this way, the data that is specific to this subtask can be saved.
- DTERTXAD - this is a field in the DTE that contains the address of a retry routine. This routine will receive control to attempt to retry. The general JES2 recovery routine issues a SETRP to a general retry routine. This general retry routine will then give control to the specified retry routine for this subtask. The subtask retry routine should issue a $SETRP to a resumption point or percolate. If the subtask is to retry or percolate, the retry routine should prepare for the event.
- DTESXAD - this is another field in the DTE that contains the address of a clean-up routine. This routine will receive control from the JES2 general subtask recovery routine to attempt subtask specific clean-up. There are two valid return codes from this recovery routine.

  1. 0 - continue normal recovery, clean-up successful
  2. 4 - unrecoverable subtask error, abend JES2 main task via a CALLRTM

Also included in the DTE is the field DTESTID which contains the subtask identifier. We will present more information on this identifier shortly.



DTE Tables ...

- DTE Tables (Related Control Blocks and Macros)...
  - DTE Chain Heads
    - ▲ Located in $HCT
    - ▲ Zero if no subtask for that type exist
    - ▲ Chain DTE Heads:
      - △ $DTEIMAG - Image DTE
      - △ $DTEALOC - Allocate DTE
      - △ $DTESPOL - Spool DTE
      - △ $DTESMF - SMF DTE
      - △ $DTEVTM - VTAM DTE
      - △ $DTEWTO - WTO DTE
      - △ $DTECNVT - Convert DTE
      - △ $DTEOFF - Offload DTE
      - △ $DTECKAP - Checkpoint application copy

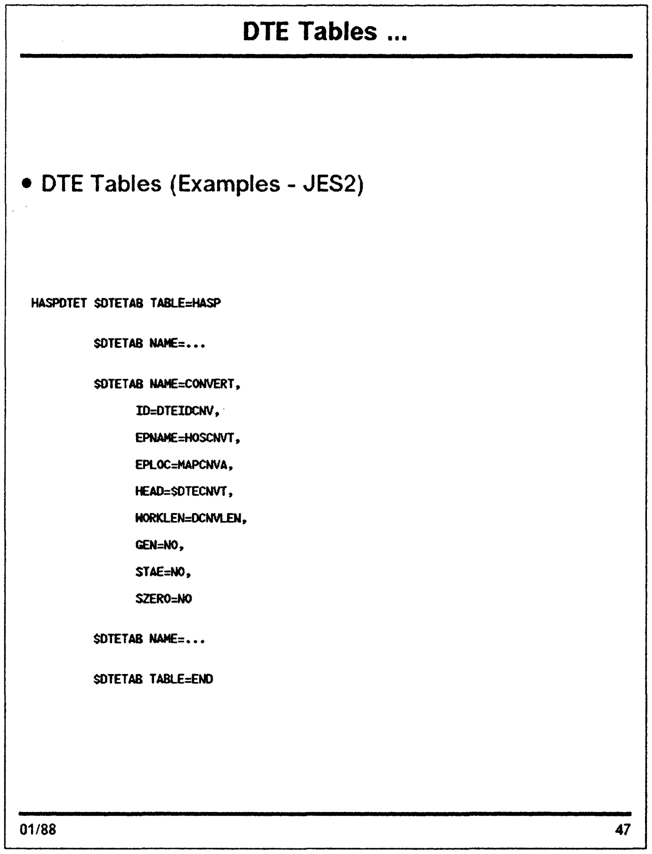
01/88 46

Just as there were pointers in the $HCT for the JES2 processors (PCEs) for each type of processor, there are pointers in the $HCT for the JES2 subtasks (DTEs) for each type of subtask. If the chain head is zero, then no subtasks of that type exists. The chain heads are:

- $DTEIMAG - points to the image subtask(s)
- $DTEALOC - points to the allocation subtask
- $DTESPOL - points to the spool subtask(s)
- $DTESMF - points to the SMF subtask
- $DTEVTM - points to the VTAM subtask
- $DTEWTO - points to the WTO subtask
- $DTECNVT - points to the converter subtask(s)
- $DTEOFF - points to the offload subtask(s)
- $DTECKAP - points to the checkpoint application copy subtask (new in 2.2.0)

We will describe the method for pointing to the installation subtask in the following foils.

A JES2 DTE Table



DTE Tables ...

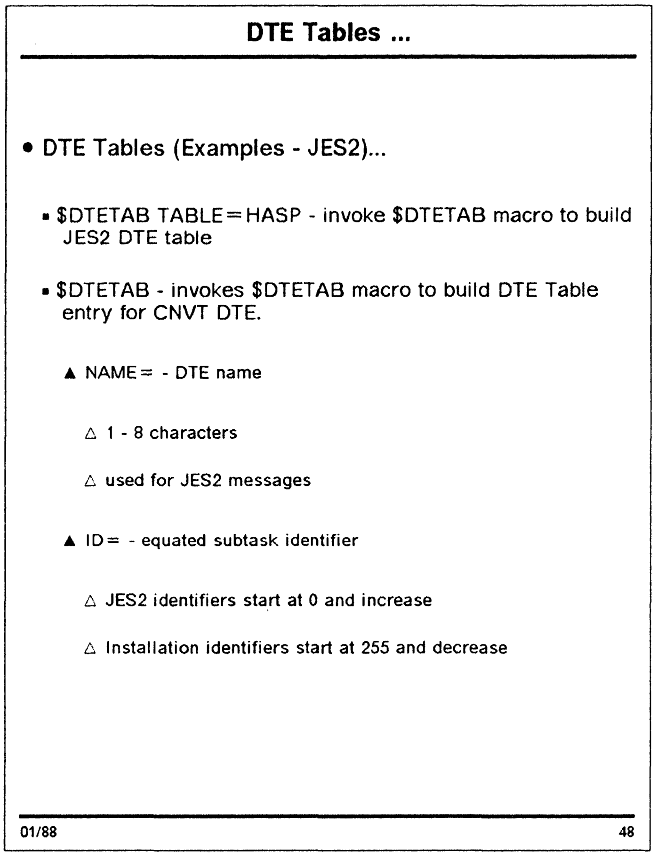
- DTE Tables (Examples - JES2)

```
HASPDTET $DTETAB TABLE=HASP

         $DTETAB NAME=...

         $DTETAB NAME=CONVERT,
               ID=DTEIDCNV,
               EPNAME=HOSCNVT,
               EPLOC=MAPCNVA,
               HEAD=$DTECNVT,
               WORKLEN=DCNVLEN,
               GEN=NO,
               STAE=NO,
               SZERO=NO

         $DTETAB NAME=...

         $DTETAB TABLE=END
```

01/88 47

The figure above is an example of what the JES2 subtask (DTE) table looks like. The table is delimited by a TABLE = HASP (to start the table) and a TABLE = END (to end the table). The table element shown represents all the information that JES2 needs to define a JES2 converter subtask. This is the table element that is passed to the $DTEDYN service to create the converter DTE. Notice that the name of the DTE table is HASPDTET, the same as that specified in the V-type address constant in the $MCT.

The following discussion will describe each operand on the $DTETAB macro and how it should be specified.



DTE Tables ...

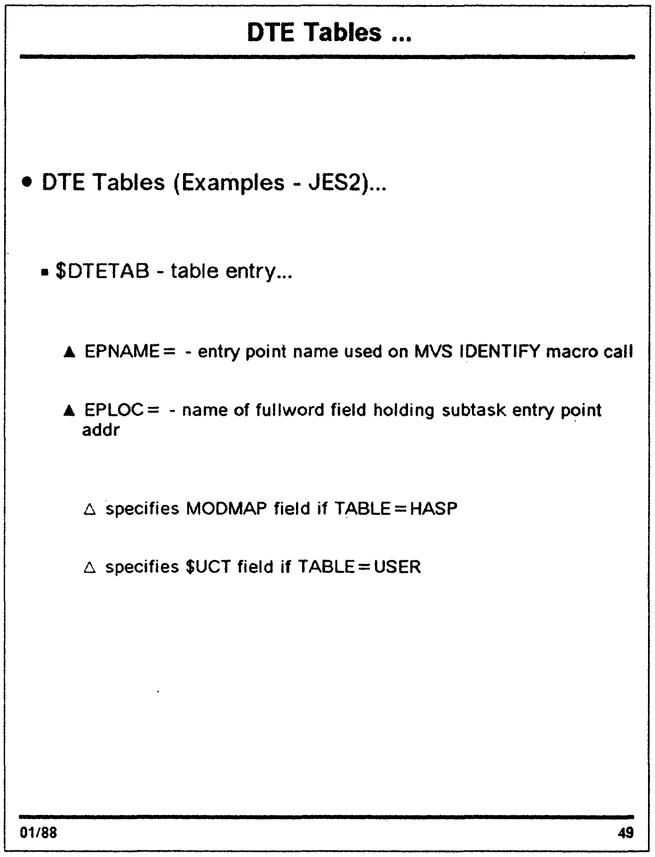
- DTE Tables (Examples - JES2)...
  - $DTETAB TABLE=HASP - invoke $DTETAB macro to build JES2 DTE table
  - $DTETAB - invokes $DTETAB macro to build DTE Table entry for CNVT DTE.
    - ▲ NAME= - DTE name
      - △ 1 - 8 characters
      - △ used for JES2 messages
    - ▲ ID= - equated subtask identifier
      - △ JES2 identifiers start at 0 and increase
      - △ Installation identifiers start at 255 and decrease

01/88 48

The JES2 tables are started by specifying 'TABLE=HASP'. This indicates to JES2 that this table is a JES2 table. You would specify 'TABLE=USER' to indicate that the table is an installation coded table. Whether it is a JES2 or installation table determines some default values for the EPLOC and HEAD $DTETAB operands. We will discuss these operands later. Specifying TABLE=HASP or TABLE=USER is the means JES2 provides to indicate the start of the table (TABLE=START) as discussed in "Concepts" on page 6.

When you specify $DTETAB with operands other than TABLE=, the macro generates a table element. In the example above, the table element that is generated is for the converter subtask.

The NAME operand specifies a one- to eight-character name. JES2 messages use this name. The ID operand specifies an equated numeric value. JES2 identifiers start at 0 and increase. Installation identifiers start at 255 and decrease. There are currently nine JES2 subtask types.

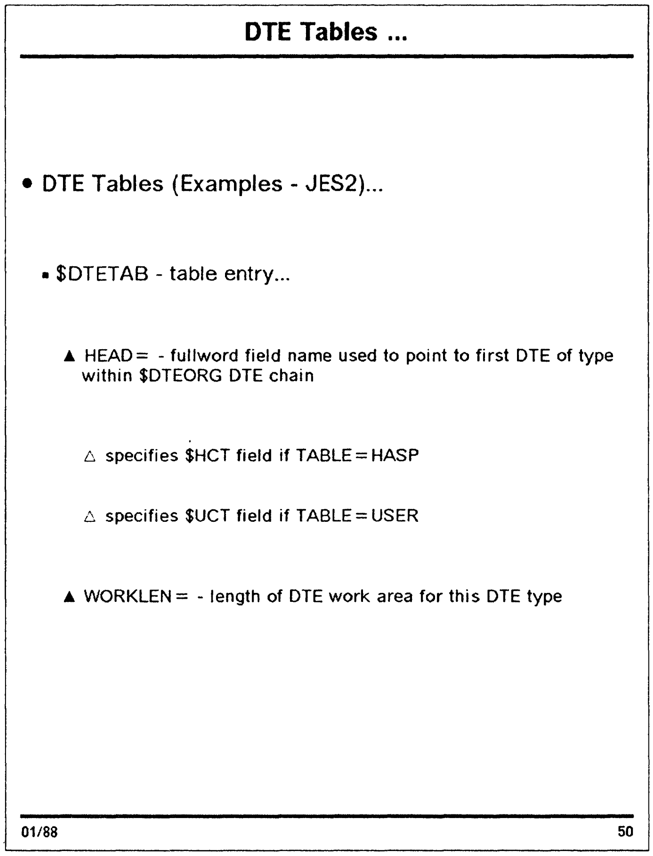


DTE Tables ...

- DTE Tables (Examples - JES2)...
  - ▪ $DTETAB - table entry...
    - ▲ EPNAME= - entry point name used on MVS IDENTIFY macro call
    - ▲ EPLOC= - name of fullword field holding subtask entry point addr
      - △ specifies MODMAP field if TABLE=HASP
      - △ specifies $UCT field if TABLE=USER

01/88 49

The EPNAME operand specifies the name of the subtask entry point. The MVS IDENTIFY macro uses this. The EPLOC operand points to the fullword field that holds the subtask entry point address. If the DTE table is a JES2 table, the default location for this entry point address is in MODMAP. If the DTE table is an installation table (i.e., TABLE = USER), the default location for this entry point address is in the $UCT. This default can be overridden by specifying 'EPNAME = (field,MODMAP)'. The field must be in either MODMAP or the $UCT.



DTE Tables ...

- DTE Tables (Examples - JES2)...

  - $DTETAB - table entry...

    ▲ HEAD= - fullword field name used to point to first DTE of type within $DTEORG DTE chain

    △ specifies $HCT field if TABLE=HASP

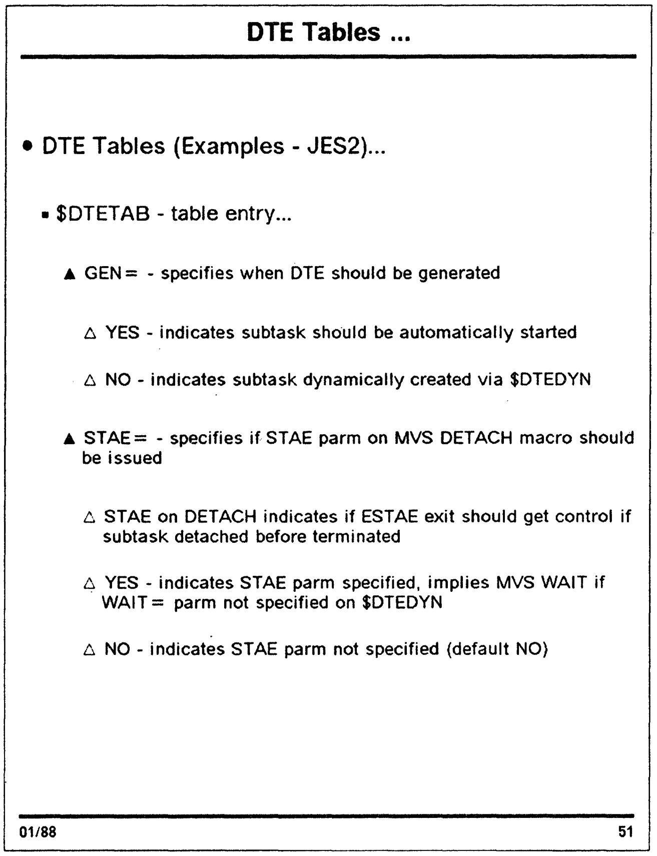
    △ specifies $UCT field if TABLE=USER

    ▲ WORKLEN= - length of DTE work area for this DTE type

01/88 50

The HEAD operand is similar to the CHAIN operand on the $PCETAB. This operand points to the fullword field used to point to the first DTE of this type. JES2 will chain the initial DTE of this type from the specified field. All DTEs can be run by starting at the $DTEORG in the $HCT. In the example the first converter subtask can be found by obtaining the address in the $DTECNVT field in the $HCT. DTEPREV and DTENEXT are fields used to chain to the next DTE. If you specify this field, it is defaulted to be in the $UCT. If you wish to override this default, you would specify 'HEAD=(field,HCT)'. The field must be in either the $HCT or the $UCT.

Just as there was a variable extension area off of PCEs, there are variable extension areas off of DTEs. The size of these extension areas can be variable between subtask types. Therefore, you specify the size for these extension areas in the $DTETAB via the WORKLEN operand. The $DTEDYN service uses this value to $GETMAIN the DTE and its extension in contiguous storage. In the example, WORKLEN=DCNVLEN, DCNVLEN is the equate for the length of the variable extension area for the converter subtask.



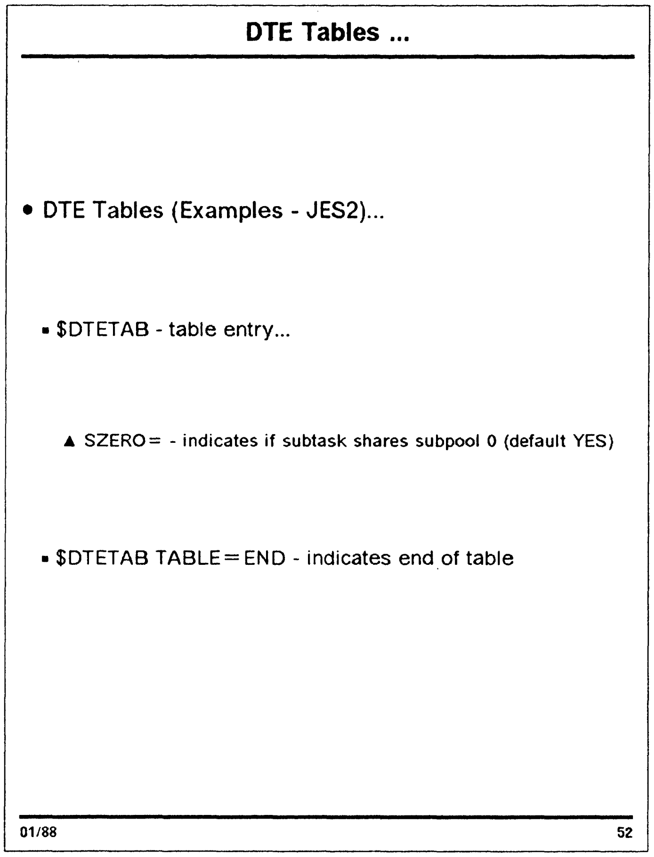
DTE Tables ...

- DTE Tables (Examples - JES2)...
  - $DTETAB - table entry...
    - ▲ GEN= - specifies when DTE should be generated
      - △ YES - indicates subtask should be automatically started
      - △ NO - indicates subtask dynamically created via $DTEDYN
    - ▲ STAE= - specifies if STAE parm on MVS DETACH macro should be issued
      - △ STAE on DETACH indicates if ESTAE exit should get control if subtask detached before terminated
      - △ YES - indicates STAE parm specified, implies MVS WAIT if WAIT= parm not specified on $DTEDYN
      - △ NO - indicates STAE parm not specified (default NO)

01/88 51

The GEN operand specifies when the subtask should be generated. GEN = YES indicates that the subtask should be automatically started. GEN = NO indicates that the subtask is dynamically created via a $DTEDYN call. In the example, you specify GEN = NO since the converter subtask is dynamically created through a $DTEDYN call by the converter processor when the subtask is needed and it is not attached.

The STAE operand indicates whether the STAE parameter should be specified on the MVS DETACH macro. If it is (i.e., STAE = YES), then the ESTAE exit will get control if the subtask detaches before it is terminated. If you specify STAE = YES this implies an MVS WAIT if WAIT = parameter is not specified on the $DTEDYN macro which creates the subtask. If you specify STAE = NO, then the STAE parameter will not be generated on the MVS DETACH macro. STAE = NO is the default.



DTE Tables ...

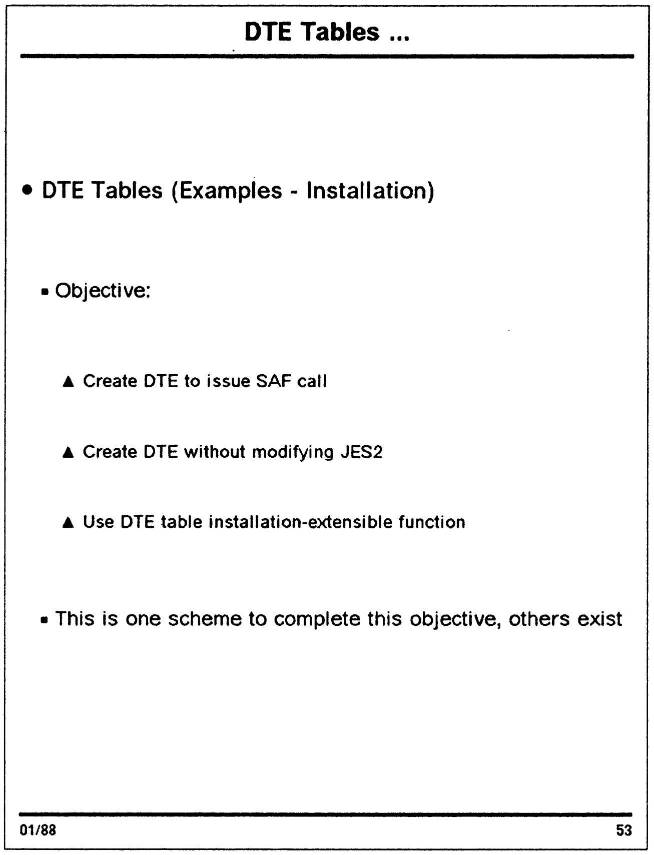
- DTE Tables (Examples - JES2)...
  - $DTETAB - table entry...
    - ▲ SZERO= - indicates if subtask shares subpool 0 (default YES)
  - $DTETAB TABLE=END - indicates end of table

01/88 52

The final operand on the $DTETAB macro is the SZERO operand. This operand tells JES2 whether this subtask should share subpool 0. The default is SZERO=YES. In the example, SZERO=NO was specified to say that the converter subtask cannot share subpool 0.

When the TABLE=END is encountered, the table is closed. This indicates the end of the JES2 DTE table. All JES2-defined subtasks are defined within this single table.

An Installation DTE Table



DTE Tables ...

- DTE Tables (Examples - Installation)
  - Objective:
    - ▲ Create DTE to issue SAF call
    - ▲ Create DTE without modifying JES2
    - ▲ Use DTE table installation-extensible function
  - This is one scheme to complete this objective, others exist

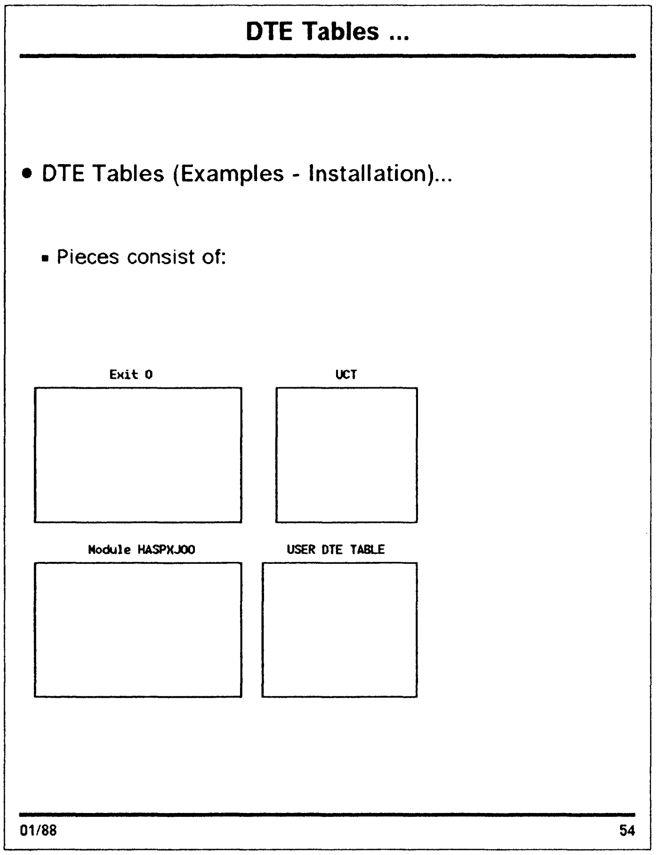
01/88 53

In order to show how you would specify an installation DTE table, we will step through the creation of an installation-defined security subtask. The security DTE is required to fulfill our security objective since the SAF call can result in an MVS WAIT.

Objective

The objective is to create a DTE to issue the SAF call on behalf of a security processor (or, as it turns out, any processor). The installation wishes to achieve this without modifying JES2 and to use the DTE tables as the means to define this DTE to JES2.

Required Pieces



DTE Tables ...

- DTE Tables (Examples - Installation)...
  - Pieces consist of:

01/88 54

To achieve the objective, you will need to code four pieces. These pieces are:

1. Exit 0

   As was discussed in "Concepts" on page 6, there are two ways to link the installation table with JES2:

   a. The first of these is to link-edit the installation DTE table with the HASJES20 load module. This requires the name of the installation table be USERDTET.

   b. If you do not wish to link-edit an installation DTE table with HASJES20 or does not wish to name the installation table USERDTET, then you must fill in the address of the installation DTE table into the $MCT at field MCTDETTU. This is the second method. This method requires that you fill in the address prior to the need to invoke the $DTEDYN service routine to create the subtask. Depending on when you will have the processor generated, you may fill in the address early in initialization or after JES2 is up and running.

   In this example, you will fill in the address of the DTE table early in initialization, specifically in Exit 0. Therefore, you require an Exit 0 that will load the module (if not already loaded) and resolve the address of the table.

2. UCT

   As has been indicated when examining the JES2 DTE table, there are certain operands that assume $UCT fields in the installation DTE table elements. Of course, you may override these assumptions, but the objective of this effort was to use the tables and not modify JES2. Therefore, a $UCT must be created that will hold certain values.

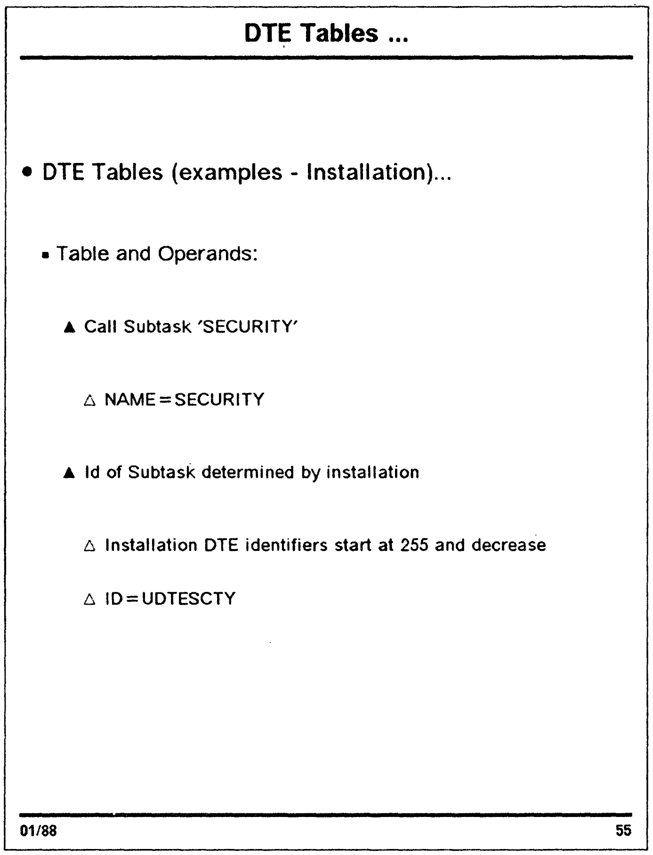
3. Module HASPXJ00

   Since you are coding a new JES2 subtask, you must write the code that is the subtask. In this example, the code resides in the module HASPXJ00. This name is one of the reserved-for-installation-use names that JES2 has in the $MODMAP control block. In this way, you can link-edit this module with HASJES20 and have its address in $MODMAP and not have to do the load of this exit from Exit 0.

4. User DTE Table

   You will have to code your own DTE table to include an element for the particular subtask. We will describe how to create this installation table element in the following section.

Coding the Installation DTE Table



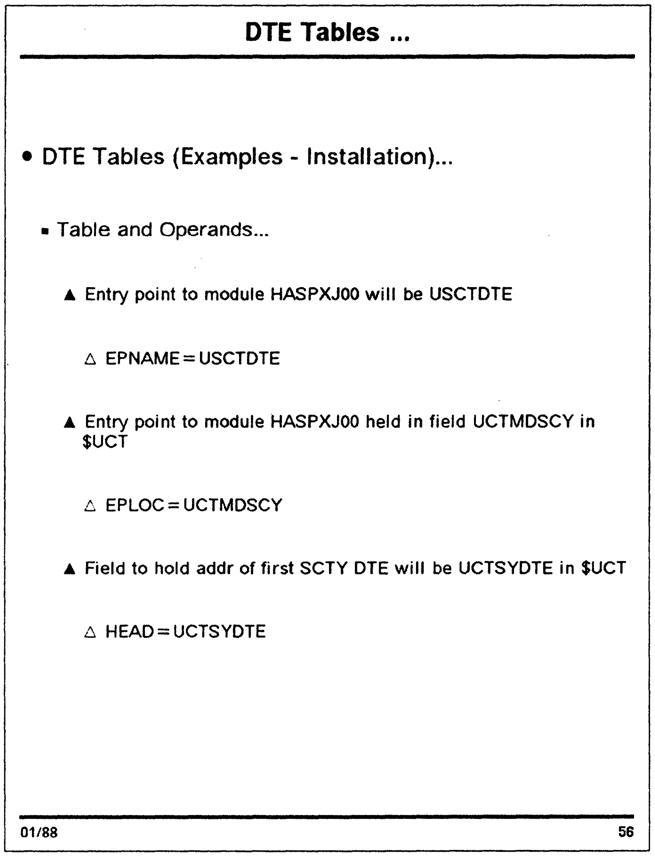
DTE Tables ...

- DTE Tables (examples - Installation)...
  - Table and Operands:
    - ▲ Call Subtask 'SECURITY'
      - △ NAME = SECURITY
    - ▲ Id of Subtask determined by installation
      - △ Installation DTE identifiers start at 255 and decrease
      - △ ID = UDTESCTY

01/88 55

In the figure above, you wanted to create a security subtask which would be called SECURITY. Therefore, NAME = SECURITY is specified to have the subtask called SECURITY in JES2 messages.

The identifier of the processor is 255. This is because installation specified identifiers should start at 255 and decrease since JES2 subtask identifiers start at 0 and work their way up. There is an equate specified in the $UCT named UDTESCTY set to 255. Therefore, we specify the operand for the identifier as 'ID = UDTESCTY'.

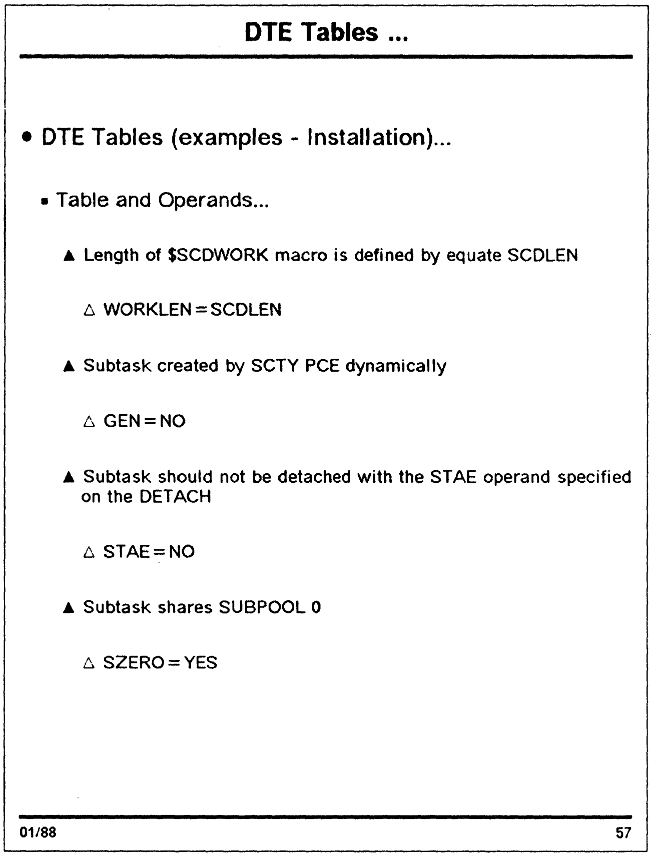


DTE Tables ...

- DTE Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Entry point to module HASPXJ00 will be USCTDTE
      - △ EPNAME = USCTDTE
    - ▲ Entry point to module HASPXJ00 held in field UCTMDSCY in $UCT
      - △ EPLOC = UCTMDSCY
    - ▲ Field to hold addr of first SCTY DTE will be UCTSYDTE in $UCT
      - △ HEAD = UCTSYDTE

01/88 56

The name of the entry point to the subtask code in module HASPXJ00 is USCTDTE. Therefore, we code 'EPNAME= USCTDTE' on the $DTETAB to tell JES2 to use USCTDTE on the MVS IDENTIFY call. The field to hold the entry point address is in the $UCT. The name of the field is UCTMDSCY. It will hold the address of the routine USCTDTE. Therefore, we code 'EPLOC= UCTMDSCY' on the $DTETAB. The $UCT field to hold the pointer to the first security subtask is UCTSYDTE. Thus, we code "HEAD= UCTSYDTE" to tell JES2 the name of the chain field. Since the table element is in the installation table, the specified field will default to being in the $UCT.



DTE Tables ...

- DTE Tables (examples - Installation)...
  - Table and Operands...
    - ▲ Length of $SCDWORK macro is defined by equate SCDLEN
      - △ WORKLEN = SCDLEN
    - ▲ Subtask created by SCTY PCE dynamically
      - △ GEN = NO
    - ▲ Subtask should not be detached with the STAE operand specified on the DETACH
      - △ STAE = NO
    - ▲ Subtask shares SUBPOOL 0
      - △ SZERO = YES

01/88 57

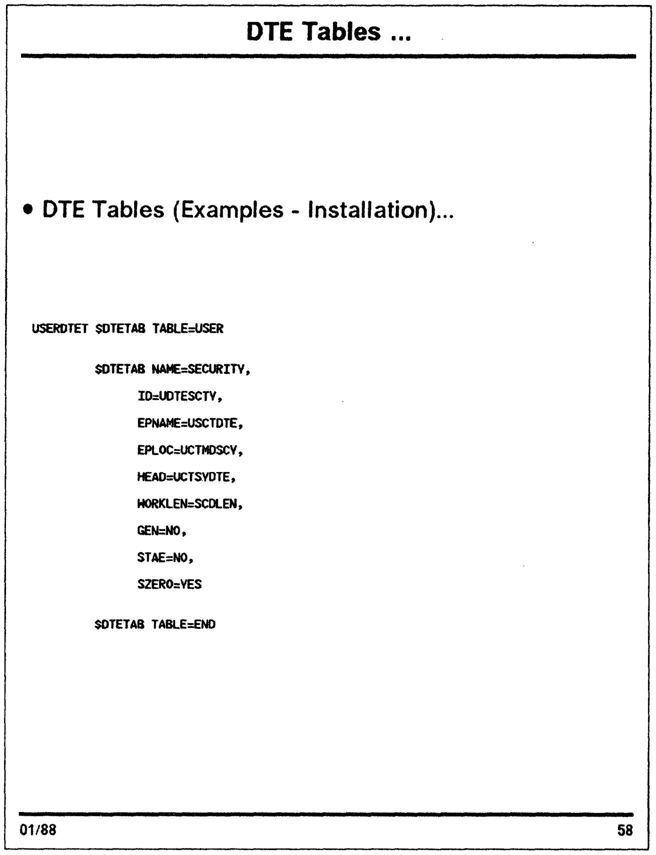
The length of the variable extension area of the security subtask is defined via an equate called SCDLEN in macro $SCDWORK. This is the value that we specify in the table: WORKLEN = SCDLEN.

You have also decided that the processor should not be generated automatically. Therefore, we specify 'GEN = NO' on the $DTETAB macro to indicate that the subtask is created dynamically via a call to $DTEDYN.

Also, you do not wish the subtask to be detached with the STAE operand specified on the MVS DETACH call. Thus, we specify 'STAE = NO'.

You would like the subtask to share subpool 0, so you specify 'SZERO = YES'.

Resulting DTE Table



DTE Tables ...

- DTE Tables (Examples - Installation)...

```
USERDTET $DTETAB TABLE=USER

         $DTETAB NAME=SECURITY,
               ID=UDTESCTY,
               EPNAME=USCTDTE,
               EPLOC=UCTMDSCV,
               HEAD=UCTSYDTE,
               WORKLEN=SCDLEN,
               GEN=NO,
               STAE=NO,
               SZERO=YES

         $DTETAB TABLE=END
```

01/88 58

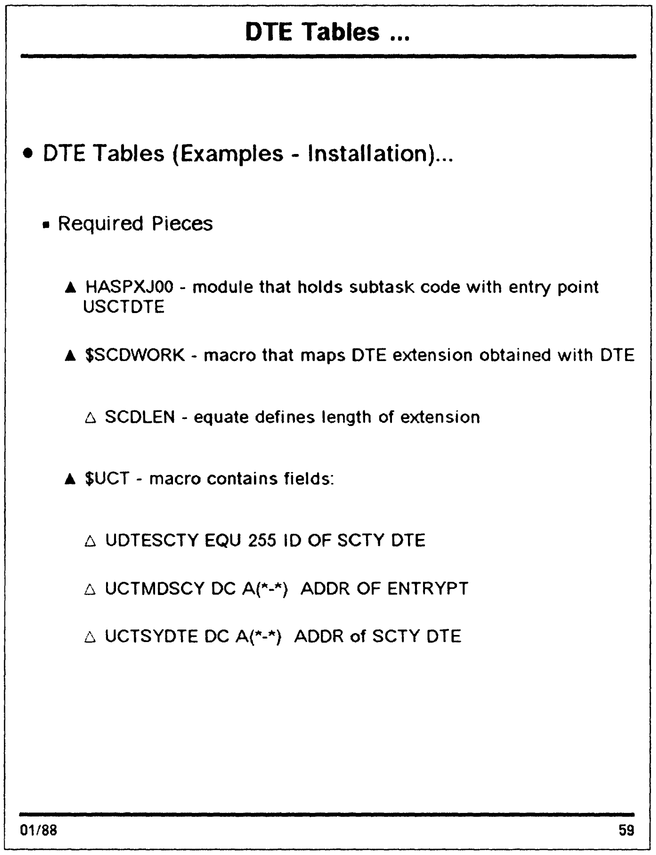
The figure above shows the resulting installation DTE table to add a security subtask to JES2. The table is begun with a TABLE = USER to tell JES2 that this is an installation table. The name of the subtask is SECURITY. The identifier of the subtask is 255, as defined by the equate UDTESCTY. The code for the subtask resides in the module HASPXJ00 and has the entry point name USCTDTE. Its entry point address is in field UCTMDSCY in the $UCT.

The first security subtask is chained from the $UCT field UCTSYDTE.

The security subtask will require its own variable extension area to the DTE that is mapped by the macro $SCDWORK with the length of SCDLEN. The subtask is generated through a specific $DTEDYN, so it will not be generated during initialization. The subtask will not be detached with the STAE option and the subtask may share subpool 0.

The table is delineated by the TABLE = END operand of the $DTETAB macro.

Coding the Other Required Pieces



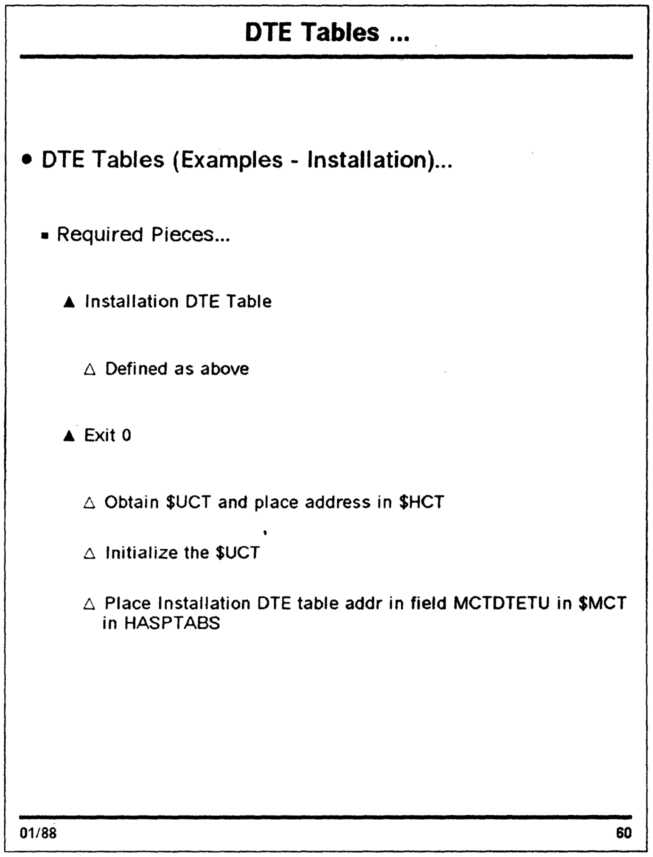
DTE Tables ...

- DTE Tables (Examples - Installation)...
  - Required Pieces
    - ▲ HASPXJ00 - module that holds subtask code with entry point USCTDTE
    - ▲ $SCDWORK - macro that maps DTE extension obtained with DTE
      - △ SCDLEN - equate defines length of extension
    - ▲ $UCT - macro contains fields:
      - △ UDTESCTY EQU 255 ID OF SCTY DTE
      - △ UCTMDSCY DC A(*-*) ADDR OF ENTRYPT
      - △ UCTSYDTE DC A(*-*) ADDR of SCTY DTE

01/88 59

Now that you have completed the installation DTE table, the other required pieces may be defined. You will have to write a HASPXJ00 module that holds the DTE subtask code. A macro must be created called $SCDWORK that will map the DTE extension. An equate named SCDLEN is required within the macro to define the length of the extension area needed.

In the installation-defined $UCT, two fields must be coded. The address of the entry point for the HASPXJ00 module for the installation DTE is in the UCTMDSCY field. The address of the first security DTE is chained from the UCTSYDTE field. Finally, we must set an equate for the identifier of the subtask. We specify the equate UDTESCTY with a value of 255.



DTE Tables ...

- DTE Tables (Examples - Installation)...
  - Required Pieces...
    - ▲ Installation DTE Table
      - △ Defined as above
    - ▲ Exit 0
      - △ Obtain $UCT and place address in $HCT
      - △ Initialize the $UCT
      - △ Place Installation DTE table addr in field MCTDTETU in $MCT in HASPTABS

01/88 60

The last two pieces that are required are the installation DTE table, as coded above, and the code for exit 0. The exit 0 code is required to do three things:

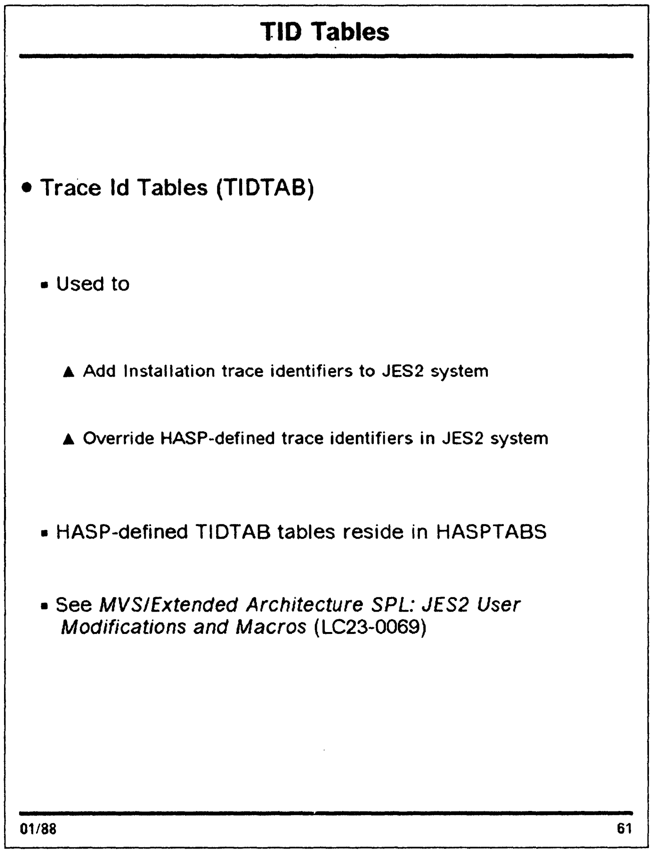
1. It must obtain the $UCT and place the $UCT's address in the $HCT.
2. It must initialize the $UCT.
3. Finally, it must place the installation DTE table address in the MCTDTETU field in the $MCT in module HASPTABS.

The code that is contained in "Appendix A. Table Pairs Coding Example" on page 147 is the code as you would be required to code it. This code includes:

- Exit 0 that obtains the $UCT, places the $UCT address in the $HCT, initializes the $UCT, and places the installation DTE table address in the $MCT.
- The HASPXJ00 module contains, among other items that we will describe shortly, the DTE code and the installation DTE table.
- The macros for the $UCT and $SCDWORK. Also, a $USERCBS macro extends the $MODULE macro so that you can use installation-created macros without modifying the $MODULE macro.

TID Tables

What Is a TID Table?



TID Tables

- Trace Id Tables (TIDTAB)
  - Used to
    - ▲ Add Installation trace identifiers to JES2 system
    - ▲ Override HASP-defined trace identifiers in JES2 system
  - HASP-defined TIDTAB tables reside in HASPTABS
  - See *MVS/Extended Architecture SPL: JES2 User Modifications and Macros* (LC23-0069)

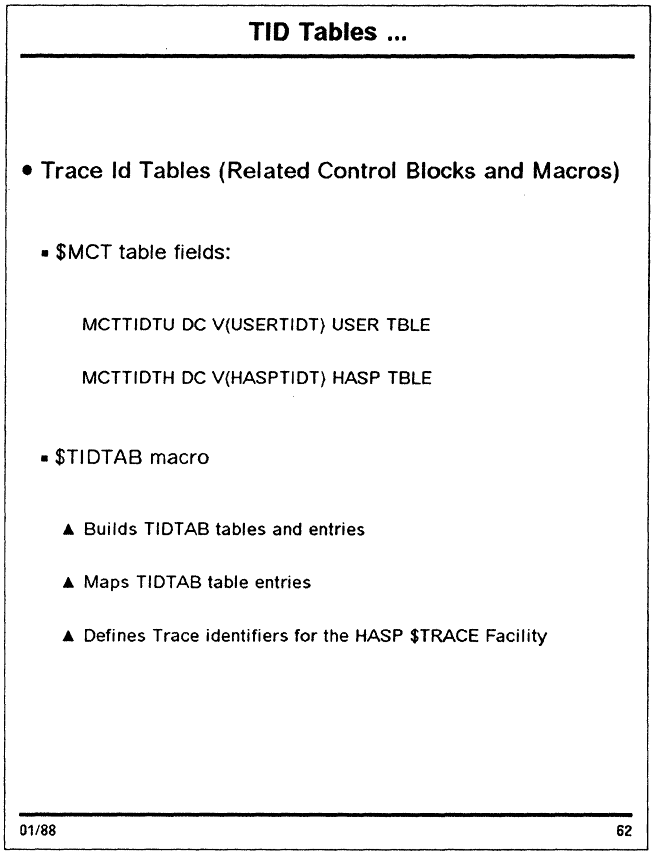
01/88 61

The Trace Id (TID) tables are used to add installation trace identifiers to a JES2 system or to override JES2-defined trace identifiers in JES2. Notice that deleting a JES2 trace identifier is not listed. This is because we recommend you do not delete any JES2 trace identifiers.

The JES2-defined TID tables reside in the JES2 module HASPTABS. Some of the following information can be found in *SPL: JES2 User Modifications and Macros.*

The TID tables are perhaps the simplest and least complete of the JES2 tables. Some of the "interfaces" need additional work. However, the function provided is useful and recommended over alternatives such as in-line modifications to JES2 source code.

TID Control Blocks and Macros



TID Tables ...

- Trace Id Tables (Related Control Blocks and Macros)
  - $MCT table fields:

```
MCTTIDTU DC V(USERTIDT) USER TBLE

MCTTIDTH DC V(HASPTIDT) HASP TBLE
```

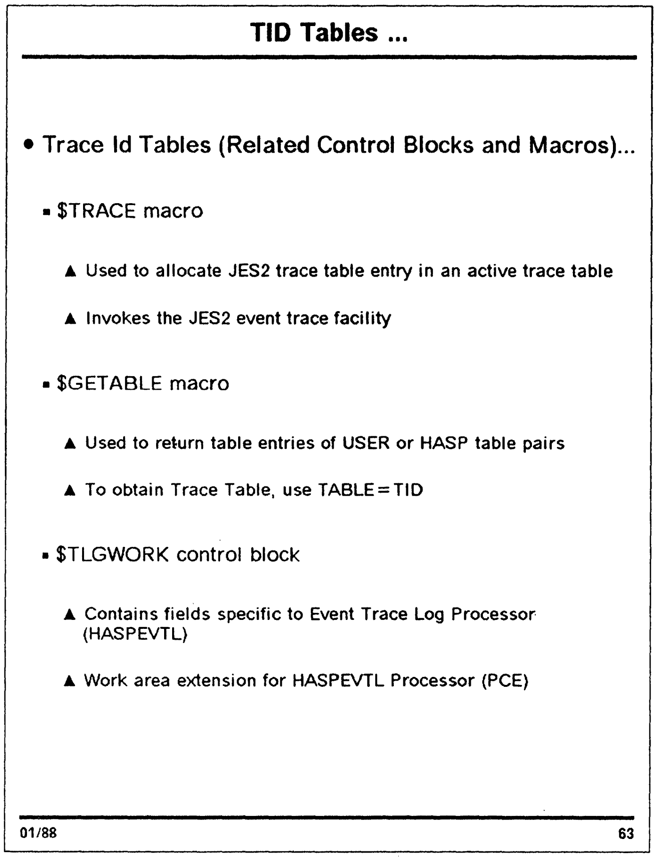
  - $TIDTAB macro
    - ▲ Builds TIDTAB tables and entries
    - ▲ Maps TIDTAB table entries
    - ▲ Defines Trace identifiers for the HASP $TRACE Facility

01/88 62

The table pair which points to the TID tables is located in the $MCT. The field MCTTIDTU will contain the address of the installation table, if such a table exists. If you want to link-edit the table with JES2 you must name the table USERTIDT and link-edit it with HASJES20. The JES2-defined TID table is pointed to from the $MCT field MCTTIDTH and is named HASPTIDT.

To aid in creating TID tables, JES2 supplies a macro named $TIDTAB. This macro builds both the JES2 and installation tables and table elements. This macro also contains the mapping macro for the TID table and element. We will describe this macro and its operands more thoroughly below.

The $TRACE facility is the user of the TID tables. It uses the tables to determine what identifiers are valid and what formatter routines will receive control (see "A JES2 TID Table" on page 74).



TID Tables ...

- Trace Id Tables (Related Control Blocks and Macros)...
  - $TRACE macro
    - ▲ Used to allocate JES2 trace table entry in an active trace table
    - ▲ Invokes the JES2 event trace facility
  - $GETABLE macro
    - ▲ Used to return table entries of USER or HASP table pairs
    - ▲ To obtain Trace Table, use TABLE=TID
  - $TLGWORK control block
    - ▲ Contains fields specific to Event Trace Log Processor (HASPEVTL)
    - ▲ Work area extension for HASPEVTL Processor (PCE)

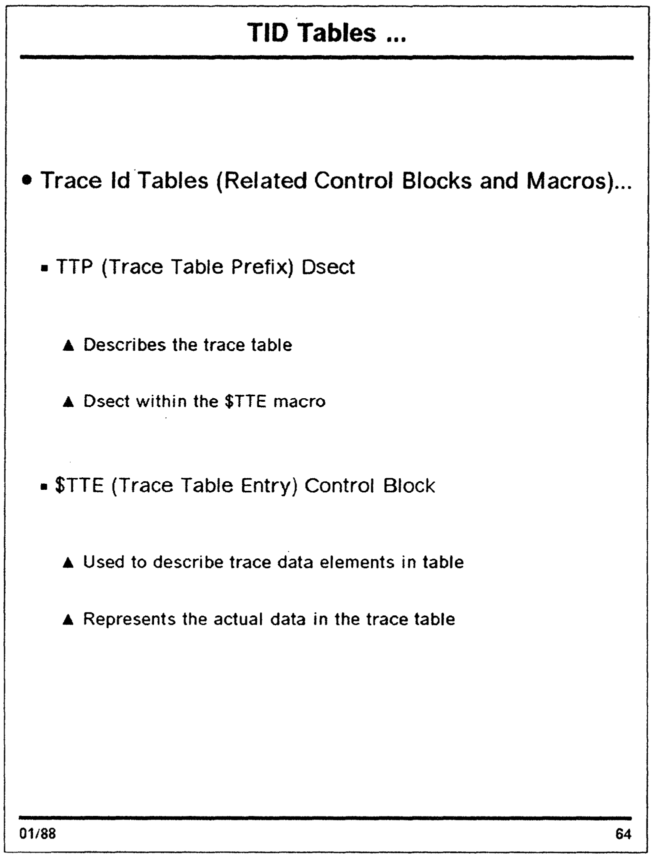
01/88 63

The $TRACE executable macro allocates a JES2 trace table entry in an active trace table and returns its address. Optionally, $TRACE initializes the Trace Table Entry (TTE) based upon parameters passed. The JES2 event trace facility is called to perform the TTE allocation.

$TRACE can be specified anywhere in the JES2 system (including the HASPSSSM load module) except in routines running as disabled interrupt exits (for example, an IOS appendage). R13 must point to a usable OS-style save area. Be certain to also code the $TRP macro on the $MODULE statement to provide the required mapping. Refer to *SPL: Modifications and Macros* for a detailed description on the use of this macro.

As with the PCETABs and DTETABs, access can be obtained to the TIDTABs via the $GETABLE macro. The $GETABLE macro invokes the $GETABLE service routine that is located in the module HASPTABS. This service obtains a table element from the user or JES2 table. To obtain a TID table, you would code the TABLE=TID operand. This macro will return the table element of the specified ID or, if LOOP is specified, it will return the next table element after the specified ID.

You will also need to specify $TLGWORK. This is the macro that maps the Event Trace Log processor variable extension area. This macro is needed because it contains fields that are specific for the processor. They will be needed by the installation format routines (which we will describe later).

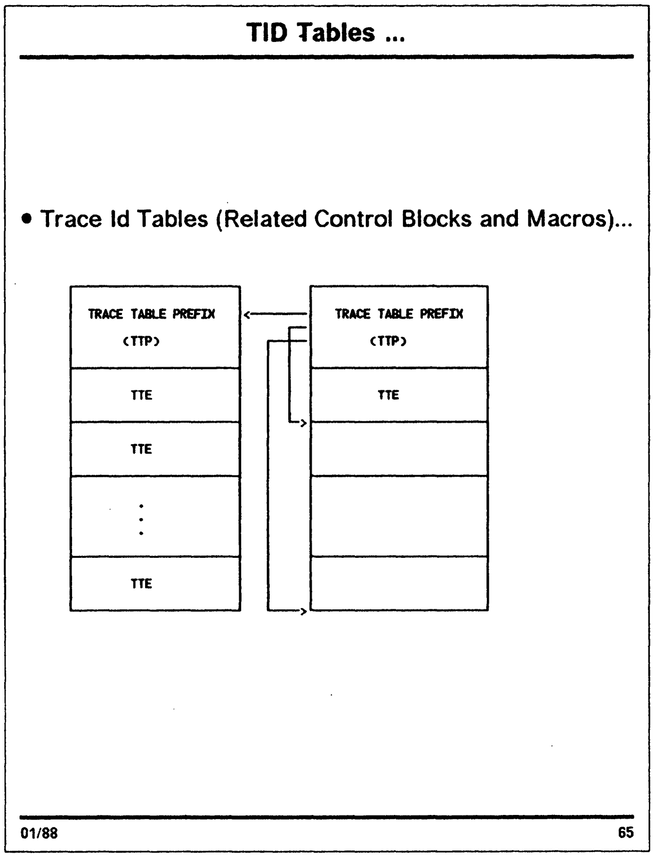


TID Tables ...

- Trace Id Tables (Related Control Blocks and Macros)...
  - TTP (Trace Table Prefix) Dsect
    - ▲ Describes the trace table
    - ▲ Dsect within the $TTE macro
  - $TTE (Trace Table Entry) Control Block
    - ▲ Used to describe trace data elements in table
    - ▲ Represents the actual data in the trace table

01/88 64

Since the trace interface is not well-defined and is rather primitive, it is necessary to understand some of the internal structures of the primary control blocks. These control blocks include the Trace Table Prefix (TTP) and the Trace Table Entry (TTE). The TTP describes the entire trace table while the TTE describes elements within the trace table. The next foil illustrates the TTP and the TTE.



TID Tables ...

- Trace Id Tables (Related Control Blocks and Macros)...

TRACE TABLE PREFIX (TTP)

TTE

TTE

TTE

TRACE TABLE PREFIX (TTP)

TTE

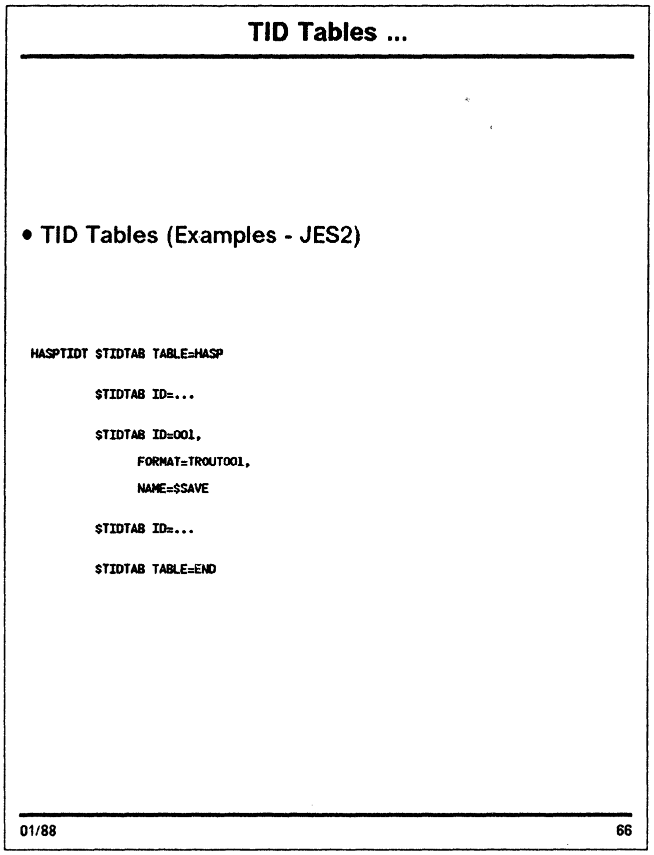
01/88

65

In the illustration above, there are two trace tables. Both contain Trace Table Prefixes. The TTP is made up of basically three pointers. The first pointer points to the previous trace table, the second pointer points to the end of the table, and the final pointer points to the next available spot in the trace table.

Trace tables are made up of as many TTEs (Trace Table Elements) as will fit in the trace table. The TTEs are not of a set size, but are the size as was specified on the $TRACE macro call. The front of the TTE contains the fields mapped by the $TTE macro that describe the data contained in the TTE.

A JES2 TID Table



TID Tables ...

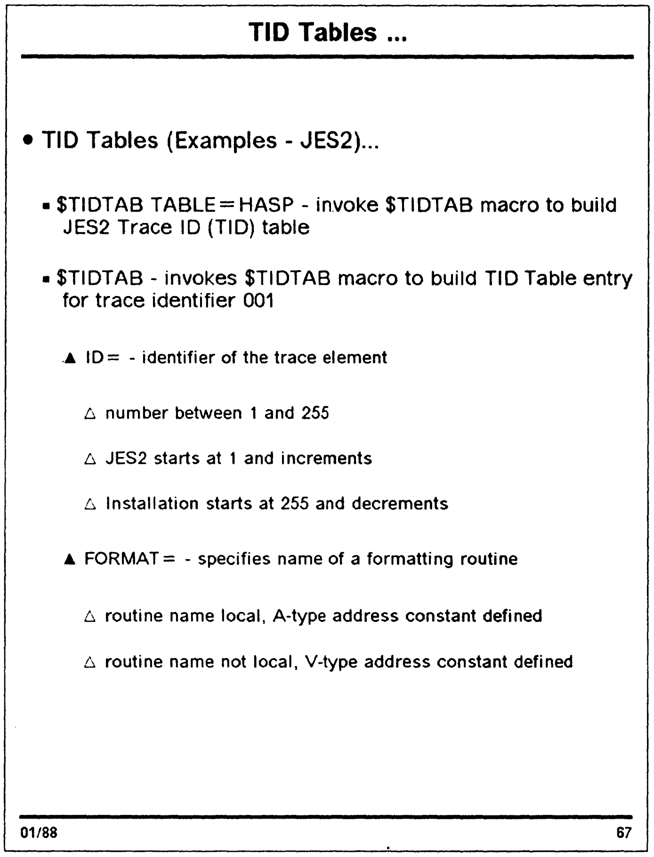
- TID Tables (Examples - JES2)

```
HASPTIDT $TIDTAB TABLE=HASP

         $TIDTAB ID=...

         $TIDTAB ID=001,
               FORMAT=TROUT001,
               NAME=$SAVE

         $TIDTAB ID=...

         $TIDTAB TABLE=END
```

01/88 66

The figure above illustrates what the JES2 TID table looks like. The table element shown represents all the information that JES2 needs to define JES2 trace identifier 1 for the tracing of $SAVEs. This is the table element that is passed to the $TRACE facility. Notice that the name of the TID table is HASPTIDT, the same as that specified in the V-type address constant in the $MCT.

The following describes each operand on the $TIDTAB macro and tells how it should be specified.



TID Tables ...

- TID Tables (Examples - JES2)...
  - $TIDTAB TABLE=HASP - invoke $TIDTAB macro to build JES2 Trace ID (TID) table
  - $TIDTAB - invokes $TIDTAB macro to build TID Table entry for trace identifier 001
    - ▲ ID= - identifier of the trace element
      - △ number between 1 and 255
      - △ JES2 starts at 1 and increments
      - △ Installation starts at 255 and decrements
    - ▲ FORMAT= - specifies name of a formatting routine
      - △ routine name local, A-type address constant defined
      - △ routine name not local, V-type address constant defined

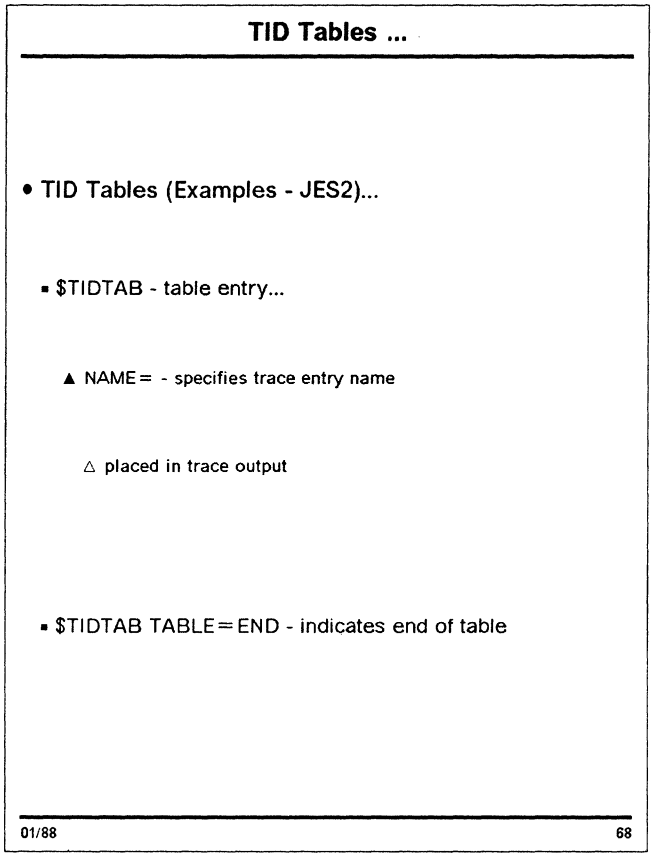
01/88 67

The JES2 tables are started by specifying 'TABLE=HASP'. This indicates to JES2 that this table is a JES2 table. You would specify 'TABLE=USER' to indicate that the table is an installation-coded table. Specifying TABLE=HASP or TABLE=USER is the means JES2 provides to indicate the start of the table (TABLE=START) as discussed in "Concepts" on page 6.

When $TIDTAB is specified with operands other than TABLE=, the macro generates a table element. In the example above, the table element that will be generated will be for trace identifier 1.

The ID operand specifies the trace identifier number that we will use to code the $TRACE macro. The number must be between 1 and 255. JES2-defined trace identifiers start at 1 and increase. Installation-defined trace identifiers should start at 255 and decrease.

The FORMAT operand specifies the name of a formatting routine to be given control when the trace table entries are being processed for printing. If the name that is specified for this operand is found to reside within the same module as the TIDTAB, then an A-type address constant is defined for the name. If the name that is specified for this operand is not found to reside within the same module as the TIDTAB, then a V-type address constant is defined for the name.



TID Tables ...

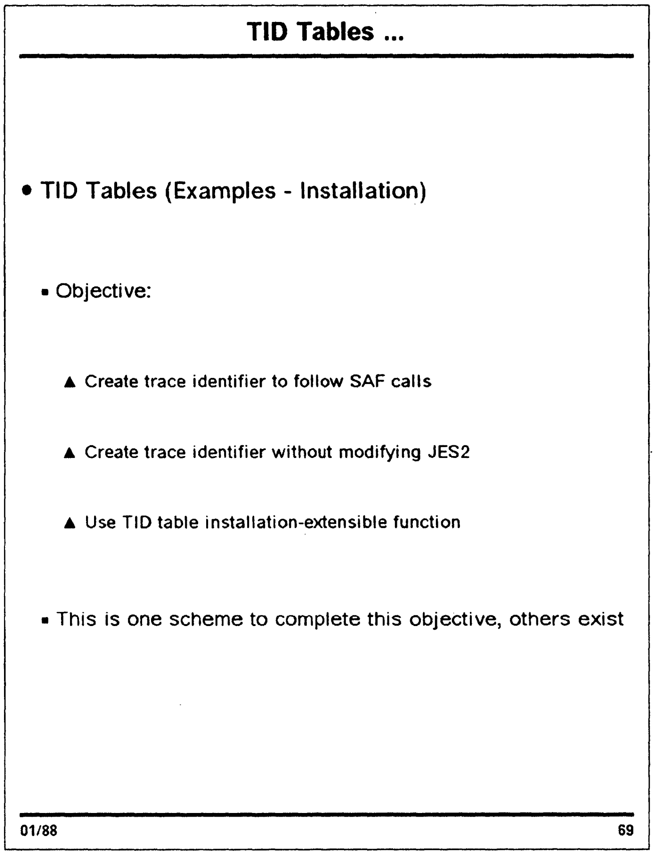
- TID Tables (Examples - JES2)...
  - $TIDTAB - table entry...
    - ▲ NAME= - specifies trace entry name
      - △ placed in trace output
  - $TIDTAB TABLE=END - indicates end of table

01/88 68

The NAME operand specifies a 1-8 character name that is associated with the specified trace id. The name will appear in the trace output to further identify the trace data.

When the TABLE=END is encountered, the table is closed. This indicates the end of the JES2 TID tables. All JES2-defined trace identifiers are defined within this single table.

An Installation TID Table



TID Tables ...

- TID Tables (Examples - Installation)
  - Objective:
    - ▲ Create trace identifier to follow SAF calls
    - ▲ Create trace identifier without modifying JES2
    - ▲ Use TID table installation-extensible function
  - This is one scheme to complete this objective, others exist

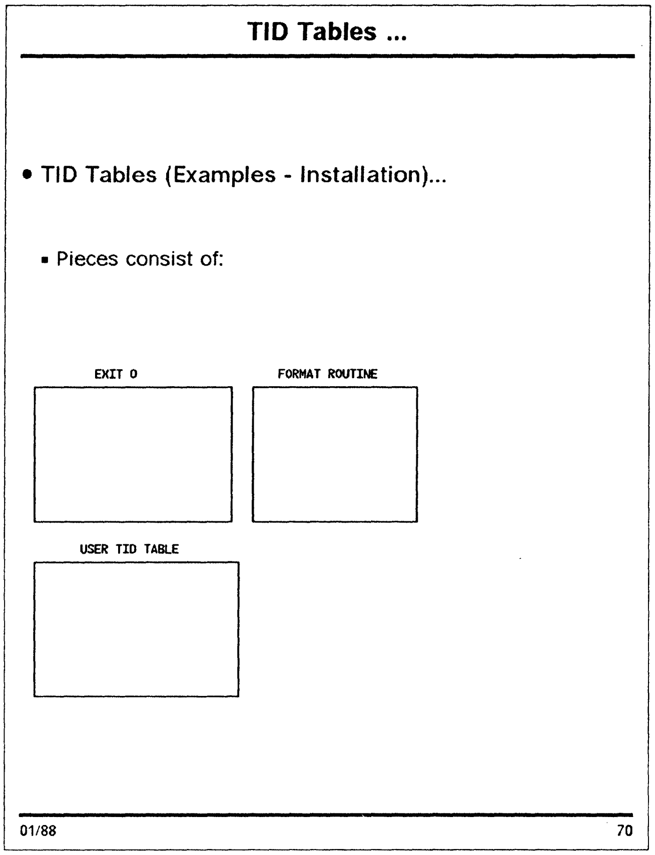
01/88 69

In order to show how you would specify a TID table, we now step through creating an installation-defined trace identifier for tracing security calls from the security PCE.

Objective

The objective is to create a trace identifier for tracing security calls. You wish to achieve this without modifying JES2 and to use the TID tables as the means to define the identifier to JES2.

Required Pieces



TID Tables ...

- TID Tables (Examples - Installation)...

  ▪ Pieces consist of:

EXIT 0

FORMAT ROUTINE

USER TID TABLE

01/88 70

To achieve the objective, you will need to code three pieces. These pieces are:

1. Exit 0

   As was discussed in "Concepts" on page 6, there are two ways to link the installation table with JES2:

   a. The first of these is to link-edit the installation TID table with the HASJES20 load module. This requires the name of the installation table as USERTIDT.

   b. If you do not wish to link-edit the installation TID table with HASJES20 or do not wish to name the installation TID table USERTIDT, then you must fill in the address of the installation TID table into the $MCT field MCTTIDTU. This is the second method. This method requires that you fill in the address before invoking the $TRACE facility to access this trace id. Depending on when the installation will use the trace id, you may fill in the address early in initialization or after JES2 is up and running.

   In this example, you will fill in the address of the TID table early in initialization, specifically in Exit 0. Therefore, you require an Exit 0 that will load the module, if not already loaded, and resolve the address of the table.

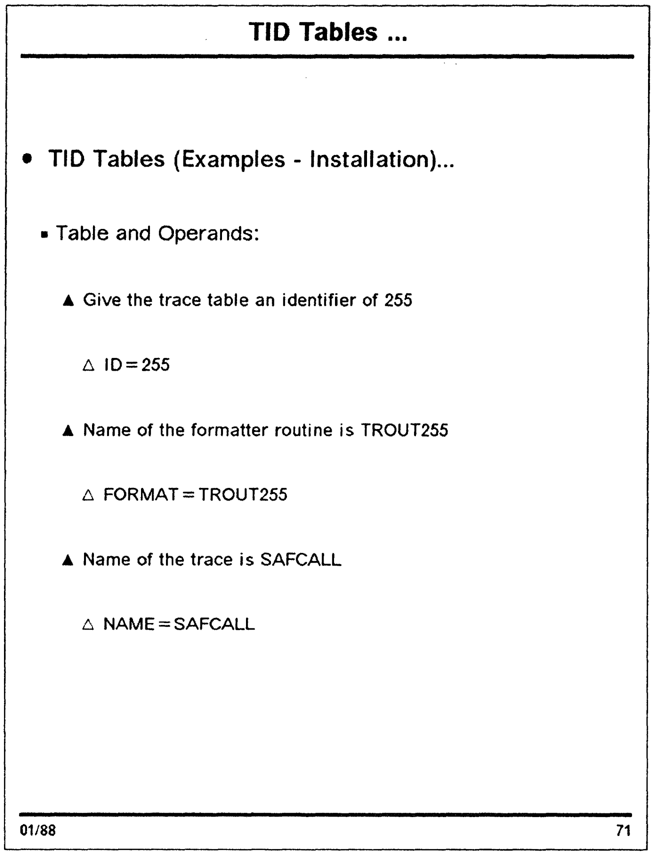
2. Format Routine

   You will need to create a format routine that will get control to format the TTE into a printable form. In this way, you can put the data into the TTE in any form or format and interpret yourself, independent of what JES2 understands or processes.

3. User TID Table

You will have to code a TID table that includes an element for the particular trace id. We will describe this installation table element in a step-wise fashion in the following section.

Coding the Installation TID Table



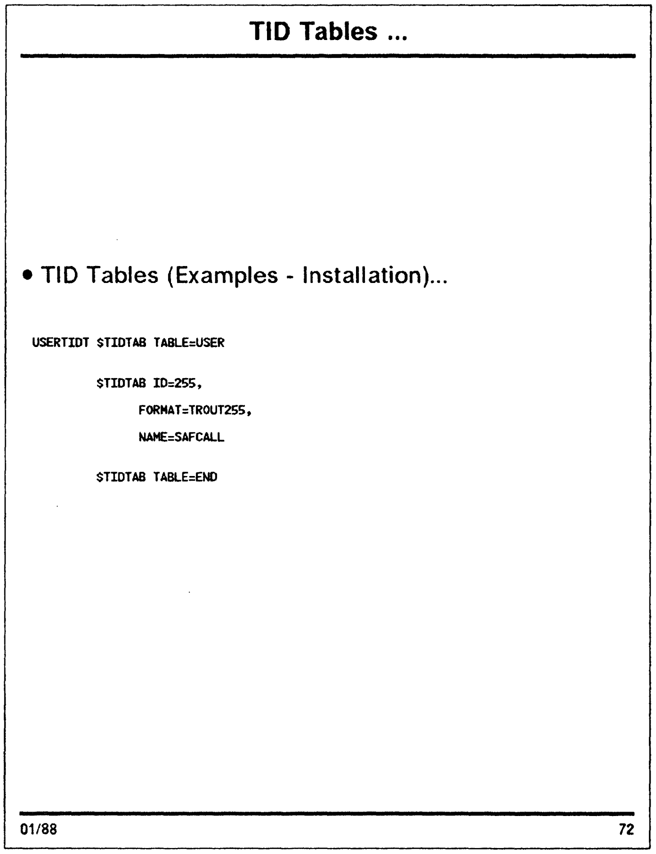
TID Tables ...

- TID Tables (Examples - Installation)...
  - Table and Operands:
    - ▲ Give the trace table an identifier of 255
      - △ ID=255
    - ▲ Name of the formatter routine is TROUT255
      - △ FORMAT=TROUT255
    - ▲ Name of the trace is SAFCALL
      - △ NAME=SAFCALL

01/88 71

Since installation identifiers should start at 255 and decrease, the ID for this installation trace table element will be 255 (ID=255). The format routine will be called TROUT255, for TRace OUTput for identifier 255. The name that should come out on the trace entry should be SAFCALL, since the function of this trace identifier is to trace the fact that a SAF call has been made. Therefore, we will code NAME=SAFCALL on the TIDTAB.

Resulting TID Table



TID Tables ...

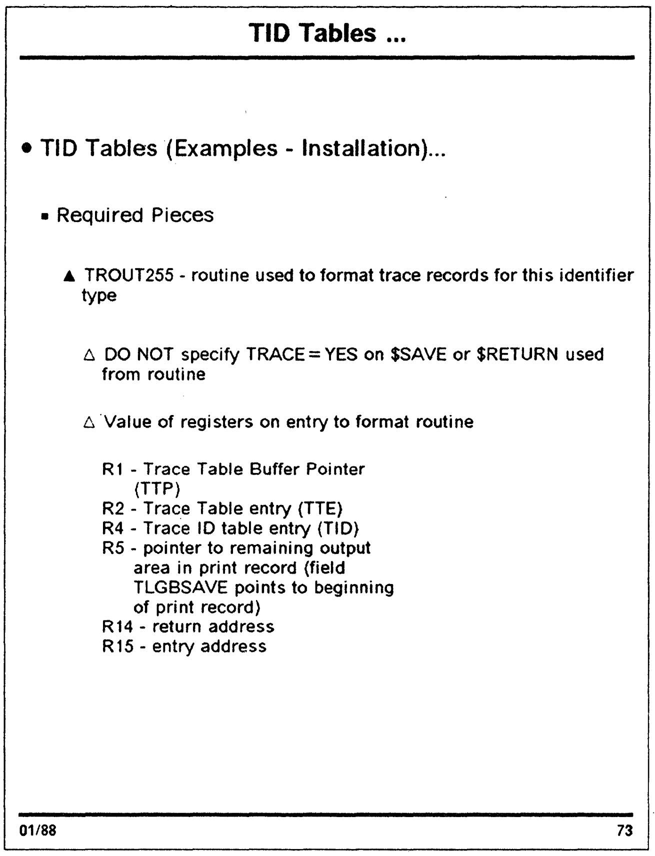
- TID Tables (Examples - Installation)...

```
USERTIDT $TIDTAB TABLE=USER

         $TIDTAB ID=255,
               FORMAT=TROUT255,
               NAME=SAFCALL

         $TIDTAB TABLE=END
```

01/88 72

The figure above shows the resulting installation TID table used to add a security trace identifier to JES2. The table is begun with TABLE = USER to tell JES2 that this is an installation table. The id of the trace element will be 255. The name of the routine that will format the trace data into a printable form will be TROUT255. The name of the trace identifier is SAFCALL. Finally, the table is delineated by the TABLE = END operand.

Coding the Other Required Pieces



TID Tables ...

- TID Tables (Examples - Installation)...
  - Required Pieces
    - ▲ TROUT255 - routine used to format trace records for this identifier type
      - △ DO NOT specify TRACE=YES on $SAVE or $RETURN used from routine
      - △ Value of registers on entry to format routine

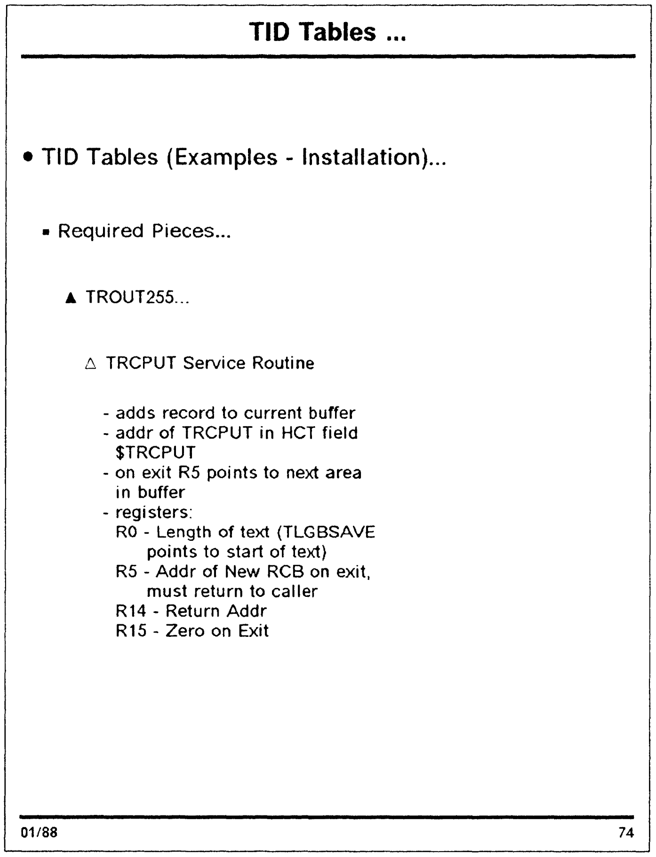
```
R1 - Trace Table Buffer Pointer
     (TTP)
R2 - Trace Table entry (TTE)
R4 - Trace ID table entry (TID)
R5 - pointer to remaining output
     area in print record (field
     TLGBSAVE points to beginning
     of print record)
R14 - return address
R15 - entry address
```

01/88 73

One of the required pieces that you would have to provide to complete the installation extension to the $TRACE facility is the format routine. This is where it becomes obvious that the trace extension facility is primitive.

The installation format routine cannot itself issue a TRACE=YES on its $SAVE or $RETURN. The registers upon entry to the format routine are as follows:

- R1 - this register points to the TTP for the trace table that contains the entry as defined by the installation TIDTAB.
- R2 - this register points to the TTE that contains the data that the installation $TRACE macro saved. This is the data to be formatted by the TROUT255 format routine.
- R4 - this register points to the TIDTAB (Trace Id Table) element that you created.
- R5 - this register points to an open area in an output area. The format routine will take the data contained in the TTE, make the data printable, and place the resulting printable data into this output area, starting at the location pointed to by R5. The field TLGBSAVE in the $TLGWORK area (the variable extension area off of the event trace log PCE) points to the beginning of this output area. The maximum size of this output is defined by an equate in $HASPEQU named TRCLRECL. Therefore, the maximum area that can be saved in this output area is TRCLRECL-1 (minus one for the carriage control). When the output area is full, a call to a routine named TRCPUT can be made to 'PUT' this line and obtain a new output area. We will describe TRCPUT shortly.
- R14 - this register contains the return address.
- R15 - this register contains the format routine entry address.



TID Tables ...

- TID Tables (Examples - Installation)...
  - Required Pieces...
    - ▲ TROUT255...
      - △ TRCPUT Service Routine

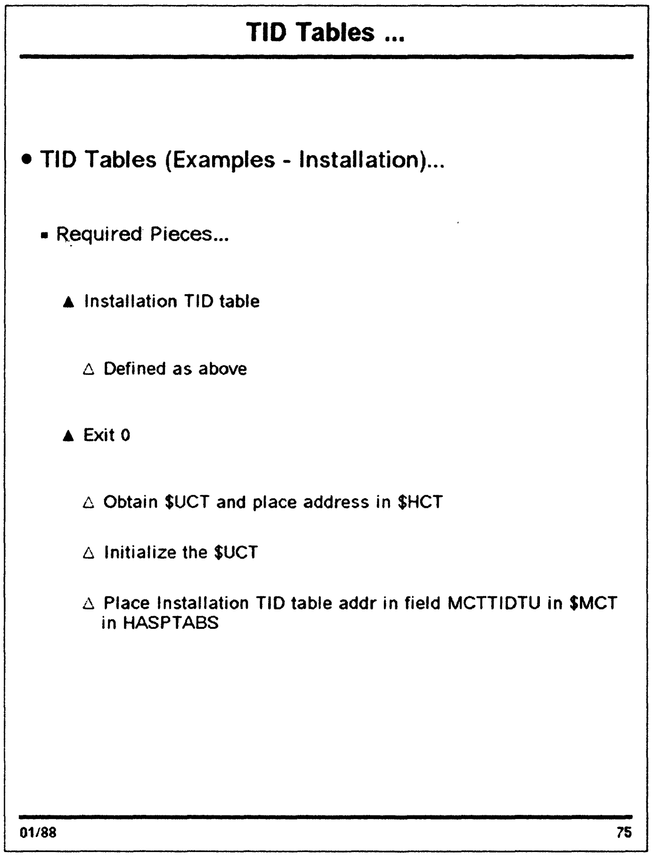
```
- adds record to current buffer
- addr of TRCPUT in HCT field
  $TRCPUT
- on exit R5 points to next area
  in buffer
- registers:
  R0 - Length of text (TLGBSAVE
       points to start of text)
  R5 - Addr of New RCB on exit,
       must return to caller
  R14 - Return Addr
  R15 - Zero on Exit
```

01/88 74

The TRCPUT service routine is an external routine available to installation format routines to "PUT" a formatted output area and obtain a new output area. The address of the TRCPUT routine is available from the $HCT field $TRCPUT.

On entry to the TRCPUT service routine, you must pass the length of the text in R0. You can calculated this by taking the ending address in the output area of the installation data and subtracting the value in TLGBSAVE. R15 must contain the address of the TRCPUT service routine and R14 must contain the return address (i.e., use standard BALR R14,R15 linkage).

On exit, the TRCPUT service routine will return in R5 the address of the new output area. This must be returned by the format routine to the caller of the installation format routine. Therefore, a $STORE of R5 should be done by the format routine upon return from the TRCPUT service routine.



TID Tables ...

- TID Tables (Examples - Installation)...
  - Required Pieces...
    - ▲ Installation TID table
      - △ Defined as above
    - ▲ Exit 0
      - △ Obtain $UCT and place address in $HCT
      - △ Initialize the $UCT
      - △ Place Installation TID table addr in field MCTTIDTU in $MCT in HASPTABS

01/88 75

The last two pieces that are required are the installation TID table, as coded above, and the code for Exit 0. The Exit 0 code is required to do three things.

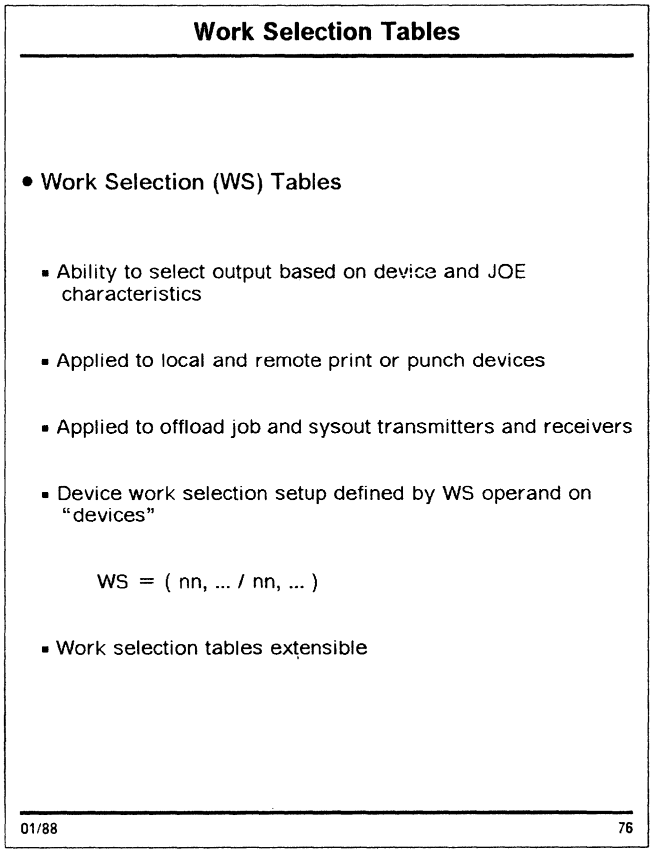
1. It must obtain the $UCT and place the $UCTs address in the $HCT.
2. It must initialize the $UCT.
3. Finally, it must place the installation TID table address in the MCTTIDTU field in the $MCT in module HASPTABS.

The code that is contained in "Appendix A. Table Pairs Coding Example" on page 147 is the code as you would be required to code it. This code includes:

- Exit 0 that obtains the $UCT, places the $UCT address in the $HCT, initializes the $UCT, and places the installation TID table address in the $MCT.
- The HASPXJ00 module contains, among other items that we will describe shortly, the TID table and the TID format routine TROUT255.

WS Tables

What Is a WS Table?



Work Selection Tables

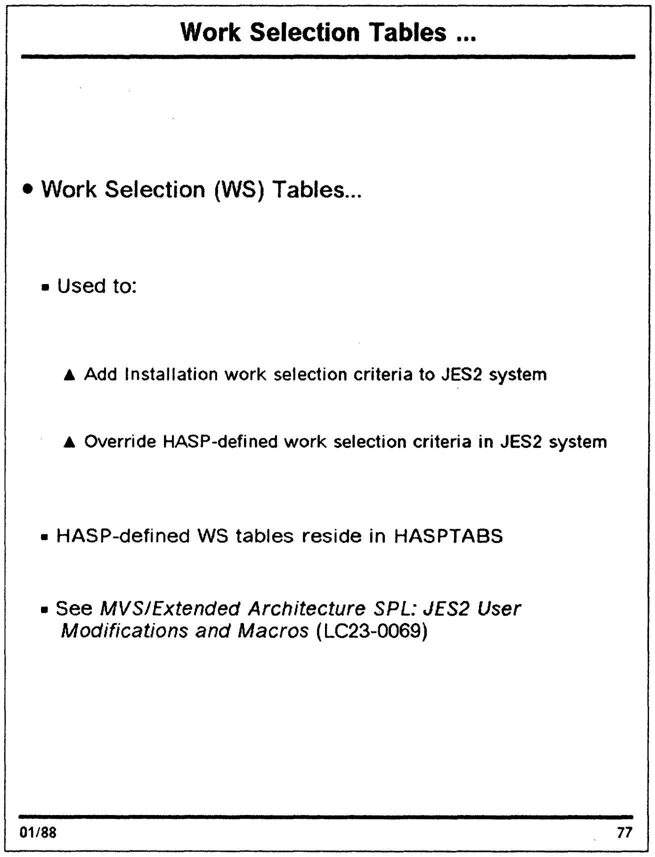
- Work Selection (WS) Tables
  - Ability to select output based on device and JOE characteristics
  - Applied to local and remote print or punch devices
  - Applied to offload job and sysout transmitters and receivers
  - Device work selection setup defined by WS operand on "devices"

    WS = (nn, ... / nn, ...)
  - Work selection tables extensible

01/88 76

Work Selection is the ability to select output based on a matching of device characteristics with output characteristics (JOE characteristics). Work Selection is available in JES2 for local and remote print and punch devices. Also, offload job and sysout transmitters and receivers make use of work selection to determine what output or job to process. Device characteristics are set via the WS (Work Selection) operand on the device. The WS operand contains a list of attributes that define the characteristics of the device. The list is made up of criteria. The position of each criterion relative to the slash in the list determines how important it is to match on that particular criterion. Criteria to the left of the slash require an exact match between the device and the output before that output is considered suitable. Criteria to the right of the slash indicate a preference for a match, but the output need not match exactly.

Additional information on the use of work selection is available in *SPL: JES2 Initialization and Tuning*, form (SC23-0065).



Work Selection Tables ...

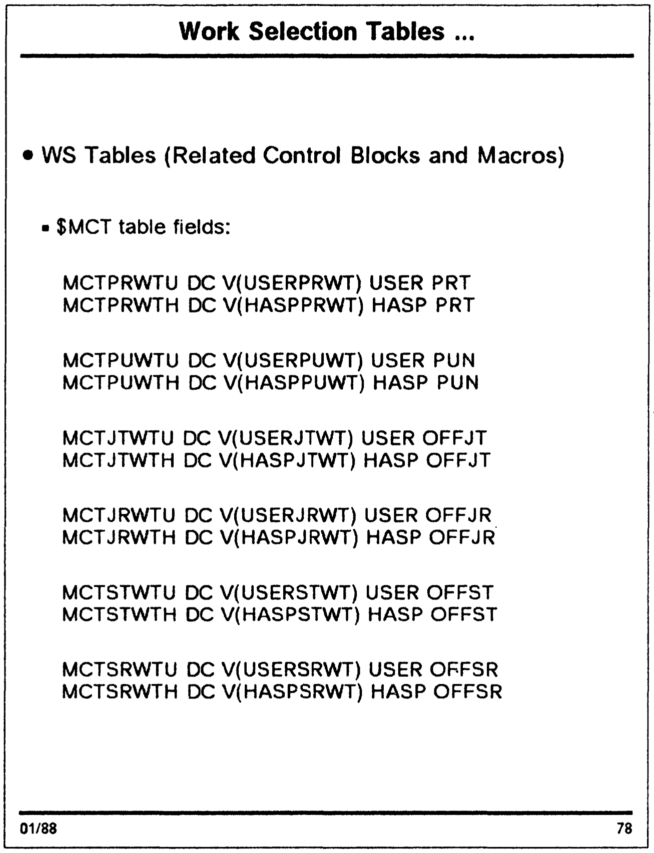
- Work Selection (WS) Tables...
  - Used to:
    - ▲ Add Installation work selection criteria to JES2 system
    - ▲ Override HASP-defined work selection criteria in JES2 system
  - HASP-defined WS tables reside in HASPTABS
  - See *MVS/Extended Architecture SPL: JES2 User Modifications and Macros* (LC23-0069)

01/88 77

The Work Selection (WS) tables are used to add installation work selection criteria to a JES2 system or override JES2-defined work selection criteria in JES2. Notice that deleting JES2 work selection criteria was not discussed. This is because we do not recommend deleting any JES2 work selection criteria.

The JES2-defined WS tables reside in the JES2 module HASPTABS. Some of the following information can be found in *SPL: JES2 User Modifications and Macros.*

WS Control Blocks and Macros



Work Selection Tables ...

- WS Tables (Related Control Blocks and Macros)
  - $MCT table fields:

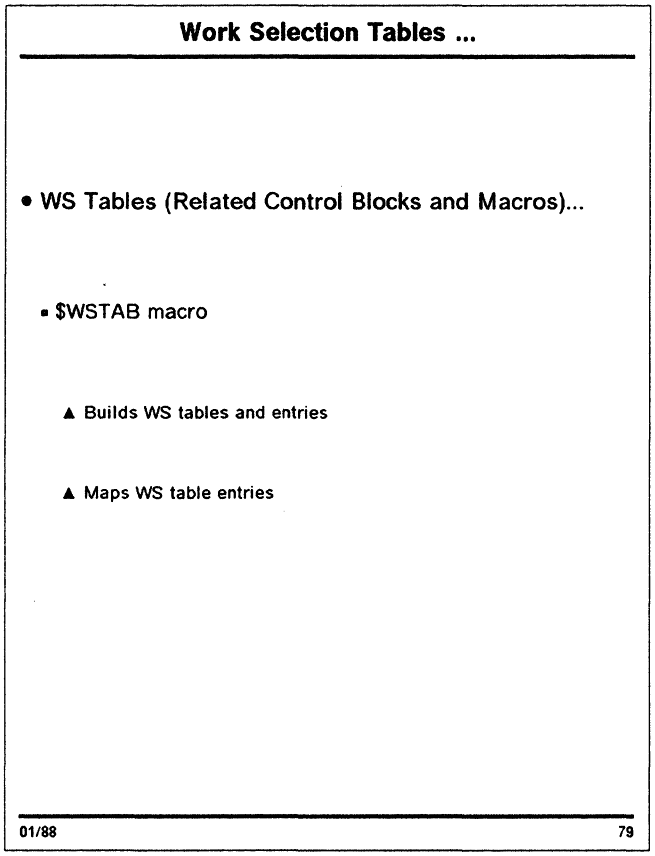
```
MCTPRWTU DC V(USERPRWT) USER PRT
MCTPRWTH DC V(HASPPRWT) HASP PRT

MCTPUWTU DC V(USERPUWT) USER PUN
MCTPUWTH DC V(HASPPUWT) HASP PUN

MCTJTWTU DC V(USERJTWT) USER OFFJT
MCTJTWTH DC V(HASPJTWT) HASP OFFJT

MCTJRWTU DC V(USERJRWT) USER OFFJR
MCTJRWTH DC V(HASPJRWT) HASP OFFJR

MCTSTWTU DC V(USERSTWT) USER OFFST
MCTSTWTH DC V(HASPSTWT) HASP OFFST

MCTSRWTU DC V(USERSRWT) USER OFFSR
MCTSRWTH DC V(HASPSRWT) HASP OFFSR
```

01/88 78

The table pairs that are used to point to the WS tables are located in the $MCT. There is one table pair for each device type which supports work selection. Therefore, there is a table pair for:

- Printers
- Punches
- Offload Job Transmitters
- Offload Job Receivers
- Offload Sysout Transmitters
- Offload Sysout Receivers

The $MCT fields for installation work selection tables are MCTPRWTU for printers, MCTPUWTU for punches, MCTJTWTU for offload job transmitters, MCTJRWTU for offload job receivers, MCTSTWTU for offload sysout transmitters, and MCTSRWTU for offload sysout receivers. If you want to link-edit an installation table with JES2 you must name your tables USERPRWT for printers, USERPUWT for punches, USERJTWT for offload job transmitters, USERJRWT for offload job receivers, USERSTWT for offload sysout transmitters, and USERSRWT for offload sysout receivers. The installation table must then be link-edited with HASJES20. The JES2-defined WS tables are pointed to from the $MCT using the MCT above and table names.



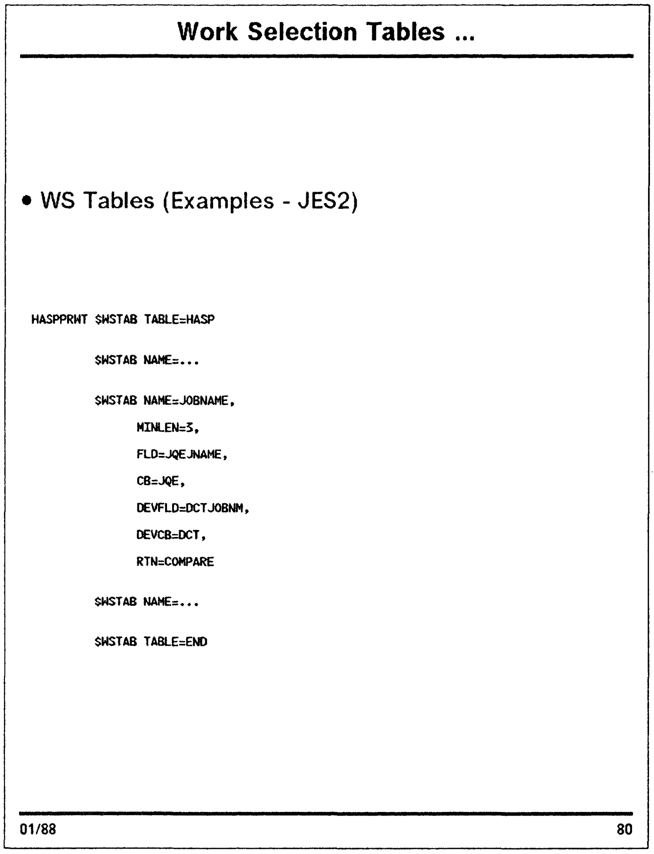
Work Selection Tables ...

- WS Tables (Related Control Blocks and Macros)...
  - $WSTAB macro
    - ▲ Builds WS tables and entries
    - ▲ Maps WS table entries

01/88 79

To aid in the creating WS tables, JES2 supplies a macro named $WSTAB. This macro builds both the JES2 and installation tables and table elements. This macro also contains the mapping macro for the WS tables and elements. We will describe this macro and its operands more thoroughly below.

A JES2 WS Table



Work Selection Tables ...

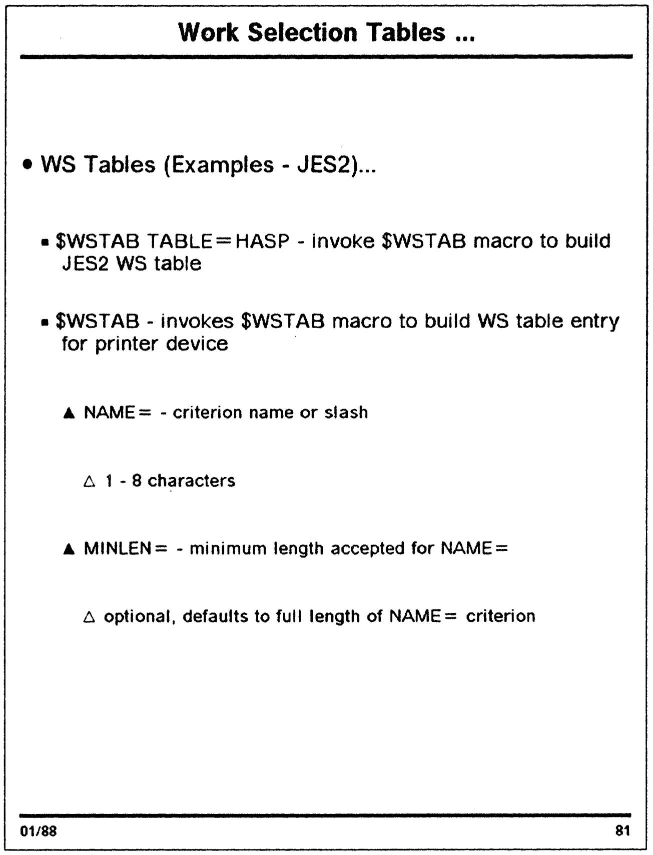
- WS Tables (Examples - JES2)

```
HASPPRWT $WSTAB TABLE=HASP

         $WSTAB NAME=...

         $WSTAB NAME=JOBNAME,
               MINLEN=3,
               FLD=JQEJNAME,
               CB=JQE,
               DEVFLD=DCTJOBNM,
               DEVCB=DCT,
               RTN=COMPARE

         $WSTAB NAME=...

         $WSTAB TABLE=END
```

01/88 80

The figure above illustrates what the JES2 work selection table looks like for the printers work selection criterion JOBNAME. The table element shown represents all the information that JES2 needs to define the JES2 criterion for JOBNAME. This is the table element that is passed to the $#GET service routine which returns eligible JOEs for processing based upon the work selection list defined for the printer. Notice that the name of the WS table is HASPPRWT, the same as that specified for the V-type address constant in the $MCT.

Now we describe each operand on the $WSTAB macro and how you should specify them.



Work Selection Tables ...

- WS Tables (Examples - JES2)...
  - $WSTAB TABLE=HASP - invoke $WSTAB macro to build JES2 WS table
  - $WSTAB - invokes $WSTAB macro to build WS table entry for printer device
    - ▲ NAME= - criterion name or slash
      - △ 1 - 8 characters
    - ▲ MINLEN= - minimum length accepted for NAME=
      - △ optional, defaults to full length of NAME= criterion

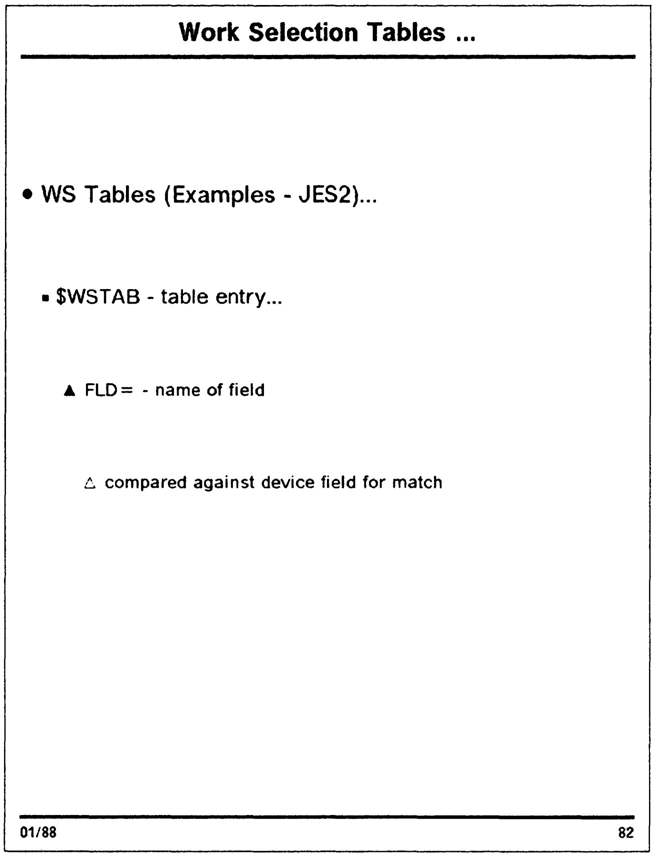
01/88 81

The JES2 tables are started by specifying 'TABLE=HASP'. This indicates to JES2 that this table is a JES2 table. You specify 'TABLE=USER' to indicate that the table is an installation-coded table. Specifying TABLE=HASP or TABLE=USER is the means JES2 provides to indicate the start of the table (TABLE=START) as discussed in "Concepts" on page 6.

When $WSTAB is specified with operands other than TABLE=, the macro generates a table element. In the example above, the table element that is generated is for the JOBNAME work selection criterion.

The NAME operand specifies the 1-8 character name of the criterion. The specified name is used to display work selection criteria as well as to specify the criteria in the work selection list. The NAME can also specify the special character '/'. The slash delineates the left section of the work selection list from the right section. Criteria to the left of the slash are required to match explicitly. Criteria to the right of the slash are not required to match. In the example above, the name of the work selection criterion is JOBNAME.

The MINLEN operand specifies the minimum length that is required for the NAME. In the example, the minimum length that can be specified for JOBNAME is 3, that is, JOB would be all that would be needed before it was recognized as JOBNAME. The default value for this field is the entire length of the value entered for the NAME operand.

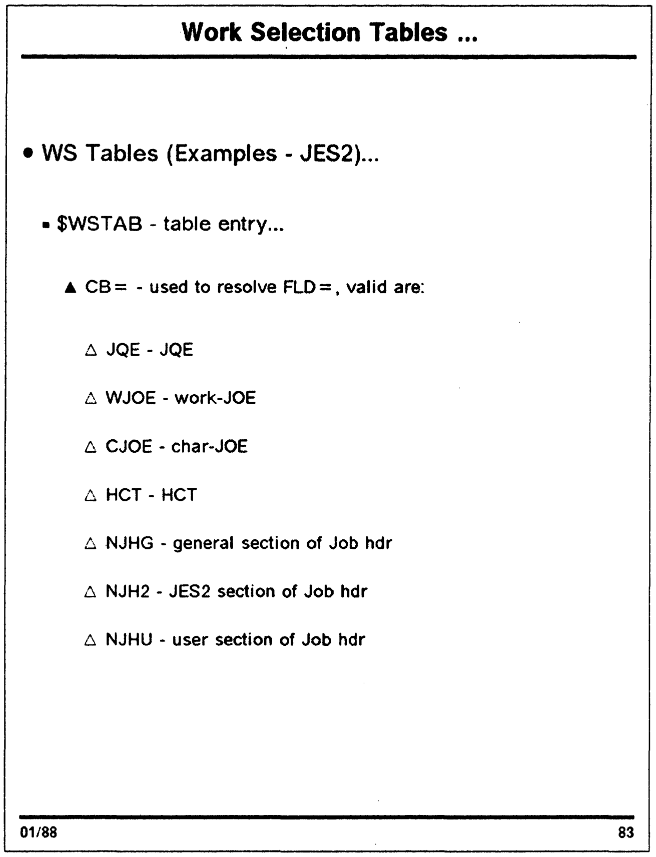


Work Selection Tables ...

- WS Tables (Examples - JES2)...
  - $WSTAB - table entry...
    - ▲ FLD= - name of field
      - △ compared against device field for match

01/88 82

The FLD operand specifies the name of the field used to determine if there is a match with the device field. In the example, the field JQEJNAME holds the job name. Thus, the job name is compared against that specified with the device to determine if this job is illegible for processing by the device.



Work Selection Tables ...

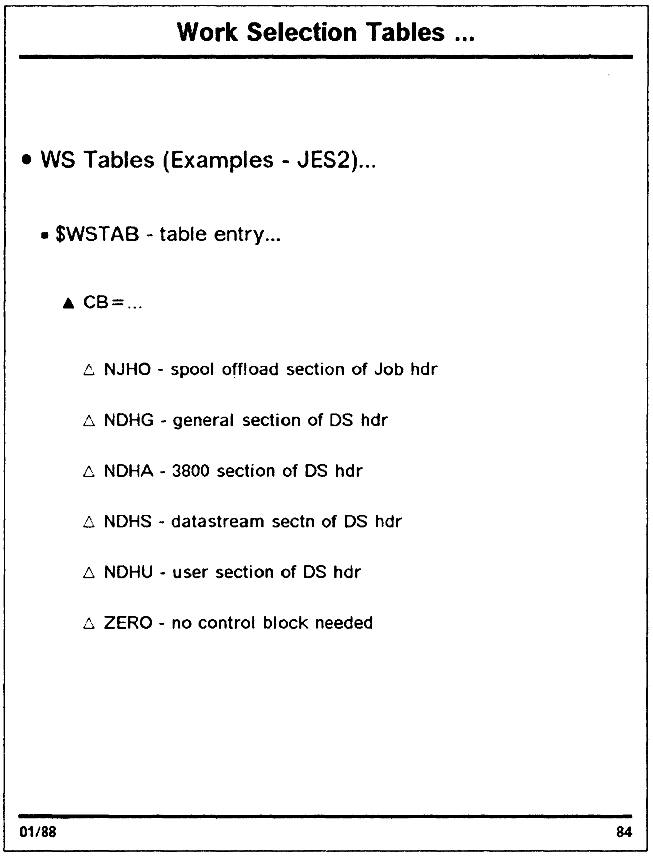
- WS Tables (Examples - JES2)...
  - $WSTAB - table entry...
    - ▲ CB= - used to resolve FLD=, valid are:
      - △ JQE - JQE
      - △ WJOE - work-JOE
      - △ CJOE - char-JOE
      - △ HCT - HCT
      - △ NJHG - general section of Job hdr
      - △ NJH2 - JES2 section of Job hdr
      - △ NJHU - user section of Job hdr

01/88 83

The CB operand tells JES2 what control block the value specified for the FLD operand is in. JES2 understands a finite number of values for the CB operand. This is because JES2 will use the CB operand to determine what control blocks should be scanned to obtain a match between the FLD/CB pair and that specified for the device. Therefore, in the example above, the CB=JQE was specified which implies that the JQE must be looked at (using the JQEJNAME field) to find a match. JES2 understands how to obtain the JQE address. JES2 would not understand all control blocks.

Those control blocks that JES2 understands how to get addresses for include:

- JQE - the control block that represents jobs to JES2.
- WJOE - the work JOE which contains information on the output to be printed.
- CJOE - the characteristics JOE which contains information on some of the characteristics of the output.
- NJHG - general section of the Job Header. This would be useful if the work selection list were to select on header fields (as for OFFLOADing).
- NJH2 - JES2 section of the Job Header.
- NJHU - user section of the Job Header. This would be useful for installations that might want to add work selection criteria of OFFLOAD (for example) where fields for selection resided in the user section of the header.



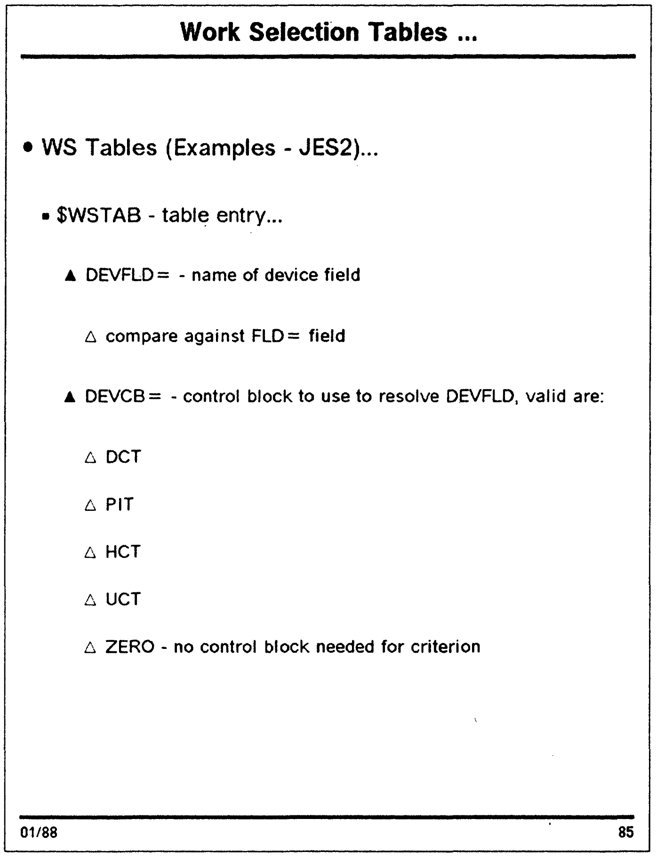
Work Selection Tables ...

- WS Tables (Examples - JES2)...
  - $WSTAB - table entry...
    - ▲ CB=...
      - △ NJHO - spool offload section of Job hdr
      - △ NDHG - general section of DS hdr
      - △ NDHA - 3800 section of DS hdr
      - △ NDHS - datastream sectn of DS hdr
      - △ NDHU - user section of DS hdr
      - △ ZERO - no control block needed

01/88 84

Some of the other control blocks known by JES2 include:

- NJHO - spool offload section of the Job header. This is the section that JES2 uses for the spool offloading and reloading of jobs.
- NDHG - general section of the dataset header. Just like the general section of the job header, might be useful for installations to create work selection criteria for selecting items from the net, tape, etc.
- NDHA - the 3800 section of the dataset header is also known by JES2.
- NDHS - datastream section of the dataset header.
- NDHU - user section of the dataset header.
- ZERO - this implies that no control block is needed and the FLD and FLAG operands are ignored.



Work Selection Tables ...

- WS Tables (Examples - JES2)...
  - $WSTAB - table entry...
    - ▲ DEVFLD= - name of device field
      - △ compare against FLD= field
    - ▲ DEVCB= - control block to use to resolve DEVFLD, valid are:
      - △ DCT
      - △ PIT
      - △ HCT
      - △ UCT
      - △ ZERO - no control block needed for criterion

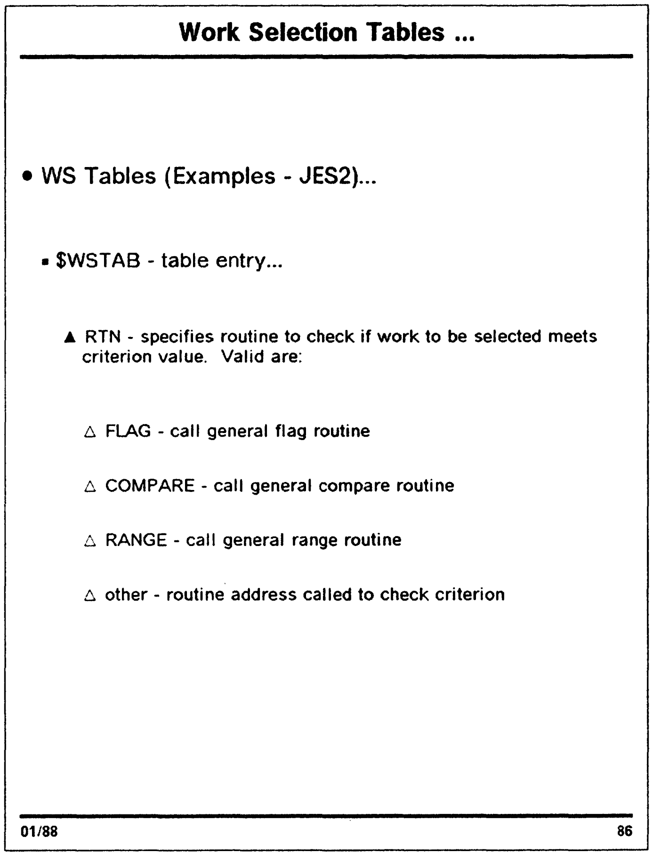
01/88 85

The DEVFLD operand specifies the name of the field used to find a match for the device. This device field is compared against the field specified for the FLD operand to determine if this device should select that item for processing. In the example, the DCTJOBNM field is compared to the JQEJNAME field in the JQE. If the fields are compatible, then the device can select that job represented by the JQE.

The DEVCB operand tells JES2 what control block the value specified for the DEVFLD operand is in. JES2 understands a finite number of values for the DEVCB operand. Like the CB operand, JES2 will use the DEVCB operand to determine what control blocks should be scanned to obtain a match between the FLD/CB pair and the DEVFLD/DEVCB pair. Therefore, in the example above, the DEVCB=DCT was specified which implies that the DCT (Device Control Table, represents the device) must be looked at (using the DCTJOBNM field) to find a match. JES2 understands how to obtain the DCT address. JES2 would not understand all control blocks.

Those control blocks that JES2 understands how to get addresses for include:

- DCT - the control block that represents devices
- PIT - the control block that represents MVS initiators
- HCT - the HASP Communication Table
- UCT - the User Communication Table
- Zero - this implies that no control block is needed. Both the DEVFLD and DEVFLAG operands are ignored.



Work Selection Tables ...

- WS Tables (Examples - JES2)...
  - $WSTAB - table entry...
    - ▲ RTN - specifies routine to check if work to be selected meets criterion value. Valid are:
      - △ FLAG - call general flag routine
      - △ COMPARE - call general compare routine
      - △ RANGE - call general range routine
      - △ other - routine address called to check criterion

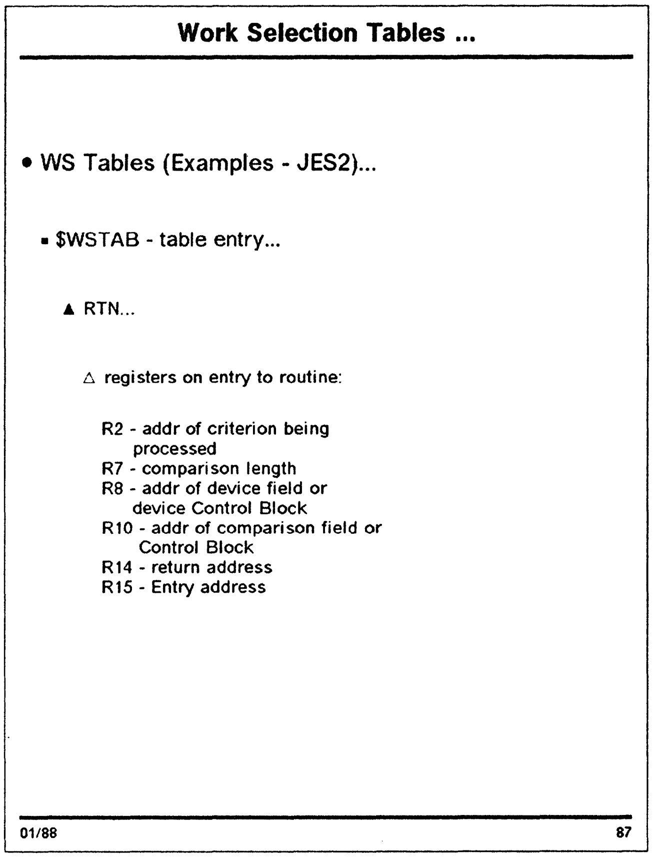
01/88 86

The RTN operand specifies a routine that is called to check if the work that has been selected satisfies the criterion value. There are three routines that JES2 provides to support work selection tables. These routines are:

- FLAG - a general routine to determine if flag values match
- COMPARE - a general routine to compare if two fields match
- RANGE - a general routine to determine if a value lies within a specified range.

In addition to these general routines, a routine name can be specified to receive control to perform the selection verification. We will say more on this shortly.

In the JES2 example, the COMPARE operand is specified to compare the JQE control block field JQEJNAME with the DCT control block field DCTJOBNM to determine if a match exists.



Work Selection Tables ...

- WS Tables (Examples - JES2)...
  - $WSTAB - table entry...
    - ▲ RTN...
      - △ registers on entry to routine:

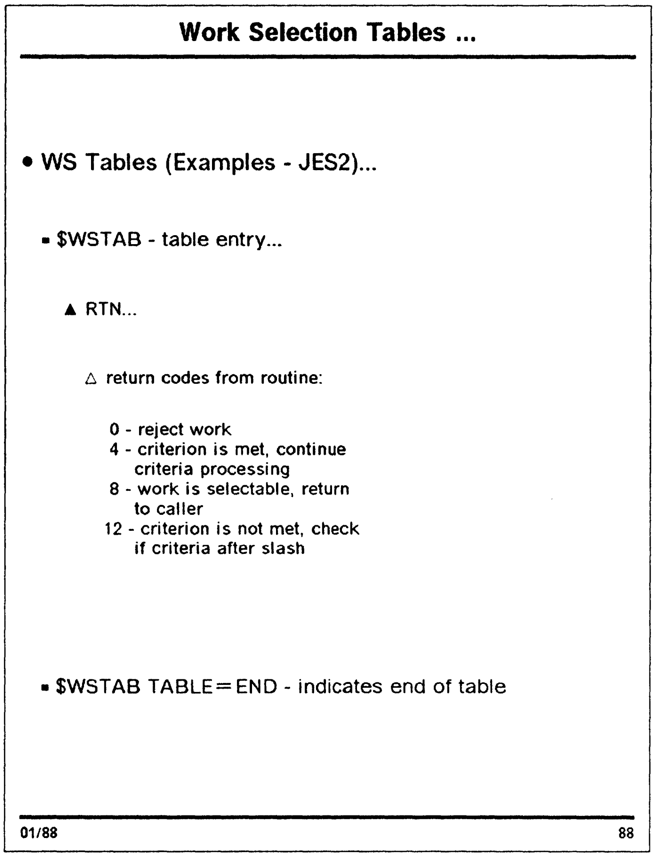
        R2 - addr of criterion being
        processed
        R7 - comparison length
        R8 - addr of device field or
        device Control Block
        R10 - addr of comparison field or
        Control Block
        R14 - return address
        R15 - Entry address

01/88 87

You can specify an installation routine to receive control to perform the validation. When the routine is given control:

- R2 will contain the address of the criterion being processed.
- R7 will contain the length of the field being compared.
- R8 contains the address of the device field (as specified via the DEVFLD operand) or device control block (as specified via the DEVCB operand).
- R10 contains the address of the comparison field (as specified via the FLD operand) or the control block (as specified via the CB operand).
- R14 contains the return address.
- R15 contains the routine entry address.

It is very important to keep in mind that the routine is called for every check of this criterion when it is in the work selection list. Therefore, this routine is in a potentially critical performance path (which is the reason for the non-standard register interface). Registers R2, R3, R4, R11, R12 and R13 must not be altered by the routine.



Work Selection Tables ...

- WS Tables (Examples - JES2)...
  - ▪ $WSTAB - table entry...
    - ▲ RTN...
      - △ return codes from routine:

```
0 - reject work
4 - criterion is met, continue
    criteria processing
8 - work is selectable, return
    to caller
12 - criterion is not met, check
    if criteria after slash
```

  - ▪ $WSTAB TABLE=END - indicates end of table

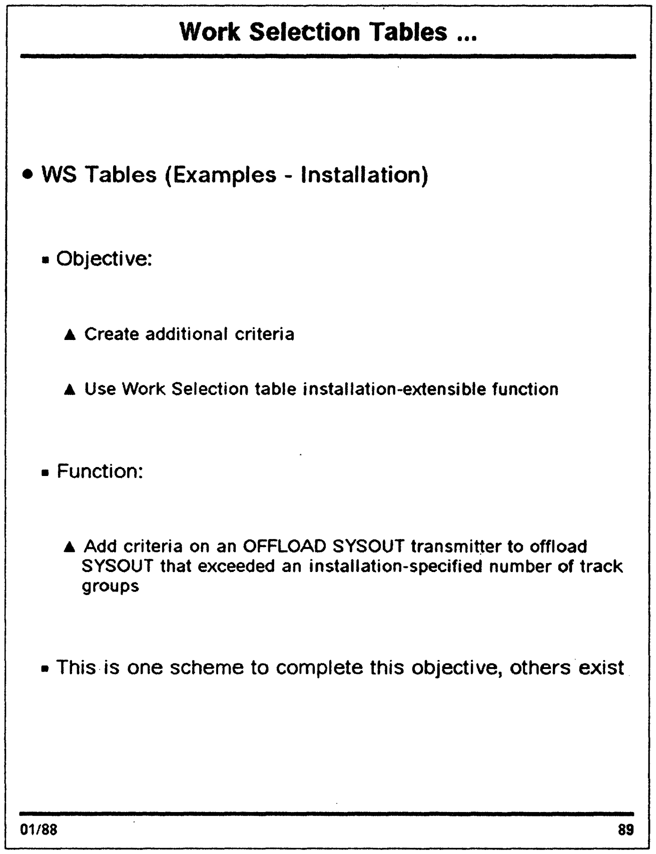
01/88 88

There are four valid return codes that can be set by the installation routine. These are:

- 0 - implies that this unit of work should be rejected, that no more scanning should be done.
- 4 - implies that the test was positive and that this unit of work **may** be acceptable depending on the tests of any other work selection criteria.
- 8 - implies that this unit of work should be selected without any further scanning of the work selection criteria.
- 12 - implies that the test for this criterion failed. However, this may be acceptable depending on the location of the criterion (before or after the slash); processing should continue to determine if this failure is acceptable.

When the TABLE=END is encountered, the table is closed. This indicates the end of the JES2 Printer Work Selection tables. All of the JES2-defined printer work selection criteria are defined within this single table.

An Installation WS Table



Work Selection Tables ...

- WS Tables (Examples - Installation)
  - Objective:
    - ▲ Create additional criteria
    - ▲ Use Work Selection table installation-extensible function
  - Function:
    - ▲ Add criteria on an OFFLOAD SYSOUT transmitter to offload SYSOUT that exceeded an installation-specified number of track groups
  - This is one scheme to complete this objective, others exist

01/88 89

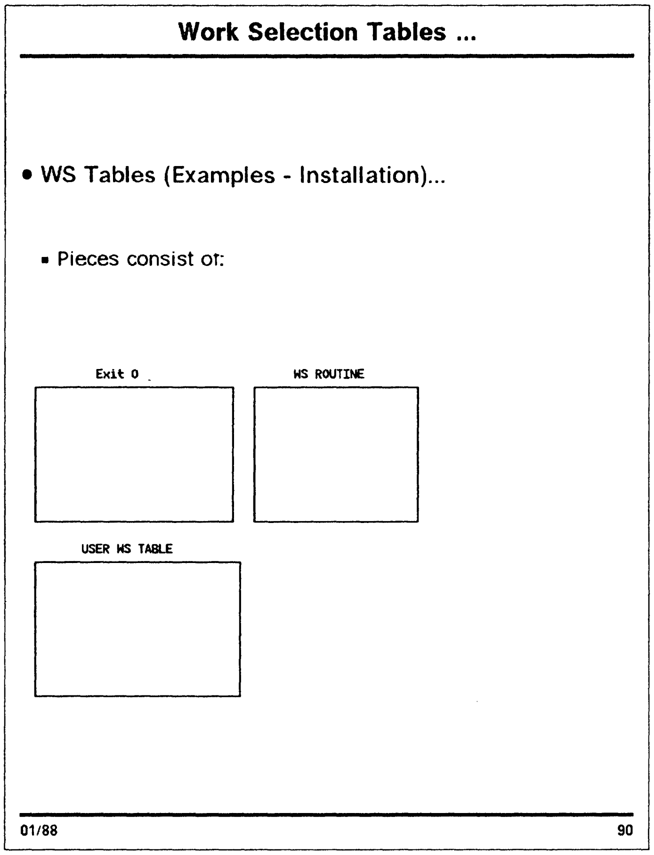
In order to show how you would specify installation Work Selection tables, the remaining description of the WS tables will step through the creation of an installation-defined work selection criteria to select output that is beyond a specified limit for offload processing.

Objective

During periods of peak spool use (e.g., end of month or end of year processing), you may be interested in using the MVS/SP JES2 2.1.5 Spool Offload facility to offload jobs that are using a large amount of JES2 spool. In order to achieve this in a way that involves the least amount of code, you would like there to be an additional work selection criterion on the Offload SYSOUT Transmitter. This operand would indicate at what spool usage threshold a job would be when it would be offloaded from the system.

To achieve this, you will add an installation table element to the work selection list for the Offload SYSOUT Transmitter. The following documents the pieces required, the coding of the table element, and the required code to "plug" the table in. This is one scheme to achieve the stated objective; others do exist.

Required Pieces



Work Selection Tables ...

- WS Tables (Examples - Installation)...
  - Pieces consist of:

01/88 90

To achieve the objective, you will need to code three pieces. These pieces are:

1. Exit 0

   As was discussed in "Concepts" on page 6, there are two ways to link the installation table with JES2:

   a. The first of these is to link-edit the installation Work Selection table with the HASJES20 load module. This requires the name of the installation table be USERSTWT.

   b. If you do not wish to link-edit your installation Work Selection table USERSTWT, then you must fill in the address of your installation Work Selection table into the $MCT field MCTSTWTU. This is the second method. This method requires that you fill in the address of your table in the $MCT before invoking the Offload SYSOUT Transmitter to access this Work Selection criterion. Depending on when you use the transmitter, you may fill in the address early in initialization or after JES2 is up and running.

   In this example, you will fill in the address of the Work Selection table early in initialization, specifically in Exit 0. Therefore, you require an Exit 0 that will load your module (if not already loaded) and resolve the address of the table.

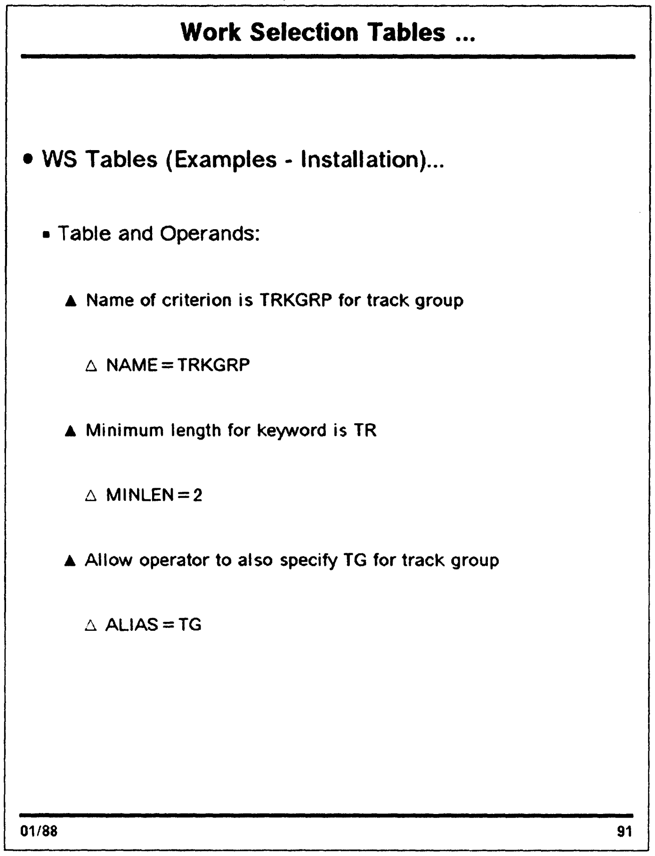
2. Work Selection Routine

   The method of deciding whether a job exceeds the specified spool usage threshold requires finding the amount of spool space used by the job. This value is held in two separate locations, depending on whether or not the job is in conversion or execution, or is elsewhere. Since this requires code more complex than that which the "canned" compare, range, or flag routines can handle, you must code a work selection routine to gain control.

3. User WS Table

   You will have to code an installation Work Selection table that includes the table element for your particular work selection criterion. We will describe this installation table element in a step-wise fashion below.

Coding the Installation WS Table



Work Selection Tables ...

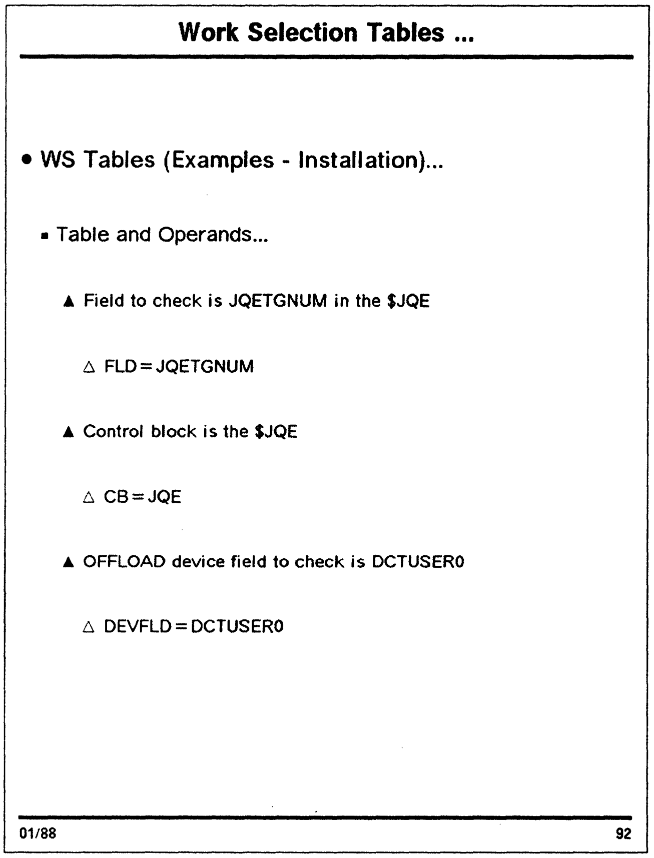
- WS Tables (Examples - Installation)...
  - Table and Operands:
    - ▲ Name of criterion is TRKGRP for track group
      - △ NAME=TRKGRP
    - ▲ Minimum length for keyword is TR
      - △ MINLEN=2
    - ▲ Allow operator to also specify TG for track group
      - △ ALIAS=TG

01/88 91

Coding the installation Work Selection table involves deciding what values you want to expose to your operators. For example, the work selection operand that is seen and entered by the operators is TRKGRP, which indicates that work is selected based on the number of track groups (spool space) that has been allocated to a job.

Since TRKGRP involves typing six characters, you may wish to make it easier for the operator by indicating that only 2 of the 6 characters need be typed. Therefore, you will specify a minimum length of 2 (MINLEN=2).

Also, when JES2 publications talk about track groups, they often refer to them in the abbreviated form of TG. In order to prevent confusion, you could specify an alias of TRKGRP that may make more sense to your operators. Thus, the alias for TRKGRP is TG.



Work Selection Tables ...

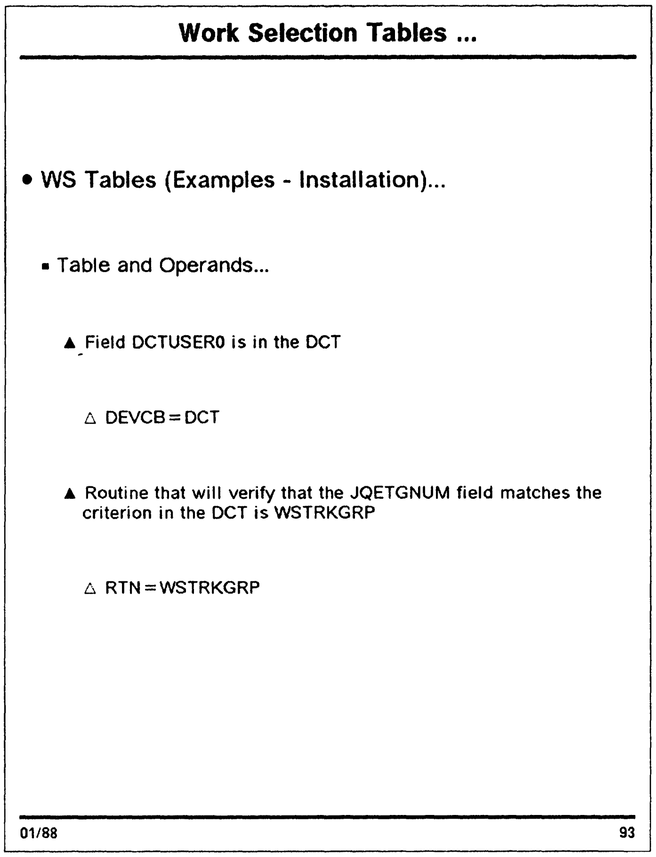
- WS Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Field to check is JQETGNUM in the $JQE
      - △ FLD=JQETGNUM
    - ▲ Control block is the $JQE
      - △ CB=JQE
    - ▲ OFFLOAD device field to check is DCTUSER0
      - △ DEVFLD=DCTUSER0

01/88 92

The field that contains the number of track groups allocated to the job is JQETGNUM. This field determines whether there is a match with the device field. Therefore, the FLD operand is set to JQETGNUM. Thus, the job's number of track groups obtained from field JQETGNUM determines whether the Offload SYSOUT Transmitter "device" should select this job for transmitting.

The field FLD=JQETGNUM is located in the control block JQE. The JQE (Job Queue Element) is a control block that represents the job while it is in the system.

So, the job's field JQETGNUM is compared against a threshold value set for the Offload SYSOUT Transmitter "device". The threshold value for the transmitter device is held in the field DCTUSER0. The DCTUSER0 field is set by the operator as the threshold value. We will discuss the setting of the field in the Installation Examples section of the $SCAN tables. Thus, the devices field is DEVFLD=DCTUSER0.



Work Selection Tables ...

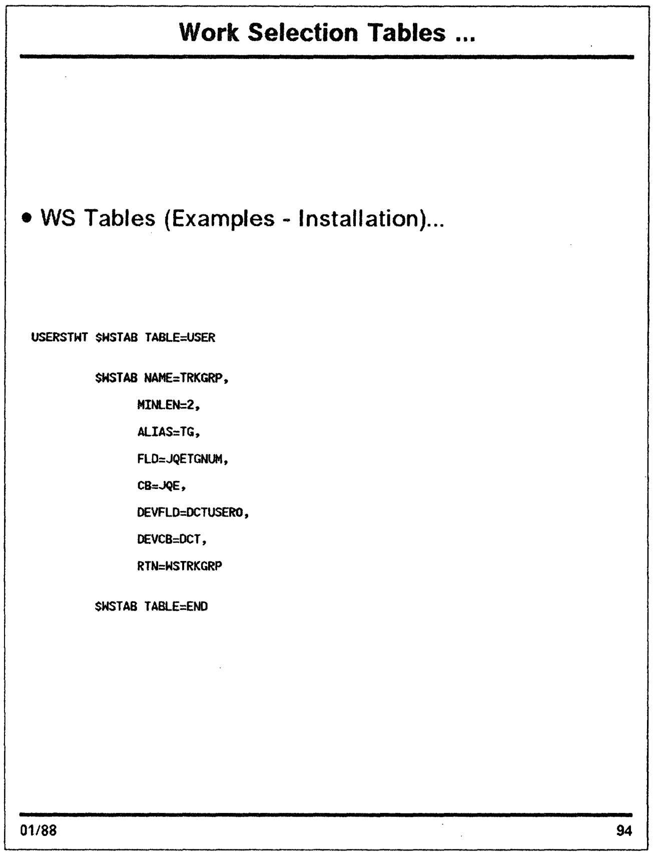
- WS Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Field DCTUSER0 is in the DCT
      - △ DEVCB=DCT
    - ▲ Routine that will verify that the JQETGNUM field matches the criterion in the DCT is WSTRKGRP
      - △ RTN=WSTRKGRP

01/88 93

The device field DCTUSER0 is located in the control block DCT (Device Control Table). DCTs define devices to JES2. Thus, every device in JES2 has a DCT; this includes Offload SYSOUT Transmitters. Therefore, the device control block is DEVCB=DCT.

As discussed earlier, a work selection routine will have to gain control to verify that the amount of spool space allocated to a job (JQETGNUM) is greater than the threshold specified by the user for the device (DCTUSER0). This is because while the job is in conversion or execution, JQETGNUM holds an offset into the checkpoint area which contains the number of track groups allocated to the job. Thus, the routine is named (WSTRKGRP). This routine must be link-edited with this table entry so that the routine's address can be resolved. See the sample code in "Appendix A. Table Pairs Coding Example" on page 147.

Resulting WS Table



Work Selection Tables ...

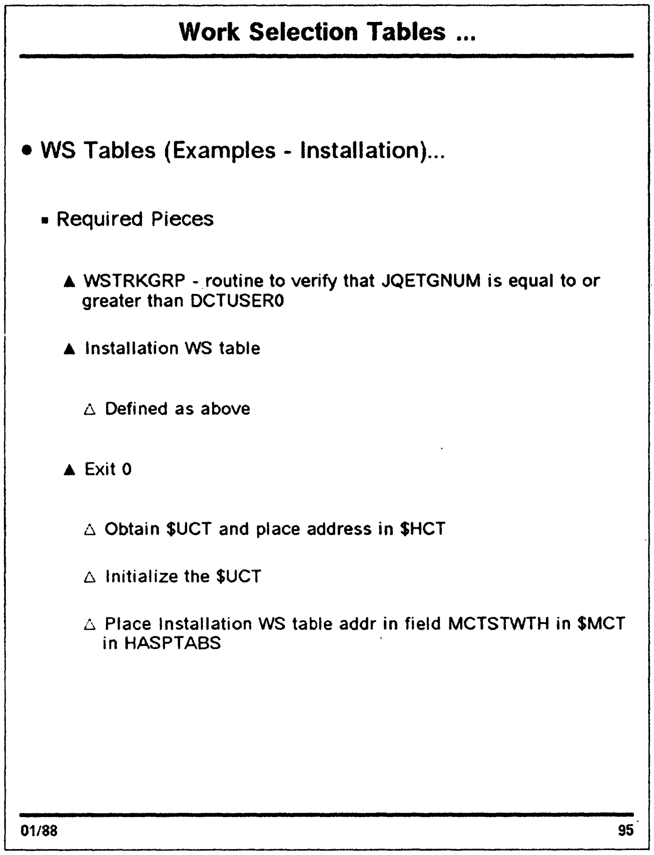
- WS Tables (Examples - Installation)...

```
USERSTWT $WSTAB TABLE=USER

         $WSTAB NAME=TRKGRP,
                MINLEN=2,
                ALIAS=TG,
                FLD=JQETGNUM,
                CB=JQE,
                DEVFLD=DCTUSER0,
                DEVCB=DCT,
                RTN=WSTRKGRP

         $WSTAB TABLE=END
```

01/88 94

The figure above shows the resulting installation Work Selection Table to add a work selection operand to the Offload SYSOUT Transmitter. The table is begun with a TABLE = USER to tell JES2 that this is an installation table. The name of the work selection criterion is TRKGRP. Only TR need be typed by the operator to indicate TRKGRP, or the operator can use the alias name of TG. The field to compare with in the job is JQETGNUM in the job's control block JQE. The field to compare against in the device is DCTUSER0 in the device control block DCT. A routine to do the actual comparison is called WSTRKGRP. Finally, the table is ended with a TABLE = END to indicate to JES2 that this installation table is completed.

Coding the Other Required Pieces



Work Selection Tables ...

- WS Tables (Examples - Installation)...
  - Required Pieces
    - ▲ WSTRKGRP - routine to verify that JQETGNUM is equal to or greater than DCTUSER0
    - ▲ Installation WS table
      - △ Defined as above
    - ▲ Exit 0
      - △ Obtain $UCT and place address in $HCT
      - △ Initialize the $UCT
      - △ Place Installation WS table addr in field MCTSTWTH in $MCT in HASPTABS

01/88 95

The pieces required to permit an installation to add a work selection operand are the installation work selection routine (WSTRKGRP), the installation work selection table (as coded above), and the code for Exit 0. The Exit 0 code is required to do three things.

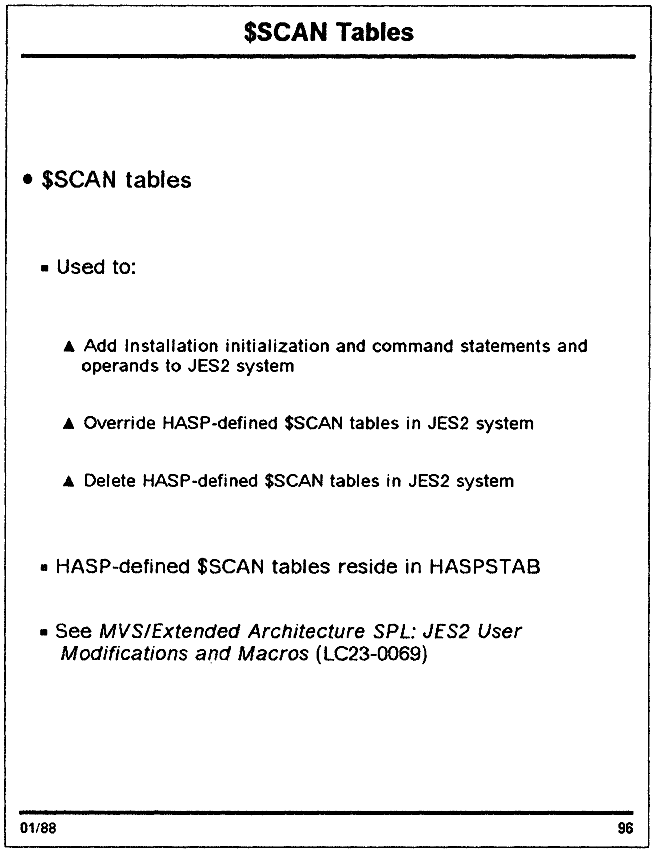
1. It must obtain the $UCT and place the $UCT's address in the $HCT.
2. It must initialize the $UCT.
3. Finally, it must place the installation Work Selection table address in the MCTSTWTU field in the $MCT in module HASPTABS.

The code that is contained in "Appendix A. Table Pairs Coding Example" on page 147 is the code as an installation would be required to code it. This code includes:

- Exit 0 that obtains the $UCT, places the $UCT address in the $HCT, initializes the $UCT, and places the installation Work Selection table address in the $MCT.
- The HASPXJ00 module contains, among other items that we will describe shortly, the Work Selection table and the Work Selection criterion routine WSTRKGRP.

$SCAN Tables

What Is a $SCAN Table?



$SCAN Tables

- $SCAN tables
  - Used to:
    - ▲ Add Installation initialization and command statements and operands to JES2 system
    - ▲ Override HASP-defined $SCAN tables in JES2 system
    - ▲ Delete HASP-defined $SCAN tables in JES2 system
  - HASP-defined $SCAN tables reside in HASPSTAB
  - See *MVS/Extended Architecture SPL: JES2 User Modifications and Macros* (LC23-0069)

01/88 96

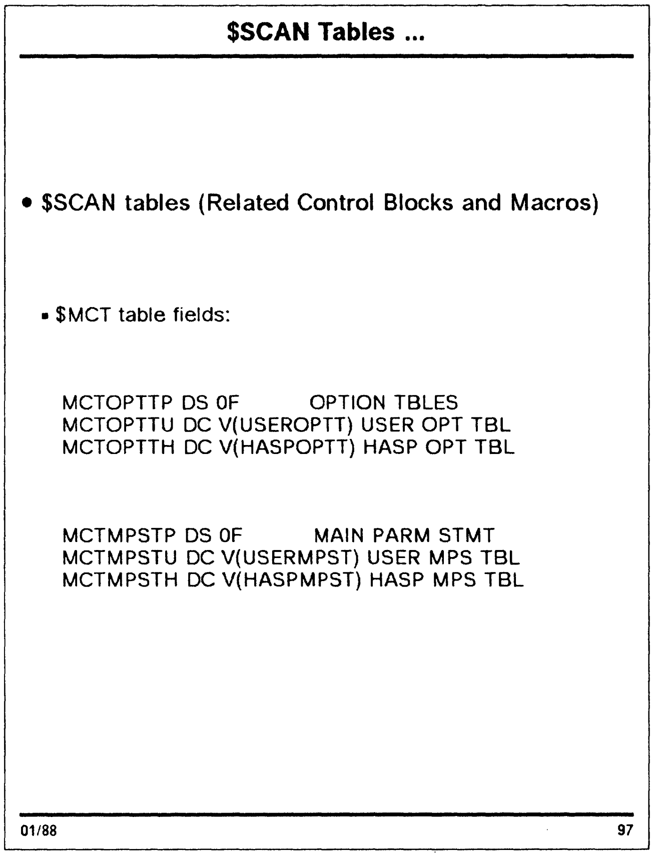
$SCAN is a facility for scanning, from left to right serially, parameter statement input (initialization statements and commands). The $SCAN facility allows the input to match a general grammar, to follow a definition that is table-defined, and to process certain input via exit routines called during the scan.

The $SCAN facility improves upon past initialization statement and command processors in that it insures that all the input to process is valid. If at any point an invalid value is encountered, the scanning is terminated and any changed values are restored to the previous values. The $SCAN facility is not a general syntax checker in that it terminates processing at the first syntax failure. It will not continue processing the statement, flushing out additional errors.

Through the $SCAN facility, you can add, override, or delete installation initialization statements and operands (to a lesser degree this includes JES2 commands). It is recommended that if you want to add commands that you call the $SCAN facility from the command exit (exit 5).

The JES2-defined $SCAN tables reside in HASPSTAB. Some of the following information can be found in the *SPL: JES2 User Modifications and Macros* (LC23-0069).

$SCAN Control Blocks and Macros



$SCAN Tables ...

- $SCAN tables (Related Control Blocks and Macros)

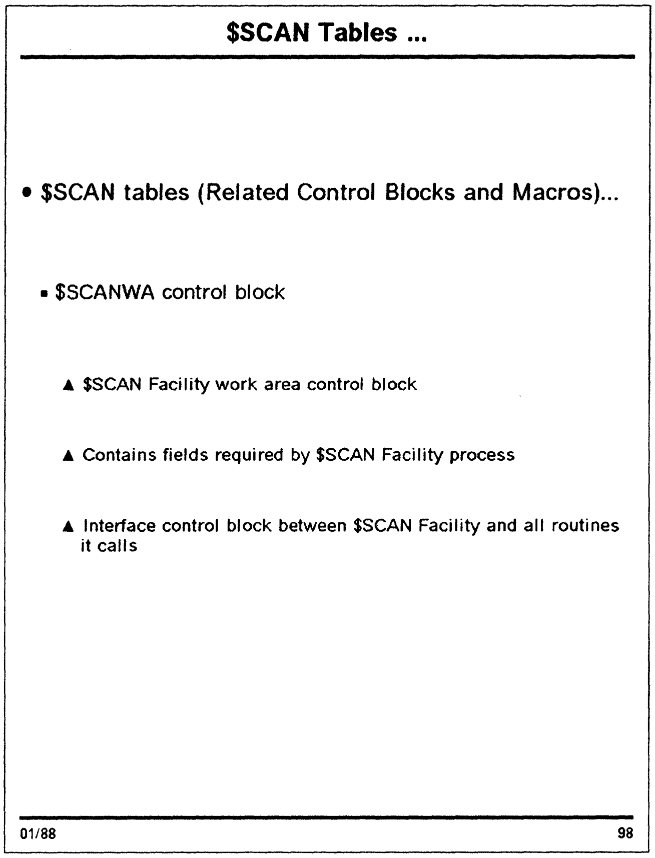
  - $MCT table fields:

```
MCTOPTTP DS 0F          OPTION TBLES
MCTOPTTU DC V(USEROPTT) USER OPT TBL
MCTOPTTH DC V(HASPOPTT) HASP OPT TBL


MCTMPSTP DS 0F          MAIN PARM STMT
MCTMPSTU DC V(USERMPST) USER MPS TBL
MCTMPSTH DC V(HASPMPST) HASP MPS TBL
```

01/88 97

The table pairs that are used to point to the initialization option tables and the initialization statement tables are located in the $MCT.[8] There is one table for the initialization options and one for the initialization statements. The $MCT field for installation tables for initialization options is MCTOPTTU; for the installation tables for the initialization statements the field is MCTMPSTU. If you want to link-edit a table with JES2 you must name the table USEROPTT for initialization options and USERMPST for initialization statements. The installation table would then need to be link-edited with HASJES20. The JES2-defined options and statements are pointed to from the $MCT using the MCT and table names as shown on the foil.

[8] Initialization **option** examples: COLD, NOREQ, WARM. Initialization **statement** examples: PRT, MASDEF.



$SCAN Tables ...

- $SCAN tables (Related Control Blocks and Macros)...
  - $SCANWA control block
    - ▲ $SCAN Facility work area control block
    - ▲ Contains fields required by $SCAN Facility process
    - ▲ Interface control block between $SCAN Facility and all routines it calls

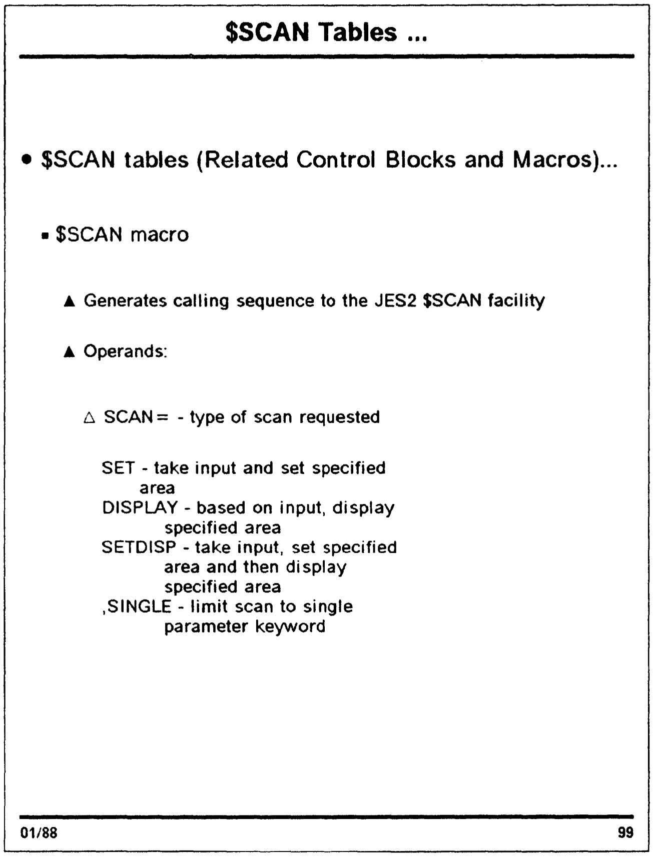
01/88 98

The $SCANWA ($SCAN work area) control block is a work area for the $SCAN request. The $SCAN facility can recursively call itself to process the input passed to it. At each invocation of $SCAN, a new $SCANWA ($SCWA) is obtained to hold information to aid the facility in the processing of the input. The $SCWA is the interface control block between the $SCAN facility and all the routines that it calls, including the pre- and post-scan exit routines and the display exit routine.

These exit routines will be given control pointing to the current $SCWA for this level of $SCAN. There are three forms of $SCWAs:

1. Work $SCWAs
2. Back-up $SCWAs
3. Display $SCWAs

The work $SCWAs are the control blocks that the exit routines will care the most about. We will discuss more on this control block and its relationship with the exit routines later in “A JES2 $SCAN Table” on page 112.



$SCAN Tables ...

- $SCAN tables (Related Control Blocks and Macros)...
  - $SCAN macro
    - ▲ Generates calling sequence to the JES2 $SCAN facility
    - ▲ Operands:
      - △ SCAN= - type of scan requested

        SET - take input and set specified
        area
        DISPLAY - based on input, display
        specified area
        SETDISP - take input, set specified
        area and then display
        specified area
        ,SINGLE - limit scan to single
        parameter keyword

01/88 99

The $SCAN facility is invoked by using the $SCAN macro. This macro generates the calling sequence to the facility and insures that the required data is passed on the call. There are several operands on this macro (all of which are documented in *SPL: JES2 User Modifications and Macros*).

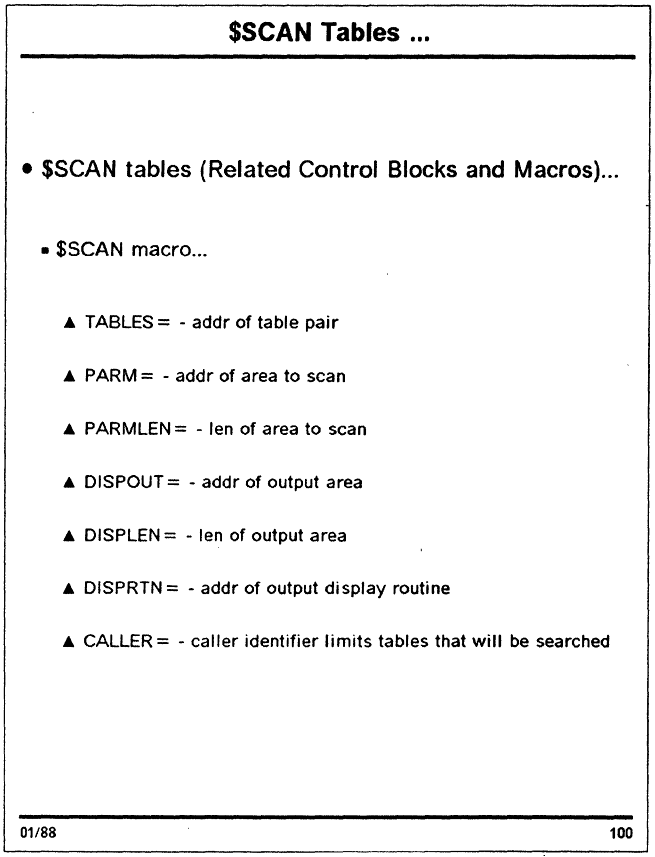
The first operand that we will discuss is the SCAN= operand. This operand indicates the type of request the caller wishes the $SCAN facility to fulfill. There are many types of calls. Three of them are:

- SET indicates that the input should be processed and the validated input should be set into fields specified by the $SCANTAB macro for the input being parsed.

- DISPLAY indicates that the input should be processed and the specified fields should be displayed using attributes specified in the $SCANTAB macro for the input being parsed.

- SETDISP indicates that a set request should be done and then, within the same $SCAN call, a display of the result should be done.

The optional second positional value that you can specify with SET, DISPLAY, or SETDISP is SINGLE. This indicates that only one initialization statement or command (with possibly many operands) may be processed on this invocation of the $SCAN facility.



$SCAN Tables ...

- $SCAN tables (Related Control Blocks and Macros)...
  - $SCAN macro...
    - ▲ TABLES = - addr of table pair
    - ▲ PARM = - addr of area to scan
    - ▲ PARMLEN = - len of area to scan
    - ▲ DISPOUT = - addr of output area
    - ▲ DISPLEN = - len of output area
    - ▲ DISPRTN = - addr of output display routine
    - ▲ CALLER = - caller identifier limits tables that will be searched

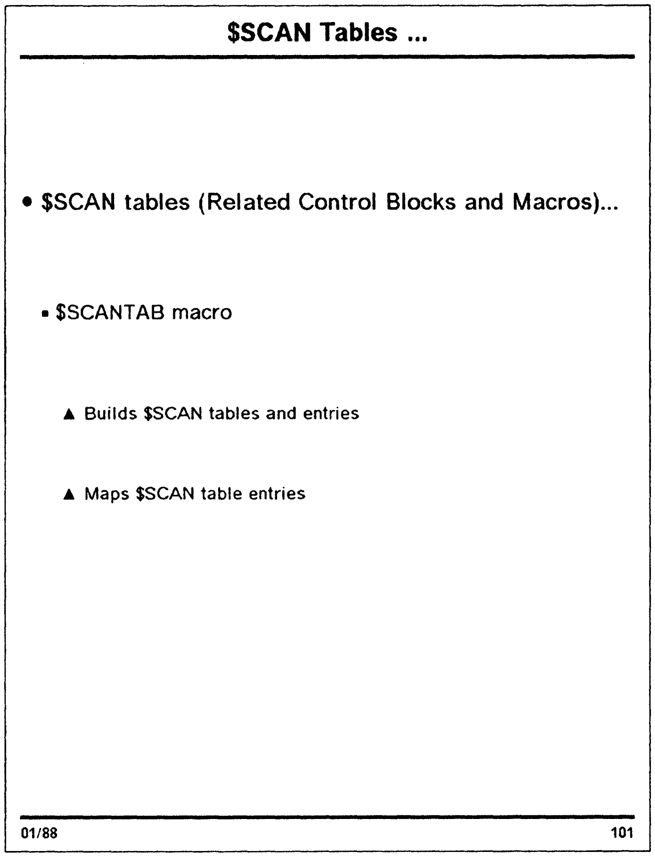
01/88 100

In addition to the SCAN= operand, there is the TABLES= operand. This operand points to the table pair in the $MCT where the $SCAN facility is to start looking for table elements that match the input that is encountered. This need not specify a table pair in the $MCT. However, the only other location where a table pair can reside is in the $UCT.

The PARM= operand points to the input that the $SCAN facility is to process. This parameter input area is required to contain the entire input plus one blanked out byte. The PARMLEN= operand specifies the length of the input plus one for the blanked out byte. Thus, if an 80-byte buffer area holds the input, and the input is only 40 bytes in length (not including the last blanked out byte), then the PARM= will point to the beginning of the buffer area and the PARMLEN= is set to 41 (includes the last blanked out byte).

The DISPOUT= operand points to the area where $SCAN facility generated display text is to be placed. DISPLEN specifies the length of the display area and DISPRTN= specifies the display routine that will get control to display the display area. The $SCAN facility does not issue any sort of display of the specified area; this is up to the routine specified in the DISPRTN field. Also note that DISPOUT=, DISPLEN=, and DISPRTN= are not required. However, if the $SCAN facility encounters an error, the diagnostic message it normally builds (using DISPOUT, DISPLEN, and DISPRTN) will not be built. Therefore, if you want to see diagnostic messages, these three display-oriented operands should be specified even for SCAN=SET calls.

The CALLER= operand is a means to specify or clarify environmental type of information on the $SCAN call. It is possible for you to code two tables for the same keyword that $SCAN should use at different times, one for initialization and one for command time, for example. Thus, a CALLER= operand is provided on the table (as we will show shortly). When the $SCAN facility is invoked with CALLER= specified on the $SCAN macro, this "caller id" is used to match the table element. Therefore, different control blocks can be specified for the same keyword so that the correct location is processed for the sets and displays. We will describe this operand further in "A JES2 $SCAN Table" on page 112.



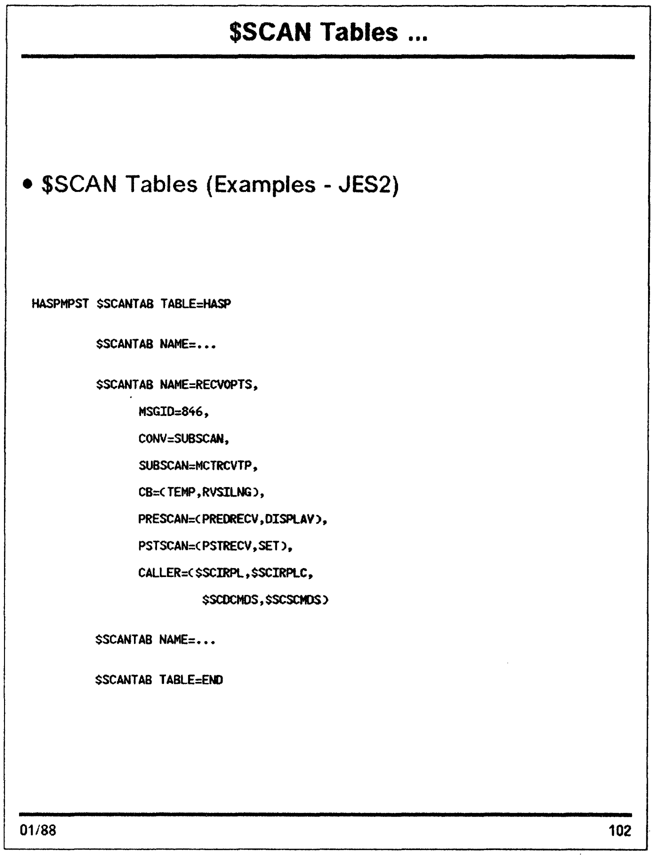
$SCAN Tables ...

- $SCAN tables (Related Control Blocks and Macros)...
  - $SCANTAB macro
    - ▲ Builds $SCAN tables and entries
    - ▲ Maps $SCAN table entries

01/88 101

Just as there were table creating macros for the PCE Tables, DTE Tables, TID Tables, and WS tables, there is a $SCANTAB macro to aid in creating $SCAN tables. This macro builds both the JES2 and installation tables and table elements. This macro also contains the mapping macro for the $SCAN tables and elements. We will describe this macro and its operand more thoroughly shortly.

A JES2 $SCAN Table



$SCAN Tables ...

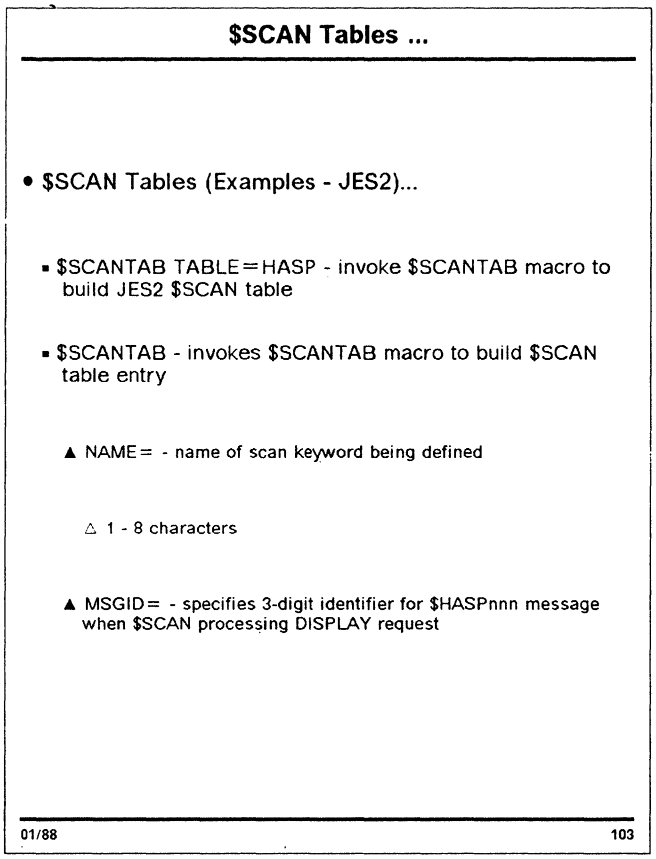
- $SCAN Tables (Examples - JES2)

```
HASPMPST $SCANTAB TABLE=HASP

         $SCANTAB NAME=...

         $SCANTAB NAME=RECVOPTS,
               MSGID=846,
               CONV=SUBSCAN,
               SUBSCAN=MCTRCVTP,
               CB=(TEMP,RVSILNG),
               PRESCAN=(PREDRECV,DISPLAV),
               PSTSCAN=(PSTRECV,SET),
               CALLER=($SCIRPL,$SCIRPLC,
                        $SCDCMDS,$SCSCMDS)

         $SCANTAB NAME=...

         $SCANTAB TABLE=END
```

01/88 102

The figure above illustrates what the $SCAN tables for JES2 main parameter statements (initialization statements) look like. The table element shown represents the table element for the RECVOPTS initialization statement. This is the table that is passed to the $SCAN facility during JES2 initialization to process the initialization statements. Notice that the name of the $SCAN table is HASPMPST, the same as that specified for the V-type address constant in the $MCT.

The following describes each of the operands on this table element as well as some other table elements. However, there are several additional operands that will not be covered. You should review the $SCANTAB macro and *SPL: JES2 User Modifications and Macros* fur a description of all the operands that you may specify on the $SCANTAB table element.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - $SCANTAB TABLE=HASP - invoke $SCANTAB macro to build JES2 $SCAN table
  - $SCANTAB - invokes $SCANTAB macro to build $SCAN table entry
    - ▲ NAME= - name of scan keyword being defined
      - △ 1 - 8 characters
    - ▲ MSGID= - specifies 3-digit identifier for $HASPnnn message when $SCAN processing DISPLAY request

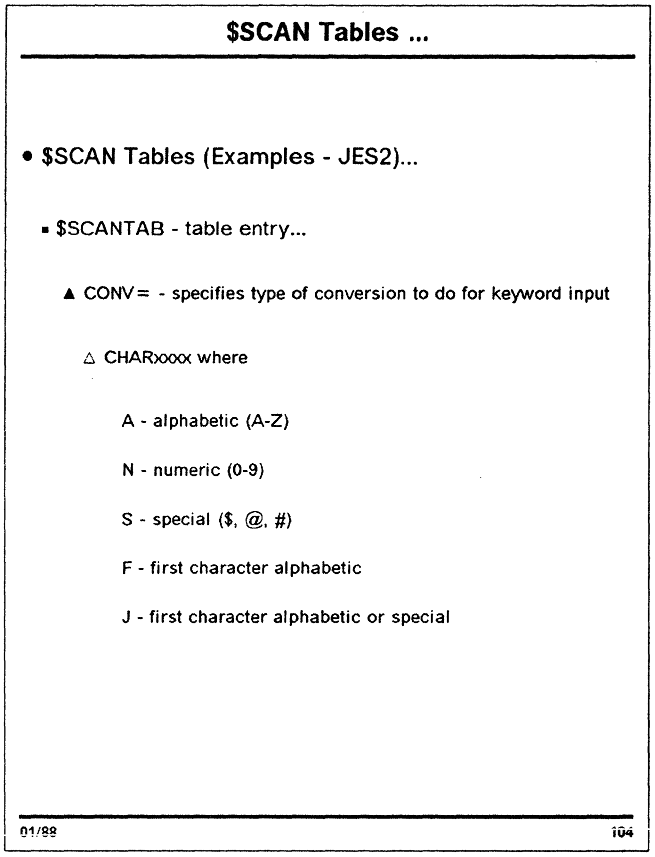
01/88 103

The JES2 $SCAN tables, like the preceding tables, are started by specifying 'TABLE=HASP'. This indicates to JES2 that this table is a JES2 table. You would specify 'TABLE=USER' to indicate that the table is an installation-coded table. Specifying whether it is a JES2 or installation table determines default values for the CALLER and SUBSCAN $SCANTAB operands. We will discuss these operands later. Specifying TABLE=HASP or TABLE=USER is the means JES2 provides to indicate the start of the table (TABLE=START) as discussed in "Concepts" on page 6.

When the $SCANTAB is specified with operands other than TABLE=, the macro generates a table element. In the example above, the table element that is generated is for the RECVOPTS initialization statement.

The NAME= operand specifies the 1-8 character name of the initialization parameter or operand. In its processing, the $SCAN facility scans the input passed to it from left to right, isolating the keywords that it encounters. The isolated keyword is then used as a matching criterion when searching through $SCAN table elements. It is the value specified for this NAME operand that $SCAN uses when attempting to match the isolated keyword.

The MSGID operand specified a three-digit message identifier that is appended to the end of $HASP to use for display requests involved with this keyword. This operand is only honored at the highest level of the keyword, the initialization statement name. In the example, for the initialization statement RECVOPTS, the message identifier that is used to respond to display requests is 846. Thus, the message identifier would be: $HASP846.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - $SCANTAB - table entry...
    - ▲ CONV= - specifies type of conversion to do for keyword input
      - △ CHARxxxx where

        A - alphabetic (A-Z)

        N - numeric (0-9)

        S - special ($, @, #)

        F - first character alphabetic

        J - first character alphabetic or special

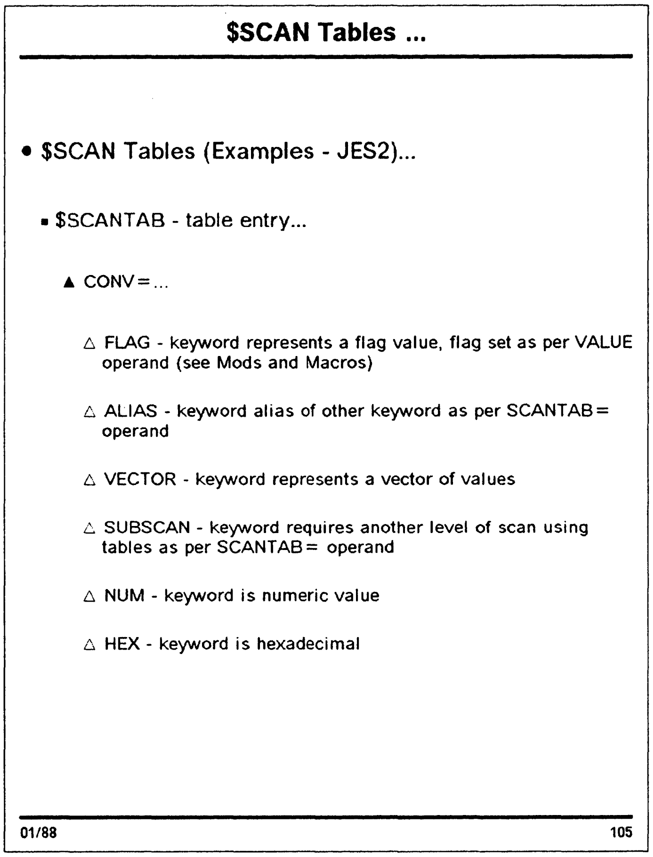
01/88 104

The conversion operand specifies the type of conversion to do with the keyword input. There are several valid values that this operand can take. The first is CHARxxxx. CHARxxxx specifies what the valid characters are that may be specified in the input, where xxxx indicates five valid types:

1. A - indicates that the input must be alphabetic.
2. N - indicates that the input must be numeric.
3. S - indicates that the input must be a special character.
4. F - indicates that the first character in the input must be alphabetic.
5. J - indicates that the first character in the input must be alphabetic or special.

Therefore, in the following examples:

- CHARJNAS - indicates that the first character in the input must be alphabetic or special and the rest of the input can be alphabetic, numeric, or special character (e.g., A$$$89A).
- CHARFNA - indicates that the first character in the input must be alphabetic and the rest of the input can be numeric or alphabetic. Special character input is not permitted. (e.g., A998AB99)
- CHARA - indicates that only alphabetic input is permitted.
- CHARN - indicates that only numeric input is permitted.
- etc.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - $SCANTAB - table entry...
    - ▲ CONV=...
      - △ FLAG - keyword represents a flag value, flag set as per VALUE operand (see Mods and Macros)
      - △ ALIAS - keyword alias of other keyword as per SCANTAB= operand
      - △ VECTOR - keyword represents a vector of values
      - △ SUBSCAN - keyword requires another level of scan using tables as per SCANTAB= operand
      - △ NUM - keyword is numeric value
      - △ HEX - keyword is hexadecimal

01/88 105

CONV=FLAG indicates that the input is a flag value. This value is then processed as the VALUE= operand indicates on the $SCANTAB. If CONV=FLAG is specified, the VALUE= operand is required. See *SPL: JES2 User Modifications and Macros* for additional information.

CONV=ALIAS indicates that this keyword is an alias name of a real keyword. The $SCANTAB table element that describes the real keyword is pointed to by the SCANTAB= operand on this alias $SCANTAB table element. This is useful for creating alternate names for initialization statements or operands. JES2 used this alias capability with the PRINTER, PRINTR, PRT initialization statements in releases at the 2.2.0 level and previous.

CONV=VECTOR indicates that the input represents a vector or list of input. The $SCANTAB(s) that describes this list of input is pointed to from the SCANTAB= operand. An example of vector input is:

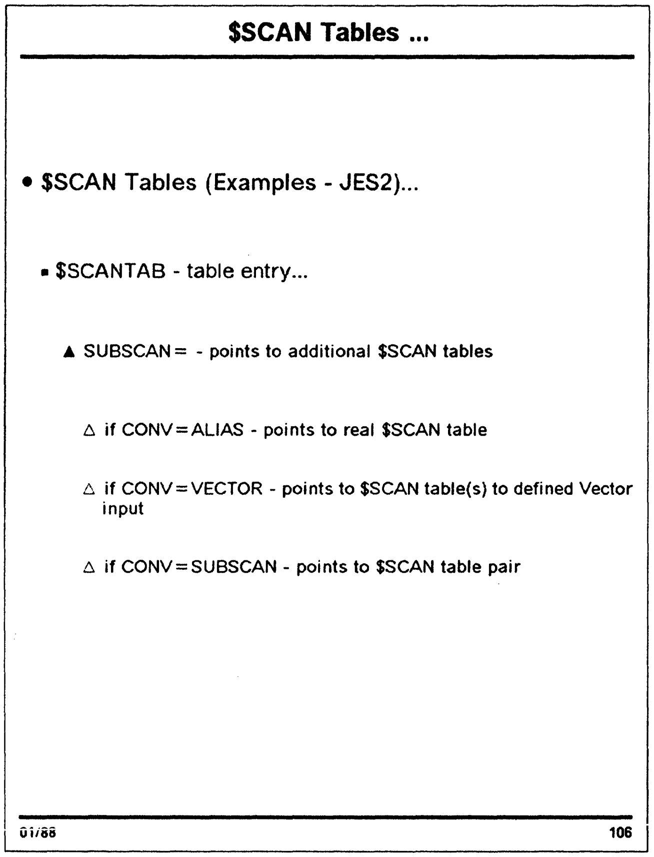
```
VOL=(SPOOL1,SPOOL2,SPOOL3)
```

CONV=SUBSCAN indicates that in order to process the rest of the input, the $SCAN facility must issue a subscan or a recursive $SCAN call. The SCANTAB= operand, in this instance, points to a $SCAN table pair (in the $MCT in JES2).

CONV=NUM indicates that the input is numeric in nature. The difference between this value and CHARN is that with this value the number is converted to hexadecimal; with CHARN, the value is character in format.

CONV=HEX indicates that the input is hexadecimal (e.g., 13EF3A).

In the JES2 example, the CONV=SUBSCAN indicates that processing the rest of the RECVOPTS initialization statement after the RECVOPTS keyword is isolated will require a recursive $SCAN call. The SUBSCAN= operand points to the $SCAN table pair that $SCAN uses to process the operands of the RECVOPTS initialization statement.



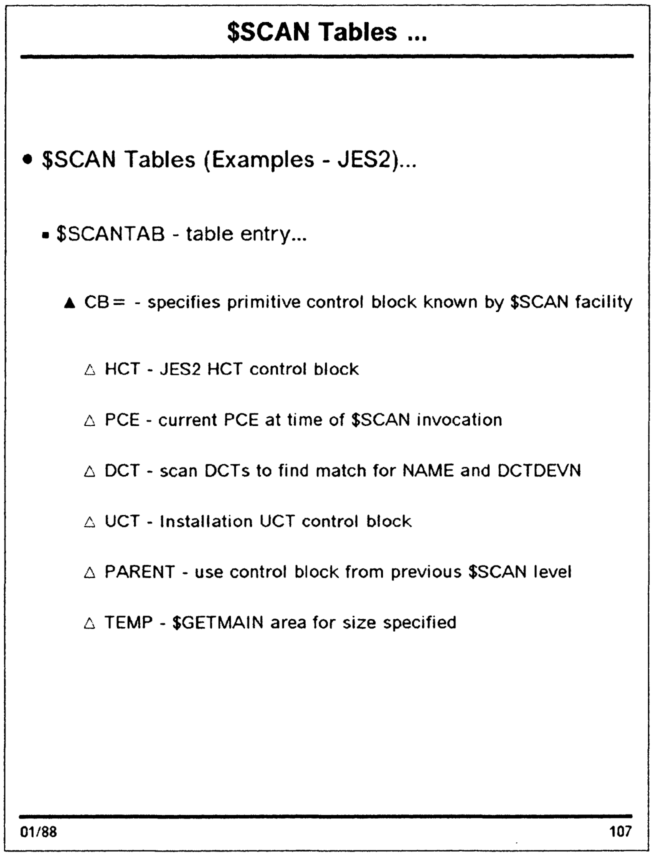
$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - $SCANTAB - table entry...
    - ▲ SUBSCAN= - points to additional $SCAN tables
      - △ if CONV=ALIAS - points to real $SCAN table
      - △ if CONV=VECTOR - points to $SCAN table(s) to defined Vector input
      - △ if CONV=SUBSCAN - points to $SCAN table pair

01/88 106

As has been stated earlier, the SUBSCAN= operand points to additional $SCAN table elements dependent upon the value of the CONV= operand. If CONV=ALIAS, the SUBSCAN operand points to a $SCAN table element that contains the "real" keyword. If CONV=VECTOR, then the SUBSCAN operand points to a table of Vector $SCAN tables. If CONV=SUBSCAN, the SUBSCAN operand points to a $SCAN table pair. For JES2 CONV=SUBSCAN, the SUBSCAN operand points to a table pair in the $MCT. In the installation table, the SUBSCAN operand would default to point to a table pair in the $UCT. With the MVS/SP JES2 2.2.0 release, the SUBSCAN operand can point to a table pair located anywhere (A-type address constant or V-type address constant).

In the JES2 example, the table pair that $SCAN uses to process the operands of the RECVOPTS initialization statement is contained in the $MCT at label MCTRCVTP.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - ▪ $SCANTAB - table entry...
    - ▲ CB= - specifies primitive control block known by $SCAN facility
      - △ HCT - JES2 HCT control block
      - △ PCE - current PCE at time of $SCAN invocation
      - △ DCT - scan DCTs to find match for NAME and DCTDEVN
      - △ UCT - Installation UCT control block
      - △ PARENT - use control block from previous $SCAN level
      - △ TEMP - $GETMAIN area for size specified

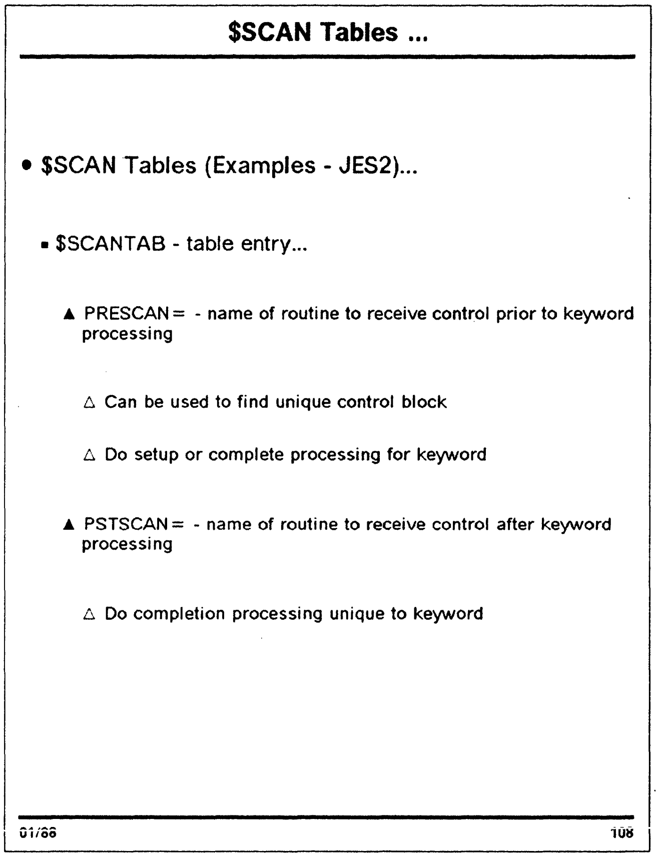
01/88 107

The CB= operand specifies the primitive control block that the $SCAN facility is to use to process this $SCAN request. The $SCAN facility is set up to know a small number of basic control blocks. These control blocks are:

1. HCT - the JES2 HCT control block.
2. PCE - the current Processor Control Element at the time of the $SCAN invocation.
3. DCT - a Device Control Table (DCT). This control block is found by calling the $DCTDYN services which scans the DCTs comparing the NAME and DCTDEVN fields for a match.
4. UCT - the User Control Table (UCT) control block.

The $SCAN facility can also be told to use the control block that was found at the previous level of $SCAN processing. Also, the facility will obtain a temporary area that can be used as a new control block. This temporary area is freed upon completing this $SCAN request, so you will have to code a POST $SCAN exit to $GETMAIN a permanent control block and copy the temporary into it. If CB=TEMP is specified, a second positional operand must be specified that states the size of the temporary control block to obtain. In the JES2 example, CB=(TEMP,RVSILNG), a temporary control block is obtained that is RVSILNG in length.

Besides identifying these basic control blocks, the $SCAN facility can be told to do control block indirection. Control block indirection is the ability to step from basic control blocks through a series of other control blocks to find the control block to use to process a request. See the CBIND= operand on the $SCANTAB macro in *SPL: JES2 User Modifications and Macros.*



$SCAN Tables ...

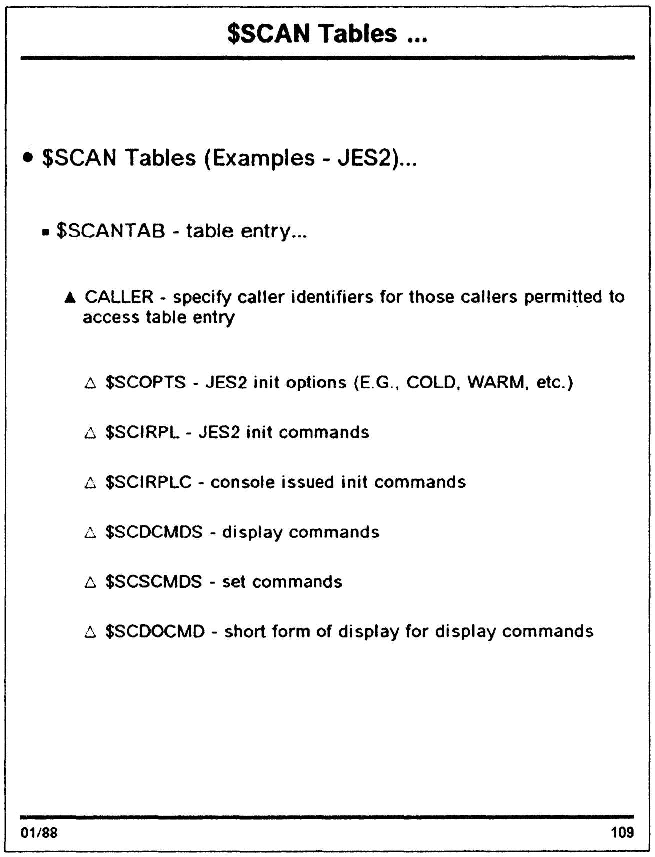
- $SCAN Tables (Examples - JES2)...
  - $SCANTAB - table entry...
    - ▲ PRESCAN= - name of routine to receive control prior to keyword processing
      - △ Can be used to find unique control block
      - △ Do setup or complete processing for keyword
    - ▲ PSTSCAN= - name of routine to receive control after keyword processing
      - △ Do completion processing unique to keyword

01/88 108

The PRESCAN= operand names a routine which will receive control prior to keyword processing. The routine address is resolved with a V-type address constant if it is determined that the routine is not in the same module as the $SCAN table element that references it. This pre-scan exit routine is given control to do unique processing to find control blocks or do setup to let the $SCAN facility complete its processing. It can also do all the processing for the keyword and pass an indicator that the $SCAN facility is finished with this keyword.

The PSTSCAN= operand names a routine that will receive control upon completing keyword processing. This routine address is resolved with a V-type address constant if it is determined that the routine is not in the same module as the $SCAN table element that references it. This post-scan exit routine is given control to do cleanup processing related to resources that may have been obtained by a pre-scan exit routine. If CB=TEMP was specified, this routine can obtain a permanent control block to place the result of the $SCAN.

We will discuss more about pre- and post-scan exits later. In the JES2 example, a pre-scan exit routine was specified with a second positional operand of DISPLAY. This means that the pre-scan exit routine PREDRECV will receive control only for DISPLAY requests to the $SCAN facility. The post-scan exit routine PSTRECV will only receive control for SET requests since the second positional operand indicates SET.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - $SCANTAB - table entry...
    - ▲ CALLER - specify caller identifiers for those callers permitted to access table entry
      - △ $SCOPTS - JES2 init options (E.G., COLD, WARM, etc.)
      - △ $SCIRPL - JES2 init commands
      - △ $SCIRPLC - console issued init commands
      - △ $SCDCMDS - display commands
      - △ $SCSCMDS - set commands
      - △ $SCDOCMD - short form of display for display commands

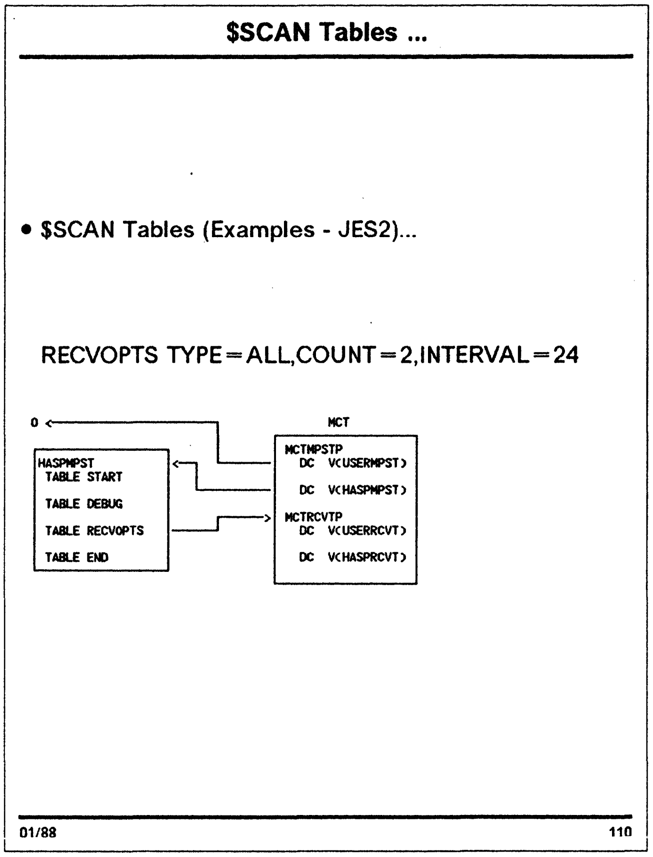
01/88 109

As was described on the $SCAN macro, the CALLER= operand is a way to indicate an environmental influence over what $SCAN table element to choose to process a keyword. The CALLER= operand on the $SCAN table element indicates what caller identifiers may access this table element.

In the JES2 example, the valid callers to access this table include:

1. JES2 initialization statement processing,
2. JES2 initialization display and set requests from the console,
3. display commands, and
4. set commands.

In order to access this table the CALLER= operand on the $SCAN macro must be one of either $SCIRPL, $SCIRPLC, $SCDCMDS, or $SCSCMDS.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...

RECVOPTS TYPE=ALL,COUNT=2,INTERVAL=24

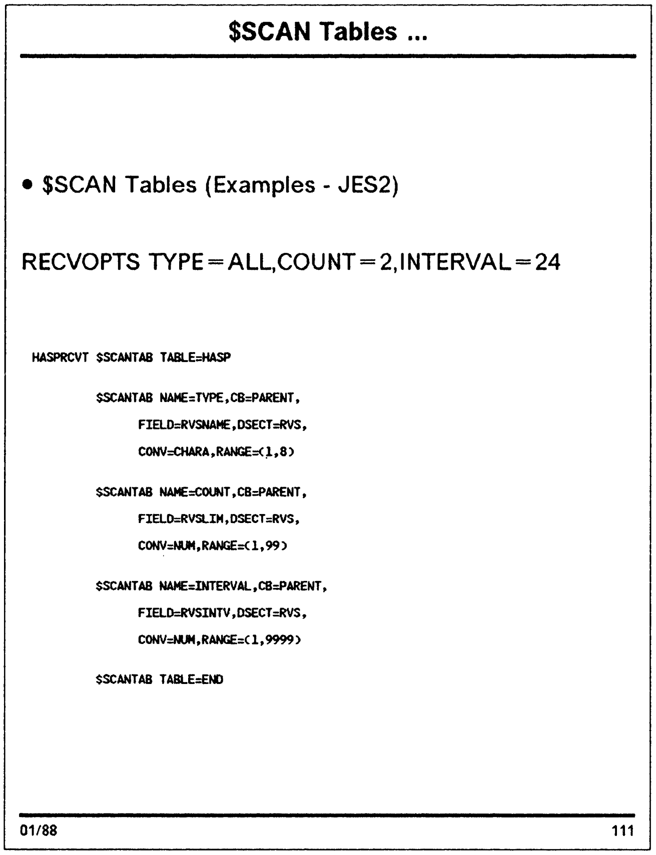
```
0 <
                           MCT
HASPMPST                   MCTMPSTP
  TABLE START                DC   V(USERMPST)
                             DC   V(HASPMPST)
  TABLE DEBUG
                           MCTRCVTP
  TABLE RECVOPTS             DC   V(USERRCVT)
                             DC   V(HASPRCVT)
  TABLE END
```

01/88 110

With the processing that has occurred so far, the $SCAN facility has taken the initialization statement described above, isolated the RECVOPTS keyword, and found the $SCAN table element for this keyword. This table element has told the $SCAN facility:

1. for display requests, use the message identifier 846 (MSGID=);
2. to complete processing for the operands on the statement a subscan (recursive $SCAN) call must be done (CONV=);
3. to use the table pair located at label MCTRCVTP in the $MCT (SUBSCAN=);
4. to obtain a temporary control block that is RVSILNG in length (CB=);
5. before processing the RECVOPTS keyword for display requests, call the PREDRECV prescan exit routine (PRESCAN=);
6. after processing the RECVOPTS keyword for set requests, call the PSTRECV post-scan exit routine (PSTSCAN=);
7. if the caller of the $SCAN facility is not from initialization or command time, don't use this table (CALLER=).

Now in order to process the operands of the RECVOPTS initialization statement, $SCAN uses the tables at HASPRCVT.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)

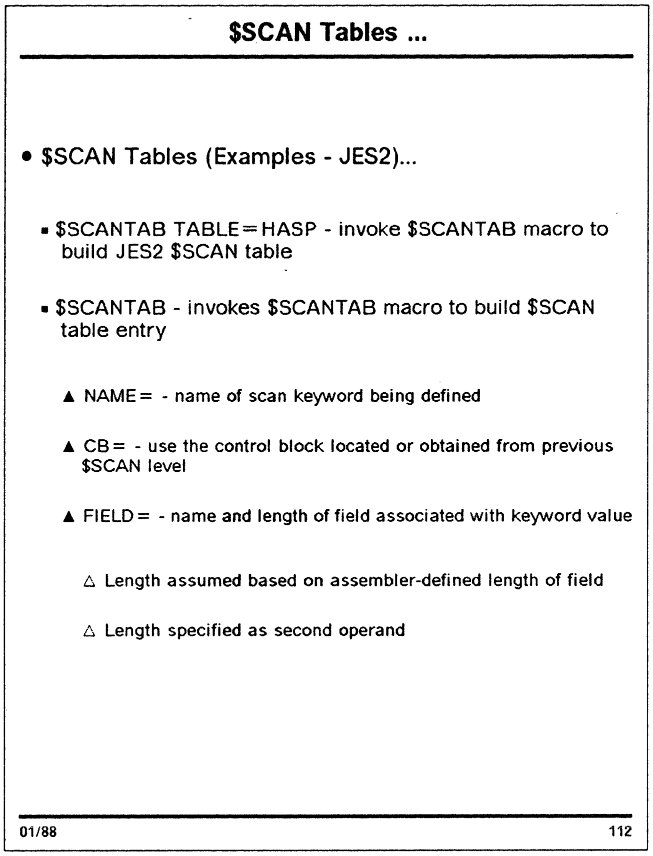
RECVOPTS TYPE=ALL,COUNT=2,INTERVAL=24

```
HASPRCVT $SCANTAB TABLE=HASP

         $SCANTAB NAME=TYPE,CB=PARENT,
               FIELD=RVSNAME,DSECT=RVS,
               CONV=CHARA,RANGE=(1,8)

         $SCANTAB NAME=COUNT,CB=PARENT,
               FIELD=RVSLIM,DSECT=RVS,
               CONV=NUM,RANGE=(1,99)

         $SCANTAB NAME=INTERVAL,CB=PARENT,
               FIELD=RVSINTV,DSECT=RVS,
               CONV=NUM,RANGE=(1,9999)

         $SCANTAB TABLE=END
```

01/88 111

The $SCAN tables that are pointed to by the RECVOPTS table element are shown above. These tables describe the valid inputs for the RECVOPTS operands and to show where the input must be placed and how it should be converted.

In order to fully explain what these table are doing, we will describe each operand below.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - $SCANTAB TABLE=HASP - invoke $SCANTAB macro to build JES2 $SCAN table
  - $SCANTAB - invokes $SCANTAB macro to build $SCAN table entry
    - ▲ NAME= - name of scan keyword being defined
    - ▲ CB= - use the control block located or obtained from previous $SCAN level
    - ▲ FIELD= - name and length of field associated with keyword value
      - △ Length assumed based on assembler-defined length of field
      - △ Length specified as second operand

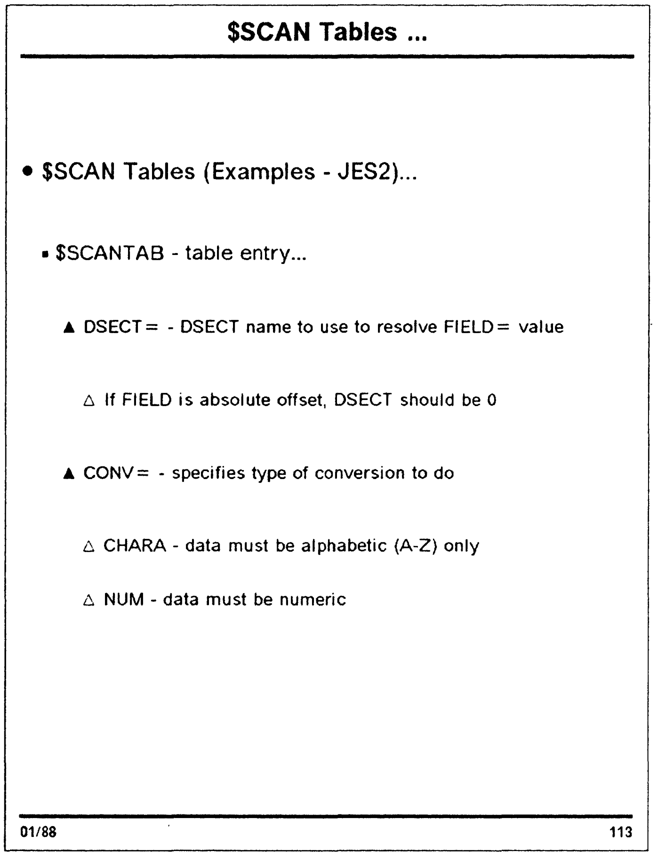
01/88 112

Once again, this is a JES2 table, as signified by TABLE=HASP. If you wish to add or override keywords described in this table, you would code a table USERRCVT with TABLE=USER and fill in the address to the table in the MCT. We will describe this process more later.

The NAME= operand is the same at this second level of scan as it was for the first level of scanning (i.e., for the RECVOPTS keyword). It indicates the one- to eight-character name of the keyword that this table element defines. In this example, there are three keywords defined by this table; TYPE, COUNT, and INTERVAL are defined by the $SCAN table elements.

The CB= operand indicates that the temporary area obtained at the previous level of scanning is to be used as the control block. This is indicated by specifying CB=PARENT.

The FIELD= operand indicates the name and length of the field that is set or displayed for the specified keyword. The length need not be specified if it defaults to its assembler-defined length. Otherwise, specify the length as a second positional operand on this FIELD= operand (e.g., FIELD=(RVSNAME,8) where 8 is the length).



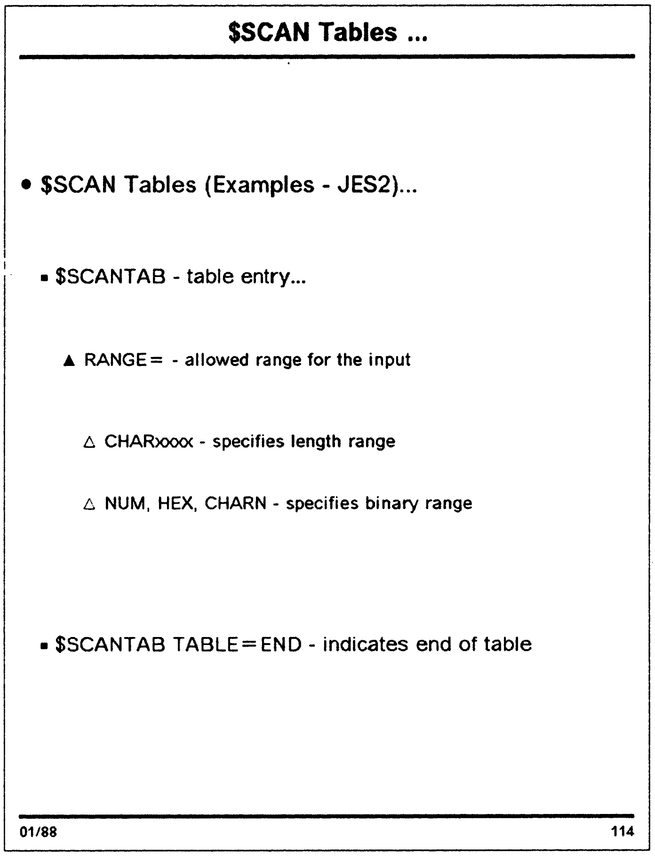
$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - $SCANTAB - table entry...
    - ▲ DSECT= - DSECT name to use to resolve FIELD= value
      - △ If FIELD is absolute offset, DSECT should be 0
    - ▲ CONV= - specifies type of conversion to do
      - △ CHARA - data must be alphabetic (A-Z) only
      - △ NUM - data must be numeric

01/88 113

In order to resolve the FIELD= offset, the DSECT that contains the field must be specified via the DSECT= operand. If the FIELD is an absolute offset, then DSECT=0 should be coded.

As was discussed earlier, the CONV= operand specifies the type of conversion for the input. The two types of conversion in the example are CHARA and NUM. CHARA indicates that the input can only be alphabetical in nature. NUM indicates that the input must be numeric.



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - $SCANTAB - table entry...
    - ▲ RANGE= - allowed range for the input
      - △ CHARxxxx - specifies length range
      - △ NUM, HEX, CHARN - specifies binary range
  - $SCANTAB TABLE=END - indicates end of table

01/88 114

RANGE= indicates the allowed range for the input. If the CONV= operand indicates CHARxxxx, then the RANGE= operand indicates the allowed length of the character input. If the CONV= operand indicates NUM, HEX, or CHARN, then the RANGE= operand indicates the allowed binary range of the input.

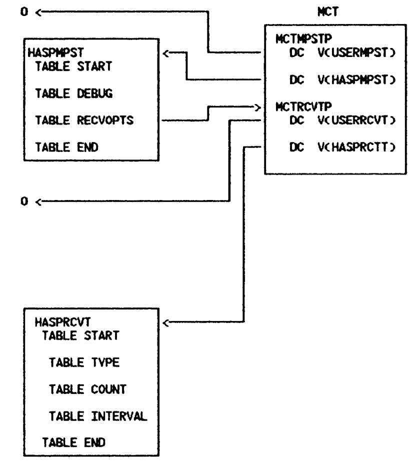
In the JES2 example, the first table entry contains CONV=CHARA and RANGE=(1,8). This means that the character input cannot be greater than eight characters and not less than one character in length. With CONV=NUM and RANGE=(1,99), this means that the numeric input cannot be less than one nor greater than 99.

TABLE=END, of course, indicates the end of this table.

$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...

RECVOPTS TYPE=ALL,COUNT=2,INTERVAL=24



```
HASPMPST
 TABLE START
 TABLE DEBUG
 TABLE RECVOPTS
 TABLE END
```

```
MCTMPSTP
  DC  V(USERMPST)
  DC  V(HASPMPST)
MCTRCVTP
  DC  V(USERRCVT)
  DC  V(HASPRCTT)
```

```
HASPRCVT
 TABLE START
  TABLE TVPE
  TABLE COUNT
  TABLE INTERVAL
 TABLE END
```

01/88 115

At this point in the $SCAN facility processing, the RECVOPTS initialization statement can be fully processed. After finding the table for RECVOPTS and recursively calling itself, the $SCAN facility completed the rest of the initialization statement by isolating the next keyword (TYPE), finding the $SCAN table element that matched this keyword, and processing the input to this operand.

The table element indicated that the input must be alphabetic, which ALL is, and it must not be less than one character and not greater than eight characters in length. Since the input passes these checks, the input is put into the temporary control block using the RVSNAME field in the DSECT RVS.

After completing the TYPE operand, the $SCAN facility continued with the COUNT operand. This was done by isolating the keyword, finding the $SCAN table element that matched this keyword, and processing the input to this operand.

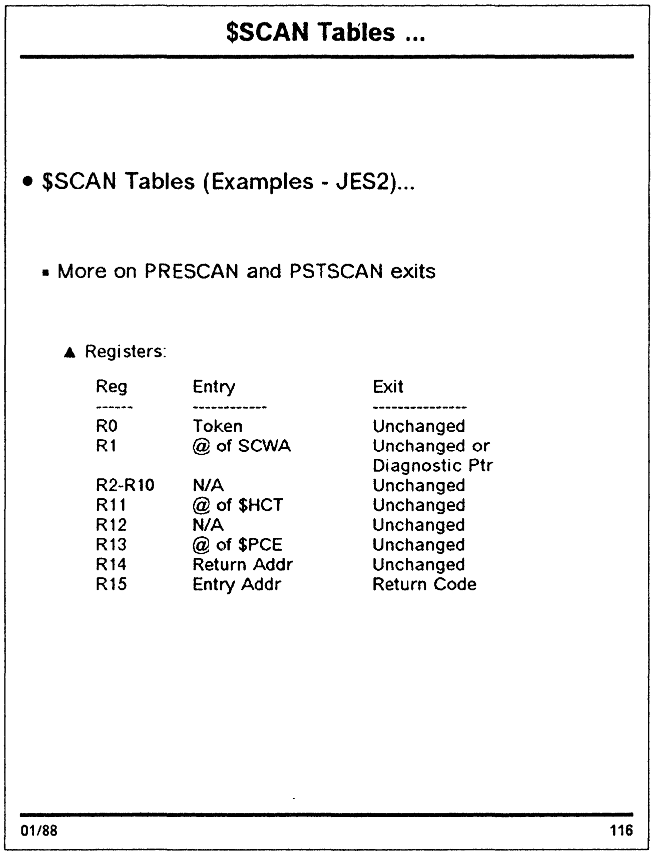
The table element indicated that the input must be numeric, which COUNT is, and it must not be less than one or greater than 99. Since the input passes these checks, the input is put into the temporary control block using the RVSLIM field in the DSECT RVS.

After completing the COUNT operand, the $SCAN facility continued with the INTERVAL operand. This was done by isolating the keyword, finding the $SCAN table element that matched this keyword, and processing the input to this operand.

The table element indicated that the input must be numeric, which INTERVAL is, and it must not be less than one or greater than 9999. Since the input passes these checks, the input is put into the temporary control block using the RVSINTV field in the DSECT RVS.

At this point processing is done with all of the operands of the RECVOPTS initialization statement. Thus $SCAN exits this level of $SCAN processing and returns to the first (RECVOPTS) level of processing. Since the request was a SET request and since the RECVOPTS table element indicated that a post-scan exit routine must be given control, the post-scan exit PSTRECV routine is given control to do some specific final processing (like obtaining a permanent control block, for example).

More about Pre-Scan and Post-Scan Exits



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - More on PRESCAN and PSTSCAN exits
    - ▲ Registers:

Reg	Entry	Exit
R0	Token	Unchanged
R1	@ of SCWA	Unchanged or Diagnostic Ptr
R2-R10	N/A	Unchanged
R11	@ of $HCT	Unchanged
R12	N/A	Unchanged
R13	@ of $PCE	Unchanged
R14	Return Addr	Unchanged
R15	Entry Addr	Return Code

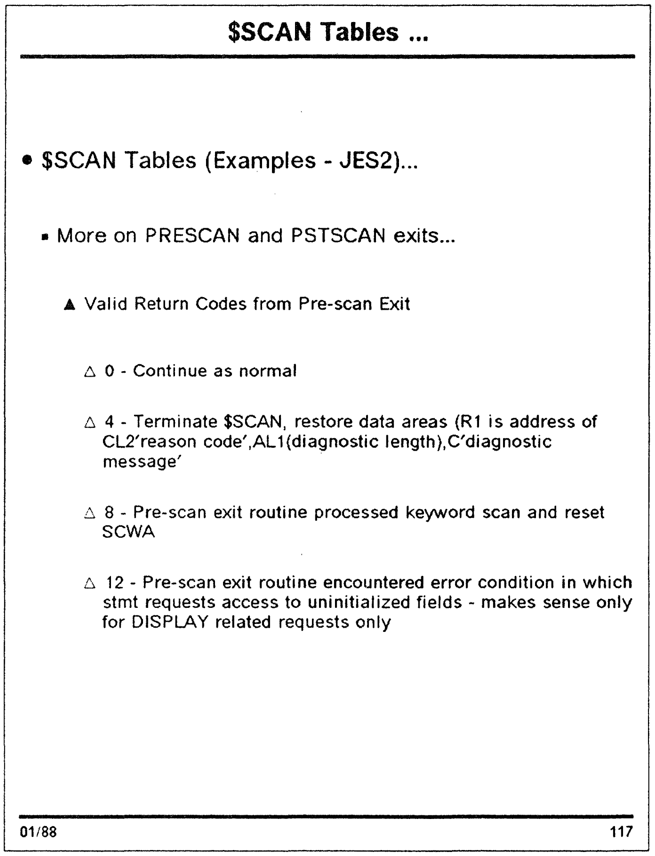
01/88 116

Above are the register conventions that pre- and post-scan exit routines can expect on entry. Before JES2 release 2.2.0, register 0 was undefined upon entry; with 2.2.0, register 0 has the token specified via TOKEN= on the $SCAN invocation. Register 1 contains the address of the $SCAN work area ($SCWA). From this work area, the exit routine is capability of obtaining the values of key fields. We will document these fields shortly.

Registers 11, 13, 14, and 15 follow the normal JES2 $EXIT type of register conventions:

- Register 11 contains the address of the $HCT.
- Register 13 contains the address of the PCE (Processor Control Element) for the processor in control at the time of the $SCAN request.
- Register 14 contains the return address.
- Register 15 contains the entry point address of the exit routine.

The registers that may be set from these pre- and post-scan exit routines also follow normal conventions. Register 1 can contain the address of a diagnostic phrase. This register is interrogated for this diagnostic phrase if the return code in register 15 indicates so. Next, we describe the valid values for register 15.



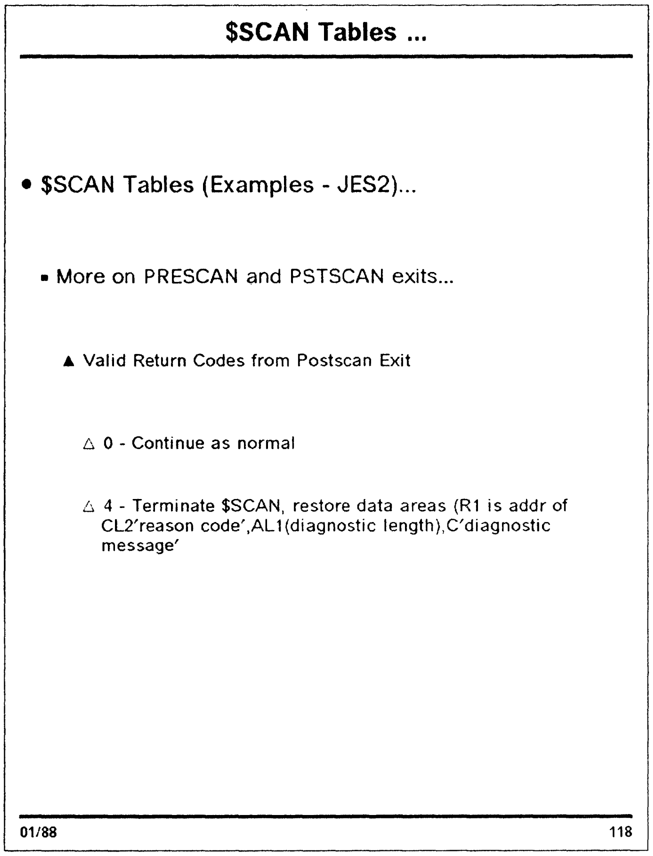
$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - More on PRESCAN and PSTSCAN exits...
    - ▲ Valid Return Codes from Pre-scan Exit
      - △ 0 - Continue as normal
      - △ 4 - Terminate $SCAN, restore data areas (R1 is address of CL2'reason code',AL1(diagnostic length),C'diagnostic message'
      - △ 8 - Pre-scan exit routine processed keyword scan and reset SCWA
      - △ 12 - Pre-scan exit routine encountered error condition in which stmt requests access to uninitialized fields - makes sense only for DISPLAY related requests only

01/88 117

From the pre-scan exit routine, the routine can specify four different return codes in register 15. Remember that pre-scan exit routines are given control after isolating the keyword from the input and finding the appropriate $SCAN table element. Exit routines are given control before any $SCAN facility processing of the keyword.

- The 0 return code indicates that the exit routine successfully completed whatever processing it was expected to do and that the $SCAN facility should continue and process the keyword. This processing may include recursive $SCAN calls.
- The 4 return code indicates that some sort of error was detected and that the $SCAN facility should terminate processing. Part of termination processing for the $SCAN facility is to restore any fields that the $SCAN facility may have altered. Also, the $SCAN facility assumes that register 1 contains the address of a diagnostic phrase. The diagnostic phrase consists of three sections that are assumed contiguous. These sections are:
  - a two-byte reason code (specified in character form - e.g., CL2'43')
  - a one-byte length of a diagnostic message (specified in hex - e.g., AL1(23))
  - a diagnostic message (specified in character form - C'msg text...')
- The 8 return code indicates that the pre-scan exit routine did all the processing for the keyword and that the $SCAN facility should continue with the next keyword at this level.
- The 12 return code indicates that the pre-scan exit routine determined that the field to be altered is not available and that $SCAN processing should be terminated. This return only makes sense for DISPLAY related requests. If a display request is made, for instance, at a time before the item to be displayed has been set, then the display is impossible and the request should be terminated. As part of this termination, the $SCAN facility will issue an internal diagnostic phrase in the format defined just above.



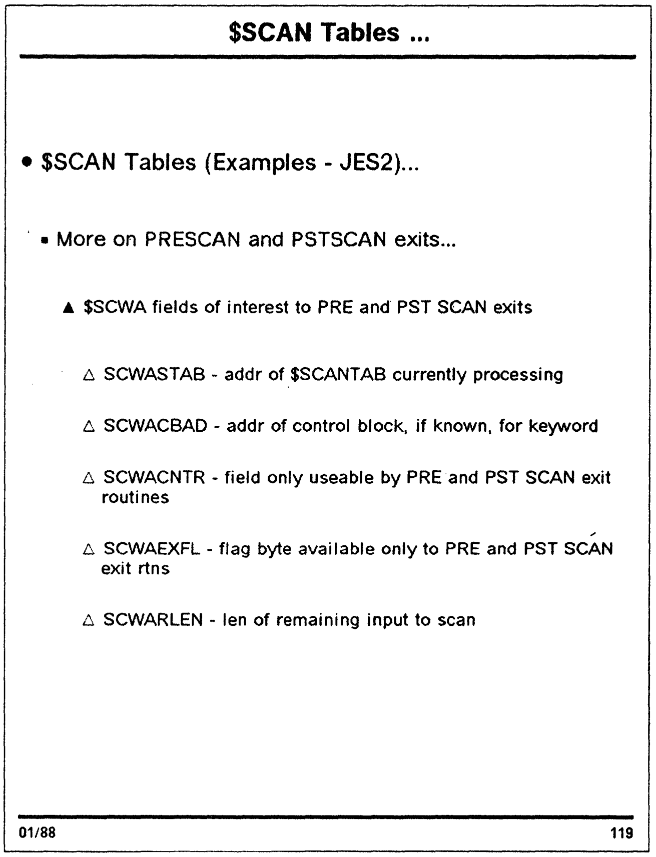
$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - More on PRESCAN and PSTSCAN exits...
    - ▲ Valid Return Codes from Postscan Exit
      - △ 0 - Continue as normal
      - △ 4 - Terminate $SCAN, restore data areas (R1 is addr of CL2′reason code′,AL1(diagnostic length),C′diagnostic message′

01/88 118

From the post-scan exit routine, the routine can specify two different return codes in register 15. Remember that post-scan exit routines are given control after all the processing for the keyword as completed. It is usually taken to "harden" control blocks (obtain permanent copies of the control block).

- The 0 return code indicates that the exit routine successfully completed whatever processing it was expected to do and that the $SCAN facility should continue with the next keyword.
- The 4 return code indicates that some sort of error was detected and that the $SCAN facility should terminate processing. Part of terminate processing for the $SCAN facility is to restore any fields that the $SCAN facility may have altered. Also, the $SCAN facility assumes that register 1 contains the address of a diagnostic phrase. The diagnostic phrase consists of three diagnostic phrase sections that are assumed contiguous. These sections are:
  - a two-byte reason code (specified in character form - e.g., CL2′43′)
  - a one-byte length of a diagnostic message (specified in hex - e.g., AL1(23))
  - a diagnostic message (specified in character form - C′msg text...′)



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - More on PRESCAN and PSTSCAN exits...
    - ▲ $SCWA fields of interest to PRE and PST SCAN exits
      - △ SCWASTAB - addr of $SCANTAB currently processing
      - △ SCWACBAD - addr of control block, if known, for keyword
      - △ SCWACNTR - field only useable by PRE and PST SCAN exit routines
      - △ SCWAEXFL - flag byte available only to PRE and PST SCAN exit rtns
      - △ SCWARLEN - len of remaining input to scan

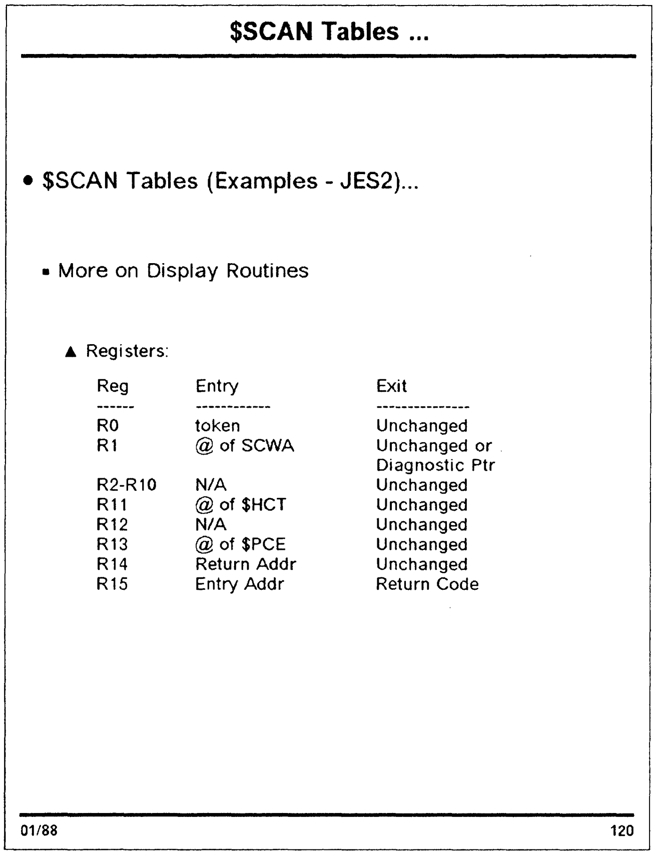
01/88 119

As stated earlier, the $SCAN work area ($SCWA) for the current level of scanning is passed to pre- and post-scan exit routines. There are a few fields that may be of interest to the pre- and post-scan exits. These fields are:

- SCWASTAB - this field contains the address of the $SCAN table element for the current keyword that is being processed.
- SCWACBAD - this field contains the address of the control block that the $SCAN facility determined was specified in the $SCAN table element (after any control block indirection (CBIND) has been applied). This field may be zero only if CONV=SUBSCAN has been specified in the table element. If the pre-scan exit finds this field zero and the CONV= operand is not equal to SUBSCAN, then the pre-scan exit routine should find the appropriate control block and set its address in this field
- SCWACNTR - this is a fullword field that is available to pre- and post-scan routines only. The $SCAN main line facility will not use it. You can use it to save an incremental value within a loop (e.g., PRT(1-8)) or to hold an address that is determined in a pre-scan exit routine.
- SCWAEXFL - this is a flag byte that is reserved for use by pre- and post-scan routines only, like the fullword field SCWACNTR. Currently, the four high-most bits are reserved for JES2 (X'11110000') and the four low bits are reserved for installations (X'00001111').
- SCWARLEN - this is a field that contains the length to scan of the remaining input. Pre- and post-scan exit routines can use it so that if these routines need to do some scanning of their own, they will be able to determine where the ending value is.

The JES2 pre- and post-scanning exit routines are located in module HASPSXIT.

More about Display Routines



$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - More on Display Routines
    - ▲ Registers:

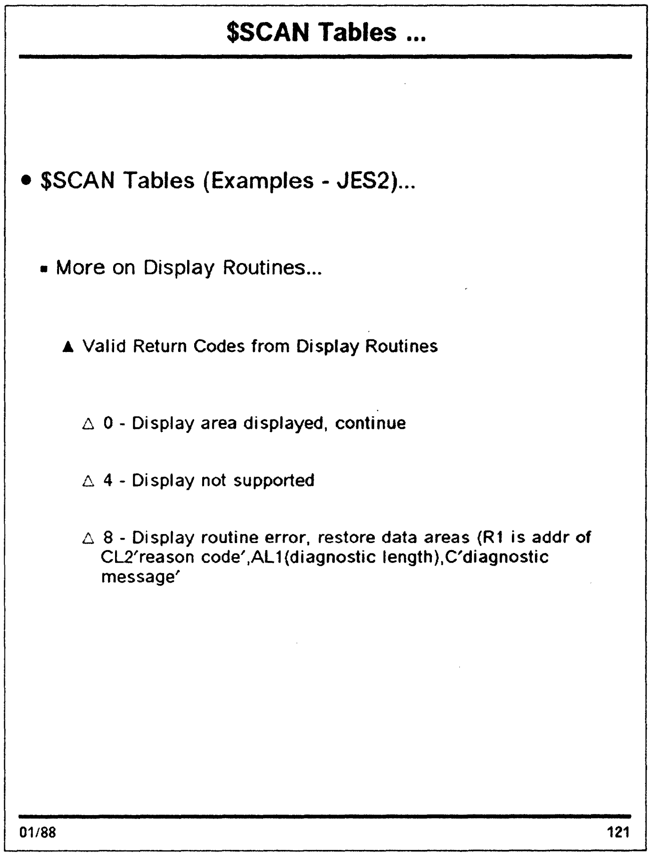
Reg	Entry	Exit
R0	token	Unchanged
R1	@ of SCWA	Unchanged or Diagnostic Ptr
R2-R10	N/A	Unchanged
R11	@ of $HCT	Unchanged
R12	N/A	Unchanged
R13	@ of $PCE	Unchanged
R14	Return Addr	Unchanged
R15	Entry Addr	Return Code

01/88 120

The display exit routine that is specified on the DISPRTN= operand of the $SCAN macro (the invoking macro of the $SCAN facility) has an interface with the $SCAN facility very similar to the pre- and post-scan exit routines. It is called whenever the $SCAN facility determines that a line of text (DISPOUT) is filled. This output area may contain the results of a display request or of a diagnostic error message.

Before JES2 release 2.2.0, register 0 was undefined upon entry; with 2.2.0, register 0 has the token specified via TOKEN= on the $SCAN invocation. Register 1 on entry to the display routine contains the address of the current $SCAN work area ($SCWA). On exit from the display routine, this register can contain a pointer to a diagnostic phrase. We will describe this shortly. Registers 11, 13, 14, and 15 follow the normal JES2 register conventions:

- Register 11 contains the address of the $HCT.
- Register 13 contains the address of the Processor Control Element (PCE) that is in control for the $SCAN request.
- Register 14 contains the return address.
- Register 15 contains the entry address of the display routine. On exit, Register 15 will contain a return code.



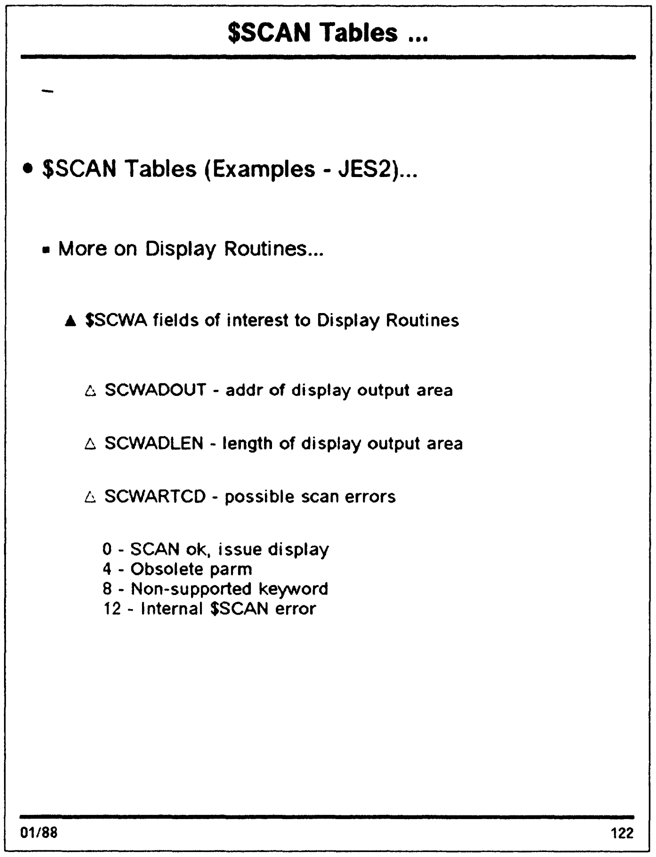
$SCAN Tables ...

- $SCAN Tables (Examples - JES2)...
  - More on Display Routines...
    - ▲ Valid Return Codes from Display Routines
      - △ 0 - Display area displayed, continue
      - △ 4 - Display not supported
      - △ 8 - Display routine error, restore data areas (R1 is addr of CL2′reason code′,AL1(diagnostic length),C′diagnostic message′

01/88 121

The valid return codes from the display routine are:

- 0 - The display area has been displayed. Continue $SCAN facility processing. Note that to continue $SCAN facility processing may be in fact to terminate the facility. This would occur if the display routine was given control to issue a diagnostic message. The following foil will explain this further.
- 4 - The display routine determined that it was not able to issue the display under the current conditions. In this case, the $SCAN facility will throw the current display line information away.
- 8 - The display routine experienced an error in issuing the display. Register 1 contains a pointer to a diagnostic phrase. The diagnostic phrase consists of three diagnostic phrase sections that are assumed contiguous. These areas are:
  - a two-byte reason code (specified in character form - e.g., CL2′43′)
  - a one-byte length of a diagnostic message (specified in hex - e.g., AL1(23))
  - a diagnostic message (specified in character form - C′msg text...′)



$SCAN Tables ...

–

- $SCAN Tables (Examples - JES2)...
  - More on Display Routines...
    - ▲ $SCWA fields of interest to Display Routines
      - △ SCWADOUT - addr of display output area
      - △ SCWADLEN - length of display output area
      - △ SCWARTCD - possible scan errors

        0 - SCAN ok, issue display
        4 - Obsolete parm
        8 - Non-supported keyword
        12 - Internal $SCAN error

01/88 122

On entry to the display routine, the current SCWA ($SCAN work area) is passed in register 1. This SCWA will contain three fields of interest to the display routine:

1. SCWADOUT - will contain the address of the display output area. This is the same output area that was specified on the $SCAN macro call on the DISPOUT= operand.

2. SCWADLEN - will contain the length of the display output area in a half word field. This is the same length that was specified on the $SCAN macro call on the DISPLEN= operand.

3. SCWARTCD - will contain the return code in a halfword field with which the $SCAN facility is currently working. This field will aid the display routine to determine if it has been given control to process the results of a display request or to process a $SCAN facility diagnostic message. The possible return code values are:

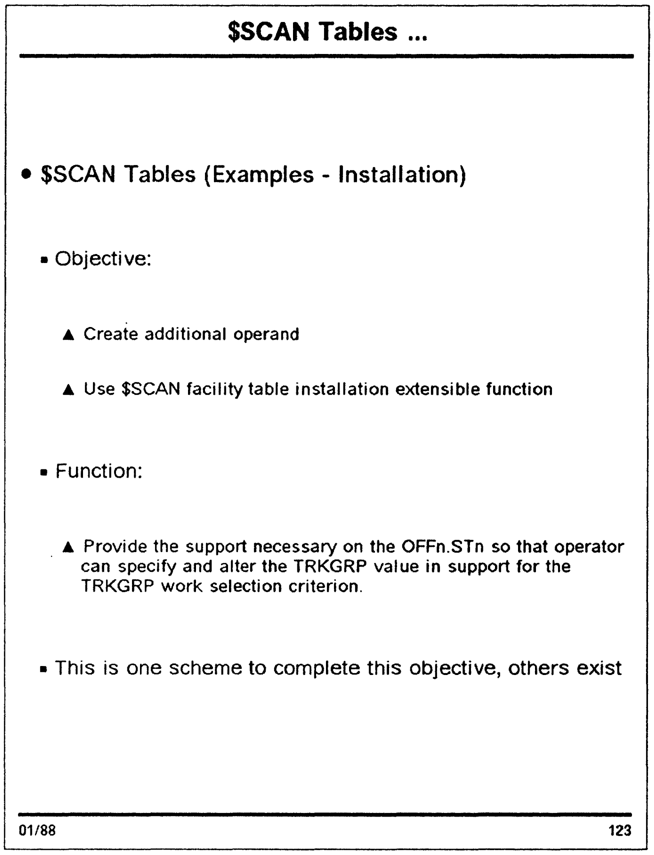
   - 0 - The $SCAN processing is okay, the display routine has been given control to issue the results of a display request.

   - 4 - The $SCAN facility has encountered an obsolete parameter (as determined from the $SCAN table element - see *SPL: JES2 User Modifications and Macros* for the $SCANTAB macro). The display routine has been given control to issue a diagnostic message.

   - 8 - The $SCAN facility has encountered a non-supported keyword. This means that some input was passed to the $SCAN facility and the facility could not locate a $SCAN table element that matched the input. The display routine has been given control to issue a diagnostic message.

   - 12 - The $SCAN facility has encountered an internal $SCAN error situation and is terminating the $SCAN request. The display routine has been given control to issue a diagnostic message.

An Installation $SCAN Table



$SCAN Tables ...

- $SCAN Tables (Examples - Installation)
  - Objective:
    - ▲ Create additional operand
    - ▲ Use $SCAN facility table installation extensible function
  - Function:
    - ▲ Provide the support necessary on the OFFn.STn so that operator can specify and alter the TRKGRP value in support for the TRKGRP work selection criterion.
  - This is one scheme to complete this objective, others exist

01/88 123

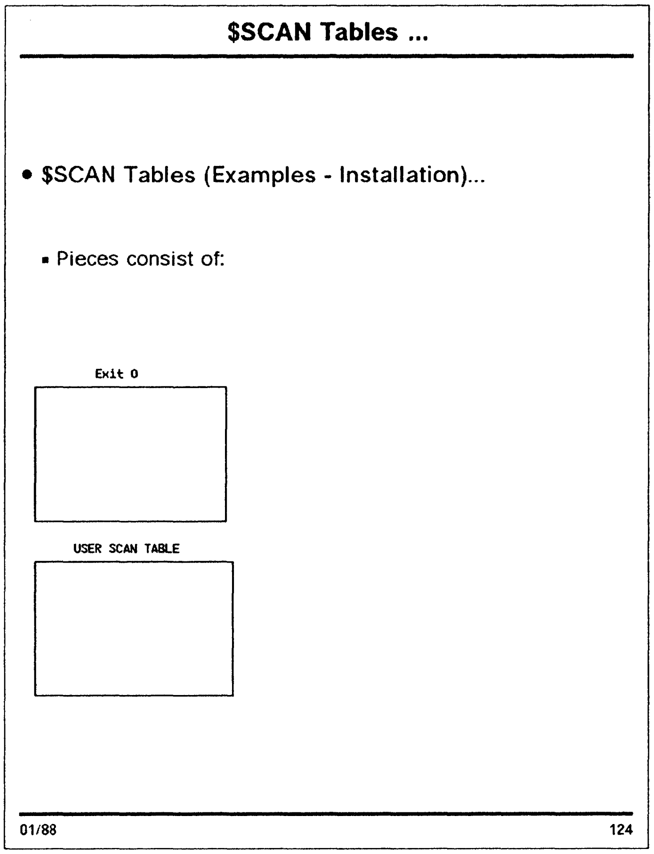
In order to show how you would specify $SCAN tables, the remaining description of the $SCAN tables will step through creating an installation-defined operand to the Offload Sysout Transmitter (OFFn.ST) initialization statement and command.

Objective

Previously in this discussion, a work selection criterion was added to the Offload Sysout Transmitter to allow selecting work based on the amount of spool space that had been allocated to a job. This work selection criterion was called TRkgrp, where only TR need be specified and an alias value of TG could be used. In the work selection table element for this criterion, the value $#GET would use to compare with was the JQE field JQETGNUM and the DCT field DCTUSER0. This example will show how to add an operand to the OFFn.ST initialization statement and command that would permit the operator to set a threshold limit in the DCTUSER0 field.

To achieve this, you will add an installation table element to the OFFn.ST list of operands. The following documents the pieces required, the coding of the table element, and the required code to "plug" the table in. This is one scheme to achieve the statement objective; others do exist.

Required Pieces



$SCAN Tables ...

- $SCAN Tables (Examples - Installation)...

  ▪ Pieces consist of:

Exit 0

USER SCAN TABLE

01/88 124

To achieve the objective, you will need to code two pieces. These pieces are:

1. Exit 0

   As was discussed in "Concepts" on page 6, there are two ways to link the installation table with JES2.

   a. The first of these is to link-edit the installation $SCAN table with the HASJES20 load module. This requires the name of the installation table be USEROSTT. This name was found by searching the $MCT for the table pair that matches the operand list table pair for the OFFn.ST initialization statement.

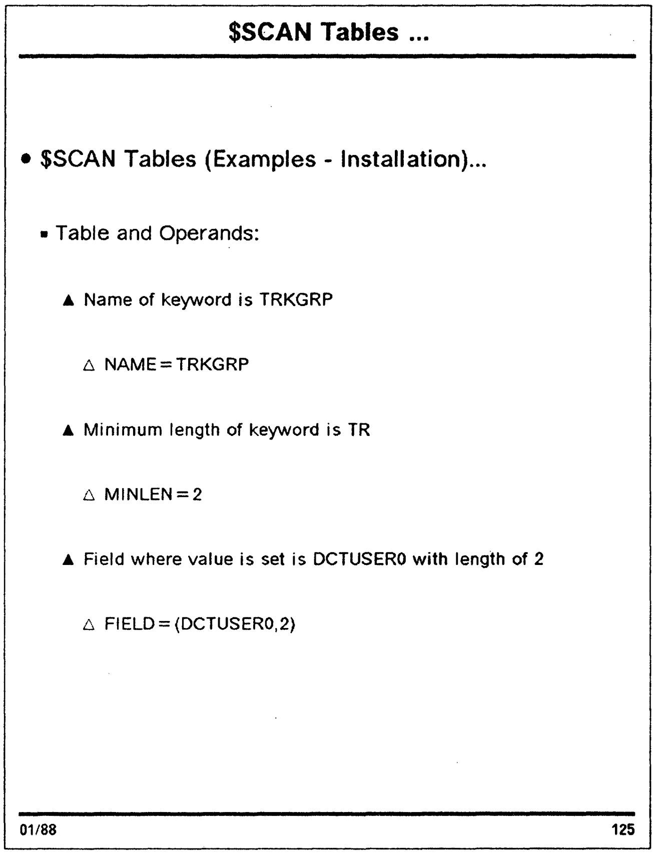
   b. If you do not wish to link-edit the installation $SCAN table USEROSTT, then you must fill in the address of the installation $SCAN table into the $MCT field MCTOSTTU. This is the second method. This method requires that you fill in the address of the table in the $MCT before invoking the OFFLOAD SYSOUT transmitter to access the DCTUSER0 field to initialize it. Depending on when you will use the transmitter, you may fill in the address early in initialization or after JES2 is up and running.

   In this example, you will fill in the address of the $SCAN table early in initialization, specifically in Exit 0. Therefore, you require an Exit 0 that will load your module (if not already loaded) and resolve the address of the table.

2. User $SCAN Table

   You will have to code a $SCAN table which includes the table element for the operand. We will describe the installation table element in a step-wise fashion below.

Coding the Installation $SCAN Table



$SCAN Tables ...

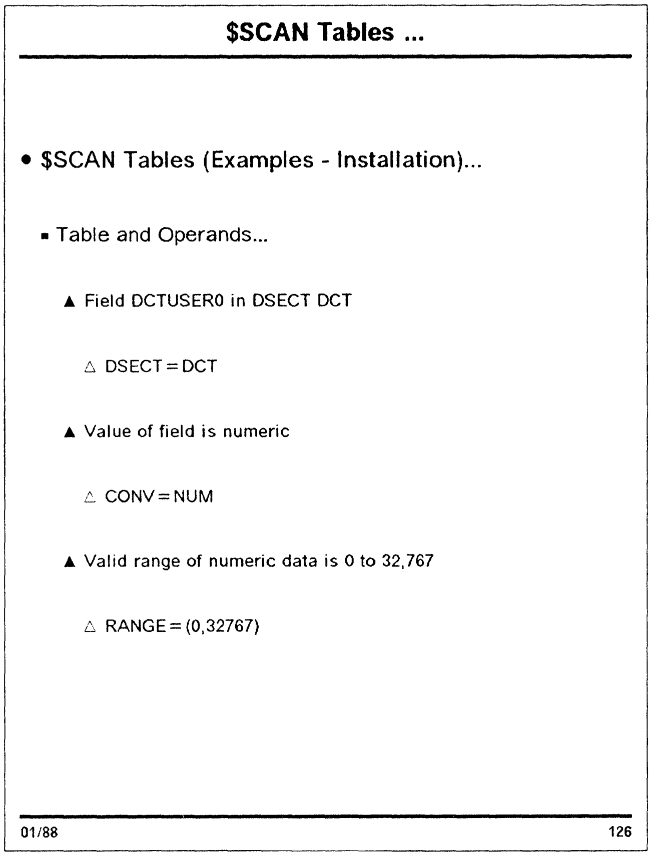
- $SCAN Tables (Examples - Installation)...
  - Table and Operands:
    - ▲ Name of keyword is TRKGRP
      - △ NAME=TRKGRP
    - ▲ Minimum length of keyword is TR
      - △ MINLEN=2
    - ▲ Field where value is set is DCTUSER0 with length of 2
      - △ FIELD=(DCTUSER0,2)

01/88 125

The name of the Offload SYSOUT transmitter operand is TRKGRP to match the work selection criterion created in the Work Selection Installation example. This is the name that the operator will specify to set the threshold value. Therefore, the NAME= operand is set to TRKGRP.

Like the work selection criterion, the minimum length that the operator will have to specify for this operand is two characters. Therefore, the MINLEN= operand is set to 2.

The field that was used as the device control block for the work selection criteria was DCTUSER0. Therefore, this $SCAN initialization operand will have to set this field. Since the JQETGNUM field that is being used as the comparing operand is only two bytes, the length of the DCTUSER0 full word field is only two bytes. This is specified by coding the length second positional operand on the FIELD operand. Therefore, the FIELD= operand is set to (DCTUSER0,2).



$SCAN Tables ...

- $SCAN Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Field DCTUSER0 in DSECT DCT
      - △ DSECT = DCT
    - ▲ Value of field is numeric
      - △ CONV = NUM
    - ▲ Valid range of numeric data is 0 to 32,767
      - △ RANGE = (0,32767)

01/88 126

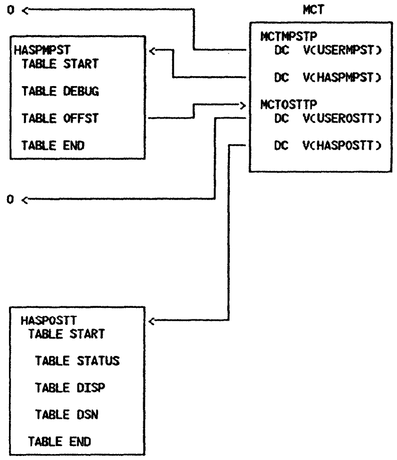
The DCTUSER0 field that is set is located in the DCT DSECT. Therefore, the DSECT= operand is set to DCT.

The conversion value that we specify for this operand is numeric, thus, the CONV= operand is set to NUM.

Since the maximum amount of spool space that can be allocated to a job must fit in a halfword field in the JQE, the maximum threshold value is 32,767. If there may be circumstances where all jobs need to be selected, the minimum value is set to 0 (it would make more sense to remove the TRKGRP criterion from the work selection list to achieve this end). Therefore, the RANGE= operand is set to (0,32767).

$SCAN Tables ...

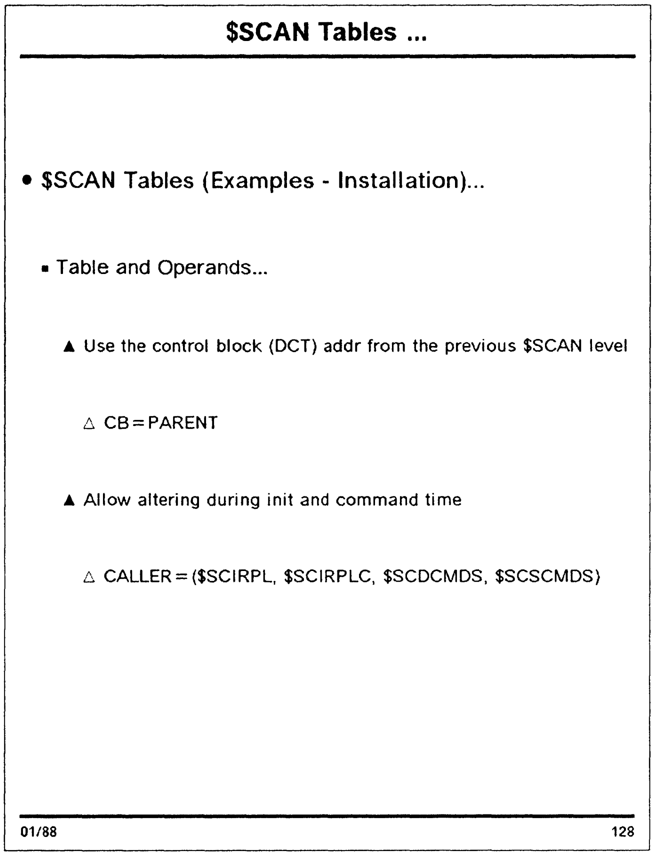
- $SCAN Tables (Examples - Installation)...



```
0
0

MCT
MCTMPSTP
  DC   V(USERMPST)
  DC   V(HASPMPST)
MCTOSTTP
  DC   V(USEROSTT)
  DC   V(HASPOSTT)

HASPMPST
 TABLE START
 TABLE DEBUG
 TABLE OFFST
 TABLE END

HASPOSTT
 TABLE START
  TABLE STATUS
  TABLE DISP
  TABLE DSN
 TABLE END
```

01/88 127

In order to set the CB= operand, you must look at the existing table structure and determine where the revised table structure will fit. Currently, there is a higher level table that contains the $SCAN table elements for the Offload devices. This table element specifies a Control Block of DCT so that the proper DCT for this device is located by the $SCAN facility. Also, this higher level table element indicates that the operands are included in a lower level table (CONV=SUBSCAN,SUBSCAN=MCTOSTTP). It is at this lower level, preceding the JES2 table of Offload SYSOUT Transmitter operands, that the installation table is placed. Therefore, the control block that is wanted at this lower level has been found by the higher level, the parent level.



$SCAN Tables ...

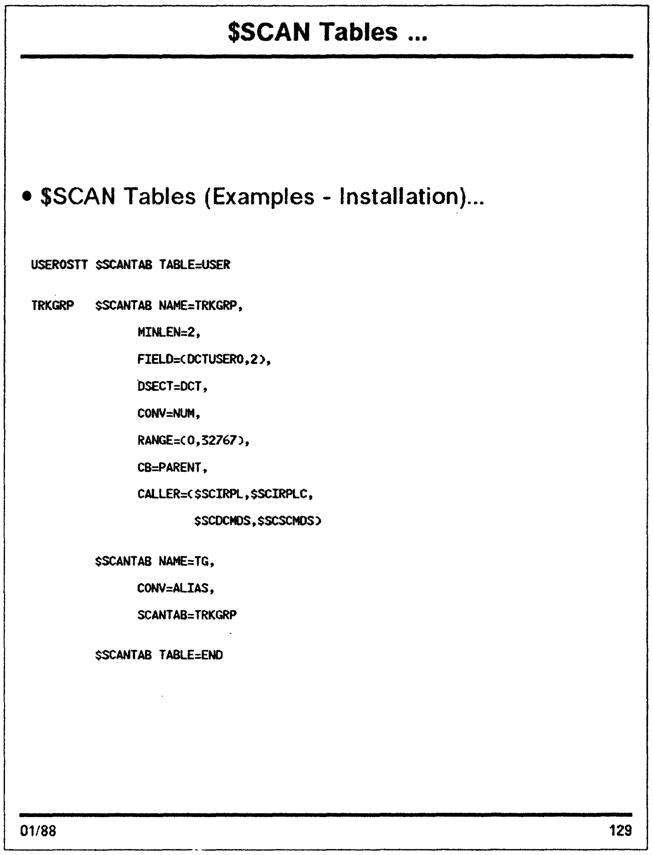
- $SCAN Tables (Examples - Installation)...
  - Table and Operands...
    - ▲ Use the control block (DCT) addr from the previous $SCAN level
      - △ CB=PARENT
    - ▲ Allow altering during init and command time
      - △ CALLER=($SCIRPL, $SCIRPLC, $SCDCMDS, $SCSCMDS)

01/88 128

Since the control block is located at the parent level, specify the CB= operand as PARENT so $SCAN uses the device DCT.

A CALLERs identifier list is specified to allow altering this operand during JES2 initialization and JES2 command time processing.

Resulting $SCAN Table



$SCAN Tables ...

- $SCAN Tables (Examples - Installation)...

```
USEROSTT $SCANTAB TABLE=USER

TRKGRP   $SCANTAB NAME=TRKGRP,
               MINLEN=2,
               FIELD=(DCTUSER0,2),
               DSECT=DCT,
               CONV=NUM,
               RANGE=(0,32767),
               CB=PARENT,
               CALLER=($SCIRPL,$SCIRPLC,
                      $SCDCMDS,$SCSCMDS)

         $SCANTAB NAME=TG,
               CONV=ALIAS,
               SCANTAB=TRKGRP

         $SCANTAB TABLE=END
```

01/88 129

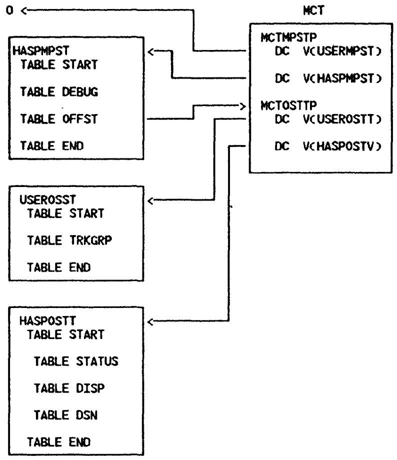
The figure above shows the resulting installation $SCAN table to add an operand to the Offload SYSOUT Transmitter initialization statement and command. Note that the name of the table is USEROSTT, so that if you wished to link-edit this table with HASJES20, the table address would be placed in the $MCT by the linkage editor. The table is begun with a TABLE=USER to tell JES2 that this is an installation table. The name of the operand is TRKGRP although the operator need only specify TR. The field that is set is the first two bytes of the fullword field DCTUSER0 located in the DSECT DCT. The conversion for the input is numeric and the numeric value cannot be less than 0 nor greater than 32767. The address of the DCT is propagated down from the parent level of $SCAN. Both JES2 initialization and JES2 command time processing may reference this table element.

An additional table is shown to provide an example of specifying an alias name for the TRKGRP operand. This table element simply indicates the name of the alias as TG (to match the work selection criteria). This table element is known to be an alias (indicated by the CONV=ALIAS) of the TRKGRP table element since the alias table element points to the TRKGRP element via the SCANTAB operand specifying the TRKGRP table element name of TRKGRP.

Finally, the table is ended with a TABLE=END to indicate to JES2 that this installation table is completed.

$SCAN Tables ...

- $SCAN Tables (Examples - Installation)...

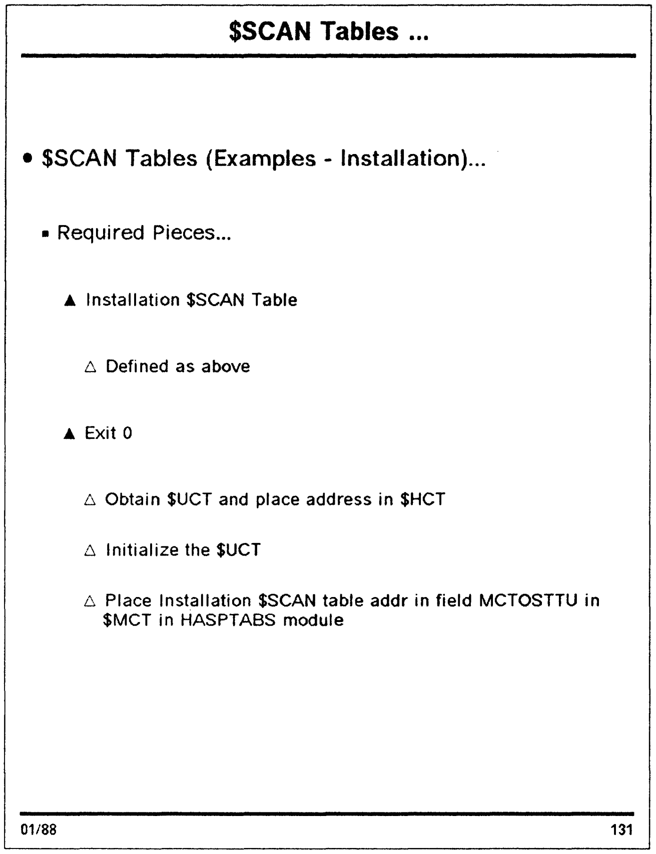


```
0 <
                                   MCT
HASPMPST                           MCTMPSTP
 TABLE START                        DC  V(USERMPST)
 TABLE DEBUG                        DC  V(HASPMPST)
 TABLE OFFST                       MCTOSTTP
 TABLE END                          DC  V(USEROSTT)
                                    DC  V(HASPOSTV)
USEROSST
 TABLE START
 TABLE TRKGRP
 TABLE END

HASPOSTT
 TABLE START
  TABLE STATUS
  TABLE DISP
  TABLE DSN
 TABLE END
```

01/88 130

The resulting table configuration is shown in the figure above. The higher level $SCAN table element for the Offload SYSOUT Transmitter points to the MCTOSTTP table pair. The first entry in this table pair will point to the installation table USEROSST which contains the installation-added operand TRKGRP. The second entry in this table still points to the JES2 table HASPOSTT which contains the JES2 operands. In this way, you have added an operand on the OFFn.ST initialization statement and command.

Coding the Other Required Pieces



$SCAN Tables ...

- $SCAN Tables (Examples - Installation)...
  - Required Pieces...
    - ▲ Installation $SCAN Table
      - △ Defined as above
    - ▲ Exit 0
      - △ Obtain $UCT and place address in $HCT
      - △ Initialize the $UCT
      - △ Place Installation $SCAN table addr in field MCTOSTTU in $MCT in HASPTABS module

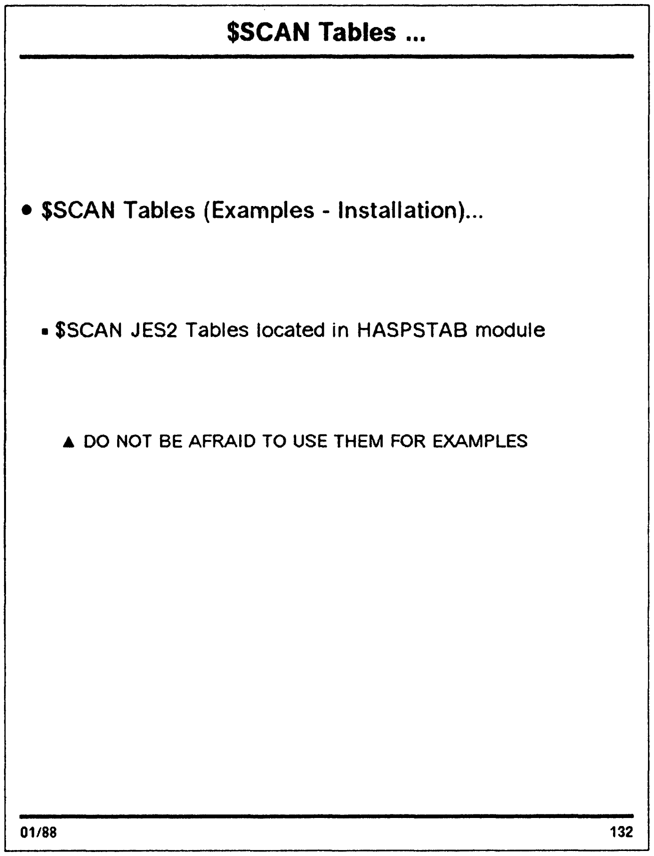
01/88 131

The pieces required to permit you to add an operand are the installation $SCAN table (as coded above) and the code for Exit 0. The Exit 0 code is required to do three things:

1. It must obtain the $UCT and place the $UCT's address in the $HCT.
2. It must initialize the $UCT.
3. Finally, it must place the installation $SCAN table address in the MCTOSTTU field in the $MCT in module HASPTABS.

The code that is contained in "Appendix A. Table Pairs Coding Example" on page 147 is the code as you would be required to code it. This code includes:

1. Exit 0 that obtains the $UCT, places the $UCT address in the $HCT, initializes the $UCT, and places the installation $SCAN table address in the $MCT.
2. The HASPXJ00 module contains, among the other items previously discussed, the $SCAN table.



$SCAN Tables ...

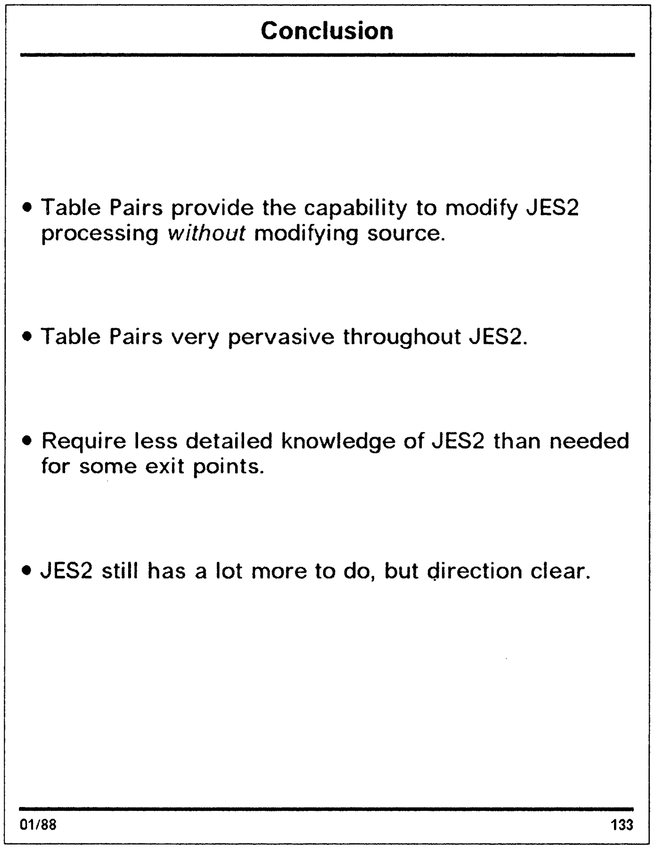
- $SCAN Tables (Examples - Installation)...
  - $SCAN JES2 Tables located in HASPSTAB module
    - ▲ DO NOT BE AFRAID TO USE THEM FOR EXAMPLES

01/88 132

The JES2 $SCAN tables are located in the module HASPSTAB. It is more than likely that there is an example in these tables that will aid you in attempting to code your first few tables. Do not hesitate to use the JES2 $SCAN tables as an example. Also, use *SPL: JES2 User Modifications and Macros*. This book contains a thorough description of the $SCAN related macros and also contains a $SCAN section that makes for useful reading.

THIS PAGE INTENTIONALLY LEFT BLANK

Conclusion



Conclusion

- Table Pairs provide the capability to modify JES2 processing *without* modifying source.
- Table Pairs very pervasive throughout JES2.
- Require less detailed knowledge of JES2 than needed for some exit points.
- JES2 still has a lot more to do, but direction clear.

01/88 133

Hopefully you now know that table pairs provide the capability to modify JES2 processing *without* modifying JES2 source. Through the ability to add, change, and even delete JES2 processors, subtasks, trace identifiers, work selection criteria, and initialization statements, you have an extremely powerful tool to tailor the JES2 component to match local needs.

Since the majority of uses for table pairs will be for adding, changing, and deleting initialization statements and operands (and, to a lesser extent, trace identifiers and work selection criteria), there is less need to have a detailed knowledge of JES2 than is required for exit coding.

However, this is not true when you need to add JES2 processors and subtasks. In this case the systems programmer will need to understand JES2 environments, dispatching, etc., to make use of these tables. However, this is not more than what a systems programmer needs to know to code a JES2 exit and the ability to add these processors and subtasks provides extremely powerful function.

Clearly, JES2 design attempts to provide an interface whereby you can tailor the JES2 product to meet business needs in a way that will not impact an your ability to migrate to newer releases of JES2. Although much more is needed in this area, with the use of table pairs you have a workable method to do this tailoring without the need to modify IBM-supplied JES2 source code.

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A. Table Pairs Coding Example

This coding example implements an installation security processor. It is made up of a JES2 initialization exit 0 and a user extension module named HASPXJ00 which contains the installation security processor, the installation security subtask, and the installation PCE, DTE, trace, work selection, and $SCAN tables. The example includes sample mapping macros $SCYWORK, $SCDWORK, and $UCT, and the macro $USERCBS which invokes the mapping macros.

Note: This code is provided as an example of installation extensions to JES2. The code is not Type 1 supported code of IBM; it is not APARable.[9] A few tests were run using JES2 at the 2.1.5 level.

The examples are inter-related to show how the tables can be used together. This is not required. That is, it is not necessary to code a PCE table (create your own processor) *and* code a DTE table (create your own subtask). In fact, it may make no sense for certain applications to design inter-related tables. Our example was contrived to show what can be done, not necessarily what should be done.

There are six pieces required for the example used in this presentation.

- HASPXJ00 - Installation extension code and tables that are required to create an installation security processor, security subtask, trace id, work selection criteria on the offload sysout transmitter work selection list, and an additional operand on the offload sysout transmitter.
- $UCT - contains required fields for table generation
- $SCDWORK - subtask DTE extension to hold fields specific to a security subtask
- $SCYWORK - processor PCE extension to hold fields specific to a security processor
- $USERCBS - control block that actually generates the above macros. This control block is known by $MODULE and is the way to get $MODULE to generate installation control blocks.
- HASPXIT0 - Exit 0 module that contains EXIT0. This exit initializes the $MCT with the addresses of the installation tables located in HASPXJ00.

[9] However, we do encourage you to use the Reader Comment Form in the back of this document to tell us about any problems you find.

$USERCBS - Generates User Control Blocks

```
         MACRO -- $USERCBS - USER CONTROL BLOCK DSECT                   00100000
         $USERCBS                                                       00200000
*********************************************************************** 00500000
*                                                                     * 00600000
*        $USERCBS - USER CONTROL BLOCK DSECT                          * 00700000
*                                                                     * 00800000
* FUNCTION:                                                           * 00900000
*                                                                     * 01000000
*        THIS DSECT IS KNOWN BY $MODULE AND WILL BE USED TO GET ALL   * 01100000
*        INSTALLATION CONTROL BLOCKS EXPANDED WITHOUT HAVING TO       * 01200000
*        MODIFY THE $MODULE MACRO.                                    * 01300000
*                                                                     * 01400000
* USED BY:                                                            * 01500000
*                                                                     * 01600000
*        ALL INSTALLATION MODULES TO GENERATE ALL INSTALLATION        * 01700000
*        DEFINED CONTROL BLOCKS.  FOR DETAILS ON THE FOLLOWING        * 01800000
*        DATA, SEE THE INDIVIDUAL CONTROL BLOCK DSECTS.               * 01900000
*                                                                     * 02000000
*        CREATED BY: N/A              FREED BY: N/A                   * 02100000
*                                                                     * 02200000
*        SUBPOOL: N/A                 KEY: N/A                        * 02300000
*                                                                     * 02400000
*        SIZE: N/A                    COMPONENT ID: CODE EXAMPLE      * 02500000
*                                                                     * 02600000
*        POINTED TO BY:  N/A                                          * 02700000
*                                                                     * 02800000
*        FREQUENCY:  N/A                                              * 02900000
*                                                                     * 03000000
*        RESIDENCY:  N/A                                              * 03100000
*                                                                     * 03200000
*        SERIALIZATION:  N/A                                          * 03300000
*                                                                     * 03400000
*        CHANGE ACTIVITY:  GUIDE 65 - CHICAGO, ILL - 7/86             * 03500000
*                                                                     * 03600000
*********************************************************************** 03700000
         GBLC  &TITLEID                                                 03800000
         LCLC  &TITL
USERCBS  DSECT                      USER CONTROL BLOCK DSECT            03900000
&TITL    SETC  '&TITLEID -- $UCT     - USER CONTROL TABLE'              04000000
         TITLE '&TITL'                                                  04100000
        $UCT      ,                 GEN THE UCT                         04200000
&TITL    SETC  '&TITLEID -- $SCDWORK - SECURITY SUBTASK WORK DSECT'     04300000
         TITLE '&TITL'                                                  04400000
        $SCDWORK  ,                 GEN THE SECURITY SUBTASK WORK DSECT 04500000
&TITL    SETC  '&TITLEID -- $SCYWORK - SECURITY PCE WORK DSECT'         04600000
         TITLE '&TITL'                                                  04700000
        $SCYWORK  ,                 GEN THE SECURITY PCE WORK DSECT     04800000
         MEND                                                           99999999
```

$SCYWORK - Processor Work Area

```
         MACRO -- $SCYWORK -- USER SECURITY PROCESSOR WORK AREA DSECT   00100000
         $SCYWORK                                                       00200000
*********************************************************************** 00500000
*                                                                     * 00600000
*        $SCYWORK - USER SECURITY PROCESSOR WORK AREA DSECT           * 00700000
*                                                                     * 00800000
* FUNCTION:                                                           * 00900000
*                                                                     * 01000000
*        HOLD FIELDS UNIQUE TO THE SECURITY PROCESSOR PCE             * 01100000
*                                                                     * 01200000
* USED BY:                                                            * 01300000
*                                                                     * 01400000
*        ALL SECURITY PROCESSOR PCE(S)                                * 01500000
*                                                                     * 01600000
*        CREATED BY: PCEDYN          FREED BY: PCEDYN                 * 01700000
*                                                                     * 01800000
*        SUBPOOL: 1                  KEY: 1                           * 01900000
*                                                                     * 02000000
*        SIZE: SEE SCYLEN EQUATE     COMPONENT ID: CODE EXAMPLE       * 02100000
*                                                                     * 02200000
*        POINTED TO BY:  UCTSYPCE FIELD OF THE $UCT DATA AREA   @MES  * 02300000
*                                                                     * 02400000
*        FREQUENCY:  ONE PER SECURITY PCE                             * 02500000
*                                                                     * 02600000
*        RESIDENCY:  VIRTUAL - ABOVE                                  * 02700000
*                    REAL - ANYWHERE                                  * 02800000
*                                                                     * 02900000
*        SERIALIZATION:  JES2 MAIN TASK SERIALIZATION                 * 03000000
*                                                                     * 03100000
*        CHANGE ACTIVITY:  GUIDE 65 - CHICAGO, ILL - 7/86             * 03200000
*                          1/88 - FIXED COMMENT                       * 03300000
*                                                                     * 03300000
*********************************************************************** 03400000
PCE      DSECT                         USER SECURITY PROCESSOR WORK AREA  03600000
         ORG   PCEWORK                  PCE WORK AREA                   03700000
         SPACE 1                                                        03800000
**********************************************************************  03900000
*                                                                    *  04000000
*        FIELDS UNIQUE TO THE SECURITY PCE                           *  04100000
*                                                                    *  04200000
**********************************************************************  04300000
SCYDTEAD DS    A                       ADDR OF THE SECURITY DTE         04400000
SCYTQE   DS    XL(TQELENG)             HASP TIMER QUEUE ELEMENT         04500000
*    FIELD GOES HERE                                                    04600000
*    FIELD GOES HERE                                                    04700000
*    FIELD GOES HERE                                                    04800000
SCYLEN   EQU   *-PCEWORK               LENGTH OF SCY                    04900000
         MEND                                                           99999999
```

$SCDWORK - Subtask Work Area

```
         MACRO -- $SCDWORK -- USER SECURITY SUBTASK WORK AREA DSECT     00100000
         $SCDWORK                                                       00200000
*********************************************************************** 00500000
*                                                                     * 00600000
*        $SCDWORK - USER SECURITY SUBTASK WORK AREA DSECT             * 00700000
*                                                                     * 00800000
* FUNCTION:                                                           * 00900000
*                                                                     * 01000000
*        HOLD FIELDS UNIQUE TO THE SECURITY SUBTASK                   * 01100000
*                                                                     * 01200000
* USED BY:                                                            * 01300000
*                                                                     * 01400000
*        ALL SECURITY SUBTASKS                                        * 01500000
*                                                                     * 01600000
*        CREATED BY: DTEDYN          FREED BY: DTEDYN                 * 01700000
*                                                                     * 01800000
*        SUBPOOL: 1                  KEY: 1                           * 01900000
*                                                                     * 02000000
*        SIZE: SEE SCDLEN EQUATE     COMPONENT ID: CODE EXAMPLE       * 02100000
*                                                                     * 02200000
*        POINTED TO BY:  UCTSYDTE FIELD OF THE $UCT DATA AREA   @MES  * 02300000
*                                                                     * 02400000
*        FREQUENCY:  ONE PER SECURITY SUBTASK                         * 02500000
*                                                                     * 02600000
*        RESIDENCY:  VIRTUAL - BELOW                                  * 02700000
*                    REAL - BELOW                                     * 02800000
*                                                                     * 02900000
*        SERIALIZATION:  SUBTASKS FOLLOW MVS SERIALIZATION CONCERNS   * 03000000
*                                                                     * 03100000
*        CHANGE ACTIVITY:  GUIDE 65 - CHICAGO, ILL - 7/86             * 03200000
*                          1/88 - ADD SCDHCT                          * 03300000
*                                                                     * 03300000
*********************************************************************** 03400000
DTE      DSECT                     USER SECURITY SUBTASK WORK AREA      03600000
         ORG   DTEWORK             DTE WORK AREA                        03700000
         SPACE 1                                                        03800000
**********************************************************************  03900000
*                                                                    *  04000000
*        FIELDS UNIQUE TO THE SECURITY SUBTASK                       *  04100000
*                                                                    *  04200000
**********************************************************************  04300000
SCDHCT   DS    A(*-*)              ADDRESS OF HCT               @SA     04500000
*    FIELD GOES HERE                                                    04500000
*    FIELD GOES HERE                                                    04600000
SCDLEN   EQU   *-DTEWORK           LENGTH OF SCD                        04700000
         MEND                                                           99999999
```

$UCT - User Communication Table

```
         MACRO -- $UCT -- USER COMMUNICATION TABLE DSECT                00100000
         $UCT                                                           00200000
*********************************************************************** 00500000
*                                                                     * 00600000
*        $UCT   - USER COMMUNICATION TABLE DSECT                      * 00700000
*                                                                     * 00800000
* FUNCTION:                                                           * 00900000
*                                                                     * 01000000
*        HOLD FIELD VARIABLES COMMON FOR INSTALLATION CODE.           * 01100000
*                                                                     * 01200000
* USED BY:                                                            * 01300000
*                                                                     * 01400000
*        ALL INSTALLATION PROCESSOR/FUNCTIONS CAN MAKE USE OF         * 01500000
*        THE $UCT.                                                    * 01600000
*                                                                     * 01700000
*        CREATED BY: HASPXITO        FREED BY: JES2 TASK TERMINATION  * 01800000
*                                                                     * 01900000
*        SUBPOOL: 0                  KEY: 1                           * 02000000
*                                                                     * 02100000
*        SIZE: SEE UCTLEN            COMPONENT ID: CODE EXAMPLE       * 02200000
*                                                                     * 02300000
*        POINTED TO BY:  $UCT FIELD OF THE $HCT DATA AREA             * 02400000
*                                                                     * 02500000
*        FREQUENCY:  ONE PER JES2 SYSTEM                              * 02600000
*                                                                     * 02700000
*        RESIDENCY:  VIRTUAL - ABOVE                                  * 02800000
*                    REAL - ANYWHERE                                  * 02900000
*                                                                     * 03000000
*        SERIALIZATION:  JES2 MAIN TASK SERIALIZATION                 * 03100000
*                                                                     * 03200000
*        CHANGE ACTIVITY:  GUIDE 65 - CHICAGO, ILL - 7/86             * 03300000
*                                                                     * 03400000
*********************************************************************** 03500000
UCT      DSECT                     USER COMMUNICATION TABLE DSECT       03700000
UCTID    DS    CL4'UCT'            UCT IDENTIFIER                       03800000
UCTSCDE  DS    A(*-*)              ADDRESS OF INSTALLATION LOAD MODULE  03900000
         SPACE 1                                                        04000000
**********************************************************************  04100000
*                                                                    *  04200000
*        FIELDS REQUIRED FOR THE PCE TABLES                          *  04300000
*                                                                    *  04400000
**********************************************************************  04500000
         SPACE 1                                                        04600000
UCTMSCTY DS    A(*-*)              ADDR OF ENTRY POINT                  04700000
UCTSYPCE DS    A(*-*)              SECURITY PROCESSORS                  04800000
UCTSYNUM DS    H'1',H'0'                                                04900000
UCTSYQUE DS    A(*-*)              ADDR OF ELEMENT TO BE VERIFIED       05000000
UPCESCTY EQU   255                 ID OF SECURITY PCE                   05100000
$DRSCTY  EQU   63                  DISPATCHER SECURITY RESOURCE         05200000
         SPACE 1                                                        05300000
**********************************************************************  05400000
*                                                                    *  05500000
*        FIELDS REQUIRED FOR THE DTE TABLES                          *  05600000
*                                                                    *  05700000
**********************************************************************  05800000
         SPACE 1                                                        05900000
UCTMDSCY DS    A(*-*)              ADDR OF ENTRY POINT                  06000000
UCTSYDTE DS    A(*-*)              ADDR OF SECURITY DTE                 06100000
UDTESCTY EQU   255                 ID OF SECURITY DTE                   06200000
         SPACE 1                                                        06300000
**********************************************************************  06400000
*                                                                    *  06500000
*        END OF UCT                                                  *  06600000
*                                                                    *  06700000
**********************************************************************  06800000
         SPACE 1                                                        06900000
UCTLEN   EQU   *-UCT               LENGTH OF UCT                        07000000
         MEND                                                           99999999
```

Exit 0 - Initialization

Prologue

```
XIT0     TITLE 'USER EXIT 0 MODULE -- PROLOG (MODULE COMMENT BLOCK)'    00010000
*********************************************************************** 00020000
*                                                                     * 00030000
* MODULE NAME = HASPXIT0 CSECT                                        * 00040000
*                                                                     * 00050000
* DESCRIPTIVE NAME = HASP EXIT 0 INITIALIZATION MODULE                * 00060000
*                                                                     * 00070000
*                                                                     * 00080000
* STATUS = OS/VS2 - SEE $MODULE EXPANSION BELOW FOR FMID, VERSION     * 00090000
*                                                                     * 00100000
* FUNCTION = THE HASPXIT0 MODULE INITIALIZES THE INSTALLATION $UCT    * 00110000
*            AND OTHER INSTALLATION DEFINED ADDRESSES AND FIELDS.     * 00120000
*                                                                     * 00130000
* NOTES = SEE BELOW                                                   * 00140000
*                                                                     * 00150000
*    DEPENDENCIES = 1) JES2 EXIT EFFECTOR                             * 00160000
*                   2) JES2 PROCESSOR AND SUBTASK DISPATCHING         * 00170000
*                                                                     * 00180000
*    RESTRICTIONS = THIS CODE IS PROVIDED AS AN EXAMPLE OF            * 00190000
*                   INSTALLATION EXTENSIONS TO JES2.  THIS CODE IS    * 00200000
*                   NOT TO BE CONSIDERED TYPE 1 SUPPORTED CODE OF     * 00210000
*                   IBM.                                              * 00220000
*                                                                     * 00230000
*    REGISTER CONVENTIONS = R0-R3   = WORK REGISTER                   * 00240000
*                           R4      = ADDRESS OF THE MTE ENTRY        * 00250000
*                           R5      = ADDRESS OF THE MCT              * 00260000
*                           R6-R9   = WORK REGISTER                   * 00270000
*                           R10     = ADDRESS OF THE UCT              * 00280000
*                           R11     = ADDRESS OF THE HCT              * 00290000
*                           R12     = LOCAL ADDRESSABILITY            * 00300000
*                           R13     = ADDRESS OF THE HASPINIT PCE     * 00310000
*                           R14-R15 = WORK AND LINKAGE REGISTER       * 00320000
*                                                                     * 00330000
*    PATCH LABEL = NONE                                               * 00340000
*                                                                     * 00350000
* MODULE TYPE = CSECT                                                 * 00360000
*                                                                     * 00370000
*    PROCESSOR = OS/VS ASSEMBLER H OR ASSEMBLER XF (370)              * 00380000
*                                                                     * 00390000
*    MODULE SIZE = SEE $MODEND MACRO EXPANSION AT END OF ASSEMBLY     * 00400000
*                                                                     * 00410000
*    ATTRIBUTES = NOT REUSABLE, NON-REENTRANT, SUPERVISOR STATE,      * 00420000
*                 PROTECT KEY OF HASP'S (1) OR 0, RMODE 24,           * 00430000
*                 AMODE 24/31                                         * 00440000
*                                                                     * 00450000
* ENTRY POINT = EXIT0                                                 * 00460000
*                                                                     * 00470000
*    PURPOSE = SEE FUNCTION                                           * 00480000
*                                                                     * 00490000
*    LINKAGE = STANDARD JES2 $SAVE/$RETURN LINKAGE                    * 00500000
*                                                                     * 00510000
* INPUT   R0  = A CODE INDICATING WHERE THE INTIALIZATION OPTIONS     * 00520000
*               WERE SPECIFIED                                        * 00530000
*         R1  = ADDRESS OF A 2-WORD PARAMETER LIST WITH THE           * 00540000
*               FOLLOWING STRUCTURE:                                  * 00550000
*               WORD 1 (+0):  ADDR OF INTIALIZATION OPTIONS STRING    * 00560000
*               WORD 2 (+4):  LENGTH OF INITIALIZATION OPTIONS STRING * 00570000
*         R11 = ADDRESS OF HCT                                        * 00580000
*         R13 = ADDRESS OF INITIALIZATION PCE                         * 00590000
*         R14 = RETURN ADDRESS                                        * 00600000
*         R15 = ADDRESS OF ENTRY POINT                                * 00610000
*                                                                     * 00620000
* OUTPUT  R15 = RETURN CODE                                           * 00630000
*         (ALL OTHERS UNCHANGED)                                      * 00640000
*                                                                     * 00650000
* EXIT-NORMAL = RETURN TO CALLER (HASPIRMA)                           * 00660000
*                                                                     * 00670000
* EXIT-ERROR = RETURN TO CALLER (HASPIRMA) WITH NON-ZERO RETURN CODE  * 00680000
*                                                                     * 00690000
* EXTERNAL REFERENCES = SEE BELOW                                     * 00700000
*                                                                     * 00710000
*    ROUTINES = MISCELLANEOUS JES2 SERVICE ROUTINES, AND              * 00720000
*               MISCELLANEOUS STANDARD SUPERVISOR SERVICE ROUTINES    * 00730000
*                                                                     * 00740000
*    DATA AREAS = SEE $MODULE MACRO EXPANSION                         * 00750000
*                                                                     * 00760000
*    CONTROL BLOCKS = SEE $MODULE MACRO EXPANSION                     * 00770000
*                                                                     * 00780000
* TABLES = SEE $MODULE MACRO DEFINITION (BELOW)                       * 00790000
*                                                                     * 00800000
* MACROS = JES2 - $ENTRY, $GETMAIN, $MODCHK, $RETURN, $SAVE           * 00810000
*                                                                     * 00820000
* MACROS = MVS  - NONE                                                * 00830000
*                                                                     * 00840000
* CHANGE ACTIVITY:  GUIDE 65 - CHICAGO, ILL - 7/86                    * 00850000
*                   CODE AT SP1.3.6/2.1.5 LEVEL                       * 00860000
*                   1/88 - VARIOUS FIXES FOR T.B.                     * 00860000
*                                                                     * 00870000
*********************************************************************** 00880000
```

Real Code

```
         TITLE 'USER XITO INITIALIZATION -- PROLOG ($HASPGBL)'          00890000
         COPY  $HASPGBL            COPY HASP GLOBALS               @133 00900000


         TITLE 'HASP XITO INITIALIZATION -- PROLOG ($MODULE)'      @133 00910000
HASPXITO $MODULE NOTICE=NONE,                                          C00920000


               TITLE='HASP XITO INITIALIZATION',                       C00930000
               $DTE,               GENERATE HASP DTE DSECT             C00940000
               $ERA,               GENERATE HASP ERA DSECT             C00950000
               $HCT,               GENERATE HASP HCT DSECT             C00960000
               $HASPEQU,           GENERATE HASP EQUATES DSECT         C00970000
               $MCT,               GENERATE HASP MCT DSECT             C00980000
               $MIT,               GENERATE HASP MIT DSECT             C00990000
               $MITETBL,           GENERATE HASP MITETBL DSECT         C01000000
               $MODMAP,            GENERATE HASP MODMAP DSECT          C01010000
               $PCE,               GENERATE HASP PCE DSECT             C01020000
               $TQE,               GENERATE HASP TQE DSECT             C01030000
               $USERCBS,           GENERATE HASP USERCB DSECT          C01040000
               $XECB               GENERATE HASP XECB DSECT             01050000
         TITLE 'USER XIT0 INITIALIZATION -- EXIT0     - OBTAIN AND SET NC01060000
               ECESSARY INFORMATION'                                    01070000
*********************************************************************** 01080000
*                                                                     * 01090000
*        EXIT0 - INSTALLATION EXIT 0 ROUTINE                          * 01100000
*                                                                     * 01110000
* FUNCTION:                                                           * 01120000
*                                                                     * 01130000
*        THIS EXIT POINT OBTAINS A $UCT CONTROL BLOCK, INITIALIZES    * 01140000
*        IT AND PLACES ITS ADDRESS IN THE $HCT.  THIS ROUTINE ALSO    * 01150000
*        INITIALIZES THE $MCT WITH THE SPECIFIED INSTALLATION TABLE   * 01160000
*        ADDRESSES.                                                   * 01170000
*                                                                     * 01180000
* LINKAGE:                                                            * 01190000
*                                                                     * 01200000
*        CALL BY JES2 INITIALIZATION                                  * 01210000
*                                                                     * 01220000
* ENVIRONMENT:                                                        * 01230000
*                                                                     * 01240000
*        JES2 MAIN TASK LIMITED (INITIALIZATION).                     * 01250000
*                                                                     * 01260000
* RECOVERY:                                                           * 01270000
*                                                                     * 01280000
*        NONE                                                         * 01290000
*                                                                     * 01300000
* REGISTER USAGE (ENTRY/EXIT):                                        * 01310000
*                                                                     * 01320000
*    REG       VALUE ON ENTRY             VALUE ON EXIT               * 01330000
*                                                                     * 01340000
*    R0        WHERE INIT OPTIONS                                     * 01350000
*               SPECIFIED                 UNCHANGED                   * 01360000
*    R1        ADDR OF PARM LIST          UNCHANGED                   * 01370000
*    R2-R10    N/A                        UNCHANGED                   * 01380000
*    R11       HCT BASE ADDRESS           UNCHANGED                   * 01390000
*    R12       N/A                        UNCHANGED                   * 01400000
*    R13       INIT PCE BASE ADDRESS      UNCHANGED                   * 01410000
*    R14       RETURN ADDRESS             UNCHANGED                   * 01420000
*    R15       ENTRY ADDRESS              RETURN CODE (SEE BELOW)     * 01430000
*                                                                     * 01440000
* PARAMETER LIST:                                                     * 01450000
*                                                                     * 01460000
*               +0 - ADDR OF INIT OPTIONS STRING                      * 01470000
*               +4 - LENGTH OF INIT OPTIONS STRING                    * 01480000
*                                                                     * 01490000
* REGISTER USAGE (INTERNAL):                                          * 01500000
*                                                                     * 01510000
*    REG       VALUE                                                  * 01520000
*                                                                     * 01530000
*    R0-R3     WORK REGISTERS                                         * 01540000
*    R4        MTE ENTRY ADDRESS                                      * 01550000
*    R5        MCT BASE ADDRESS                                       * 01560000
*    R6-9      WORK REGISTER                                          * 01570000
*    R10       UCT BASE ADDRESS                                       * 01580000
*    R11       HCT BASE ADDRESS                                       * 01590000
*    R12       LOCAL BASE ADDRESS                                     * 01600000
*    R13       INIT PCE BASE ADDRESS                                  * 01610000
*    R14       LINK/WORK REGISTER                                     * 01620000
*    R15       LINK/WORK REGISTER                                     * 01630000
*                                                                     * 01640000
* RETURN CODES (R15 ON EXIT):                                         * 01650000
*                                                                     * 01660000
*        0  -  PROCESSING SUCCESSFUL (NO ERRORS)                      * 01670000
*       12  -  PROCESSING FAILED, TERMINATE JES2                      * 01680000
*                                                                     * 01690000
* OTHER CONSIDERATIONS:                                               * 01700000
                                                                      * 01710000
*        N/A                                                          * 01720000
*                                                                     * 01730000
*********************************************************************** 01740000
```

```
         SPACE 1                                                        01750000
         USING UCT,R10             ESTABLISH UCT ADDRESSABILITY         01760000
         SPACE 1                                                        01770000
EXITO   $ENTRY BASE=R12            DEFINE HASPXITO ENTRY POINT          01780000
         SPACE 2                                                        01790000
        $SAVE  TRACE=NO,NAME=EXITO  GET NEW SAVE AREA, SAVE REGS        01800000
         LR    R12,R15             ESTABLISH BASE REGISTER              01810000
         CLC   $UCT,$ZEROS         ALREADY OBTAINED $UCT...             01820000
         BNE   XITRETO              YES, RETURN TO JES2                 01830000
         EJECT                                                          01840000
*********************************************************************** 01850000
*                                                                     * 01860000
*        OBTAIN AND INITIALIZE THE UCT                                * 01870000
*                                                                     * 01880000
*********************************************************************** 01890000
         SPACE 1                                                        01900000
        $GETMAIN RC,LV=UCTLEN,SP=0,LOC=ANY   OBTAIN THE $UCT            01910000
         LTR   R15,R15             GETMAIN SUCCESSFUL...                01920000
         BNZ   XITGTERR             NO, INDICATE ERROR ALLOCATING STOR  01930000
         SPACE 1                                                        01940000
         LR    R2,R1               SET TO                               01950000
         LA    R3,UCTLEN            CLEAR THE                           01960000
         SLR   R15,R15               STORAGE FOR                        01970000
         MVCL  R2,R14                 THE $UCT                          01980000
         SPACE 1                                                        01990000
         ST    R1,$UCT             SET UCT ADDRESS IN $HCT              02000000
         LR    R10,R1              SET UCT ADDRESSABILITY               02010000
         MVC   UCTID,=CL4'UCT'     SET UCT ID                           02020000
         MVC   UCTSYNUM,$H1        SET NUMBER OF PCE(S) TO DEFINE       02030000
         EJECT                                                          02040000
*********************************************************************** 02050000
*                                                                     * 02060000
*        LOAD MODULE THAT CONTAINS THE SECURITY PCE, SECURITY DTE,    * 02070000
*        AND THE NECESSARY TABLES TO INSTALL INSTALLATION TAILORING   * 02080000
*                                                                     * 02090000
*********************************************************************** 02100000
         SPACE 1                                                        02110000
         L     R1,$HASPMAP         GET THE HASP MODMAP ADDRESS          02120000
         ICM   R1,B'1111',MAPADDR+MAPJXMOD-MAP(R1)  IF HASPXJ00 IN      02130000
         BNZ   XITMODAD             HASJES20, SKIP LOAD                 02140000
         SPACE 1                                                        02150000
        $MODCHK NAME='HASPXJ00',LOAD=YES,TEST=(MIT,VERSION),          C02160000
               MESSAGE=YES,ERRET=XITGTERR  LOAD THE INSTALLATION MODULE 02170000
         SPACE 1                                                        02180000
         LR    R1,R0               GET EP ADDRESS IN R1
XITMODAD ST    R1,UCTSCDE          SAVE THE LOAD MODULE ADDRESS    @MES 02190000
         EJECT                                                          02200000
*********************************************************************** 02210000
*                                                                     * 02220000
*        SEARCH THROUGH MODULE TO FIND ENTRY POINTS FOR THE SECURITY * 02230000
*        PCE, SECURITY DTE, PCE TABLE, DTE TABLE, TID TABLE, WORK     * 02240000
*        SELECTION TABLE, AND THE $SCAN TABLE.                        * 02250000
*                                                                     * 02260000
*********************************************************************** 02270000
         SPACE 1                                                        02280000
         USING MTE,R4              ESTABLISH MTE ADDRESSABILITY         02290000
         USING MCT,R5              ESTABLISH MCT ADDRESSABILITY         02300000
         SPACE 1                                                        02310000
         L     R5,$MCT             OBTAIN THE MCT ADDRESS               02320000
         L     R4,MITENTAD-MIT(,R1)  OBTAIN THE MITABLE ADDRESS         02330000
XITOLP   LA    R6,XITOTBL1         OBTAIN THE TBL OF ENTRY POINTS ADDR  02340000
         LA    R7,XITOTBLL         GET THE NUMBER OF ENTRIES IN TABLE   02350000
         CLI   MTENAME,X'FF'       FOUND END OF TABLE...                02360000
         BE    XITENDT              YES, GO VERIFY ADDRESSES            02370000
XITOMTL  LH    R1,TBLFLDOF(,R6)    OBTAIN THE OFFSET TO THE FIELD       02380000
         CLC   MTENAME,TBLNAME(R6)  ENTRY IN MIT MATCH REQUEST IN TABLE 02390000
         BNE   XITOTB                NO, INCREMENT TO NEXT TABLE ENTRY  02400000
         CLC   TBLFLDCB(L'TBLFLDCB,R6),$ZEROS  YES, CB THE UCT...       02410000
         BE    XITOUCT               YES, GO SET FIELD ADDRESS IN UCT   02420000
         ALR   R1,R5               SET THE FIELD ADDRESS IN THE MCT     02430000
         B     XITOMVC             GO SET ENTRY ADDRESS IN MCT          02440000
         SPACE 1                                                     02450000
XITOUCT  ALR   R1,R10            SET FIELD ADDRESS IN THE UCT        02460000
XITOMVC  MVC   0(4,R1),MTEADDR   MOVE ENTRY ADDR INTO CONTROL BLOCK  02470000
         B     XITOLPC           GO CHECK NEXT MIT ENTRY             02480000
         SPACE 1                                                     02490000
XITOTB   LA    R6,TBLENTYL(,R6)  INCREMENT TO NEXT TABLE ENTRY       02500000
         BCT   R7,XITOMTL        CHECK NXT TABLE ENTRY AGAINST MITABL 02510000
XITOLPC  LA    R4,MTELEN(,R4)    INCREMENT TO NEXT MITABLE ENTRY     02520000
         B     XITOLP            CONTINUE SEARCH FOR ENTRY POINTS    02530000
         EJECT                                                       02540000
```

```
***********************************************************************
*                                                                     *
*        VERIFY THAT THE NECESSARY ADDRESSES HAVE BEEN FOUND          *
*                                                                     *
***********************************************************************
         SPACE 1
XITENDT  LA    R6,XITOTBL1         SET THE ADDRESS TO TABLE
         LA    R7,XITOTBLL         SET THE NUMBER OF ENTRIES
XITCLCLP LH    R1,TBLFLDOF(,R6)    OBTAIN THE OFFSET INTO THE CB
         CLC   TBLFLDCB(L'TBLFLDCB,R6),$ZEROS  CONTROL BLOCK THE UCT...
         BE    XITUCT               YES, GO CHECK IT
         AL    R1,$MCT              NO, GET THE MCT FIELD ADDRESS
         B     XITCLC              GO CHECK IF ADDRESS SET
         SPACE 1
XITUCT   ALR   R1,R10              GET THE UCT FIELD ADDRESS
XITCLC   CLC   0(4,R1),$ZEROS      FIELD SET...
         BE    XITGTERR             NO, EXIT WITH AN ERROR
         LA    R6,TBLENTYL(,R6)    BUMP TO NEXT TABLE ENTRY
         BCT   R7,XITCLCLP         GO CHECK NEXT TABLE ENTRY
         SPACE 1
***********************************************************************
*                                                                     *
*        SET GOOD RETURN CODE AND RETURN                              *
*                                                                     *
***********************************************************************
         SPACE 1
XITRETO  SLR   R15,R15             INDICATE GOOD RETURN
         B     XITRET              GO RETURN TO JES2
         SPACE 1
***********************************************************************
*                                                                     *
*        SET ERROR RETURN AND RETURN TO JES2.                         *
*                                                                     *
***********************************************************************
         SPACE 1
XITGTERR LA    R15,12              INDICATE ERROR RETURN
         SPACE 1
XITRET  $RETURN TRACE=NO,RC=(R15)  END OF EXITO INITIALIZATION
         EJECT
************************************************************************
*                                                                      *
*        BUILD THE TABLE OF ENTRY POINTS THAT ARE TO BE FOUND.         *
*        THE TABLE CONSISTS OF:                                        *
*                                                                      *
*                CL8'NAME OF ENTRY POINT',                             *
*                AL2(OFFSET INTO EITHER UCT OR MCT OF FIELD TO SET)    *
*                AL2(0 IF UCT OR 1 IF MCT)                             *
*                                                                      *
************************************************************************
         SPACE 1
         DS    0F
XITOTBL1 DC    CL8'USCTPCE',AL2(UCTMSCTY-UCT),AL2(0)
         DC    CL8'USCTDTE',AL2(UCTMDSCY-UCT),AL2(0)
         DC    CL8'USERPCET',AL2(MCTPCETU-MCT),AL2(1)
         DC    CL8'USERDTET',AL2(MCTDTETU-MCT),AL2(1)
         DC    CL8'USERTIDT',AL2(MCTTIDTU-MCT),AL2(1)
         DC    CL8'USERSTWT',AL2(MCTSTWTU-MCT),AL2(1)
         DC    CL8'USEROSTT',AL2(MCTOSTTU-MCT),AL2(1)
XITOTBLL EQU   (*-XITOTBL1)/12     CALC NUMBER OF ENTRIES
         SPACE 1
TBLNAME  EQU   0,8                 NAME OF ENTRY POINT
TBLFLDOF EQU   8,2                 FIELD OFFSET
TBLFLDCB EQU   10,2                FIELD CONTROL BLOCK
TBLENTYL EQU   12                  LENGTH OF TABLE ENTRY
```

Epilog

```
         TITLE 'HASP XITO INITIALIZATION -- EPILOG ($MODEND)'            99990000
        $MODEND ,                                                        99991010
APARNUM  DC     CL7'XXXXXXX'         APAR NUMBER                         99999999
         END    ,                    END OF HASPXITO                     99999999
```

User Extension Code and Tables

Prologue

```
XJ00     TITLE 'USER EXTENSION MODULE -- PROLOG (MODULE COMMENT BLOCK)' 00010000
************************************************************************ 00020000
*                                                                      * 00030000
* MODULE NAME = HASJES20 ( HASPXJ00 CSECT )                            * 00040000
*                                                                      * 00050000
* DESCRIPTIVE NAME = HASPXJ00 CSECT OF JES2 MAIN MODULE                * 00060000
*                                                                      * 00070000
*                                                                      * 00080000
* STATUS = OS/VS2 - SEE $MODULE EXPANSION BELOW FOR FMID, VERSION      * 00090000
*                                                                      * 00100000
* FUNCTION = THE HASPXJ00 CSECT CONTAINS THE INSTALLATION SECURITY     * 00110000
*            PROCESSOR, THE INSTALLATION SECURITY SUBTASK, AND         * 00120000
*            THE INSTALLATION PCE, DTE, TRACE, WORK SELECTION,         * 00130000
*            AND $SCAN TABLES.                                         * 00140000
*                                                                      * 00150000
* NOTES = SEE BELOW                                                    * 00160000
*                                                                      * 00170000
*    DEPENDENCIES = JES2 PROCESSOR AND SUBTASK DISPATCHING             * 00180000
*                                                                      * 00190000
*    RESTRICTIONS = THIS CODE IS PROVIDED AS AN EXAMPLE OF             * 00200000
*                   INSTALLATION EXTENSIONS TO JES2.  THIS CODE IS     * 00210000
*                   NOT TO BE CONSIDERED TYPE 1 SUPPORTED CODE OF      * 00220000
*                   IBM.                                               * 00230000
*                                                                      * 00240000
*    REGISTER CONVENTIONS = SEE ENTRY POINT DOCUMENTATION              * 00250000
*                                                                      * 00260000
* MODULE TYPE = PROCEDURE, TABLE ( CSECT TYPE )                        * 00270000
*                                                                      * 00280000
*    PROCESSOR = OS/VS ASSEMBLER H OR ASSEMBLER XF (370)               * 00290000
*                                                                      * 00300000
*    MODULE SIZE = SEE $MODEND MACRO EXPANSION AT END OF ASSEMBLY      * 00310000
*                                                                      * 00320000
*    ATTRIBUTES = HASP REENTRANT, RMODE 24, AMODE 24/31.               * 00330000
*                                                                      * 00340000
* ENTRY POINT =    USCTPCE  - INITIAL ENTRY TO SECURITY PROCESSOR      * 00350000
*                  USCTDTE  - INITIAL ENTRY TO THE SUBTASK USED FOR    * 00360000
*                             AUTHORIZATION CHECKS                     * 00370000
*                  USERPCET - ENTRY FOR INSTALLATION PCE TABLE         * 00380000
*                  USERDTET - ENTRY FOR INSTALLATION DTE TABLE         * 00390000
*                  USERTIDT - ENTRY FOR INSTALLATION TRACE ID TABLE    * 00400000
*                  USERSTWT - ENTRY FOR INSTALLATION OFFLOAD SYSOUT    * 00410000
*                             TRANSMITTER WORK SELECTION TABLE         * 00420000
*                  USEROSTT - ENTRY FOR INSTALLATION OFFLOAD SYSOUT    * 00430000
*                             TRANSMITTER OPERAND TABLE                * 00440000
*                                                                      * 00450000
*    PURPOSE = SEE FUNCTION                                            * 00460000
*                                                                      * 00470000
*    LINKAGE = SEE ENTRY POINT DOCUMENTATION                           * 00480000
*                                                                      * 00490000
*    INPUT = SEE ENTRY POINT DOCUMENTATION                             * 00500000
*                                                                      * 00510000
*    OUTPUT = SEE ENTRY POINT DOCUMENTATION                            * 00520000
*                                                                      * 00530000
*    EXIT-NORMAL = SEE ENTRY POINT DOCUMENTATION                       * 00540000
*                                                                      * 00550000
*    EXIT-ERROR = SEE ENTRY POINT DOCUMENTATION                        * 00560000
*                                                                      * 00570000
* EXTERNAL REFERENCES = SEE BELOW                                      * 00580000
*                                                                      * 00590000
*    ROUTINES = NONE                                                   * 00600000
*                                                                      * 00610000
*    DATA AREAS = SEE $MODULE MACRO SPECIFICATION                      * 00620000
*                                                                      * 00630000
*    CONTROL BLOCKS = SEE $MODULE MACRO SPECIFICATION                  * 00640000
*                                                                      * 00650000
* TABLES = SEE $MODULE MACRO SPECIFICATION                             * 00660000
*                                                                      * 00670000
* MACROS = JES2 - $ACTIVE, $AMODE, $CALL, $DECODE, $DORMANT, $DTEDYN,  * 00680000
*                 $ENTRY, $MODULE, $PCETAB, $REGS, $RETURN, $SAVE,     * 00690000
*                 $SCANTAB, $STIMER, $STORE, $TIDTAB, $TRACE, $WAIT,   * 00700000
*                 $WSTAB                                               * 00710000
*                                                                      * 00720000
* MACROS = MVS  - ATTACH, DEQ, ENQ, ESTAE, POST, SDUMP, WAIT           * 00730000
*                                                                      * 00740000
* CHANGE ACTIVITY:  GUIDE 65 - CHICAGO, ILL - 7/86                     * 00750000
*                   CODE AT SP1.3.6/2.1.5 LEVEL                        * 00760000
*                   1/88 VARIOUS FIXES BY BDB, SA, JK, MES, SWW FOR TB* 00760000
*                                                                      * 00770000
*********************************************************************** 00780000


         TITLE 'USER EXTENSION MODULE -- PROLOG ($HASPGBL)'             00790000
         COPY  $HASPGBL            COPY HASP GLOBALS                    00800000
         TITLE 'USER EXTENSION MODULE -- PROLOG ($MODULE)'              00810000
HASPXJ00 $MODULE NOTICE=NONE,                                          C00820000
               ENTRIES=(USERPCET,USERDTET,USERTIDT,USERSTWT,USEROSTT), C00830000
               TITLE='USER EXTENSION MODULE',                          C00840000
               $DCT,                 GENERATE HASP DCT DSECT           C00850000
               $DTE,                 GENERATE HASP DTE DSECT           C00860000
               $DTETAB,              GENERATE HASP DTETAB DSECT        C00870000
               $ERA,                 GENERATE HASP ERA DSECT           C00880000
               $HASPEQU,             GENERATE HASP EQUATES DSECT       C00890000
               $HCT,                 GENERATE HASP HCT DSECT           C00900000
               $JQE,                 GENERATE HASP JQE DSECT           C00910000
               $MIT,                 GENERATE HASP MIT DSECT           C00920000
               $PCE,                 GENERATE HASP PCE DSECT           C00930000
               $PCETAB,              GENERATE HASP PCETAB DSECT        C00940000
               $RDRWORK,             GENERATE HASP RDRWORK DSECT       C00950000
               $SCANTAB,             GENERATE HASP SCANTAB DSECT       C00960000
               $SCAT,                GENERATE HASP SCAT DSECT          C00970000
               $SVT,                 GENERATE HASP SVT DSECT           C00980000
               $TIDTAB,              GENERATE HASP TIDTAB DSECT        C00990000
               $TLGWORK,             GENERATE HASP TLGWORK DSECT       C01000000
               $TQE,                 GENERATE HASP TQE DSECT           C01010000
               $TRP,                 GENERATE HASP TRP DSECT           C01020000
               $TTE,                 GENERATE HASP TTE DSECT           C01030000
               $USERCBS,             GENERATE USER DSECTS              C01040000
               $WSTAB,               GENERATE HASP WSTAB DSECT         C01050000
               $XECB                 GENERATE HASP XECB DSECT           01060000
```

Overview

```
         TITLE 'USER EXTENSION MODULE -- INTRO    - BRIEF OVERVIEW OF MC01070000
                FUNCTION AND RELATED PIECES'                            01080000
*********************************************************************** 01090000
*                                                                     * 01100000
* FUNCTION -- THIS MODULE CONTAINS THE INSTALLATION EXTENSION CODE    * 01110000
*             AND TABLES THAT ARE REQUIRED TO CREATE AN INSTALLATION  * 01120000
*             SECURITY PROCESSOR, SECURITY SUBTASK, TRACE ID, WORK    * 01130000
*             SELECTION CRITERIA ON THE OFFLOAD SYSOUT TRANSMITTER    * 01140000
*             WORK SELECTION LIST, AND AN ADDITIONAL OPERAND ON THE   * 01150000
*             OFFLOAD SYSOUT TRANSMITTER.                             * 01160000
*                                                                     * 01170000
* REQUIRED PIECES -- HASPXJ00 - THIS MODULE                           * 01180000
*                    $UCT     - CONTAINS REQUIRED FIELDS FOR TABLE    * 01190000
*                               GENERATION                            * 01200000
*                    $SCDWORK - SUBTASK DTE EXTENSION TO HOLD FIELDS  * 01210000
*                               SPECIFIC TO A SECURITY SUBTASK        * 01220000
*                    $SCYWORK - PROCESSOR PCE EXTENSION TO HOLD       * 01230000
*                               FIELDS SPECIFIC TO A SECURITY         * 01240000
*                               PROCESSOR                             * 01250000
*                    $USERCBS - CONTROL BLOCK THAT ACTUALLY GENERATES * 01260000
*                               THE ABOVE MACROS.  THIS CONTROL BLOCK * 01270000
*                               IS KNOWN BY $MODULE AND IS THE WAY    * 01280000
*                               FOR AN INSTALLATION TO GET $MODULE TO * 01290000
*                               GENERATE THEIR CONTROL BLOCKS         * 01300000
*                    HASPXIT0 - EXIT 0 MODULE THAT CONTAINS EXIT0.    * 01310000
*                               THIS EXIT INITIALIZES THE $MCT WITH   * 01320000
*                               THE ADDRESSES OF THE INSTALLATION     * 01330000
*                               TABLES LOCATED IN HASPXJ00.           * 01340000
*                                                                     * 01350000
*********************************************************************** 01360000
```

USCTPCE - Initial Entry Point

```
         TITLE 'USER EXTENSION MODULE -- USCTPCE  - SECURITY PROCESSOR,C01370000
               INITIAL ENTRY POINT'                                     01380000
*********************************************************************** 01390000
*                                                                     * 01400000
* PROCESSOR NAME -- USCTPCE                                           * 01410000
*                                                                     * 01420000
* DESCRIPTIVE NAME -- USER SECURITY PROCESSOR                         * 01430000
*                                                                     * 01440000
* FUNCTION -- MANAGE THE INSTALLATION SECURITY SAF CALLS BY PASSING   * 01450000
*             A REQUEST TO THE SECURITY PROCESSOR'S SECURITY          * 01460000
*             SUBTASK TO ISSUE THE SAF CALL.                          * 01470000
*                                                                     * 01480000
* NOTES --     BECAUSE A JES2 PROCESSOR IS NOT ALLOWED TO DIRECTLY    * 01490000
*              ISSUE AN OS WAIT, USCTPCE ATTACHES A SUB-TASK TO       * 01500000
*              PERFORM THOSE FUNCTIONS REQUIRING WAITS.  THE SUB-TASK,* 01510000
*              USCTDTE, PERFORMS THE CALL TO THE SECURITY             * 01520000
*              AUTHORIZATION FACILITY (SAF).                          * 01530000
*                                                                     * 01540000
*                                                                     * 01550000
* REGISTER CONVENTIONS -- R0 - R2 -- WORK REGISTERS                   * 01560000
*                         R3      -- ADDRESS OF $DTE                  * 01570000
*                         R4      -- ADDRESS OF WORK ELEMENT          * 01580000
*                         R5 - R9 -- WORK REGISTERS                   * 01590000
*                         R10     -- ADDRESS OF $UCT                  * 01600000
*                         R11     -- ADDRESS OF $HCT                  * 01610000
*                         R12     -- BASE ADDRESSABILITY              * 01620000
*                         R13     -- ADDRESS OF PCE                   * 01630000
*                         R14     -- LINKAGE REGISTER                 * 01640000
*                         R15     -- LINKAGE REGISTER                 * 01650000
*                                                                     * 01660000
*********************************************************************** 01670000
         EJECT                                                          01680000
*********************************************************************** 01690000
*                                                                     * 01700000
*        USCTPCE INITAL ENTRY POINT                                   * 01710000
*                                                                     * 01720000
*********************************************************************** 01730000
         SPACE 2                                                        01740000
         USING UCT,R10             ESTABLISH UCT ADDRESSABILITY         01750000
         SPACE 1                                                        01760000
USCTPCE  $ENTRY BASE=R12           PROVIDE PROCESSOR ENTRY POINT        01770000
         SPACE 1                                                        01780000
         L     R10,$UCT            OBTAIN THE UCT ADDRESS               01790000
         EJECT                                                          01800000
*********************************************************************** 01810000
*                                                                     * 01820000
*        MAIN LOOP OF THE SECURITY PROCESSOR.                         * 01830000
*                                                                     * 01840000
*********************************************************************** 01850000
         SPACE 1                                                        01860000
USCTYLOP $ACTIVE                   INDICATE PROCESSOR ACTIVE            01870000
         ICM   R3,B'1111',SCYDTEAD  SUBTASK ATTACHED...                 01880000
         BZ    USCATACH              NO, GO ATTACH IT                   01890000
         TM    DTEFLAG1-DTE(R3),DTE1ACTV  SUBTASK ACTIVE...             01900000
         BO    USCTEST              YES, GO QUEUE UP MEMBER             01910000
         SPACE 1                                                        01920000
*********************************************************************** 01930000
*                                                                     * 01940000
*        DETACH THE SECURITY SUBTASK (ABENDED)                        * 01950000
*                                                                     * 01960000
*********************************************************************** 01970000
         SPACE 1                                                        01980000
        $DTEDYN DETACH,ID=UDTESCTY,DTE=(R3),WAIT=XECB                  C01990000
                                   DETACH ABENDED SUB-TASK              02000000
         XC    SCYDTEAD,SCYDTEAD   CLEAR DTE ADDR                       02010000
         EJECT                                                          02020000
*********************************************************************** 02030000
*                                                                     * 02040000
*        (RE)-ATTACH THE SECURITY SUBTASK                             * 02050000
*                                                                     * 02060000
*********************************************************************** 02070000
         SPACE 1                                                        02080000
USCATACH $DTEDYN ATTACH,ID=UDTESCTY,WAIT=XECB,ERRET=USCATERR           C02090000
                                                 ATTACH USCTDTE         02100000
         ST    R1,SCYDTEAD         STORE SUBTASK DTE ADDRESS            02110000
         MVC   XECBECB-XECB+DTEIXECB-DTE(,R1),$ZEROS CLEAR             C02120000
                                                 COMMUNICATION ECB      02130000
         LR    R3,R1               SET THE SUBTASK DTE ADDRESS          02140000
         ST    R11,SCDHCT(,R3)     STORE HCT ADDRESS IN DTE XTNSN  @SA  02140000
**********************************************************************  02160000
*                                                                    *  02170000
*        DETERMINE IF THERE IS WORK TO BE DONE                       *  02180000
*                                                                    *  02190000
**********************************************************************  02200000
         SPACE 1                                                        02150000
USCTEST  ICM   R4,B'1111',UCTSYQUE  ANY WORK TO DO...                   02220000
         BNZ   USCWORK               YES, GO DO IT                      02230000
         SPACE 1                                                        02240000
        $DORMANT                   INDICATE THAT PROCESSOR COMPLETE     02250000
         SPACE 1                                                        02270000
        $WAIT  SCTY,INHIBIT=NO     WAIT FOR WORK                        02280000
         B     USCTYLOP            GO CHECK FOR WORK TO DO              02290000
         EJECT                                                          02300000
*********************************************************************** 02310000
*                                                                     * 02320000
*        SETUP FOR SUB-TASK TO PROCESS JOB                            * 02330000
*                                                                     * 02340000
* INSTALLATION CODE WOULD GO HERE TO PASS TO SUBTASK THE NECESSARY    * 02350000
* INFORMATION (THROUGH THE DTE EXTENSION THAT IS UNIQUE FOR THE       * 02360000
* SECURITY SUBTASK).                                                  * 02370000
*                                                                     * 02380000
*********************************************************************** 02390000
         SPACE 1                                                        02400000
USCWORK  DS    0H                                                       02410000
         XC    UCTSYQUE,UCTSYQUE   INDICATE WORK BEING PROCESSED (IN   C02420000
                                   REALITY THIS WOULD PROBABLY UNCHAIN C02430000
                                   THE REQUEST, NOT CLEAR THE QUEUE)    02440000
         EJECT                                                          02450000
*********************************************************************** 02460000
*                                                                     * 02470000
*        MVS POST THE SUBTASK FOR WORK TO DO AND $WAIT FOR IT TO      * 02480000
*        COMPLETE.  NOTE THAT THE CALL TO THE SUBTASK IS $TRACE'D,    * 02490000
*        IF TRACING IS ACTIVE.                                        * 02500000
*                                                                     * 02510000
*********************************************************************** 02520000
         SPACE 1                                                        02530000
         MVC   XECBECB-XECB+DTEIXECB-DTE(,R3),$ZEROS CLEAR ECB         C02540000
                                                     FOR $WAIT          02550000
         LA    R1,DTEWECB-DTE(,R3) POINT TO THE WORK ECB                02560000
         SPACE 1                                                        02570000
         POST  (1)                 POST SECURITY SUBTASK FOR WORK       02580000
         SPACE 1                                                        02590000
        $TRACE ID=255,LEN=USCSAFML,OFF=USCTROFF,NAME=SAFCALL            02600000
         MVC   0(USCSAFML,R1),USCSAFM  SET INFORMATION TO BE TRACED     02610000
         SPACE 1                                                        02620000
USCTROFF LR    R1,R3               GET DTE ADDRESS                      02630000
        $WAIT  OPER,XECB=DTEIXECB-DTE(,R1) $WAIT FOR SUB-TASK          C02640000
                                            TO POST US                  02650000
         EJECT                                                          02660000
**********************************************************************  02670000
*                                                                    *  02680000
*        SUBTASK HAS POSTED US BACK                                  *  02690000
*                                                                    *  02700000
*   INSTALLATION CODE WOULD GO HERE TO VALIDATE THE SUCCESS OF THE   *  02710000
*   SECURITY CALL AND TO DO ANY PROCESSING RELEVANT TO THE SUCCESS   *  02720000
*   OR FAILURE OF THE CALL.                                          *  02730000
*                                                                    *  02740000
**********************************************************************  02750000
         SPACE 1                                                        02760000
         DS    0H                  VALIDATE THE RESULT OF THE SECURITY C02770000
                                    CALL.                               02780000
         SPACE                                                          02790000
*********************************************************************** 02800000
*                                                                     * 02810000
*        BRANCH TO OBTAIN THE NEXT ITEM TO VERIFY                     * 02820000
*                                                                     * 02830000
*********************************************************************** 02840000
         SPACE 1                                                        02850000
         B     USCTEST              GO CHECK FOR MORE WORK              02860000
         EJECT                                                          02870000
*********************************************************************** 02880000
*                                                                     * 02890000
*        AN ERROR WAS ENCOUNTERED ON THE ATTACH OF THE SUBTASK.       * 02900000
*        WAIT FOR 30 SECONDS AND ATTEMPT TO TRY AGAIN.                * 02910000
*                                                                     * 02920000
*********************************************************************** 02930000
         SPACE 1                                                        02940000
USCATERR LA    R1,SCYTQE            GET ADDRESS OF PCE TQE              02950000
         LA    R0,30                SET TIME INTERVAL                   02960000
         ST    R0,TQETIME(,R1)       IN TQE                             02970000
         ST    R13,TQEPCE(,R1)      STORE PCE ADDRESS IN TQE            02980000
        $STIMER (R1)                CHAIN THIS TQE                      02990000
        $WAIT  WORK                  AND WAIT FOR INTERVAL TO ELAPSE    03000000
         B     USCATACH             GO ATTACH SUBTASK                   03010000
         SPACE 1                                                        03020000
*********************************************************************** 03030000
*                                                                     * 03040000
*        LIST LITERALS AND SUSPEND ADDRESSABILITIES.                  * 03050000
*                                                                     * 03060000
*********************************************************************** 03070000
         SPACE 1                                                        03080000
         LTORG                                                          03090000
         SPACE 1                                                        03100000
         DROP  R10,R12,R13          SUSPEND UCT, BASE, AND PCE ADDRESS  03110000
```

USCTDTE - Security Subtask, Initial Entry Point

```
         TITLE 'USER EXTENSION MODULE -- USCTDTE  - SECURITY SUBTASK, IC
               NITIAL ENTRY POINT'
***********************************************************************
*                                                                     *
*        USCTDTE - USER SECURITY SUBTASK                              *
*                                                                     *
* FUNCTION:                                                           *
*                                                                     *
*        THIS IS AN EXAMPLE OF A USER CODED SECURITY SUBTASK.  THIS   *
*        SUBTASK IS DEFINED BY THE USERDTET DTE TABLE.  THIS SUBTASK  *
*        IS ATTACHED BY THE USCTPCE SECURITY PROCESSOR.  THE          *
*        PURPOSE OF THIS SUBTASK IS TO CODE THE SAF CALL TO VERIFY    *
*        THE ELEMENT THAT WAS PASSED TO IT FROM THE SECURITY          *
*        PROCESSOR.                                                   *
*                                                                     *
* LINKAGE:                                                            *
*                                                                     *
*        CONTROL GIVEN BY MVS VIA AN ATTACH MVS CALL.                 *
*                                                                     *
* ENVIRONMENT:                                                        *
*                                                                     *
*        JES2 SUBTASK                                                 *
*                                                                     *
* RECOVERY:                                                           *
*                                                                     *
*        MVS ESTAE ESTABLISH UPON ENTRY.  THE RECOVERY ROUTINE IS     *
*        PROVIDED BY THE $STABEND ROUTINE LOCATED IN HASPRAS.         *
*                                                                     *
* REGISTER USAGE (ENTRY/EXIT):                                        *
*                                                                     *
*    REG       VALUE ON ENTRY             VALUE ON EXIT               *
*                                                                     *
*    R0        N/A                        UNPREDICTABLE               *
*    R1        DTE ADDRESS AS SPECIFIED                               *
*               ON THE ATTACH CALL        UNPREDICTABLE               *
*    R2-R14    N/A                        UNPREDICTABLE               *
*    R15       ENTRY ADDRESS              UNPREDICTABLE               *
*                                                                     *
* PARAMETER LIST:                                                     *
*                                                                     *
*        ALL NECESSARY INFORMATION LOCATED IN THE DTE, AS PASSED      *
*        BY THE ATTACHING PROCESSOR.                                  *
*                                                                     *
* REGISTER USAGE (INTERNAL):                                          *
*                                                                     *
*    REG       VALUE                                                  *
*                                                                     *
*    R0-R10    WORK REGISTERS                                         *
*    R11       HCT BASE ADDRESS                                       *
*    R12       LOCAL BASE ADDRESS                                     *
*    R13       DTE BASE ADDRESS                                       *
*    R14       LINK/WORK REGISTER                                     *
*    R15       LINK/WORK REGISTER                                     *
*                                                                     *
* RETURN CODES (R15 ON EXIT):                                         *
*                                                                     *
*        N/A                                                          *
*                                                                     *
* OTHER CONSIDERATIONS:                                               *
*                                                                     *
*        N/A                                                          *
*                                                                     *
***********************************************************************
         SPACE 1
         USING HCT,R11             ESTABLISH HCT ADDRESSABILITY
         USING DTE,R13             ESTABLISH DTE ADDRESSABILITY
         SPACE 1
USCTDTE  $ENTRY BASE=R12           USER SECURITY SUB-TASK
         LR    R12,R15             SET LOCAL BASE
         LR    R13,R1              SET DTE BASE
         L     R11,SCDHCT          SET HCT BASE                    @SA  03810000
*********************************************************************** 03850000
*                                                                  @BDB 03820000
*USCXA  $AMODE 31,RELATED=(USC37)  FORCE 31-BIT MODE FOR UDTESCTY       03830000
*                                                                  @BDB 03820000
*    REMOVED THE $AMODE BECAUSE THE $MODULE ENVIRONMENT IS JES2.  @BDB 03820000
*    THIS CAUSES THE EXPANSION TO GENERATE A CONSTANT $HIBITON     @BDB 03820000
*    WHICH RESIDES IN THE HCT.  SINCE WE DON'T AUTOMATICALLY      @BDB 03820000
*    HAVE ADDRESSABILITY TO THE HCT IN A SUBTASK WE ABEND IN      @BDB 03820000
*    EXECUTION.                                                    @BDB 03820000
*    THIS IS NOT A PROBLEM IF THIS ROUTINE IS COPIED INTO ITS      @BDB 03820000
*    OWN MODULE AND THEN CODE THE $MODULE WITH ENVIRON=SUBTASK.    @BDB 03820000
*                                                                  @BDB 03820000
*********************************************************************** 03850000
USCXA    LA    R15,USCXA01         PSEUDO $AMODE           $AMODE  @BDB 03830000
         O     R15,HIGHON          SET HI BIT ON           $AMODE  @BDB
         BSM   R0,R15              SET MODE                $AMODE  @BDB
HIGHON   DC    0F'0',X'80000000'   MASK FOR 31 BIT MODE    $AMODE  @BDB
USCXA01  DS    0H                  RESUME                  $AMODE  @BDB
         SPACE 1                                                        03840000
*********************************************************************** 03850000
*                                                                    *  03860000
*        SET THE RETRY ROUTINE, THE CLEAN-UP ROUTINE, AND THE        *  03870000
*        VRA EXIT ROUTINE ADDRESSES.                                 *  03880000
*                                                                    *  03890000
*   INSTALLATION SHOULD SET THE DTERTXAD, DTEESXAD, AND DTEVRXAD     *  03900000
*   FOR THE RETRY ROUTINE ADDRESS, THE CLEAN-UP ROUTINE ADDRESS      *  03910000
*   AND THE VRA EXIT ROUTINE ADDRESS RESPECTIVELY, IF THESE          *  03920000
*   ROUTINES ARE NEEDED.                                             *  03930000
*                                                                    *  03940000
*********************************************************************** 03950000
         SPACE 1                                                        03960000
         L     R2,$STABNDA         GET SUBTASK ESTAE RTN ADDRESS        03970000
         LR    R3,R13              COPY DTE ADDRESS                     03980000
         EJECT                                                          03990000
************************************************************************ 04000000
*                                                                     * 04010000
*        E S T A B L I S H   E S T A E   E N V I R O N M E N T        * 04020000
*                                                                     * 04030000
************************************************************************ 04040000
         SPACE 1                                                        04050000
         MVC   DTEAWRKA(USCSTLN),USCABND  MOVE ESTAE PARM LIST          04060000
         SPACE 1                                                        04070000
         ESTAE (2),PARAM=(3),RECORD=YES,MF=(E,DTEAWRKA)                C04080000
                                   ESTABLISH RECOVERY ENVIRONMENT       04090000
         SPACE 1                                                        04100000
         OI    DTEFLAG1,DTE1ACTV   SHOW SUBTASK ACTIVE                  04110000
*                                                                       04120000
*  INSTALLATION SHOULD INITIALIZE THE DTE EXTENSION FOR THE SUBTASK     04130000
*     HERE                                                              04140000
*                                                                       04150000
```

USCTDTE - Security Subtask, Main Processing

```
         TITLE 'USER EXTENSION MODULE --          - SECURITY SUBTASK, MC04160000
               AIN PROCESSING'                                          04170000
*********************************************************************** 04180000
*                                                                     * 04190000
*        NOTIFY PROCESSOR THAT WORK NEEDED AND WAIT FOR A RESPONSE    * 04200000
*                                                                     * 04210000
*********************************************************************** 04220000
         SPACE 1                                                        04230000
USCPOST  XC    DTEWECB,DTEWECB     CLEAR WORK ECB                       04240000
         SPACE 1                                                        04250000
         POST  DTEIXECB            POST PROCESSOR FOR WORK              04260000
         SPACE 1                                                        04270000
         TM    DTEFLAG1,DTE1TERM   SUBTASK SHUTDOWN REQUESTED...        04280000
         BO    USCRET               YES, EXIT TO DELETE SECURITY SUBT   04290000
         SPACE 1                                                        04300000
         WAIT  ECB=DTEWECB          ELSE WAIT FOR WORK TO DO            04310000
         SPACE 1                                                        04320000
         TM    DTEFLAG1,DTE1TERM   SUBTASK SHUTDOWN REQUESTED...        04330000
         BO    USCRET               YES, EXIT TO DELETE SECURITY SUBT   04340000
         EJECT                                                          04350000
**********************************************************************  04740000
*                                                                  @BDB 04730000
*         ISSUE A MVS WTO TO INDICATE THAT THE SUBTASK IS          @BDB 04730000
*         EXECUTING.                                               @BDB 04730000
*                                                                  @BDB 04730000
**********************************************************************  04740000
         SPACE 1                                                        04410000
         LA    R1,USMSG901                                         @BDB
         WTO   MF=(E,(1))                                          @BDB 04420000
         SPACE 1                                                        04430000
**********************************************************************  04440000
*                                                                    *  04450000
*        GO POST THE PROCESSOR FOR WORK                              *  04460000
*                                                                    *  04470000
**********************************************************************  04480000
         SPACE 1                                                        04490000
         B     USCPOST             GO POST PROCESSOR FOR WORK           04500000
```

USCTDTE - Security Subtask, Termination

```
         TITLE 'USER EXTENSION MODULE --          - SECURITY SUBTASK, TC04510000
               ERMINATION'                                              04520000
*********************************************************************** 04530000
*                                                                     * 04540000
*        TERMINATE SECURITY SUBTASK                                   * 04550000
*                                                                     * 04560000
*        NOTE THAT THE MAIN TASK TERMINATION CODE WAITS 30 SECONDS    * 04570000
*        FOR THE SUBTASK TO GO AWAY BEFORE CONTINUING.  IF THE MAIN   * 04580000
*        TASK COMPLETES TERMINATION BEFORE THE SUBTASK DOES (DUE TO   * 04590000
*        DEBUG TRACING IN THE SUBTASK), AN A03 ABEND WILL RESULT.     * 04600000
*                                                                     * 04610000
*********************************************************************** 04620000
         SPACE 1                                                        04630000
USCRET   DS    0H                                                       04640000
USC37   $AMODE 24,RELATED=(USCXA)   AMODE 24 FOR SECURITY TERMINATION   04650000
         SPACE 1                                                        04660000
         ESTAE 0                   CANCEL ESTAE                         04670000
         SVC   3                   THEN RETURN TO SYSTEM                04680000
         EJECT                                                          04690000
**********************************************************************  04700000
*                                                                    *  04710000
*        CREATE THE ESTAE PARAMETER LIST AND TRACED INFORMATION      *  04720000
*                                                                    *  04730000
**********************************************************************  04740000
         SPACE 1                                                        04750000
USCABND  ESTAE ,CT,PURGE=NONE,ASYNCH=YES,TERM=NO,MF=L                   04760000
USCSTLN  EQU   *-USCABND           LENGTH OF ESTAE PARAMETER LIST       04770000
         SPACE 1                                                        04780000
USCSAFM  DC    C'THIS IS TRACE DATA THAT SHOULD BE FILLED IN FOR INSTALC04790000
               LATION USE IN TRACING SECURITY CALLS'                    04800000
USCSAFML EQU   *-USCSAFM                                                04810000
         SPACE 1                                                        04820000
        $MID   901                                                 @BDB
USMSG901 WTO   '&MID. SECURITY SUBTASK INVOKED',                   @BDBC
               MF=L,ROUTCDE=10,DESC=6                              @BDB
         SPACE 1                                                        04820000
         DROP  R13                 DROP DTE ADDRESSABILITY              04830000
```

TROUT255 - Tracing Routine for SAF Call

```
         TITLE 'USER EXTENSION MODULE -- TROUT255 - TRACING ROUTINE FORC04840000
                SAFCALL ID=255'                                         04850000
*********************************************************************** 04860000
*                                                                     * 04870000
*        TROUT255 - TRACING ROUTINE IN SUPPORT OF THE TRACE ID 255.   * 04880000
*                                                                     * 04890000
* FUNCTION:                                                           * 04900000
*                                                                     * 04910000
*        THIS ROUTINE WILL BE CALLED TO FORMAT THE TRACE RECORD FOR   * 04920000
*        THE INSTALLATION TRACE ID 255.  THIS ROUTINE SHOULD BE       * 04930000
*        ALTERED BY THE INSTALLATION TO FORMAT THE INFORMATION THAT   * 04940000
*        WAS SAVED ON THE TRACING OF THIS ID.                         * 04950000
*                                                                     * 04960000
* LINKAGE:                                                            * 04970000
*                                                                     * 04980000
*        BALR R14,15 TO BY HASPMISC                                   * 04990000
*                                                                     * 05000000
* ENVIRONMENT:                                                        * 05010000
*                                                                     * 05020000
*        THIS ENVIRONMENT IS CALL FROM THE JES2 MAIN TASK.            * 05030000
*                                                                     * 05040000
* RECOVERY:                                                           * 05050000
*                                                                     * 05060000
*        NONE                                                         * 05070000
*                                                                     * 05080000
* REGISTER USAGE (ENTRY/EXIT):                                        * 05090000
*                                                                     * 05100000
*    REG       VALUE ON ENTRY             VALUE ON EXIT               * 05110000
*                                                                     * 05120000
*    R0        N/A                        UNCHANGED                   * 05130000
*    R1        TRACE TABLE BUFFER ADDR    UNCHANGED                   * 05140000
*    R2        TRACE TABLE ENTRY (TTE)    UNCHANGED                   * 05150000
*    R3        N/A                        UNCHANGED                   * 05160000
*    R4        TRACE ID TABLE ENTRY       UNCHANGED                   * 05170000
*    R5        POINTER TO REMAINING OUT-  POINTER TO LOCATION IN OUT- * 05180000
*               PUT AREA IN PRINT RECORD   PUT AREA AFTER THIS ENTRY  * 05190000
*    R6-R10    N/A                        UNCHANGED                   * 05200000
*    R11       HCT BASE ADDRESS           UNCHANGED                   * 05210000
*    R12       N/A                        UNCHANGED                   * 05220000
*    R13       PCE BASE ADDRESS           UNCHANGED                   * 05230000
*    R14       RETURN ADDRESS             UNCHANGED                   * 05240000
*    R15       ENTRY ADDRESS                0                         * 05250000
*                                                                     * 05260000
* PARAMETER LIST:                                                     * 05270000
*                                                                     * 05280000
*        NONE                                                         * 05290000
*                                                                     * 05300000
* REGISTER USAGE (INTERNAL):                                          * 05310000
*                                                                     * 05320000
*    REG       VALUE                                                  * 05330000
*                                                                     * 05340000
*    R0-R1     WORK REGISTERS                                         * 05350000
*    R2        TTE ADDRESS                                            * 05360000
*    R3        LOCATION IN TTE                                        * 05370000
*    R4        WORK REGISTER                                          * 05380000
*    R5        LOCATION IN OUTPUT AREA                                * 05390000
*    R6-R8     WORK REGISTER                                          * 05400000
*    R9        *** RESERVED ***                                       * 05410000
*    R10       WORK REGISTER                                          * 05420000
*    R11       HCT BASE ADDRESS                                       * 05430000
*    R12       LOCAL BASE ADDRESS                                     * 05440000
*    R13       PCE BASE ADDRESS                                       * 05450000
*    R14       LINK/WORK REGISTER                                     * 05460000
*    R15       LINK/WORK REGISTER                                     * 05470000
*                                                                     * 05480000
* RETURN CODES (R15 ON EXIT):                                         * 05490000
*                                                                     * 05500000
*        0  -  PROCESSING SUCCESSFUL (NO ERRORS)                      * 05510000
*                                                                     * 05520000
* OTHER CONSIDERATIONS:                                               * 05530000
*                                                                      * 05540000
*       MUST RETURN THE NEW VALUE OF R5 ON EXIT (I.E., $STORE (R5))  * 05550000
*                                                                      * 05560000
*********************************************************************** 05570000
         SPACE 1                                                        05580000
         USING TTE,R2              ESTABLISH TTE ADDRESSABILITY         05590000
         USING PCE,R13             ESTABLISH PCE ADDRESSABILITY         05600000
         SPACE 1                                                        05610000
TROUT255 $ENTRY BASE=R12           ID=255 TRACE FORMATOR ROUTINE        05620000
        $SAVE  NAME=TROUT255,TRACE=NO  SAVE CALLERS REGISTERS           05630000
         LR    R12,R15             ESTABLISH BASE ADDRESS               05640000
         SPACE 1                                                        05650000
         LA    R3,TTEDATA          POINT TO THE TTE DATA                05660000
         MVC   0(USCSAFML,R5),0(R3)  SET THE TRACED INFO IN OUTPUT AREA 05670000
         LA    R0,USCSAFML(,R5)    POINT BEYOND INFORMATION             05680000
         SL    R0,TLGBSAVE          AND FIND LENGTH OF PRINT LINE       05690000
         L     R15,$TRCPUT         GET TRCPUT ROUTINE ADDRESS AND  @BDB
        $CALL  (R15)               GO PRINT THE LINE              @JK   05700000
        $STORE (R5)                INSURE NEW BUFFER IS PASSED BACK     05710000
         SPACE 1                                                        05720000
        $RETURN TRACE=NO           RETURN TO CALLER                     05730000
         SPACE 1                                                        05740000
         DROP  R2,R12,R13          SUSPEND TTE, LOCAL, AND PCE ADDRESS  05750000
```

WSTRKGRP - Work Selection Routine

```
         TITLE 'USER EXTENSION MODULE -- WSTRKGRP - WORK SELECTION ROUTC05760000
               INE FOR TRKGRP CRITERIA'                                 05770000
*********************************************************************** 05780000
*                                                                     * 05790000
*        WSTRKGRP - WORK SELECTION ROUTINE TO COMPARE THE DCT'S       * 05800000
*                   AND JQE'S NUMBER OF TRACK GROUPS.                 * 05810000
*                                                                     * 05820000
* FUNCTION:                                                           * 05830000
*                                                                     * 05840000
*        THIS ROUTINE WILL BE CALLED TO INSURE THAT THE JOB'S NUMBER  * 05850000
*        OF TRACK GROUPS IS EQUAL TO OR BEYOND THE DCT'S THRESHOLD.   * 05860000
*                                                                     * 05870000
* LINKAGE:                                                            * 05880000
*                                                                     * 05890000
*        BALR R14,15 TO BY HASPSERV                                   * 05900000
*                                                                     * 05910000
* ENVIRONMENT:                                                        * 05920000
*                                                                     * 05930000
*        THIS ENVIRONMENT IS CALL FROM THE JES2 MAIN TASK.            * 05940000
*                                                                     * 05950000
* RECOVERY:                                                           * 05960000
*                                                                     * 05970000
*        NONE                                                         * 05980000
*                                                                     * 05990000
* REGISTER USAGE (ENTRY/EXIT):                                        * 06000000
*                                                                     * 06010000
*    REG       VALUE ON ENTRY             VALUE ON EXIT               * 06020000
*                                                                     * 06030000
*    R0        N/A                        UNCHANGED                   * 06040000
*    R1        N/A                        UNPREDICTABLE               * 06050000
*    R2        ADDR OF CRITERION BEING                                * 06060000
*               PROCESSED                 UNCHANGED                   * 06070000
*    R4-R5     N/A                        UNCHANGED                   * 06080000
*    R6        N/A                        UNPREDICTABLE               * 06090000
*    R7        COMPARISON LENGTH          UNPREDICTABLE               * 06100000
*    R8        ADDR OF DEVICE FIELD       UNCHANGED                   * 06110000
*    R9        N/A                        UNCHANGED                   * 06120000
*    R10       ADDR OF COMPARISON FIELD   UNCHANGED                   * 06130000
*    R11       HCT BASE ADDRESS           UNCHANGED                   * 06140000
*    R12       N/A                        UNCHANGED                   * 06150000
*    R13       PCE BASE ADDRESS           UNCHANGED                   * 06160000
*    R14       RETURN ADDRESS             UNCHANGED                   * 06170000
*    R15       ENTRY ADDRESS                0                         * 06180000
*                                                                     * 06190000
* PARAMETER LIST:                                                     * 06200000
*                                                                     * 06210000
*        NONE                                                         * 06220000
*                                                                     * 06230000
* REGISTER USAGE (INTERNAL):                                          * 06240000
*                                                                     * 06250000
*    REG       VALUE                                                  * 06260000
*                                                                     * 06270000
*    R0        N/A                                                    * 06280000
*    R1        ADDR OF JQE                                            * 06290000
*    R2        ADDR OF CRITERION BEING                                * 06300000
*               PROCESSED                                             * 06310000
*    R4-R5     N/A                                                    * 06320000
*    R6        N/A                                                    * 06330000
*    R7        COMPARISON LENGTH                                      * 06340000
*    R8        ADDR OF DEVICE FIELD                                   * 06350000
*    R9        N/A                                                    * 06360000
*    R10       ADDR OF COMPARISON FIELD                               * 06370000
*    R11       HCT BASE ADDRESS                                       * 06380000
*    R12       N/A                                                    * 06390000
*    R13       PCE BASE ADDRESS                                       * 06400000
*    R14       LINKAGE REGISTER                                       * 06410000
*    R15       LINKAGE REGISTER                                       * 06420000
*                                                                     * 06430000
* RETURN CODES (R15 ON EXIT):                                         * 06440000
*                                                                     * 06450000
*        4  -  CONTINUE CRITERIA PROCESSING, ACCEPTABLE CONDITION     * 06460000
*       12  -  UNACCEPTABLE CONDITION, CRITERIA DO NOT MATCH          * 06470000
*                                                                     * 06480000
* OTHER CONSIDERATIONS:                                               * 06490000
*                                                                     * 06500000
*        $SAVE AND $RETURN NOT USED FOR PERFORMANCE REASONS           * 06510000
*                                                                     * 06520000
*********************************************************************** 06530000
         SPACE 1                                                        06540000
         ENTRY WSTRKGRP            ESTABLISH ENTRY POINT                06550000
         USING WSTRKGRP,R6         ESTABLISH ADDRESSABILITY             06560000
         USING PCE,R13             ESTABLISH PCE ADDRESSABILITY         06570000
         SPACE 1                                                        06580000
WSTRKGRP LR    R6,R15              SET ADDRESSABILITY                   06590000
         BCTR  R7,0                PREPARE LENGTH FOR EXECUTES          06600000
         LR    R15,R10             SET THE JQE FIELD ADDRESS            06610000
         SL    R15,=A(JQETGNUM-JQE)  TO OBTAIN THE JQE ADDRESS          06620000
         LR    R1,R10              OBTAIN THE FIELD ADDRESS             06630000
         TM    JQEFLAG5-JQE(R15),JQE5XUSD  NUM OF TGS IN EXT AREA...    06640000
         BNO   WSTTGN                        NO, GO DO COMPARISON       06650000
         LH    R1,JQETGNUM-JQE(,R15)  GET THE OFFSET INTO EXT AREA      06660000
         AL    R1,$JQEEXT              AND OBTAIN THE ADDRESS OF TGN    06670000
WSTTGN   LA    R15,12              ASSUME TG NUMBER NOT AT THRESHOLD    06680000
         EX    R7,WSTCLC           TG NUMBER AT THRESHOLD...            06690000
         BLR   R14                  NO, RETURN INDICATING NO MATCH      06700000
         LA    R15,4                YES, INDICATE MATCH                 06710000
         BR    R14                 RETURN TO CALLER                     06720000
         SPACE 1                                                        06730000
WSTCLC   CLC   0(*-*,R1),0(R8)   *** EXECUTE ONLY ***                   06740000
         SPACE 1                                                        06750000
         DROP  R6,R13              SUSPEND LOCAL AND PCE ADDRESSABILITY 06760000
```

Tables

```
         TITLE 'USER EXTENSION MODULE -- USERPCET - TABLE FOR INSTALLATC06770000
               ION SECURITY PROCESSOR'                                  06780000
*********************************************************************** 06790000
*                                                                     * 06800000
*        DEFINE THE PROCESSOR TABLE                                   * 06810000
*                                                                     * 06820000
*********************************************************************** 06830000
         SPACE 1                                                        06840000
USERPCET $PCETAB TABLE=USER                                             06850000
SCTYPCET $PCETAB NAME=SCTY,DESC='SECURITY',MODULE=HASPXJ00,            C06860000
               ENTRYPT=UCTMSCTY,CHAIN=UCTSYPCE,COUNTS=UCTSYNUM,        C06870000
               MACRO=$SCYWORK,WORKLEN=SCYLEN,GEN=INIT,DISPTCH=WARM,    C06880000
               PCEFLGS=0,FSS=NO,PCEID=(0,UPCESCTY),DCTTAB=*-*           06890000
         $PCETAB TABLE=END                                              06900000
```

```
         TITLE 'USER EXTENSION MODULE -- USERDTET - TABLE FOR INSTALLATC06910000
               ION SECURITY SUBTASK'                                    06920000
*********************************************************************** 06930000
*                                                                     * 06940000
*        DEFINE THE SUBTASK TABLE                                     * 06950000
*                                                                     * 06960000
*********************************************************************** 06970000
         SPACE 1                                                        06980000
USERDTET $DTETAB TABLE=USER                                             06990000
         $DTETAB NAME=SECURITY,ID=UDTESCTY,EPNAME=USCTDTE,             C07000000
               EPLOC=UCTMDSCY,HEAD=UCTSYDTE,WORKLEN=SCDLEN,            C07010000
               GEN=NO,STAE=NO,SZERO=YES                                 07020000
         $DTETAB TABLE=END                                              07030000
```

```
         TITLE 'USER EXTENSION MODULE -- USERTIDT - TABLE FOR INSTALLATC07040000
               ION TRACE ID TABLE(S)'                                   07050000
*********************************************************************** 07060000
*                                                                     * 07070000
*        DEFINE THE TRACE ID TABLE                                    * 07080000
*                                                                     * 07090000
*********************************************************************** 07100000
         SPACE 1                                                        07110000
USERTIDT $TIDTAB TABLE=USER                                             07120000
         $TIDTAB ID=255,FORMAT=TROUT255,NAME=SAFCALL                    07130000
         $TIDTAB TABLE=END                                              07140000
```

```
         TITLE 'USER EXTENSION MODULE -- USERSTWT - TABLE FOR INSTALLATC07150000
               ION WORK SELECTION CRITERIA'                             07160000
*********************************************************************** 07170000
*                                                                     * 07180000
*        DEFINE THE WORK SELECTION CRITERIA TABLE                     * 07190000
*                                                                     * 07200000
*********************************************************************** 07210000
         SPACE 1                                                        07220000
USERSTWT $WSTAB TABLE=USER                                              07230000
         $WSTAB NAME=TRKGRP,MINLEN=2,ALIAS=TG,FLD=JQETGNUM,CB=JQE,     C07240000
               DEVFLD=DCTUSER0,DEVCB=DCT,RTN=WSTRKGRP                   07250000
         $WSTAB TABLE=END                                               07260000
```

```
         TITLE 'USER EXTENSION MODULE -- USEROSTT - TABLE FOR INSTALLATC07270000
               ION SCAN TABLE FOR OFFN.STN'                             07280000
*********************************************************************** 07290000
*                                                                     * 07300000
*        DEFINE THE OFFLOAD SYSOUT TRANSMITTER OPERAND TABLE          * 07310000
*                                                                     * 07320000
*********************************************************************** 07330000
```

```
         SPACE 1                                                        07340000
USEROSTT $SCANTAB TABLE=USER                                            07350000
TRKGRP   $SCANTAB NAME=TRKGRP,MINLEN=2,FIELD=(DCTUSER0,2),DSECT=DCT,   C07360000
               CONV=NUM,RANGE=(0,32767),CB=PARENT,CALLERS=($SCIRPL,    C07370000
               $SCIRPLC,$SCDCMDS,$SCSCMDS)                              07380000
         $SCANTAB NAME=TG,CONV=ALIAS,SCANTAB=TRKGRP
         $SCANTAB TABLE=END                                             07390000
         EJECT                                                          07400000
*********************************************************************** 07410000
*                                                                     * 07420000
*        LIST THE LITERALS FOR THE HASPXJ00 MODULE.                   * 07430000
*                                                                     * 07440000
*********************************************************************** 07450000
         SPACE 1                                                        07460000
         LTORG ,                                                        07470000


         TITLE 'USER EXTENSION MODULE -- EPILOG ($MODEND)'              99990000
        $MODEND ,                                                       99991000
APARNUM  DC    CL7'OZXXXXX'        APAR NUMBER                          99999997
         END   ,                   END OF HASPXJ00                      99999999
```

Index

READER'S COMMENT FORM

Title: Extending JES2
Using Table Pairs
Washington Systems Center
Technical Bulletin GG66-0282-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Please state your occupation: ______________________

Comments:

Please mail to:

Scott W. Wood
IBM Washington Systems Center
JES2 Support
18100 Frederick Pike
ISG/Building 183 Room 2T74
Gaithersburg, MD 20879

Reader's Comment Form

Cut or Fold Along Line

Fold and tape

Please Do Not Staple

Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

S. W. Wood
IBM Corporation
Washington Systems Center
18100 Frederick Pike
Gaithersburg, MD 20879

Fold and tape

Please Do Not Staple

Fold and tape

READER'S COMMENT FORM

Title:
Extending JES2
Using Table Pairs
Washington Systems Center
Technical Bulletin GG66-0282-00

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Please state your occupation: ______________________

Comments:

Please mail to:
Scott W. Wood
IBM Washington Systems Center
JES2 Support
18100 Frederick Pike
ISG/Building 183 Room 2T74
Gaithersburg, MD 20879

Cut or Fold Along Line

Reader's Comment Form

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

S. W. Wood
IBM Corporation
Washington Systems Center
18100 Frederick Pike
Gaithersburg, MD 20879

Fold and tape Please Do Not Staple Fold and tape

IBM®

GG66-0282-00

Extending JES2 Using Table Pairs

Printed in U.S.A.

GG66-0282-00

IBM

GG66-0282-0