NORTHERN ILLINOIS UNIVERSITY

CSCI 466: DATA BASE/DATA COMMUNICATIONS
NOTES AND PROGRAMMING EXAMPLES

A Thesis Submitted to the
University Honors Program
in Partial Fulfillment of the
Requirements of the Baccalaureate Degree
With University Honors

Department of Computer Science
by
Fiona A. McCall
DeKalb, Illinois
May 1987
Approved:  Lee M. Gaines

Department of:  Computer Science

Date:  May 12, 1987
TABLE OF CONTENTS

The contents have been divided into the following sections:

COURSE SYLLABUS
COMMON MODULE REQUIREMENTS
NIU STRUCTURED MACROS
DOCUMENTATION STANDARDS
ACCESS METHOD SERVICES (AMS)
JCL AND PARAMETER CODING
VSAM ALTERNATE INDEX
VSAM IN ASSEMBLER
INFORMATIONAL MANAGEMENT SYSTEM (IMS)
CUSTOMER INFORMATION CONTROL SYSTEM (CICS)
COURSE SYLLABUS
COMMON MODULE REQUIREMENTS
COMMON MODULE REQUIREMENTS

You will be executing the following modules throughout the semester. Given below are the module names, region requirements, all required data sets with their specified DD names, and other notes.

COBOL COMPILER

Module name: IKFCBL00
Required region: 128K

Required Data Sets:

Input: STEPLIB - Location of the COBOL compiler.
       DSN=SYS2.VSC.V124.VSCOLIB,DISP=SHR
       SYSIN - COBOL source program.
Output: SYSLIN - Object module created by the COBOL compiler.
         Typical space is (800,(400,100)).
         SYSPRINT - Source listing and compiler error messages.
Utility: SYSUT1 thru 4 - Temporary work data sets used by the compiler.
         Typical space is (460,(600,200)).

IBM ASSEMBLER

Module name: IEV90
Required region: 256K
Usual parm options: NODECK,OBJECT

Required Data Sets:

Input: STEPLIB - Location of the Assembler.
       DSN=SYS2.ASMH.V210.LOADLIB,DISP=SHR
       SYSIN - Assembler source program.
       SYSLIB - Macro libraries. For NIU these are:
                 DSN=SYS2.MACLIB,DISP=SHR
                 DSN=SYS1.MACLIB,DISP=SHR
Output: SYSLIN - Object module created by the Assembler.
         Typical space is (CYL,(1,1)).
         SYSPRINT - Source listing and messages from the Assembler.
Utility: SYSUT1 - Temporary work data set for the Assembler.
         Typical space is (CYL,(5,5))
LOADERS

Module name: IEWLDRGO (Alias: LOADER)
Required region: 192K
Parm format: 'LOADERPARM/USERPARM' where the information before
the slash is for the loader and the information
after the slash is for the source program. A
loader parm of MAP,LET is recommended.

Required Data Sets:

Input: SYSLIN - One or more object modules.
SYSLIB - Libraries containing load and/or object
modules referenced by the SYSLIN object
module(s). For NIU the libraries are as
follows for Assembler produced object
modules:
DSN=SYS2.CALLIB,DISP=SHR
For Compiler produced object modules:
DSN=SYS2.VSC.V124.VSCLLIB,DISP=SHR
DSN=SYS2.CALLIB,DISP=SHR

Output: SYSLOUT - Loader information and error messages.

LINKAGE EDITOR

Module name: IEWL
Required region: 192K

Required Data Sets:

Input: SYSLIN - One or more object modules.
SYSLIB - As specified in Loader.

Output: SYSLMOD - Load module created by the Linkage Editor.
                Must be PDS member.
                (ie. - DSN=MYLIB(THISONE),...)
SYSPRINT - Messages from the Linkage Editor.
The NIU structured macros have been designed to facilitate the implementation of the standard structured programming constructs in the IBM assembler language. These constructs include

**IF THEN ELSE**

```
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
```

**DO WHILE**

```
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
```

For those familiar with IBM's concept 14 macros, implementation of the above constructs may be accomplished by coding the NIU structured macros exactly as you would code the concept 14 IF ELSE ENDIF and DO WHILE (only operand supported) macros.
The macros have also been designed to allow the following extensions to the standard structured constructs.

IF CASE ANALYSIS

DO WHILE CASE ANALYSIS

All of the above structures can be implemented using the following macros:

IF, ELSE, ENDIF, DO, ENDDO, COND

An explanation and examples of each of these constructs is listed in the following pages.
IF MACRO SET

The IF THEN ELSE construct can be implemented as follows:

    IF condition THEN
        code if condition is true
    ELSE (optional)
        code if condition is false
    ENDIF

A condition may be specified as any legitimate operand of the
COND macro (except the ELSE operand). A discussion of conditions
and the COND macro is presented in a later section. THEN may be
coded following the condition, but it is only treated as a
comment. If used, it must be preceded by a space. ELSE is
optional; if the condition is not true and ELSE is not coded,
control is passed to the next instruction following the ENDIF.

IF COND COND (ELSE) ENDIF.

This macro set supports a case structure of the following
form:

```
* *
* **********************
* *
* **********************
* *
* **********************
* *
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
**
```


Shown below is an example of how this structure may be implemented in coding sequence.

```
IF
  COND condition-1
    code if condition-1 is true
  COND condition-2
    code if condition-2 is true
  .
  .
  .
  COND condition-n
    code if condition-n is true
  COND ELSE (implies NOT all of the previous conditions)
    code for ELSE condition
ENDIF
```

The coding rules for the IF COND COND (ELSE) ENDIF structure are:

1. Coding IF with no operands signifies the beginning of the IF COND COND (ELSE) ENDIF structure.

2. At least one COND with any operand other than ELSE (see COND description) must be coded immediately following IF macro.

3. Every IF, whether in the IF COND COND (ELSE) ENDIF structure or IF (THEN) ELSE structure, must have a corresponding ENDIF which signifies the end of the sequence of code.

4. The rules for the COND ELSE are:
   a. At least one other COND (not having ELSE as an operand) must precede the COND ELSE.
   b. The COND ELSE must be the last COND coded in the particular sequence. Therefore, only one COND ELSE per sequence is allowed.

General Information about the IF COND COND (ELSE) ENDIF

1. Only the code associated with (i.e., following) the first true condition will be executed. It is your responsibility to ensure that this will produce the desired results.

2. The COND ELSE is optional; however, if none of the COND conditions is true and the COND ELSE is specified, the code following the COND ELSE will be executed. If none of the COND conditions is true and COND ELSE was not
Refer to the COND section for a discussion of conditions. The WHILE operand of the DO, however, should be coded either

WHILE=simple condition

as a keyword parameter for simple conditions (i.e., conditions not containing an AND, OR, ANDIF, or ORIF) or

WHILE,compound condition

as a positional parameter for compound conditions.

DO WHILE Case Analysis

This macro set supports a loop case structure of the following form:

```
*                          *
***************>** ** ** ********************* * * * ** * * * ** ************ * ************** * * * ** * * * ** * * * ** * * * ** ************** ** * ** * ** * ****************************************************
```

It may be implemented in a coding sequence of the following type.

```
DO
COND  condition-1
  code if condition-1 is true
COND  condition-2
  code if condition-2 is true
...  ...
COND  condition-n
        code if condition-n is true
ENDDO
```

The loop is terminated when none of the COND conditions is associated with the DO ENDDO is true. In this case, control is passed to the first instruction following the ENDDO.

The coding rules are

1. A DO with no operands signifies the beginning of a DO WHILE case structure sequence.
2. Every DO must have a corresponding ENDDO.
3. COND within a DO may not be coded with the ELSE parameter. An assembly error will be generated if you attempt this.
4. Only the code associated with the first true COND condition will be executed on each pass through the loop. It's your responsibility to make sure that this will produce the desired results.

**CODING OF CONDITIONS**

A condition may be coded as an operand of the DO or IF macros for simple IF THEN ELSE or DO WHILE structures, or as an operand of the COND macro in the IF or DO WHILE case analysis.

**Simple Conditions**

A simple condition may be coded as any of the following four types:

1. A condition code value in parentheses—this would be used to test the value of the condition code as set by a previous instruction. For example,

   ```
   SR  R15,R15
   CLC  FIELD1,FIELD2
   IF  (2)  THEN
   ```
LA R15,4
ENDIF

After execution of this code, if the condition code was set to 2 as a result of the CLC instruction (i.e., if FIELD1 > FIELD2) then R15 would contain a 4; otherwise R15 would remain 0.

IF CASE ANALYSIS EXAMPLE

Suppose we wish to compare FIELD1 to FIELD2 and set R15 to 0 if the fields are equal, to 4 if FIELD1 < FIELD2, or to 8 if FIELD1 > FIELD2.

CLC FIELD1,FIELD2
IF
COND (8) (FIELD1 = FIELD2?)
SR R15,R15
COND (4) (FIELD1 < FIELD2?)
LA R15,4
COND (2) (FIELD1 > FIELD2?)
LA R15,8
ENDIF

The above code would produce the desired results.

2. The second type of simple condition is an extended mnemonic in parentheses. The ones provided by the macros are presented in Table 1. User-defined mnemonics can also be used provided that they have been equated previously in the assembly.

Table 1--Extended Mnemonics

<table>
<thead>
<tr>
<th>CASE</th>
<th>MNEMONICS</th>
<th>MEANING</th>
<th>complement</th>
</tr>
</thead>
<tbody>
<tr>
<td>after compare</td>
<td>H, GT</td>
<td>high, greater than</td>
<td>NH, LE</td>
</tr>
<tr>
<td>instructions</td>
<td>L, LT</td>
<td>low, less than</td>
<td>NL, GE</td>
</tr>
<tr>
<td></td>
<td>E, EQ</td>
<td>equal</td>
<td>NE</td>
</tr>
<tr>
<td>after arithmetic</td>
<td>P</td>
<td>plus</td>
<td>NP</td>
</tr>
<tr>
<td>instructions</td>
<td>M</td>
<td>minus</td>
<td>NM</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>zero</td>
<td>NZ</td>
</tr>
<tr>
<td></td>
<td>O</td>
<td>overflow</td>
<td>NO</td>
</tr>
<tr>
<td>after test under</td>
<td>O</td>
<td>ones</td>
<td>NO</td>
</tr>
<tr>
<td>mask instructions</td>
<td>M</td>
<td>mixed</td>
<td>NM</td>
</tr>
<tr>
<td></td>
<td>Z</td>
<td>zeros</td>
<td>NZ</td>
</tr>
</tbody>
</table>
Consider the following DO WHILE example:

```
LA    R9,40
S     R9,='F'4'
DO    WHILE=(NM)
     CALL SUBROUT,((R9))
S     R9,='F'4'
ENDDO
```

The effect of this code is to test the value of the condition code in the DO macro following each subtract instruction and to execute the code within the loop until R9 is negative. When R9 does become negative, control would be passed to the first instruction following the ENDDO.

3. The third type of simple condition is any instruction setting the condition code (other than a comparison instruction) followed by a desired condition code value. Note that in the previous example, it was necessary to include the subtract instruction both just before the loop was entered and also as the last instruction within the loop. This type of situation can frequently be avoided by using an arithmetic instruction as a simple condition. The format of this type of condition is

(cc setting instruction, mnemonic or condition code value)

For example

```
LA    R9,40
DO    WHILE=(S,R9,='F'4',NM)
     CALL SUBROUT,((R9))
ENDDO
```

This example has the same effect as the previous example. Notice that the opcode and all operands are separated by commas.

(Use in the DO WHILE case structure.)

Suppose we want a loop to continue until R10 contains a negative number. Furthermore, every time we enter the loop, we wish to call ZEROPROC if R10 equals 0 or POSPROC if R10 is positive. The following code would implement this idea.

```
DO
  COND (LTR,R10,R10,Z)
  CALL ZEROPROC
  COND (LTR,R10,R10,P)
  CALL POSPROC,((R10))
ENDDO
```
4. The fourth condition and last type of simple condition is a compare instruction. This is coded for a two operand compare instruction as shown on the following page.

\[(\text{comparison, operand1, extended mnemonic, operand2})\]

\[
\begin{align*}
\text{opcode} & \quad \text{or} \quad \text{condition code value}
\end{align*}
\]

A comparison instruction is any instruction that has an opcode mnemonic beginning with 'C' (i.e., CLI, CP, etc) and sets the condition code.

A three operand compare instruction is also allowed. It is coded the same as a two operand comparison with the condition code value (or mnemonic) placed between the second and third operands.

An example of the use of this type of condition might be to process update transactions based on a field called TRANCODE, where A=add, C=change, and D=delete.

The following code could implement the processing of a transaction:

\[
\text{IF} \quad \text{COND} \quad \text{CLI, TRANCODE, EQ, C'A'} \quad \text{(TRANCODE = A?)}
\]

\[
\text{CALL ADD} \quad \text{(IF YES CALL ADD)}
\]

\[
\text{COND} \quad \text{CLI, TRANCODE, EQ, C'C'} \quad \text{(TRANCODE = C?)}
\]

\[
\text{CALL CHANGE} \quad \text{(IF YES CALL CHANGE)}
\]

\[
\text{COND} \quad \text{CLI, TRANCODE, EQ, C'D'} \quad \text{(TRANCODE = D?)}
\]

\[
\text{CALL DELETE} \quad \text{(IF YES CALL DELETE)}
\]

\[
\text{COND ELSE} \quad \text{(NOT A, C, OR D?)} \quad \text{(IF NOT, CALL ERROR)}
\]

\[
\text{ENDIF}
\]

The code beneath the COND ELSE is executed when none of the previous conditions is true. In this case, this would be an error condition. If none of the conditions is true and the COND ELSE is omitted, the program would abend.

**Compound Conditions**

A compound condition is two or more simple conditions separated by logical operators. The valid logical operators are AND, OR, ANDIF, and ORIF. The compound conditions can be broken into three types. **TYPE 1** conditions are those containing only the AND and OR operators. **Type 1** conditions will satisfy most condition configurations. A valid **Type 1** condition would be

\[(\text{TM, BYTE, B'00110001', NZ}), \text{AND}, (\text{CLI, FLAG, EQ, C'Y'})\]
Conditions can be further compounded as follows:

A OR B AND C

The logical operators AND and OR are evaluated in the logical hierarchy—AND before OR; therefore, Type 1 conditions are disjunctive. Thus, the above condition has implied parentheses as

A OR (B AND C)

Similarly,

A AND B OR C AND D AND E OR F

would evaluate with parentheses implied as follows:

(A AND B) OR (C AND D AND E) OR (F)

This hierarchy of evaluation can be altered with the use of Type 2 and Type 3 conditions. Type 2 conditions are those containing the logical operator ANDIF. This operator has the effect of putting parentheses around the Type 1 group on either side of the ANDIF. The operator has no logical difference from the AND operator unless at least one of the Type 1 group on either side contains an OR. Thus, the previous example can be altered as in the following:

(A) ANDIF (B OR C) ANDIF ((D AND E) OR F)

(A) ANDIF (B OR C) ANDIF (D) ANDIF (E OR F)

As you can see, when ANDIF is used exclusively instead of AND, the result is a purely conjunctive condition.

A Type 3 condition is one containing the logical operator ORIF. This operator has the effect of placing parentheses around the Type 1 or Type 2 condition on either side of the ORIF. It has no logical difference from the OR operator unless at least one of the groups on either side of the ORIF is a Type 2 condition. The following examples show the effect of the ORIF.

(((A OR B) ANDIF (C OR D)) ORIF ((E AND F) OR G)

(((A AND B) OR (C AND D)) ORIF ((E OR F) ANDIF (G OR H))

ORIF (((I) ANDIF (J OR K OR L) ANDIF (M)))

The following examples illustrate when the ANDIF and ORIF have no logical difference from the AND and OR operators. These situations should be avoided.

(((A AND B) ANDIF ((C AND D))

(((A AND B) OR C) ORIF ((D AND E) OR F)
To illustrate the use of compound conditions, we will extend the example of handling an update transaction to include processing a series of update transactions (until an end-of-file condition occurs). Recall that TRANCODE=A for an add transaction, TRANCODE=C for a change transaction, TRANCODE=D for a delete transaction, and EOFFLAG=N means not end of transaction file.

```plaintext
DO
  COND (CLI,EOFFLAG,EQ,C'N'),AND,(CLI,TRANCODE,EQ,C'A')
    CALL ADD
  COND (CLI,EOFFLAG,EQ,C'N'),AND,(CLI,TRANCODE,EQ,C'C')
    CALL CHANGE
  COND (CLI,EOFFLAG,EQ,C'N'),AND,(CLI,TRANCODE,EQ,C'D')
    CALL DELETE
  COND (CLI,EOFFLAG,EQ,C'N')
    CALL ERROR
ENDDO
```

**NESTING**

The occurrence of any structured sequence is allowed in the code associated with another "higher level" structured sequence. This nesting may occur up to 50 levels deep.

As an example, we can further "clean up" the update example by the following use of nesting.

```plaintext
GET NEXTTRAN,TRANBUFF
DO WHILE=(CLI,EOFFLAG,EQ,C'N')
  IF
    COND (CLI,TRANCODE,EQ,C'A')
      CALL ADD
    COND (CLI,TRANCODE,EQ,C'C')
      CALL CHANGE
    COND (CLI,TRANCODE,EQ,C'D')
      CALL DELETE
    ELSE
      ENDIF
  ENDIF
GET NEXTTRAN,TRANBUFF
ENDDO
```

The code within each structured sequence should always be indented in order to improve readability.
DOCUMENTATION STANDARDS

Specific documentation standards to be followed in all of the CSCI 446 programming assignments are outlined in the following pages.

In general, all programming output should be double spaced (where appropriate) and should be centered on the page to make it more readable. Descriptive page and column headings should be printed at the top of each page of output and must include the current processing date and consecutive page numbers. For inquiry processing, appropriate and complete messages must be given for each inquiry performed (ie. - error messages, "not found" messages, informational messages, etc.).

Please remember that this is not your first programming class and you are not beginning programmers anymore. You should realize that once you are employed, you will be producing reports for users who are interested in the contents of the report but are not familiar with the process of producing the report. In other words, you will be expected to produce output that is complete and written in English so that it can be easily understood by any end user. Since we are trying to prepare you for future employment, the assignments are not always going to explain in detail what you are to do, so use your common sense in completing them.
JCL DOCUMENTATION

1. All JCL documentation must be placed within star boxes.

2. There must be a main documentation box following the JOB card statement that explains what the job does.

3. Following each EXEC statement, there must be a documentation box that:
   - Gives the step name.
   - Explains the function of the step.
   - Lists the DD card names with a description.

The following is an example of JCL documentation:

```plaintext
//EXAMPLE JOB,'JCL DOC',TIME=(0,5),REGION=256K

FUNCTION: THIS JOB CONSISTS OF THE FOLLOWING JOB

STEPS:

STEP1 - TO COMPILE A COBOL PROGRAM.

STEP2 - TO LINK EDIT THE OBJECT MODULE.

STEP3 - TO EXECUTE THE LOAD MODULE.

*********************************************************************

STEP1 EXEC PGM=IKFCBLO0,PARM='APOST,STATE,FLOW=10'

STEP1 COMPILE THE COBOL SOURCE CODE PRODUCING

AN OBJECT MODULE.

*********************************************************************

STEPLIB INPUT: LOCATION OF COBOL COMPILER PROGRAM

SYSIN INPUT: THE COBOL SOURCE CODE

SYSLIN OUTPUT: OBJECT MODULE CREATED BY COMPILER

SYSPRINT OUTPUT: SOURCE LISTING & COMPILER MESSAGES

SYSUT1-4 WORK: COMPILER WORK AREAS

*********************************************************************

STEPLIB DD DSN=SYS2.VSC.V124.VSCOLIB,DISP=SHR

SYSPRINT DD SYSOUT=A

SYSUT1 DD UNIT=PUB,SPACE=(460,(600,200))

SYSUT2 DD UNIT=PUB,SPACE=(460,(600,200))

SYSUT3 DD UNIT=PUB,SPACE=(460,(600,200))

SYSUT4 DD UNIT=PUB,SPACE=(460,(600,200))

SYSLIN DD DSN=&&OBJMOD,UNIT=PUB,DISP=(NEW,PASS),

SPACE=(800,(400,100))

*** (Continued on next page) ***
```
**COBOL SOURCE CODE**

**STEP2** EXEC PGM=IEWL,COND=(0,NE,STEP1)

**********************************************************************

**STEP2** LINK EDIT THE OBJECT MODULE CREATED IN STEP1 INTO A LOAD MODULE.

**SYSLIN** INPUT: OBJECT MODULE CREATED IN STEP1

**SYSLIB** INPUT: REFERENCE LIBRARY

**SYSPRINT** OUTPUT: DIAGNOSTIC MESSAGES

**SYSLMOD** OUTPUT: LOAD MODULE CREATED

**********************************************************************

**SYSLIN** DD DSN=&OBJMOD,DISP=(OLD,DELETE)

**SYSLIB** DD DSN=SYS2.VSC.V124.VSCLIB,DISP=SHR

**SYSLIB** DD DSN=SYS2.CALLIB,DISP=SHR

**SYSLMOD** DD DSN=T901234.MYLIB(PGM1),UNIT=DISK,

**OUTFILE** DD DSN=T901234.OUTPUT.FILE,DISP=(,KEEP,DELETE),

**OUTFILE** DD DSN=T901234.OUTPUT.FILE,DISP=(,KEEP,DELETE),

DCB=(RECFM=FB,LRECL=80,BLKSIZE=800)
COBOL DOCUMENTATION

IDENTIFICATION DIVISION

There must be a main documentation box that contains:

1. Program name
2. Function of the program
3. Input with JCL DD card names
4. Output with JCL DD card names
5. Entry and exit conditions
6. Notes listing special features

The following is an example of IDENTIFICATION DIVISION documentation:

****************************************************************************************************
* *
* PROGRAM: PGM1 *
* *
* FUNCTION: TO PRINT A REPORT USING AN OS FILE. *
* *
* INPUT: OSDATA - AN OS FILE CONTAINING EMPLOYEE INFORMATION. *
* *
* OUTPUT: PRINTER - A REPORT LISTING EMPLOYEE INFORMATION. *
* *
* ENTRY/EXIT CONDITIONS: NONE. *
* *
* NOTES: NONE. *
*
****************************************************************************************************

DATA DIVISION

Before each file definition (FD) there must be a documentation box.

Within the documentation box there must be a brief description of the file.

The following is an example of FD documentation:

****************************************************************************************************
* *
* THIS IS AN OS INPUT FILE CONTAINING EMPLOYEE RECORDS THAT CONTAIN INFORMATION TO BE PRINTED ON A REPORT. *
* *
****************************************************************************************************

FD OS-INFILE PIC X(80).
WORKING-STORAGE SECTION

- A documentation box is to be place at the beginning of the WORKING-STORAGE SECTION.

- Each field defined in the WORKING-STORAGE SECTION must be listed in the documentation box with a brief description.

The following is an example of WORKING-STORAGE SECTION documentation:

******************************************************************************
* *
* INPUT-EOF-FLAG IS SET TO 'Y' WHEN END OF INPUT FILE OCCURS. *
* *
* WS-INPUT-REC DEFINES THE OS INPUT RECORD FORMAT. *
* *
******************************************************************************

01 INPUT-EOF-FLAG.
  .

01 WS-INPUT-REC.
  .

PROCEDURE DIVISION

- At the beginning of the PROCEDURE DIVISION, there must be a documentation box for the main routine.

- Within this documentation box, there must be the
  1. Name
  2. Function
  3. Logic - pseudocode.

The following is an example of a documentation box for the main routine:

******************************************************************************
* *
* NAME: 000-MAIN ROUTINE *
* *
* FUNCTION: TO CONTROL THE FLOW OF EXECUTION AND INVOKE A SUBROUTINE TO PROCESS THE RECORDS. *
* *
* LOGIC: READ FIRST RECORD *
*   INVOKE 010-PROCESS-RTN *
* *
******************************************************************************
- There must also be documentation boxes at the beginning of each subroutine.

- A subroutine documentation box must include the same information as the main routine documentation box; however, if the subroutine consists of a simple process (i.e. building a print line), the logic may be omitted.

The following is an example of a documentation box for a subroutine that requires logic/pseudocode:

```
**************************************************************************
* *
* NAME: 010-PROCESS-RTN *
* *
* FUNCTION: TO CONTROL RECORD PROCESSING AND TO INVOLVE SUBROUTINES TO PRINT PAGE HEADERS, BUILD DETAIL LINES, AND PRINT DETAIL LINES. *
* *
* LOGIC: INVOKE 090-HEADER-RTN READ FIRST INPUT RECORD DOWHILE NOT END OF FILE INVOKE 030-BUILD-RTN INVOKE 040-PRINT-RTN READ NEXT RECORD ENDDO *
* *
**************************************************************************
```

The following is an example of a documentation box for a subroutine that does not require logic/pseudocode:

```
**************************************************************************
* *
* NAME: 030-BUILD-RTN *
* *
* FUNCTION: TO BUILD THE DETAIL LINE OF THE REPORT. *
* *
**************************************************************************
```

This subroutine would consist of a series of MOVE statements.
IDCAMS DOCUMENTATION

1. Follow the JCL documentation standards.
2. Line document each IDCAMS parameter.
3. Each IDCAMS stepname should reflect the function of the step (ie. - DLET, DEFINE, REPRO, etc.).

The following is an example of IDCAMS line documentation:

```
DEFINE CLUSTER (       /* DEFINE VSAM KSDS CLUSTER */
  NAME(T901234.FA6.KSDS) /* KSDS CLUSTER'S NAME */
  VOLUMES(ACA004)       /* RESIDES ON VOL=SER=ACA004 */
  TRACKS(1 1)           /* SPACE ALLOCATION */
  INDEXED               /* CLUSTER IS INDEXED */
  KEYS(9 0)             /* PRIME KEY LEN=9 DISPLACEMENT=0 */
  RECORDSIZE(80 80))    /* RECORD LENGTH IS FIXED 80 BYTES */
```

VSAM COBOL DOCUMENTATION

1. Follow the COBOL documentation standards.
2. When referring to a file, be sure to describe what type of file it is (ie. - OS; VSAM KSDS, ESDS, or RRDS).

The following is an example of IDENTIFICATION DIVISION documentation for a VSAM COBOL program:

```
*** PROGRAM:       PGM2.      *
*** FUNCTION:      TO LOAD AN OS FILE INTO A VSAM KSDS FILE. *
*** INPUT:         OSFILE - AN OS FILE CONTAINING EMPLOYEE INFORMATION. *
*** OUTPUT:        KSDSFILE - A VSAM KSDS FILE WHICH IS INITIALLY EMPTY. *
*** ENTRY/EXIT CONDITIONS:   NONE. *
*** NOTES:         NONE. *
```
Subroutines are not required for the VSAM in Assembler program, but internal subroutines may be used. If you are using internal subroutines, a documentation box must be placed at the top of a new page at the beginning of each subroutine. If you are not using internal subroutines, divide the program into logical sections and place a documentation box at the beginning of each section. Each logical section with its documentation must begin at the top of a new page.

- At the beginning of the program, there must be a main documentation box containing:

1. Function — a brief description of what the program does.
2. Input — a brief description of the input file(s) with JCL DD card name(s).
3. Output — a brief description of the output file(s) with JCL DD card name(s).
4. Entry/Exit Conditions.
5. Notes — any special feature(s) of the program.
6. Pseudocode.

The following is an example of a main documentation box:

```
************************************************************
* *
* FUNCTION: TO PERFORM OS FILE TRANSACTIONS AGAINST A VSAM KSDS FILE. *
* *
* INPUT: OSFILE - AN OS TRANSACTION FILE. *
* INKSDS - A VSAM KSDS FILE CONTAINING EMPLOYEE INFORMATION. *
* *
* OUTPUT: PRINT - A REPORT LISTING THE TRANSACTIONS PERFORMED AGAINST THE VSAM KSDS FILE. *
* *
* ENTRY/EXIT CONDITIONS: NONE. *
* *
* NOTES: NONE. *
*
*** (Continued on following page) ***
```
* PSEUDOCODE: OPEN INPUT/OUTPUT FILES
  * IF OPEN NOT SUCCESSFUL
  * BUILD AND WRITE ERROR MESSAGE
  * ELSE
  * READ FIRST TRANSACTION RECORD
  * DOWHILE NOT END OF FILE
  * PERFORM TRANSACTION ON VSAM KSDS FILE
  * IF TRANSACTION UNSUCCESSFUL
  * BUILD AND WRITE ERROR LINE
  * ELSE
  * BUILD AND WRITE DETAIL LINE
  * ENDF
  * READ NEXT TRANSACTION RECORD
  * ENDDO
  * ENDF
  * CLOSE INPUT/OUTPUT FILES
  *
******************************************************************************

- Each line of code must also have line documentation.

- There must be a documentation box at the beginning of each internal subroutine or logical section of the program containing:
  1. Function.
  2. Pseudocode.

- For any of the special routines (ie. - LERAD or SYNAAD routines), there must be a documentation box at the beginning containing:
  1. Name.
  2. Function.
  3. Pseudocode.

- Each ACB (Access Control Block) with its EXLST, RPL (Request Parameter List), and DCB (Data Control Block) must have a documentation box containing a brief description of what it is.

The following is an example of ACB documentation:

******************************************************************************
* THIS IS AN ACB (ACCESS CONTROL BLOCK) WITH ITS EXLIST FOR A VSAM KSDS FILE WHICH IS TO BE PROCESSED BY ALTERNATE KEY.
******************************************************************************

- Finally, the storage area of the program must also be line documented.
IMSCONTROL BLOCK GENERATION DOCUMENTATION

- At the beginning of the DBD (Data Base Descriptor) control block generation, there must be a documentation box containing:

1. Name - of the DBD.
2. Function.
3. Hierarchical Structure.

The following is an example of DBD generation documentation:

*************************************************************** **
* NAME:    MYDBD*
* FUNCTION: TO GENERATE A DBD (DATA BASE DESCRIPTOR) USING ASSEMBLER MACROS.
* HIERARCHICAL STRUCTURE:
* CONSULTANT
* |
* CLIENT   DEPARTMENT   EMPLOYEE HISTORY
* CONTRACT
* SERVICES PERFORMED
** ***************
- For each PSB (Program Status Block) control block generation, there must be a documentation box containing:

1. Name - of the PSB.
2. Function.

The following is an example of PSB generation documentation:

```
*************************************************************** **
* NAME: CONSULTL                                    *
* FUNCTION: TO GENERATE A PSB (PROGRAM SPECIFICATION   *
* BLOCK) CONTAINING ONE PCB (PROGRAM COMMUNICATION   *
* BLOCK) USING ASSEMBLER Macros. THIS PSB             *
* WILL BE USED FOR LOADING THE DATA BASE              *
* SEGMENTS.                                          *
*************************************************************** **
```

IMS COBOL PROGRAM DOCUMENTATION

- Follow the COBOL documentation standards with a few additions and modifications.
- The main documentation box must also include the ENTRY/EXIT CONDITIONS of the program.

The following is an example of IDENTIFICATION DIVISION documentation for an IMS COBOL program:

```
*************************************************************** **
* PROGRAM: PGM3.                                        *
* FUNCTION: TO PERFORM INQUIRIES AGAINST A DATA BASE      *
* USING A TRANSACTION FILE CONTAINING RECORDS            *
* WITH THE INFORMATION NEEDED TO MAKE AN INQUIRY. A REPORT IS GENERATED LISTING THE RESULTS OF THE INQUIRIES.
* INPUT: TRANSACT - AN OS TRANSACTION FILE.              *
* MYDBD - A CONSULTANT DATA BASE.                        *
* OUTPUT: PRINTER - A REPORT LISTING THE INFORMATION     *
* PERTAINING TO THE INQUIRIES.                           *
* ENTRY/EXIT CONDITIONS: PARAMETER LIST CONTAINS ADDRESS *
* OF THE PCB-MASK.                                      *
* NOTES: USING THE IMS SIMULATOR.                        *
*************************************************************** **
```
- This program contains a LINKAGE SECTION which must have a documentation box that explains the contents of this section.

The following is an example of LINKAGE SECTION documentation:

```
**************************************************************
* PCB-MASK THIS IS A PROGRAM COMMUNICATION BLOCK MASK USED *
* TO ACCESS THE STATUS INFORMATION GENERATED *
* FOLLOWING EACH CALL TO THE DATA BASE. *
* 
**************************************************************
```

01 PCB-MASK.

- The logic/pseudocode for an IMS program must contain the actual commands used to access the data base. (ie. - GU, GN, etc.)

The following is an example of an IMS subroutine documentation box:

```
**************************************************************
* NAME: 050-LOOP-RTN. *
* 
* FUNCTION: TO CONTINUE AN INQUIRY AGAINST THE DATA BASE *
* BY PERFORMING CALLS TO THE DATA BASE IN A LOOP. *
* 
* LOGIC: GN CONSULTANT*U *
* CLIENT *
* IF CALL UNSUCCESSFUL *
* SET EXIT FLAG *
* ELSE *
* BUILD AND WRITE DETAIL LINE *
* ENDIF *
* 
**************************************************************
```
CICS BMS MAP DOCUMENTATION

- For each map definition, there must be a documentation box containing:

  1. Name - of the map.
  2. Function.
  3. List of macros used.

The following is an example of BMS (Basic Mapping Support) map documentation:

***************************************************************
* *
* NAME: MAP1. *
* *
* FUNCTION: TO DEFINE THE FIELDS OF A MAIN MENU SCREEN. *
* *
* MACROS: DFHMSD - TO DEFINE THE MAP SET. *
* DFHMDI - TO DEFINE A MAP. *
* DFHMDF - TO DEFINE A FIELD. *
* *
************************************************************

CICS COBOL PROGRAM DOCUMENTATION

- Follow the COBOL documentation standards with the following additions:
- The main documentation box must also include the ENTRY/EXIT CONDITIONS of the program.

The following is an example of IDENTIFICATION DIVISION documentation for a CICS COBOL program:

*****************************************************************
* *
* PROGRAM: PGM4. *
* *
* FUNCTION: TO SEND THE MAIN MENU MAP TO THE SCREEN OF A TERMINAL AND ASK THE USER TO ENTER A SELECTION. *
* *
* INPUT: COB.SYSLIB - A COPY LIBRARY CONTAINING THE MAP. *
* *
* OUTPUT: A FORMATTED SCREEN. *
* *
*** (Continued on following page) ***
**ENTRY/EXIT CONDITIONS:** DEPENDS ON COMMUNICATION AREA.  
**NOTES:** USING CICS AND CODED IN PSEUDOCONVERSATIONAL MODE.

- A CICS program requires a LINKAGE SECTION that must have a documentation box containing a brief description of each field.

The following is an example of a LINKAGE SECTION documentation box:

```
    ******************************************
    * DFHCOMMAREA THE COMMUNICATION AREA USED BY THE PROGRAM *
    * TO PASS A 1-BYTE PARAMETER WHICH IS USED AS *
    * A FLAG. *
    ******************************************

    01 DFHCOMMAREA PIC X.
    .
    .
    .
```

ACCESS METHOD SERVICES (AMS)
JCL AND PARAMETER CODING
ACCESS METHOD SERVICES (AMS)
JCL AND PARAMETER CODING

Access Method Services (AMS) is the single "utility" for creating, maintaining, and deleting VSAM data sets. The program AMS executes is called IDCAMS.

The JCL necessary to use AMS is simple; all a user needs is an EXEC statement executing IDCAMS, a SYSPRINT DD statement for AMS message output, and a SYSIN DD statement identifying the control statements. Depending on the type of operation being performed, the user might also have to include additional DD statements referencing clusters; of course, non-VSAM data sets must be specified as usual. The following skeleton JCL can be used to run AMS at NIU:

```plaintext
//jobname JOB ,'name',REGION=nnnK
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//ddname DD (parameters as required)
//SYSIN DD *

(IDCAMS commands and parameters)
```

Most AMS operations require 200K or more; 240K should be sufficient. Coding the various IDCAMS parameters is also easy. Selecting which parameters to code is the hard part. All IDCAMS operations consist of a command (e.g. DEFINE), a positional parameter (CLUSTER) and additional keyword parameters. (I combine the command and positional parameter into the term 'command'; 'parameter' refers to keyword parameters.) Examples of these commands include DEFINE CLUSTER and DEFINE USERCATALOG, which do exactly what they say. An example of a parameter is the NAME parameter, which can be used along with the DEFINE CLUSTER command to assign a name to the cluster. An example of the coding of this much is

```plaintext
//SYSIN DD *
DEFINE CLUSTER (NAME(VSAMKSDS))
```

Note that IDCAMS control statements must begin between columns 2 and 16 inclusive. Note also that the parameters of the DEFINE CLUSTER command are themselves enclosed in parentheses. This is not always the case; the parameters used in other commands are not always enclosed. Pay careful attention to this matter.

To allow for a large number of parameters (and to increase readability), control statements can be continued on the next line. To do this, place a hyphen (-) as the last non-blank
character on the line that is to be continued. For example, the above control statement could (and at NIU should) be recoded as

```c
//SYSIN  DD *
DEFINE CLUSTER (-
  NAME(VSAMKSDS))
```

Documentation can be inserted almost anywhere within a control statement. It must begin with the characters /* and end with */. The following example shows the NIU standard (Note the location of the hyphen):

```c
//SYSIN  DD *
/***************************************************************************/
/*
/* THIS ILLUSTRATES IDCAMS CONTROL STATEMENT DOCUMENTATION */
/*
***************************************************************************/
DEFINE CLUSTER ( /* DEFINE A CLUSTER */ -
  NAME(VSAMKSDS) /* CLUSTER NAME VSAMKSDS */ -
  other parm /* other parm doc */ -
  last parm /* last parm doc */ -
)
```

Two things to remember: (1) the */ cannot begin in column 1, and (2) do not put a hyphen after the comment of the final line of the control statement.

**DEFINING A CLUSTER**

The JCL for defining a cluster is

```c
//
// JOB
// EXEC PGM=IDCAMS
// SYSPRINT DD SYSOUT=A
//SYSIN DD *
DEFINE CLUSTER (parameters)
```

The required parameters are

- **NAME (name)**
  - Defines the name of the cluster (1-44 characters)
- **VOLUMES (volser list)**
  - Identifies the volume(s) on which the cluster is to be defined
- **TRACKS(primary secondary) or CYLINDERS(primary secondary) or RECORDS(primary secondary)**

One of these must be coded so that IDCAMS will know how big the cluster needs to be. The primary space is allocated once, the secondary up to 15 times.
INDEXED|NONINDEXED|NUMBERED

The parameter identifies the type of cluster being defined:

INDEXED --KSDS; NONINDEXED--ESDS; and NUMBERED--RRDS

Some optional parameters (these all have defaults) are

KEYS(length offset)    default KEYS(64 0)

This applies to KSDS only; it specifies the length of each record's key and its displacement into the record. It defaults to a key length of 64 with an offset of 0.

REPLICATE|NOREPLICATE    default NOREPLICATE

The parameter specifies that each index record should be written on its track as many times as it will fit. This reduces rotational delay by a significant degree, but at the expense of DASD space. The default yields slower access but better use of space. This parameter applies only the KSDS.

IMBED|NOIMBED    default NOIMBED

IMBED specifies that the sequence block for a control area is to be written to the first track of the control area, as many times as it will fit. A control area is never bigger than one cylinder; therefore, seeking to a control area's sequence set block automatically seeks to the control area when IMBED is specified. NOIMBED negates the effect of IMBED. It is used with KSDS only.

RECORDSIZE(average maximum)    default RECORDSIZE(4089 4089)

It defines the average and maximum record sizes for the cluster. For fixed-length records, 'average' and 'maximum' must be equal. If you know the cluster's record size(s), code this parameter; otherwise omit it and consider the cluster to have U(ndefined) type records.

SPANNED|NONSPPANNNED    default NONSPANNNED

SPANNED allows a single record to span over control-interval boundaries. NONSPANNNED, the default, disallows spanned records.

FREESPACE(ci% ca%)    default FREESPACE(0 0)

This parameter specifies the amount of distributed free space to be left in the cluster after initial load and after any split occurs. For clusters that will have many records inserted, some free space should be specified; for static files, use the default of no free space.
REUSE|NOREUSE  default NOREUSE

REUSE allows the cluster to be used over and over as a temporary cluster. Each time it is opened, the data currently there is effectively deleted. REUSE is useful if the cluster is to be used as a temporary workfile over and over. NOREUSE specifies that the cluster is permanent.

CONTROLINTERVALSIZE(size)

Specifies the size, in bytes, of each control interval. The allowable sizes are 512, 1024, 2048, and 4096 bytes. The default is the size IDCAMS thinks will be best, based on the cluster's RECORDSIZE and BUFFERSPACE. Unless you are sure of a size to choose, it is usually better to let IDCAMS make the decision. Remember that the unit transferred between auxiliary and virtual storage is the control interval.

BUFFERSPACE(size)

Specifies the amount of space to be allocated for buffers. The default is an attempt by VSAM to get enough buffer space for two control intervals. Essentially, the more random the access to the cluster, the more buffer space should be specified. A large BUFFERSPACE provides faster access at the expense of main memory.

MASTERPW(password) CONTROLPW(password)
UPDATEPW(password) READPW(password)

Specifies that the cluster is to be password-protected and also tells IDCAMS what the passwords are. Each password is one to eight characters long, enclosed in quotes if it contains any special characters. If a cluster is password-protected, the correct password must be specified before access is allowed. If the job's JCL (or IDCAMS control statements) do not contain the correct password, the system operator will be prompted for it. The READPW password must be specified for reading the cluster; the UPDATEPW for writing to it, the CONTROLPW for using control-interval access to the cluster, and the MASTERPW for moving or deleting the cluster. Specifying a high-level password automatically allows all types of 'lower' access. For example, correctly supplying the UPDATEPW allows READ access also. Do not use any password here at NIU.

ATTEMPTS(number)  default ATTEMPTS(2)

Indicates the number of attempts the system operator is to get in supplying the password.

CODE(codename)

When the system operator is prompted for a password, the prompting message contains the name of the cluster the password is for. If you don't want the operator to know what cluster is being
entered, include CODE(codename). The code name, which is coded like a password, will appear in the prompt message to the operator instead of the cluster name.

OWNER(owner id)

Identifies the owner of the cluster. 'Owner id' is coded like a password. The parameter is used for documentation purposes only.

TO(Julian date)

Specifies an expiration date for the cluster. The cluster cannot be deleted until the Julian date has passed (but remember this when you read about the PURGE parameter of DELETE a few pages later).

FOR(number of days)

Specifies an expiration date of 'number of days' from today

WRITECHECK|NOWRITECHECK default NOWRITECHECK

WRITECHECK causes VSAM to reread every record written to the cluster, and to check what was read back with what should have been written. For critical data, specifying WRITECHECK is reasonable; however, it triples the overhead. NOWRITECHECK negates WRITECHECK.

EXCEPTIONEXIT(module name)

Indicates that an installation-provided error module is to receive control in event of an I/O error.

MODEL(entryname)

Specifies that this cluster's attributes are to be modeled after those of the already-existing cluster specified in the MODEL parameter. This is very useful if, for example, you wish to create a temporary copy of a production cluster to test new programs against.

An ESDS and RRDS contain only one component--the data component. A KSDS, however, contains two separate components--data and index. When defining a KSDS, you can define each component separately. This results in a more efficiently-accessed cluster, since the attributes of a cluster's data component and its index component tend to be different. For example, a control interval size of 1024 might prove to be optimal for a cluster's data component but a terrible size for the index. If you do not define a cluster's data and index components separately, any parameters and values you pick apply equally to both. Defining
the data and index components separately is easy, as the following example illustrates:

```plaintext
//SYSIN DD *
DEFINE CLUSTER (-
  NAME(KSCLUS) /* CLUSTER'S NAME IS KSCLUS */-
  VOLUMES(ACA007) /* RESIDES ON VOL=SER=ACA007 */-
  CYLINDERS(10 5) /* SPACE ALLOCATION */-
  DATA ( /* DEFINE DATA COMPONENT */-
    NAME(KSDATA) /* DATA COMPONENT NAMED KSDATA */-
    KEYS(21 37) /* KEY INFORMATION */-
    RECORDSIZE(121 121) /* FIXED-LENGTH 121 BYTES */-
    FREESPACE(20 10) /* CI = 20%; CA = 10% */-
    CONTROLINTERVALSIZE (2048) /* CI SIZE */-
  )
  INDEX ( /* DEFINE INDEX COMPONENT */-
    NAME(KSNDX) /* INDEX'S NAME IS KSNDX */-
    REPLICATE /* REPLICATE */-
    IMBED /* AND IMBED FOR SPEED */-
    CONTROLINTERVALSIZE(512) /* CI SIZE */-
)```

DELETING A CLUSTER

The DELETE command is used to delete a cluster. Deleting a KSDS deletes both the index and data components. Entire catalogs can also be deleted if you are authorized to do so. (You can delete only your clusters).

The DELETE JCL is

```plaintext
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
```

Required DELETE parameters are

`ENTRYNAME` | `ENTRYNAME/PASSWORD`

Specifies the name of the entry to be deleted. Examples of "entries" are clusters and catalogs. If the name is not unique, you must tell IDCAMS what you are deleting by including a parameter (e.g., CLUSTER or USERCATALOG). The second format is used for password-protected entries.

Some optional parameters are

- `PURGE` | `NOPURGE` (default `NOPURGE`)
- `ERASE` | `NOERASE` (no default)

`PURGE` tells IDCAMS to delete the entry even if its expiration date (see `TO` and `FOR` above) has not yet passed.

`ERASE` causes IDCAMS to physically rewrite the cluster's data com-
ponent with binary zeroes. NOERASE merely removes the cluster's entry from the catalog. In theory, it is possible to read a cluster that was deleted.

COPYING FROM OR INTO A CLUSTER

The REPRO command is used to copy data from a data set to another VSAM or non-VSAM data set, and vice versa. With REPRO, you can do the following:

* copy a VSAM data set to another VSAM data set
* copy a VSAM data set to a sequential data set
* copy a sequential data set to another sequential data set
* copy a sequential or ISAM data set to a VSAM data set
* copy an ISAM data set to a sequential data set
* copy an alternate index (AIX) as if it were a VSAM data set
* copy a sequential, ISAM or VSAM data set into an RRDS

The REPRO JCL is

```
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//inputdd DD required in some instances
//outputdd DD required in some instances
//SYSIN DD *
```

REPRO parameters

If the data set identified in either the 'inputdd' or the 'outputdd' statement is a VSAM cluster, it must have been previously defined.

The required parameters are

INFILE(ddname[/password])|ININDATASET(entryname[/password])

Identifies the input data set or cluster. INFILE must be used if the input data set is non-VSAM.

OUTFILE(ddname[/password])|OUTOUTDATASET(entryname[/password])

Identifies the output data set or cluster; OUTFILE must be used if the output data set is non-VSAM.

ININDATASET/OUTOUTDATASET identifies a VSAM cluster by its DSN (entryname in the VSAM catalog). If this parameter is used, no corresponding DD in the JCL is required.

OUTOUTDATASET(entryname)|OUTOUTDATASET(entryname/password)

Identifies a VSAM cluster by its DSN (entry name in the VSAM catalog). If this parameter is used, no outdd DD is required.
There are a rather large number of optional parameters for the REPRO command, none of which are discussed here.

PRINTING A CLUSTER

The PRINT command is used to list the contents of a cluster. It doesn't matter what cluster type it is; IDCAMS will figure this out and adjust the printout accordingly.

The PRINT JCL is

```
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//inputdd DD optional
//output DD optional
//SYSIN DD *
```

The INDATASET (or INFILE) parameter is used as in REPRO, and it is the only required parameter. An OUTFILE parameter can also be included; if it is omitted, the output is sent to SYSPRINT. There are a dozen or so other optional parameters which can be specified, but we are interested in only one. Unless you specify otherwise, the cluster being printed comes out in hex format (similar to IEBPTPCH TOTCONV=XE). In order to get it out in more easily readable format, include the CHARACTER parameter.

DISPLAYING CATALOG INFORMATION

The LISTCAT command is used to display information contained in a VSAM catalog. You can list all entries in the catalog, or just the entry for a specific cluster.

The LISTCAT JCL is

```
// EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
```

LISTCAT parameters

The LISTCAT command has no required parameters

Some optional commands are

CATALOG(name)

Indicates which VSAM catalog contains the entries to be listed.

NAME|VOLUME|ALLOCATION|HISTORY|ALL

Tells IDCAMS how much information you want displayed. Specifying
NAME, for example, displays only name information. HISTORY displays the history of the entry. ALL displays all information. The parameter defaults to NAME.

ENTRIES(entryname)|ENTRIES(entryname/password)

Names one or more entries to be displayed. If ENTRIES is omitted, all entries will be listed.

CLUSTER|DATA|INDEX|USERCATALOG|etc.

If the name of one of the entries specified in ENTRIES is not unique, you must pick one of these to tell VSAM which entry type is to be displayed.

SUMMARY

This section is by no means an attempt to replace the OS/VS Access Method Services manual. Moreover, it is intended as a quick reference guide that once understood will enable you to read the manual more profitably.
VSAM ALTERNATE INDEX
Three IDCAMS commands are required to create an alternate index (AIX). A description of these commands with parameters and subparameters follows:

1. **DEFINE ALTERNATEINDEX**
   
   ```plaintext
   NAME(AIX name) -
   RELATE(base cluster name) -
   RECORDS/TRACKS/CYLINDERS(ps) -
   KEYS(length displacement) -
   UNIQUEKEY/NONUNIQUEKEY -
   UPGRADE/NOUPGRADE -
   VOLUMES(AIX volser) -
   RECORDSIZE(avg max))
   ```
   
   - **Define the AIX**
   - **Space allocation for AIX**
   - **Alternate key field**
   - **Are there duplicate alternate key values?**
   - **Should AIX be modified when base cluster is modified?**
   - **Volume location**
   - **AIX record size**
   - **(required at NIU)**

2. **DEFINE PATH**
   
   ```plaintext
   NAME(path name) -
   PATHENTRY(AIX name))
   ```
   
   - **Link AIX to base cluster**

3. **BLDINDEX**
   
   ```plaintext
   INDATASET(base cluster name) -
   OUTDATASET(AIX name)
   ```
   
   - **Generate AIX using base cluster**

The following is a coded example demonstrating the use of the above commands:

```plaintext
//STEP1 EXEC PGM=IDCAMSS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
DEFINE AIX (*/
  NAME(T901234.AIX) -
  RELATE(T901234.KSDS) -
  VOLUMES(ACA008) -
  RECORDSIZE(35 60) -
  KEYS(20 7) -
  TRACKS(1 1))
DEFINE PATH (*/
  NAME(T901234.PATH) -
  PATHENTRY(T901234.AIX))
/*
//STEP2 EXEC PGM=IDCAMSS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
BLDINDEX (*/
  INDATASET(T901234.KSDS) -
  OUTDATASET(T901234.AIX)
/*
```
Several coding requirements for processing a VSAM file through an alternate index in a COBOL program are either not documented or are discussed in obscure manuals. The first requirement has to do with JCL. As you know, the ASSIGN clause on the SELECT statement identifies a DDname for a data set. Consider the following SELECT statement:

SELECT MASTER ASSIGN TO XYZ...

If this cluster is to be processed through an alternate index, two DD statements must be included at execution time. The first is DD name XYZ for the KSDS file; the second must also be XYZ immediately followed by a 1 to identify it as the path defined for the alternate index associated with that KSDS:

```cobol
//XYZ DD DSN= . . .
//XYZ1 DD DSN= . . .
```

The second requirement pertains to sequential processing using the alternate key. The programmer must indicate that the alternate key is being used. When you are performing random processing, the KEY IS option on the READ statement can be used. However, this option is not valid for sequential processing. Therefore, a START statement with a KEY IS option must be issued before any records can be read. Assuming you have an alternate key field named AIX-KEY, this START statement would look something like this:

```cobol
MOVE SPACES TO AIX-KEY.
START MASTER KEY NOT LESS THAN AIX-KEY.
```

These statements will establish current position using the alternate index. Records may then be read sequentially in the same manner you are accustomed to.
THE COBOL SELECT STATEMENT

The following is an example of the COBOL SELECT statement and the corresponding JCL:

SELECT KSDSFILE ASSIGN TO DA-MASTER . . .

On the GO step:

//MASTER  DD  DSN=T901234.KSDS
//MASTER1 DD  DSN=T901234.PATH1  (Do not code in program
//MASTER2 DD  DSN=T901234.PATH2  (Do not code in program

The format of the SELECT statement is as follows:

SELECT filename ASSIGN TO DA-ddname
ORGANIZATION IS INDEXED
ACCESS MODE IS [SEQUENTIAL/RANDOM/DYNAMIC]
RECORDKEY IS primekey [IN recordname]
* ALTERNATE RECORD KEY IS aixkey [WITH DUPLICATES]
FILE STATUS IS ws-field. (PIC XX)

* Note: There should be one ALTERNATE RECORD KEY IS . . .
for each alternate key and each must be specified
in the same order in which it was defined with the
IDCAMS utility program.
CALCULATING ALTERNATE INDEX RECORDSIZE

The formula for calculating the "max" and "avg" values of the RECORDSIZE subparameter of the IDCAMS DEFINE AIX is as follows:

1. Maximum: \( \text{RECORDSIZE} = A + B + (n \times C) \)
   
   \( A = 5 \), the record control bytes  
   \( B = \) the length of the alternate key in bytes  
   \( C = \) the length of the prime key in bytes  
   \( n = 1 \) if the alternate keys are unique  
   or  
   \( n = \) the estimated maximum number of nonunique alternate keys

2. Average: \( \text{RECORDSIZE} = \) approximately 1/2 of the maximum RECORDSIZE

The following are examples for calculating the alternate index RECORDSIZE value:

1. If unique alternate key length = 15 bytes and prime key length = 25 bytes
   
   Then maximum RECORDSIZE = \( 5 + 15 + (1 \times 25) \)  
   \( = 45 \) bytes  
   
   And average RECORDSIZE = 30 bytes.

2. If nonunique alternate key length = 10 bytes and estimated number of nonunique alternate keys = 5 and prime key length = 15 bytes
   
   Then maximum RECORDSIZE = \( 5 + 10 + (5 \times 15) \)  
   \( = 60 \) bytes  
   
   And average RECORDSIZE = 35 bytes.
VSAM IN ASSEMBLER
There are many similarities between using VSAM and QSAM. However, VSAM is a more powerful access method than QSAM, and discussing all of its features would make the subject confusing. Therefore, we restrict the discussion to the portions of VSAM which can be used to perform the standard master file processing.

Access Control Block (ACB)

VSAM clusters are identified in an Assembler program through an Access Control Block (ACB) instead of the DCB which is used by QSAM. The following is a macro used to generate an ACB:

```
label ACB AM=VSAM, Indicates VSAM is the access method
  DDNAME=ddname, Gives the JCL DD statement name for the VSAM file (if using an AIX, specify the path DD statement)
  MACRF=(. . .), Specifies all processing options that might be used for this VSAM file
  EXLST=exlist@ Indicates the address of the EXLST macro that specifies error handling routines used for this file (see Error Handling)
```

Processing options that can be used with MACRF are:

- ADR: If cluster will be accessed by RBA
- KEY: If cluster will be accessed by key field
- DIR: Indicates direct access to records by key
- SEQ: Indicates sequential processing; the order is determined by the type of VSAM cluster being processed
- SKP: Indicates skip sequential processing, which is keyed access with the keys in sequential order
- IN: Indicates retrieval processing (GET records)
- OUT: Indicates output processing (PUT records)

You can code as many of these options as are needed for your application program. For instance, if you are going to perform both direct and sequential access to a cluster in the same program, you would want to code both DIR and SEQ. If you wanted to read and write to the file in the same program, you would code both IN and OUT. You may find a need in some cases to use all the options in a program. The example on the following page illustrates the use of the ACB macro.
Request Parameter List (RPL)

To read from and write to a VSAM file, you use the same macros as in QSAM: GET and PUT. The format of the GET macro in VSAM looks like:

```
label GET RPL=rpl@ where @ = address
```

Notice the absence of a buffer location. Also, a question you might ask is, "If we are doing keyed access, how do we specify the key of the record we are looking for?" VSAM uses the RPL (Request Parameter List) control block to hold all detailed information about a processing request. The RPL contains all information for the request being performed, and VSAM checks the RPL when the request is issued to decide how it should process the request, where it should return the record retrieved, etc. The RPL macro is used to create an RPL.

```
label RPL AM=VSAM, ACB=acb@, AREA=buffer@, AREALEN=n, ARG=key@, KEYLEN=n, RECLEN=n, OPTCD=(. . .) 
```

Indicates access method
Address of the ACB used with this RPL
Address of the buffer used for I/O in the program
Specifies the buffer length
Address of the key used for direct access
Key field length (used only for generic retrieval)
Record length (used when PUTting a record to the data set)
Processing options in effect for this request

The OPTCD parameter tells VSAM what processing options to use for the particular request being issued. It indicates how you want the request satisfied (i.e. - whether you want direct or sequential processing, whether you are using locate or move mode, etc.).

The possible options for the OPTCD parameter are as follows (only one option per group may be chosen and the defaults are underlined):

1. ADR  Processing an ESDS or KSDS by RBA.
   KEY  Processing a RRDS or KSDS by key.
2. **DIR** Retrieve records directly. The key field is specified by the ARG= parameter; the key length by KEYLEN=.

**SEQ** Retrieve the record following the last record retrieved (ie. - retrieve records in sequential order).

**SKP** Use skip sequential processing to retrieve the record whose key and key length are specified as above for DIR. To use this method, keys must be presented in key order.

3. **NUP** No records are to be updated. Using a PUT macro with this option indicates a new record is being stored to the file.

**UPD** Records are to be updated/deleted. VSAM remembers position on retrieval.

**NSP** Used with OPTCD=DIR only. VSAM remembers position after a direct retrieval to allow a switch to sequential processing from that point on following the direct retrieval.

4. **KEQ** The request is satisfied only if a record key exactly matches the key argument.

**KGE** The request is satisfied by a record key equal to the key argument or the record with the next higher key.

5. **FKS** The full key field (ie. - as specified in the DEFINE CLUSTER) is being used as a search argument.

**GEN** The generic key (leftmost n bytes of the full key) is being used as the search argument. The value of n is specified in the KEYLEN= parameter. The first n bytes of the ARG field are used to match the keys of records in the file.

6. **LOC** Locate mode processing. The address of the retrieved record is placed in the AREA= field. The AREALEN= parameter should be specified as 4.

**MVE** Retrieved record is moved to the AREA= field. This option must be used for PUT or ERASE.

The following is an example of a RPL macro:

```
VSAMRPL RPL AM=VSAM,
          ACB=VSAMACB,
          AREA=VSAMBUFF,
          AREALEN=80,
          ARG=KEYARG,
          KEYLEN=6,
          RECLEN=80,
          OPTCD=(KEY,DIR,FKS,KEQ,MVE)
```
MODIFYING THE RPL

VSAM gives you the ability to change the type of processing you are performing against a cluster while your program is executing. This means you can change from direct to sequential processing, no update to update processing, full key search to generic key search, etc. These changes to the RPL are made through the use of the MODCB macro which modifies the specified control block. The format of this macro is:

```
label MODCB RPL=rpladdr, Address of RPL to be modified
      ... other parameters ...
```

The other parameters that can be specified in the MODCB macro are the same ones that can be specified in the RPL macro. Once you modify the RPL, the new parameters remain in effect until the next change is made to that RPL with another MODCB macro. Thus, you can change the method used to access a VSAM cluster during the execution of your program.

You must be careful in modifying the RPL: the parameters coded on the RPL macro must be allowable processing options for the cluster. Remember that the allowable processing options were specified in the ACB macro for the cluster. If you change the RPL to indicate a processing option not allowed by the ACB, your next I/O request generates an error. An example of this is if you code in the ACB MACRF=(SEQ,IN) and modify the RPL with OPTCD=DIR to perform direct processing of the cluster. This would cause an error because the ACB for the cluster allows only sequential processing.

The following is an example of the MODCB macro:

```
MODCB RPL=VSAMRPL,OPTCD=(SEQ)
```
OPENING AND CLOSING VSAM DATA SETS

Before you can access a VSAM cluster, you must OPEN it. This is done with an OPEN macro, which is coded in the same manner as a QSAM OPEN, except that VSAM files do not require the INPUT/OUTPUT option to be coded (it has already been specified in the ACB for the cluster) and an ACB address is supplied, rather than a DCB address. After the OPEN executes, R15 contains a return code indicating whether or not the file opened successfully.

You can open more than one file with a single OPEN macro, and you can also open both VSAM and non-VSAM files with a single OPEN macro.

The following are return codes with their meanings:

0 - Open was successful.
4 - Successful, but there are warning messages.
8 - At least 1 data set did not open.
12 - A non-VSAM data set did not open
(while opening both VSAM and non-VSAM data sets).

When you have completed the processing of a cluster, that cluster should be closed. The format of the CLOSE macro is the same as it is for QSAM.

The following is an example containing both the OPEN and CLOSE macros:

OPEN (CARDDCB,(INPUT),VSAMACB,,PRNTDCB,(OUTPUT))
IF (LTR,R15,R15,NZ) DID THE OPEN FAIL?
   ABEND 2222,DUMP IF YES, ABEND
ENDIF
CLOSE (CARDDCB,VSAMACB,PRNTDCB)
TESTING RETURN CODES/FEEDBACK INFORMATION
AFTER I/O REQUESTS

After the execution of any macro used to process a VSAM data set, a status field is set in the RPL to indicate the success of the operation. The return code placed in R15 indicates, in a general manner, whether or not the operation was successful (0 = success, anything else = error). More detailed information about an error is given in the RPL field called the feedback code. If R15 is not zero, the feedback code can be tested to determine the exact cause of the error. This test is carried out by using the TESTCB macro. The format of this macro is:

```
label TESTCB RPL=rpladdr, FDBK=number  Value to be matched against feedback code
```

This TESTCB macro compares the feedback value specified in the FDBK parameter against the feedback code in the RPL and sets the condition code. Following the macro, you can use a conditional branch instruction to determine the results of the test.

The following is an example of the TESTCB macro:

```
TESTCB RPL=VSAMRPL, FDBK=4  END OF FILE FEEDBACK CODE
BNE OTHERERR  IF NOT EQUAL, TRY ANOTHER CODE
MVI EOFLAG,C'Y'  ELSE SET EOF FLAG
```

The above tests the feedback code for end of file and sets the EOFLAG to 'on' if end of file occurred. The TESTCB macro should be used to determine the type of processing error that has occurred and to handle the error appropriately. Keep in mind that issuing the TESTCB macro causes R15 to be set to indicate if the macro executed properly (i.e. - destroys the I/O return code).

The following is a (very abbreviated) list of the feedback code values which can be used to test the feedback code:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>End of data set</td>
</tr>
<tr>
<td>8</td>
<td>Duplicate record (for store)</td>
</tr>
<tr>
<td>12</td>
<td>Sequence error when loading file</td>
</tr>
<tr>
<td>16</td>
<td>Record not found</td>
</tr>
</tbody>
</table>
ERROR HANDLING Routines

The EXLST macro is used to indicate the location of routines that are used to handle error conditions that occur while processing a VSAM data set. The error routines are needed to handle end of data processing, logical error processing, and physical errors. If a physical error occurs, processing should always be terminated. A logical error indicates that the preceding VSAM request was not successful. This would include end of file (if no EODAD exit is specified), no record found, duplicate record (for store operation), and/or an attempt to perform any type of processing that is not valid for the VSAM data set. The format of the EXLST macro is:

label EXLST EODAD=addr, Address of end of file routine
LERA D=addr, Address of logical error routine
SYNAD=addr Address of physical error routine

The address specified in the above operands is the label of the error handling routine in the program. On entry to an error handling routine, R1 contains the address of the RPL in use at the time of the error, R15 contains the address of the beginning of the error routine, and R14 contains the return link address. If you use a VSAM macro in the error routine, R15 is changed by the macro; therefore, it should not be used as the base register for the routine. These routines should be placed between the main logic of the program and the storage areas used by the program.

The following is an example of the EXLST macro with a corresponding error routine:

ERRROUT EXLST LERAD=LOGERR
* LOGERR DS 0D
USING *,R15
STM R0,R15,TEMPSAVE TEMP BASE REG. (TESTCB ALTERS R13)
LA R13,SAVE2 SAVE THE REGISTERS
LR R12,R15 POINT R13 AT 18-F SAVEAREA
USING LOGERR,R12 SET UP BASE REG. (USE 2-12 ONLY)
DROP R15 DROP R15 AS BASE REGISTER
TESTCB RPL=VSAMRPL, TEST FOR END OF FILE X
FDBK=4
IF (NE)
ABEND 1111,DUMP ABEND WITH A DUMP
ELSE
MVI ERRFLAG,C'E'
SET EOF FLAG
ENDIF
* LM R0,R15,TEMPSAVE RETURN TO INSTRUCTION FOLLOWING
* BR R14 MACRO THAT CAUSED THE ERROR
* LTORG DROP R12 DROP R12 AS BASE REGISTER

NOTE: If you do not code an EODAD routine, you can use your LERAD
routine to test for end of file. To do this, you must check the value of the RPL feedback code. If you do not code any error handling routines (ie. - you omit the EXLST parameter from the ACB), you must test R15 after each VSAM macro to make sure that the operation was successful.

RETRIEVING RECORDS

As indicated earlier, records are retrieved using the VSAM GET macro. To retrieve records directly (ie. - by key), the key value of the record must be placed in the field specified as the ARG field in the RPL, and the options must indicate DIR retrieval. The format of the GET macro is:

label GET RPL=rpladdr

Following the GET, if the record does not exist, R15 is set to a value of 8, and the feedback field of the RPL is set to 16 (no record found). If a LERAD routine has been specified on the EXLST macro, it gets control following the unsuccessful GET to allow any error processing required before continuing with the main logic of the program. To retrieve records sequentially, the GET macro is used with the SEQ option. The current record pointer (used only for sequential processing) must have been set to a valid record for sequential retrieval to be successful. If the sequential request causes an error (end of file), R15 again contains a value of 8, and the EODAD routine on the EXLST macro, if specified, gets control. If no EODAD is specified, the LERAD routine gets control.

UPDATING RECORDS

Updating records in Assembler requires that the record to be modified be retrieved (GET) using UPD and MOV options prior to the PUT. The record can be modified in the buffer and PUT back to the data set. Be sure the RPL has the correct options specified before the GET. If you decide not to update the record, another GET cancels the previous GET for update. Remember to make sure that the GET was successful before trying to update the record. The format of the PUT macro is:

label PUT RPL=rpladdr

If an error occurs while the PUT is executing, R15 contains a non-zero value, and the LERAD routine, if specified, gets control. You should always have a method of verifying that the PUT was successful.
DELETING RECORDS

Deleting a record also requires a prior GET for update. Once the record is retrieved for update, the ERASE macro is issued to delete the record from the data set. Again, check for a successful GET before issuing the ERASE. The format of the ERASE macro is:

label ERASE RPL=rpladdr

If an error occurs while the ERASE is executing, R15 will contain a non-zero value, and the LERAD routine, if specified, gets control. Once again, you should always have a method of verifying that the ERASE was successful.

ADDING RECORDS TO A DATASET

To insert a new record into a data set, the RPL must have the NUP option specified. The record to be added to the data set must be contained in the buffer specified in the AREA parameter. The RECLEN parameter of the RPL must contain the length of the record being added to the data set. By issuing a PUT macro with these options specified, the record contained in the buffer is stored in the data set unless a record with the same key already exists. The format of the PUT macro is the same as it is for an update. If there is a duplicate record, R15 will contain a non-zero value and the LERAD routine, if specified, gets control.

POSITIONING FOR SEQUENTIAL RETRIEVAL

The POINT macro is used to position the current record pointer to a requested record in a VSAM KSDS or RRDS file, but it does not cause a record to be retrieved. Before issuing the POINT, the RPL must be set up for SEQ processing, and the key of the record to be pointed to must be stored in the field specified by the RPL's ARG parameter. The POINT macro is then issued to position the current record pointer. The format of the POINT macro is:

label POINT RPL=rpladdr

Following execution of the POINT macro, R15 will contain a non-zero value if the request was unsuccessful, and the LERAD routine, if specified, gets control.
XSAVE AND XRETURN MACROS

The X-macro definitions are contained in the macro library SYS2.MACLIB and may be invoked in a program which makes this library available at assembly time. If you use a JCL procedure, this should be taken care of by the procedure. If you do not use a JCL procedure, you must be sure to include SYS2.MACLIB as one of your SYSLIB data sets for the assembly step.

XSAVE
- Generates standard entry code upon entry to a routine. If used, this must be the first instruction coded in your routine (except possibly for a preceding CSECT or ENTRY statement).
- Keyword operands include:
  OPT= OPT=CSECT causes the CSECT statement to be generated.
  BR= Specifies register number of base register. Default value is 12. The generated code will include the USING statement together with instructions which place the basepoint address in the base register.
  TR= Specifies whether or not trace feature is to be used. Default is YES. To turn off the trace, specify TR=NO.

XRETURN
- Generates standard exit code when a routine is exited.
- Keyword operands include:
  TR= For the trace feature. See description given for XSAVE above.
  RC= Specifies return code value, if one is to be used. The specified value will be placed in R15 on exit. May be indicated as a decimal constant, or as the contents of a register if register number is enclosed in parentheses.
    (ie. - RC=4 or RC=(R9) )
  RGS= Specifies numbers of registers to be restored before exiting. Coded as a sublist. Can use individual register numbers or a range of register numbers separated by a hyphen.
    (ie. - RGS=(14,2-8,10-12) )
    Default is RGS=(14-12).

Some of the other X-macros are XSET, XCALL, XSTOP, and XSNAP. You can copy them into a source listing by using the COPY statement and you are encouraged to do this.

When you obtain the macro definitions, study them and try to uncover some of the other options which they offer you. You will learn a lot about coding macros by doing this.
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJECT CODE</th>
<th>ADDR1</th>
<th>ADDR2</th>
<th>STMT</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRINT NGEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>EQUREGS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>VSAMEX</td>
<td>XSAVE_OPT=CSECT, BR=13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>OPEN (CUSTACB,PRNTDCB,(OUTPUT))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>* THE CUSTOMER FILE HAS A PRIME KEY LENGTH OF 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>* THE RPL IS INITIALLY SET UP FOR DIRECT ACCESS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>* WE'LL RETRIEVE THE RECORD WITH KEY '11111'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>MVI ERRFLAG,C' '</td>
<td>RESET ERROR FLAG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>MVC KEYARG,+C'11111'</td>
<td>MOVE KEY TO ARG FIELD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>GET RPL=CUSTRPL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>IF (CLI,ERRFLAG,EQ,C'N') WAS GET SUCCESSFUL?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>PUT PRNTDCB,NOTFOUND PRINT ERROR MESSAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>ELSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>PUT PRNTDCB,CUSTBUF-1 PRINT RECORD RETRIEVED</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>ENDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>NOW CHANGE RPL TO PROCESS SEQUENTIALLY. NOTE USE OF NSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>* OPTION IN ORIGINAL RPL TO REMEMBER POSITION. ALSO WE'LL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>* USE A DIFFERENT BUFFER.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>MODCB RPL=CUSTRPL,OPTCD=(SEQ),AREA=CUSTBUF2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>GET RPL=CUSTRPL</td>
<td>GET RECORD FOLLOWING '11111'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>124</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>125</td>
<td>IF (CLI,ERRFLAG,EQ,C'E') DID EOF OCCUR?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>129</td>
<td>PUT PRNTDCB,EOFMSG PRINT EOF MESSAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>134</td>
<td>ELSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>137</td>
<td>PUT PRNTDCB,CUSTBUF2-1 PRINT RECORD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>142</td>
<td>ENDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>144</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>145</td>
<td>NOW ADD A RECORD WITH KEY '18888'. IT'S FOUND IN 'NEWREC'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>146</td>
<td>* STORAGE AREA. WE MUST SET RECORD LENGTH IN THE RPL.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>147</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>148</td>
<td>L R2,+F'187'</td>
<td>SET REC LENGTH OF NEW RECORD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>149</td>
<td>MODCB RPL=CUSTRPL,OPTCD=(DIR,NUP),AREA=NEWREC, X RECLEN=(R2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177</td>
<td>PUT RPL=CUSTRPL</td>
<td>ADD RECORD 188888</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>184</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>185</td>
<td>IF (CLI,ERRFLAG,EQ,C'D') IF DUPLICATE RECORD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>189</td>
<td>PUT PRNTDCB,DUPMSG PRINT DUP MESSAGE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>194</td>
<td>ELSE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>197</td>
<td>PUT PRNTDCB,RECADDED PRINT SUCCESS MSG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>ENDF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>204</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>205</td>
<td>NOW DELETE RECORD WITH KEY '23456'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>207</td>
<td>MODCB RPL=CUSTRPL,OPTCD=(DIR,UPD)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>225</td>
<td>MVC KEYARG,+C'23456' MOVE KEY TO ARG FIELD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>226</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>227</td>
<td>GET RPL=CUSTRPL</td>
<td>RETRIEVE THE RECORD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOC</td>
<td>OBJECT CODE</td>
<td>ADDR1</td>
<td>ADDR2</td>
<td>STMT</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>-------</td>
<td>-------</td>
<td>-------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>234</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>235</td>
<td></td>
<td></td>
<td></td>
<td>IF (CLI,ERRFLAG,EQ,C'N') IF RECORD NOT FOUND</td>
<td></td>
</tr>
<tr>
<td>239</td>
<td></td>
<td></td>
<td></td>
<td>PUT PRNDCB.NOTFOUND PRINT NOT FOUND MSG</td>
<td></td>
</tr>
<tr>
<td>244</td>
<td></td>
<td></td>
<td></td>
<td>ELSE</td>
<td></td>
</tr>
<tr>
<td>247</td>
<td></td>
<td></td>
<td></td>
<td>ERASE RPL=CUSTRPL IF FOUND, DELETE RECORD</td>
<td></td>
</tr>
<tr>
<td>254</td>
<td></td>
<td></td>
<td></td>
<td>* IF ERASE FAILS, PGM WILL BE TERMINATED BY LERAD OR SYNAD RTN</td>
<td></td>
</tr>
<tr>
<td>255</td>
<td></td>
<td></td>
<td></td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>0002E0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002E0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002E0</td>
<td>0000000BB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002E4</td>
<td>F1F1F1F1F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002E9</td>
<td>F2F3F4F5F6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>289</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>290</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002F0</td>
<td></td>
<td></td>
<td>002F0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002F0</td>
<td>0000F0F0EB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002F4</td>
<td>18CF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002F6</td>
<td>41DO C128</td>
<td>00418</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>297</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>298</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>304</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>305</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>323</td>
<td></td>
<td></td>
<td></td>
<td>TESTCB RPL=CUST RPL,FDBK=4 TEST FOR EOF</td>
<td></td>
</tr>
<tr>
<td>326</td>
<td></td>
<td></td>
<td></td>
<td>IF (NE) TESTCB RPL=CUST RPL,FDBK=16 TEST FOR NO REC FOUND</td>
<td></td>
</tr>
<tr>
<td>344</td>
<td></td>
<td></td>
<td></td>
<td>IF (NE) TESTCB RPL=CUST RPL,FDBK=8 TEST FOR DUPLICATE</td>
<td></td>
</tr>
<tr>
<td>347</td>
<td></td>
<td></td>
<td></td>
<td>IF (NE) ABEND 1000,DUMP ABEND WITH DUMP</td>
<td></td>
</tr>
<tr>
<td>375</td>
<td></td>
<td></td>
<td></td>
<td>ELSE</td>
<td></td>
</tr>
<tr>
<td>381</td>
<td></td>
<td></td>
<td></td>
<td>ELSE</td>
<td></td>
</tr>
<tr>
<td>384</td>
<td></td>
<td></td>
<td></td>
<td>ELSE</td>
<td></td>
</tr>
<tr>
<td>385</td>
<td></td>
<td></td>
<td></td>
<td>ELSE</td>
<td></td>
</tr>
<tr>
<td>393</td>
<td></td>
<td></td>
<td></td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>399</td>
<td></td>
<td></td>
<td></td>
<td>ENDIF</td>
<td></td>
</tr>
<tr>
<td>0003BB</td>
<td>92C4 C4CO</td>
<td>007BO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003CO</td>
<td>92D5 C4CO</td>
<td>007BO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003CB</td>
<td>92C5 C4CO</td>
<td>007BO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003CC</td>
<td>980F COEB</td>
<td>003D8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003DO</td>
<td>07FE</td>
<td>003D8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003DB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>397</td>
<td></td>
<td></td>
<td></td>
<td>TEMPSAVE DS 16F TEMP SAVE AREA</td>
<td></td>
</tr>
<tr>
<td>398</td>
<td></td>
<td></td>
<td></td>
<td>TEMPSAVE DS 18F SAVE AREA FRO VSAM MACROS</td>
<td></td>
</tr>
<tr>
<td>399</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOC</td>
<td>OBJECT CODE</td>
<td>ADDR1</td>
<td>ADDR2</td>
<td>STMT</td>
<td>SOURCE</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>-------</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>000460</td>
<td>90OF F090</td>
<td>00460</td>
<td>00530</td>
<td>408</td>
<td>LA R13,PHYSV2</td>
</tr>
<tr>
<td>000464</td>
<td>18CF</td>
<td>00460</td>
<td>00530</td>
<td>409</td>
<td>*</td>
</tr>
<tr>
<td>000466</td>
<td>41DO C090</td>
<td>00460</td>
<td>00530</td>
<td>410</td>
<td>SHOWCB RPL=CUSTRL,AREA=ERRSTAT,LENGTH=8,</td>
</tr>
<tr>
<td>0004EA</td>
<td>98OF C090</td>
<td>00460</td>
<td>00530</td>
<td>412</td>
<td>IF (LTR,R15,R15,NZ)</td>
</tr>
<tr>
<td>0004EE</td>
<td>07FE</td>
<td>00460</td>
<td>00530</td>
<td>433</td>
<td>ENDIF</td>
</tr>
<tr>
<td>000530</td>
<td>00460</td>
<td>00530</td>
<td>464</td>
<td>BR R14</td>
<td>SAVE AREA FRO VSAM MACROS</td>
</tr>
<tr>
<td>000578</td>
<td>40</td>
<td>00460</td>
<td>00530</td>
<td>473</td>
<td>KEYARG DS CL5</td>
</tr>
<tr>
<td>00057D</td>
<td>40</td>
<td>00460</td>
<td>00530</td>
<td>474</td>
<td>CUSTBUF DS CL187</td>
</tr>
<tr>
<td>000633</td>
<td>40</td>
<td>00460</td>
<td>00530</td>
<td>475</td>
<td>CUSTBUF2 DS CL187</td>
</tr>
<tr>
<td>0006F5</td>
<td>F1F8F8F8C3E4E2</td>
<td>00460</td>
<td>00530</td>
<td>476</td>
<td>NEWREC DC CL187′18888CUSTOMER INFORMATION,...' NEW RECORD</td>
</tr>
<tr>
<td>000780</td>
<td>40</td>
<td>00460</td>
<td>00530</td>
<td>483</td>
<td>ERRFLAG DC CL17'</td>
</tr>
<tr>
<td>000781</td>
<td>40C509D9D6040940</td>
<td>00460</td>
<td>00530</td>
<td>485</td>
<td>PHYINFO DC CL26′ ERROR STATUS CODES FOLLOW'</td>
</tr>
<tr>
<td>0007CC</td>
<td>40C4E47D3C9C3C1</td>
<td>00460</td>
<td>00530</td>
<td>493</td>
<td>NOUSFOUND DC CL133′ CUSTOMER RECORD NOT FOUND'</td>
</tr>
<tr>
<td>000839</td>
<td>40C3E42E30604045</td>
<td>00460</td>
<td>00530</td>
<td>494</td>
<td>PHYERMSG DC CL133′ VSAM PHYSICAL ERROR HAS OCCURRED'</td>
</tr>
<tr>
<td>00088A</td>
<td>40E5E2C104D07C8</td>
<td>00460</td>
<td>00530</td>
<td>495</td>
<td>DUPMSG DC CL133′ DUPLICATE RECORD FOUND'</td>
</tr>
<tr>
<td>00093F</td>
<td>40C50C440D6C640</td>
<td>00460</td>
<td>00530</td>
<td>496</td>
<td>EOFMSG DC CL133′ END OF FILE ENCOUNTERED DURING SEQUENTIAL READ'</td>
</tr>
<tr>
<td>000A49</td>
<td>40D9E5C30D6C440</td>
<td>00460</td>
<td>00530</td>
<td>497</td>
<td>READDAD DC CL133′ RECORD SUCCESSFULLY ADDED'</td>
</tr>
<tr>
<td>500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOC</td>
<td>OBJECT CODE</td>
<td>ADDR1</td>
<td>ADDR2</td>
<td>STMT</td>
<td>SOURCE</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>501</td>
<td>000AD0</td>
<td>003+</td>
<td>054+</td>
<td>CUSTACB</td>
<td>DS</td>
</tr>
<tr>
<td>502</td>
<td>X'AO', ACB</td>
<td>054+</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>503</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>504</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>505</td>
<td>O00AD0 AO</td>
<td>003+</td>
<td>054+</td>
<td>AL1(16)</td>
<td>DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>506</td>
<td>O00AD1 1C</td>
<td>003+</td>
<td>054+</td>
<td>AL2(76+1)</td>
<td>DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>507</td>
<td>O00AD4 00000000</td>
<td>003+</td>
<td>054+</td>
<td>A(o)</td>
<td>DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>508</td>
<td>O00AD8 00000000</td>
<td>003+</td>
<td>054+</td>
<td>A(o)</td>
<td>DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>509</td>
<td>O00AD9 00000000</td>
<td>003+</td>
<td>054+</td>
<td>B'100110'</td>
<td>DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>510</td>
<td>O00ADE 00000000</td>
<td>003+</td>
<td>054+</td>
<td>B'00000000'</td>
<td>DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>511</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>513</td>
<td>O00AEO 00000000</td>
<td>003+</td>
<td>054+</td>
<td>AL1(1)</td>
<td>DC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>514</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>515</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>516</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>517</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>518</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>519</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>520</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>521</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>522</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>523</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>524</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>525</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>526</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>527</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>528</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>529</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>530</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>531</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>532</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>533</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>534</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>535</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>536</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>537</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>538</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>539</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>540</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>541</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>542</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>543</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>544</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>545</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>546</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>547</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>548</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>549</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>550</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>551</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DC = Directly Connectible
DS = Directly Storable
X = Yes
O = No
X = X

**Notes:**
- DDNAME=CUSTFILE
- MACRF=KEY,DIR,SEQ,IN,OUT
- EXLIST=ERROUT
- AMB=VSAM
<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJECT CODE</th>
<th>ADDR1</th>
<th>ADDR2</th>
<th>STMT</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>552</td>
<td>553 CUSRPL</td>
<td>RPL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>008B30</td>
<td>554+CUSRPL</td>
<td>DS</td>
<td>OF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>008B30</td>
<td>00</td>
<td>DC</td>
<td>AL1(0)</td>
<td></td>
<td>RPL IDENTIFICATION</td>
</tr>
<tr>
<td>008B31</td>
<td>00</td>
<td>DC</td>
<td>AL1(16)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B32</td>
<td>00</td>
<td>DC</td>
<td>AL1(16)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B33</td>
<td>0C</td>
<td>DC</td>
<td>AL1(76)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B34</td>
<td>00000000</td>
<td>DC</td>
<td>A(0)</td>
<td>02-RPLVS</td>
<td>X04SVVS</td>
</tr>
<tr>
<td>008B35</td>
<td>00000000</td>
<td>DC</td>
<td>A(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B3C</td>
<td>00</td>
<td>DC</td>
<td>BL1 'O'</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B3D</td>
<td>00000000</td>
<td>DC</td>
<td>XL3 'O'</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B40</td>
<td>00000000</td>
<td>DC</td>
<td>AL2(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B42</td>
<td>00000000</td>
<td>DC</td>
<td>AL2(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B44</td>
<td>00000000</td>
<td>DC</td>
<td>A(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B48</td>
<td>00000ADD0</td>
<td>DC</td>
<td>A(CUSTACB)</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B4C</td>
<td>00000000</td>
<td>DC</td>
<td>A(0)</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B50</td>
<td>0000057E</td>
<td>DC</td>
<td>A(CUSTBUF)</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B54</td>
<td>00000000</td>
<td>DC</td>
<td>A(KEYARG)</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B58</td>
<td>0400</td>
<td>DC</td>
<td>B'10000000'</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B59</td>
<td>81</td>
<td>DC</td>
<td>B'10000001'</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B5A</td>
<td>00</td>
<td>DC</td>
<td>AL1(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B5B</td>
<td>00</td>
<td>DC</td>
<td>AL1(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B5C</td>
<td>00000000</td>
<td>DC</td>
<td>A(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B60</td>
<td>00000000</td>
<td>DC</td>
<td>A(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B64</td>
<td>000000BB</td>
<td>DC</td>
<td>A(187)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B68</td>
<td>00</td>
<td>DC</td>
<td>B'00000000'</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B69</td>
<td>00</td>
<td>DC</td>
<td>B'00000000'</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B6A</td>
<td>00</td>
<td>DC</td>
<td>B'00000000'</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B6B</td>
<td>00</td>
<td>DC</td>
<td>B'00000000'</td>
<td>02-RPLVS</td>
<td>X02-RPLVS</td>
</tr>
<tr>
<td>008B6C</td>
<td>0000000000000000</td>
<td>DC</td>
<td>XL8 'O'</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B74</td>
<td>40</td>
<td>DC</td>
<td>B'01000000'</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B75</td>
<td>00</td>
<td>DC</td>
<td>AL1(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B76</td>
<td>00000000</td>
<td>DC</td>
<td>AL2(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>008B78</td>
<td>00000000</td>
<td>DC</td>
<td>A(0)</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>587</td>
<td>588 PRTDCH</td>
<td>DC</td>
<td>OF 'O'</td>
<td>02-RPLVS</td>
<td>X04SVHS</td>
</tr>
<tr>
<td>590**</td>
<td>591** DATA CONTROL BLOCK</td>
<td>01-DCB</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LOC</th>
<th>OBJECT CODE</th>
<th>ADDR1</th>
<th>ADDR2</th>
<th>STMT</th>
<th>SOURCE STATEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>592</td>
<td>PRTDCH DCB</td>
<td>DSCOR+PS,DDNAME=PRTOUT,MACRF=PM,RECFC=FA,LRECL=133</td>
<td>OF 'O'</td>
<td>ORIGIN ON WORD BOUNDARY</td>
<td>01-DCB</td>
</tr>
<tr>
<td>LOC</td>
<td>OBJECT CODE</td>
<td>ADDR1</td>
<td>ADDR2</td>
<td>STMT</td>
<td>SOURCE STATEMENT</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>------------------</td>
</tr>
<tr>
<td>594+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DIRECT ACCESS DEVICE INTERFACE</td>
</tr>
<tr>
<td>000B7C 0000000000000000000000</td>
<td>596+</td>
<td>DC</td>
<td>BL16'O'</td>
<td>FOAD, DVTBL</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000B8C 0000000000000000000000</td>
<td>597+</td>
<td>DC</td>
<td>A(0)</td>
<td>KEYLE, DEV'T, TRBAL</td>
<td>01-DCB</td>
</tr>
<tr>
<td>599+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>COMMON ACCESS METHOD INTERFACE</td>
</tr>
<tr>
<td>000B90 00</td>
<td>601+</td>
<td>DC</td>
<td>AL1(O)</td>
<td>BUFD</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000B91 000001</td>
<td>602+</td>
<td>DC</td>
<td>AL2(1)</td>
<td>BUFBC</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000B94 0000</td>
<td>603+</td>
<td>DC</td>
<td>AL2(0)</td>
<td>BUFL</td>
<td>01-DCB</td>
</tr>
<tr>
<td>000B96 4000</td>
<td>604+</td>
<td>DC</td>
<td>BL2'01000000000000000'</td>
<td>DSORG</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000B98 0000000000000000000000</td>
<td>605+</td>
<td>DC</td>
<td>A(1)</td>
<td>ICQE/IOBAD</td>
<td>01-DCB</td>
</tr>
<tr>
<td>607+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FOUNDATION EXTENSION</td>
</tr>
<tr>
<td>000B9C 00</td>
<td>609+</td>
<td>DC</td>
<td>BL1'00000000000'</td>
<td>EODAD</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000B9D 00000001</td>
<td>610+</td>
<td>DC</td>
<td>AL3(1)</td>
<td>EODAD</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BA0 84</td>
<td>611+</td>
<td>DC</td>
<td>BL1'10000100'</td>
<td>RECFM</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BA1 0000000000000000000000</td>
<td>612+</td>
<td>DC</td>
<td>AL3(0)</td>
<td>EXLIST</td>
<td>01-DCB</td>
</tr>
<tr>
<td>614+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FOUNDATION BLOCK</td>
</tr>
<tr>
<td>000BA4 D7D9E36D6E4E34040</td>
<td>616+</td>
<td>DC</td>
<td>CLB'PRTOUT'</td>
<td>DDNAME</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BAC 02</td>
<td>617+</td>
<td>DC</td>
<td>BL1'000000010'</td>
<td>OFLGS</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BAD 00</td>
<td>618+</td>
<td>DC</td>
<td>BL1'000000000'</td>
<td>IFLG</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BAE 0050</td>
<td>619+</td>
<td>DC</td>
<td>BL2'0000000001010000'</td>
<td>MACR</td>
<td>01-DCB</td>
</tr>
<tr>
<td>621+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>BSAM-BPAM-QSAM INTERFACE</td>
</tr>
<tr>
<td>000BB0 00</td>
<td>623+</td>
<td>DC</td>
<td>BL1'O0000000'</td>
<td>CHECK, GERR, PERR</td>
<td>RER1</td>
</tr>
<tr>
<td>000BB1 000001</td>
<td>624+</td>
<td>DC</td>
<td>AL3(1)</td>
<td>CHECK, GERR, PERR</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BB4 0000000000000000000000</td>
<td>625+</td>
<td>DC</td>
<td>A(0)</td>
<td>SYNAD</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BB8 0000000000000000000000</td>
<td>626+</td>
<td>DC</td>
<td>H'O'</td>
<td>CIND1, CIND2</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BBA 0000000000000000000000</td>
<td>627+</td>
<td>DC</td>
<td>AL2(0)</td>
<td>BLKSIZE</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BBC 0000000000000000000000</td>
<td>628+</td>
<td>DC</td>
<td>F'O'</td>
<td>WCP0, WCPL, OFFSR, OFFSW</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BC0 0000000000000000000000</td>
<td>629+</td>
<td>DC</td>
<td>A(1)</td>
<td>IDBA</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BC4 00</td>
<td>630+</td>
<td>DC</td>
<td>AL1(0)</td>
<td>NCP</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BC5 0000000000000000000000</td>
<td>631+</td>
<td>DC</td>
<td>AL3(1)</td>
<td>EOB, EOBAD</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>633+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>QSAM INTERFACE</td>
</tr>
<tr>
<td>000BC8 0000000000000000000000</td>
<td>635+</td>
<td>DC</td>
<td>A(1)</td>
<td>RECADD</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BCB 0000000000000000000000</td>
<td>636+</td>
<td>DC</td>
<td>H'O'</td>
<td>QSWS</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BCE 0085</td>
<td>637+</td>
<td>DC</td>
<td>AL2(133)</td>
<td>LRECL</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BD0 00</td>
<td>638+</td>
<td>DC</td>
<td>BL1'O00000000'</td>
<td>EROPT</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BD1 0000000000000000000000</td>
<td>639+</td>
<td>DC</td>
<td>AL3(1)</td>
<td>CNTRL</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BD4 0000000000000000000000</td>
<td>640+</td>
<td>DC</td>
<td>F'O'</td>
<td>PRECL</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>000BD8 0000000000000000000000</td>
<td>641+</td>
<td>DC</td>
<td>A(1)</td>
<td>EOB</td>
<td>O1-DCB</td>
</tr>
<tr>
<td>642</td>
<td>END</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
INFORMATIONAL MANAGEMENT SYSTEM (IMS)
A procedure named NWIMSSSEG (New IMS Simulator Link-edit and Go) resides in the cataloged procedure library T90MSTR.JJ.IMSSIM.PROCLIB. To use this library, you must include a //PROCLIB DD statement defining this library. The procedure is designed to be executed following a compile step. It will link-edit your program, DBD, and PSB, creating your IMS application module and run the IMS simulator program to execute your IMS program. Your JCL should resemble the following example:

```plaintext
//PROCLIB DD DSN=T90MSTR.JJ.IMSSIM.PROCLIB, DISP=SHR
// EXEC COBUC
// COB.SYSIN DD *
//   [Your source code]
//   /*
//   // EXEC NWIMSSSEG,COND=(O,NE), PSBNAME=psbname,
//   //   LOADMEM='(dbname)', WRITE='(dbname.PUNCH)' //
//   // DBDLINK.SYSLIN DD [Identifies your DBD object module]
//   //   DD *
//   //   NAME dbname
//   /*
//   // PSBLINK.SYSLIN DD [Identifies your PSB object module]
//   // IMS.ddnames DD [DD cards required by your program]
//   // IMS.ddnames DD [DD cards required for SYMDMP, if used]
//   /*
```

**LOADING A DATA SET**

The LOADMEM and WRITE symbolic parameters are required when loading a data base. In addition, you must include an IMS.PUNCH DD card, which defines the data set your data base is written to upon completion of your program. You should use DUMMY until you are fairly certain your program works. This data set may be blocked but must have an LRECL of 80. The following is an example of the JCL for loading a data base:

```plaintext
//PROCLIB DD DSN=T90MSTR.JJ.IMSSIM.PROCLIB, DISP=SHR
// EXEC COBUC
// COB.SYSIN DD *
//   [COBOL load program]
//   /*
//   // EXEC NWIMSSSEG,COND=(O,NE), PSBNAME=LOADPSB,
//   //   LOADMEM='(MYDBD)', WRITE='(MYDBD.PUNCH)' //
//   // DBDLINK.SYSLIN DD DSN=T901234.IMSPDS(MYDBD),UNIT=DISK,
//   //   VOL=SER=ACA008, DISP=SHR
//   // DD *
//   // NAME MYDBD
//   /*
//   // PSBLINK.SYSLIN DD DSN=T901234.IMSPDS(LOADPSB),UNIT=DISK,
//   //   VOL=SER=ACA008, DISP=SHR

*** (Continued on following page) ***
RETRIEVING RECORDS FROM THE DATA BASE

When executing a program which performs read-only processing from the database, the LOADMEM and WRITE symbolic parameters are not used. In addition, the IMS.PUNCH DD card is not included. Instead, you must include a DD card defining the data set containing your data base. The DDname for this DD card must be the same as your DBD name. The following is an example of the JCL for retrieving records from the data base:

//PROCLIB DD DSN=T90MSTR.JJ.IMSSIM.PROCLIB,DISP=SHR
// EXEC COBUC
// COB.SYSIN DD *
  [COBOL retrieval program]
  /*
   // EXEC NWIMSSEG,COND=(O,NE),PSBNAME=MYPsb
   // DBDLINK.SYSLIN DD DSN=T901234.IMSPDS(MYDBD),UNIT=DISK,
   //   VOL=SER=AC008,DISP=SHR
   // DD *
   NAME MYDBD
   */
// PSBLINK.SYSLIN DD DSN=T901234.IMSPDS(MYPSB),UNIT=DISK,
//   VOL=SER=AC008,DISP=SHR
// IMS.MYDBD DD DSN=T901234.DATABASE,UNIT=DISK,
//   VOL=SER=AC008,DISP=SHR
//

UPDATING THE DATA BASE

When updating the data base, you must have your original data base already loaded and should write out the modified version of the data base. To avoid problems that could arise due to the problem program (ie. - a bug), the modified data base should be written to a different data set, which requires a combination of the two previous JCL streams.

The WRITE symbolic parameter is required, but the LOADMEM is not. You must identify the data base to be loaded by including a DD card with your DBD name as the DD name. You must also include an IMS.PUNCH DD card identifying the data set to which the modified data base is to be written. An example is given on the following page.
//PROCLIB DD DSN=T90MSTR.JJ.IMSSIM.PROCLIB,DISP=SHR
//EXEC COBUC
//COB.SYSIN DD *
[COBOL update program]
/*
  EXEC NWIMSSSEG,COND=(0,NE),PSBNAME=MYPSB,
  WRITE='(MYDBD.PUNCH)
  DBDLINK.SYSLIN DD DSN=T901234.IMSPDS(MYDBD),UNIT=DISK,
  VOL=SER=ACA008,DISP=SHR
  DD *
  NAME MYDBD
  */
//PSBLINK.SYSLIN DD DSN=T901234.IMSPDS(MYPDS),UNIT=DISK,
//  VOL=SER=ACA008,DISP=SHR
//IMS.MYDBD DD DSN=T901234.DATABASE,UNIT=DISK,
//  VOL=SER=ACA008,DISP=SHR
//IMS.PUNCH DD DSN=T901234.NEW.DATABASE,UNIT=DISK,
//  VOL=SER=ACA008,DISP=SHR,SPACE=(TRK,(1,1)),
//  DCB=(LRECL=80,BLKSIZE=800,DSORG=PS,RECFM=FB)
//

NOTES

1. You must use one PSB when loading the database and another for all other processing. This affects the PSBNAME symbolic parameter.

2. Your program's name (PROGRAM-ID) must be the same as the name assigned to the program's PSB.

3. If a SOC4 occurs, you have not established addressability to the LINKAGE SECTION PCB-MASK.

4. In order to use the standard entry point (ENTRY 'DLITCBL' USING PCB-MASK) instead of a PROCEDURE DIVISION USING ... statement in the COBOL program, you must include the following four lines of JCL in the above JCL streams (these lines are to be coded directly after the EXEC NWIMSSSEG statement):

//PGMLINK.SYSLIN DD
//   DD *
//   ENTRY DLITCBL
/*

5. The DFSDUMP procedure is available in the IMS procedure library. It can be used to print out a formatted dump of the IMS database image. The following JCL is needed to execute the DFSDUMP procedure (code after the JCL for the NWIMSSSEG procedure):

//EXEC DFSDUMP,COND=(0,NE),DBD=MYDBD
//SYSUT1 DD DSN=T901234.IMSPDS(MYDBD),UNIT=DISK,
//  VOL=SER=ACA008,DISP=(OLD,KEEP)
6. Another way to identify your DBD object module in the NWIMSSEG procedure JCL is to use the INCLUDE statement. The first set of JCL shows how it was done in the above JCL examples, and the second set of JCL uses the INCLUDE statement:

```
//DBDLINK.SYSLIN DD DSN=T901234.IMSPDS(MYDBD),UNIT=DISK,
//   VOL=SER=A008,DISP=SHR
//   DD *
//   NAME MYDBD
/*

//DBDLINK.SYSLIN DD *
/*
   INCLUDE DBDLIB(MYDBD)
   NAME(MYDBD)
/*
//DBDLINK.DBDLIB DD DSN=T901234.IMSPDS,UNIT=DISK,
//   VOL=SER=A008,DISP=SHR
```
IMS ACCESS RULES

LOAD RULES:

1. Segments must be loaded in hierarchical sequence.
2. Only one segment can be added on a single call.
3. The call should have a single unqualified SSA containing the segment name of the segment being added.

RETRIEVAL RULES:

1. A call may or may not have SSA's.
2. For any level being retrieved with a SSA, the SSA may or may not have command codes and/or a qualifier.
3. If a SSA at any level is unqualified, any occurrence of that segment type will satisfy the call.
4. A GU call with an unqualified SSA at the root segment level will cause IMS to begin the search at the beginning of the data base.
5. If a SSA is omitted for one or more levels above the lowest level specified, unqualified SSA's are implied for the missing levels except in the following situation:

If the call prior to the GU call established position on a segment of the type for which there is a missing SSA, and there are no qualified SSA's at any higher level, a SSA qualified with current position is assumed for the missing SSA.

REPLACE AND DELETE RULES:

1. The segment (or path of segments) to be deleted/replaced must have been retrieved using one of the get hold calls.
2. Key (sequence) fields cannot be changed.
3. No other data base calls (for the PCB used) can be issued between the get hold and the delete/replace.
4. SSA's are used in a delete/replace call only when the corresponding get hold call retrieved a path of segments.
5. When segments are retrieved using a path call, and no SSA is specified on a replace, all retrieved segments are replaced. SSA's with the N command code indicate that certain segment(s) should not be replaced.
6. When segments are retrieved using a path call, and no SSA is specified on a delete, the highest level segment retrieved and all of its dependents are deleted. When a single SSA is included on the delete, it indicates which segment (and its dependents) is to be deleted.
INSERT RULES:

1. An insert call must have one unqualified SSA.
2. If the D command code is not used, the lowest level SSA specifies the segment type being inserted, and it must be unqualified.
3. The D command code can be used to insert a path of segments. The SSA containing the D command code and all lower level SSA's must be unqualified.
4. If a SSA at any level is unqualified, any occurrence of that segment type will satisfy the call. Omitting the SSA has the same effect.
5. An insert call with an unqualified SSA at the root level will cause IMS to begin searching from the beginning of the data base.
6. If a SSA is omitted for one or more levels above the lowest level specified, unqualified SSA's are implied for missing levels except in the following situation:

   If the call prior to the insert call established position on a segment of the type for which there is a missing SSA, and there are no qualified SSA's at any higher level, a SSA qualified with current position is assumed for the missing SSA.
IMS SIMULATOR ABEND RETURN CODES

U0001 - Invalid parameters were passed into IMS. Check the syntax of the PSBNAME, WRITE, and LOADMEM parameters.

U0002 - Invalid debug options in the debug parameters. Check for open and close parentheses and periods between module numbers.

U0003 - An error exists in the LOADMEM parameter list.

U0004 - An error exists in the WRITE parameter list.

U0005 - A problem exists in a SDB. The PSB and DBD may not match (ie. - check for a misspelled segment name or wrong parent).

U0006 - There is no JCB for what was passed to DLI as a PCB. The parameters may be out of order on a DLI call, or the PCB may have been written over.

U0007 - Too many or too little parameters were specified on a DLI call.

U0111 - Could not find the application program in the program load library created in the PGMLINK step of the NWIMSEG proc. Check your naming conventions - the PSB and the program must have the same name.

U0112 - Could not find the DDname needed for a database that must be read into. This occurs when the LOADMEM parameter is forgotten or when the DDname was typed in wrong.

U0804 - Could not get enough memory.

U0999 - The IMS simulator is having program checks internally that cannot be handled. Check to see if the application program is writing to memory outside of its region.

U1000 - DFSDUMP could not open the database. Check for a missing or misspelled SYSUT1 data set.

U1002 - DFSDUMP could not open the database or is having internal problems.
NIU does not have or use IMS; therefore, we must use an IMS simulator written and modified by students. There are some minor discrepancies between the real macros used to generate IMS control blocks and those provided by the simulator. This means that the description of these macros in the text is not totally applicable to NIU. What follows is the accurate description of the NIU IMS simulator macros:

I. DBD GENERATION MACROS

The DBD (Data Base Descriptor) generation process is used to tell IMS exactly what the data base looks like and the name of the data base. The data base is defined by coding a set of macro calls in IBM Assembler. These macros are used to define the names of the segments, the names of the fields within the segments, which fields are to be used as keys, and the lengths of the fields and segments.

DBD MACRO

The DBD macro is used to define the name of the data base and the type of structure that it is to be stored in.

Parameters:

NAME= The name of the DBD; one to eight characters long; required.

ACCESS= The access method to be used for the data base. If coded, it must be HIDAM; HIDAM is also the default.

Example: DBD NAME=MYDBD,ACCESS=HIDAM

DATASET MACRO

The DATASET macro is used to define the DDname of the data set that will be used to access the data base. This macro is also used to define the type of DASD device that the data base will/does reside on.

Parameters:

DD1= The DDname of the data set containing the data base.
  This must be the same as the DBD name.

DEVICE= The device type on which the data base resides. If coded, it must be 3350; 3350 is also the default.

Example: DATASET DD1=MYDBD,DEVICE=3350
SEGMMACRO

The SEGM macro is used to define the name, size, position of the segment in the database, and the rules to be followed when inserting new occurrences of this segment type into the database. There may be up to 256 segment types (SEGM macros) in one DBD.

Parameters:

NAME= The name of the segment; one to eight characters.

PARENT= The name of the parent segment type for this segment; for a root segment, PARENT=0; 0 is the default.

BYTES= A two-operand sublist: the first value is the minimum length of the segment, and the second is the maximum length. For fixed-length segments, the two values are the same.

RULES= A two-operand sublist: the first value is used only with logical relationships and should be omitted for our purposes. The second value is used for segments with no sequence field or non-unique sequence fields. This value determines where, within the twin chain, new segments are inserted. It can have the values of FIRST, LAST, or HERE; LAST is the default.

Example: SEGM NAME=MYSEG,PARENT=ROOTSEG,BYTES=(80,80),RULES=(,FIRST)

FIELD MACRO

The FIELD macro is used to define the name and position of a field that may be used by the application program to locate a segment based on the field's contents. There may be up to 1000 fields defined in one DBD, and each field in the DBD must have a unique name.

Parameters:

NAME= A three-operand sublist: the first value is the name of the field; if this field is not a sequence (key) field, it is the only value coded. If the field is a sequence field, two additional values must be included. The second value SEQ follows the field name, and it identifies this field as a sequence field. The third value is U or M, and it indicates if the contents of the sequence field are unique (U) or if multiple values (M) can occur.

BYTES= The length of the field; it must be greater than 0.

START= The position of the field within the segment; it must be greater than 0.

Example: FIELD NAME=(MYFIELD,SEQ,M),BYTES=25,START=1
DBDGEN MACRO

The DBDGEN macro signals the end of the DBD segment and field definitions and handles the generation of the DBD control blocks.

Parameters: None.

Example: DBDGEN

FINISH MACRO

Parameters: None.

Example: FINISH

II. PSB GENERATION MACROS

The PSB (Program Specification Block) generation is used to tell IMS what information it is to allow a specific application program to access, the name of the application program, and the language in which the application program is coded.

PCB MACRO

The PCB macro is used to specify the name of the DBD, the type of processing options for this PCB, and whether single or multiple positioning is to be used. There must be at least one PCB macro coded for a PSB. The PCB pointers passed to the user program will be in the order that they were defined in the PSBGEN.

Parameters:

TYPE= The type of PCB being generated: DB for data base; TP for data communications; the default is DB.

DBDNAME= The name of the associated DBD.

PROCOPT= The processing options for applications using this PCB. The following options are available:

G - allows read access.
I - allows inserts.
R - allows segment replacement.
A - allows any or all of the above (G, I, R); the default.
P - allows path calls (not yet implemented).
LS - for initial loading.

KEYLEN= The length of the longest possible concatenated key; default is 0.
POS= The positioning option: S (or SINGLE) for single
positioning (you want this); M (or MULTIPLE) for
multiple positioning. The default is S.

Example: PCB TYPE=DB,DBDNAME=MYDBD,PROCOPT=A,KEYLEN=21,POS=S

SENSEG MACRO

The SENSEG macro is used to specify what segments the application
program is allowed to access and what information the application
program is allowed to get out of those segments. The purpose of this
macro is to let the application program view part or all of the
hierarchy.

Parameters:

NAME= The name specified in the DBD for this segment.
PARENT= The name specified in the DBD for this segment's
parent. If NAME= names a root segment, then use
PARENT=0; 0 is the default.
PROCOPT= The processing options allowed on this segment by
application programs. For the simulator, the
following options are available:

D - allows access to segment data; the default.
K - allows access to the segment key field only.

Example: SENSEG NAME=MYSEG,PARENT=ROOTSEG,PROCOPT=D

PSBGEN MACRO

The PSBGEN macro is used to perform the generation of the control
block(s) needed to define the PCB(s). There is only one PSBGEN macro
coded for the PSB generation process no matter how many PCB macros are
coded. This macro is also used to specify the language that the
application program was coded in, and its name.

Parameters:

LANG= The application program source language; it can be
ASSEM for Assembler programs or COBOL for COBOL
programs. The default is COBOL.

PSBNAME= The name of this PSB; one to eight characters long.
It must be the same as the name assigned to the
application program.

Example: PSBGEN LANG=COBOL,PSBNAME=MYPSB
IMS CONTROL BLOCK GENERATION EXAMPLE

CONSULTANT KEY: SSN

CLIENT KEY: NAME

DEPARTMENT KEY: CODE

EMPLOYEE HISTORY KEY: POSITION

CONTRACT KEY: NUMBER

SERVICES (NO KEY)

EMPLOYEE HISTORY

CLIENT

CONSULTANT

IMS CONTROL BLOCK GENERATION EXAMPLE

01 CONSULT.
  05 SSN PIC X(9).  --- UNIQUE KEY
  05 NAME PIC X(20).
  05 ADDRESS PIC X(30).
  05 PHONE PIC X(10).
  05 SPECIALT PIC X(20).

01 CLIENT.
  05 CNAME PIC X(20).  --- NONUNIQUE KEY
  05 CADDRESS PIC X(30).
  05 CPHONE PIC X(10).

01 CONTRACT.
  05 NUMBER PIC X(7).  --- UNIQUE KEY
  05 BEGINDATE PIC X(8).  --- SEARCH
  05 ENDDATE PIC X(8).  --- SEARCH
  05 AMOUNT PIC X(5).

01 SERVICES.
  05 TOPIC PIC X(20).  --- SEARCH
  05 COMPUTER PIC X(20).

01 DEPT.
  05 CODE PIC X(5).  --- UNIQUE KEY
  05 MANAGER PIC X(20).  --- SEARCH
  05 MGRPHONE PIC X(10).

01 EMPHIST.
  05 POSITION PIC X(20).  --- UNIQUE KEY
  05 SALARY PIC X(7).
  05 STARTDT PIC X(8).
IMS CONTROL BLOCK GENERATION EXAMPLE

1. DBD (Data Base Descriptor) definition example:

    PRINT NOGEN
    *
    DBD    NAME=MYDBD,ACCESS=HIDAM
    *
    DATASET DD1=MYDBD,DEVICE=3350
    *
    SEGM    NAME=CONSULT,PARENT=0,BYTES=(89,89)
    FIELD    NAME=(SSN,SEQ,U),BYTES=9,START=1
    *
    SEGM    NAME=CLEINT,PARENT=CONSULT,BYTES=(60,60),
    RULES=(,LAST)
    FIELD    NAME=(CNAME,SEQ,M),BYTES=20,START=1
    *
    SEGM    NAME=CONTRACT,PARENT=CLIENT,BYTES=(28,28)
    FIELD    NAME=(NUMBER,SEQ,U),BYTES=7,START=1
    FIELD    NAME=BEGINDAT,BYTES=8,START=8
    FIELD    NAME=ENDDATE,BYTES=8,START=16
    *
    SEGM    NAME=SERVICES,PARENT=CONTRACT,BYTES=(40,40),
    RULES=(,LAST)
    FIELD    NAME=TOPIC,BYTES=20,START=1
    *
    SEGM    NAME=DEPT,PARENT=CONSULT,BYTES=(35,35)
    FIELD    NAME=(CODE,SEQ,U),BYTES=5,START=1
    FIELD    NAME=MANAGER,BYTES=20,START=6
    *
    SEGM    NAME=EMPHIST,PARENT=CONSULT,BYTES=(35,35)
    FIELD    NAME=(POSITION,SEQ,U),BYTES=20,START=1
    *
    DBDGEN
    *
    FINISH
    *
    END
IMS CONTROL BLOCK GENERATION EXAMPLE

2. PSB (Program Status Block) definition example:

```
PRINT NOGEN
PCB TYPE=DB,DBDNAME=MYDBD,PROCOPT=LS,KEYLEN=217,POS=S
SENSEG NAME=CONSULT,PARENT=0,PROCOPT=D
SENSEG NAME=CLIENT,PARENT=CONSULT,PROCOPT=D
SENSEG NAME=CONTRACT,PARENT=CLIENT,PROCOPT=D
SENSEG NAME=SERVICES,PARENT=CONTRACT,PROCOPT=D
SENSEG NAME=DEPT,PARENT=CONSULT,PROCOPT=D
SENSEG NAME=EMPHIST,PARENT=CONSULT,PROCOPT=D
PSBGEN LANG=COBOL,PSBNAME=CONSULTL
END
```
<table>
<thead>
<tr>
<th>STATUS CODE</th>
<th>DATA BASE CALLS</th>
<th>MSG CALLS</th>
<th>SYSTEM SERVICE CALLS</th>
<th>CALL COMPLETED</th>
<th>ERROR IN CALL</th>
<th>I/O OR SYST ERROR</th>
<th>CATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AB</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AC</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AD</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AH</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AI</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AJ</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AK</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AL</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AM</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AO</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AP</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AQ</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AT</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AU</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AU</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>AZ</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>A1</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>A2</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>A3</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>A4</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>A5</td>
<td>X X X X X X X X X X X X</td>
<td>X X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

**Description**

- CHNG CALL FOR RESPONSE ALTERNATE PCB CAN ONLY SPECIFY LOGICAL TERMINAL DESTINATION TRANSACTION CODE DESTINATION SPECIFIED.
- SEGMENT I/O AREA REQUIRED, NONE SPECIFIED IN CALL.
- HIERARCHICAL ERROR IN SSA.
- INVALID FUNCTION PARAMETER.
- CALL REQUIRES SSA, NONE PROVIDED.
- DATA MANAGEMENT OPEN ERROR.
- INVALID SSA QUALIFICATION FORMAT.
- INVALID FIELD NAME IN CALL.
- CALL USING LT PCB IN BATCH PG.
- CALL FUNCTION NOT COMPATIBLE WITH PROCESSING OPTION OR SMTP SENSITIVITY.
- I/O ERROR ISAM, OSAM, BSAM, OR VSAM.
- MORE THAN FOUR USER CALL PARAMETERS FOR A TPPCB ARE INVALID.
- READ I/O ERROR, MESSAGE CHAIN CANNOT BE FOLLOWED, MINIMUM OF ONE MESSAGE LOST.
- USER I/O AREA TOO LONG.
- SSA TOO LONG.
- RESPONSE ALTERNATE PCB REFERENCED BY ISRT CALL HAS MORE THAN ONE PHYSICAL TERMINAL ASSIGNED FOR INPUT PURPOSES. NOTIFY MASTER TERMINAL.
- THE CONVERSATIONAL PROGRAM HAS ISSUED A PURGE CALL TO A PCB THAT CANNOT BE PURGED.
- CALL ATTEMPTED WITH B-CHAR LOGICAL TERMINAL NAME NOT KNOWN TO SYSTEM.
- CHANGE ATTEMPTED WITH INVALID PCB.
- INSERT/PURGE ATTEMPTED TO A MOD TP PCB WITH NO DESTINATION SET.
- SECURITY VIOLATION.
- FORMAT NAME SPECIFIED ON 2ND OR SUBSEQUENT MSG ISRT OR PURG.
<table>
<thead>
<tr>
<th>STATUS CODE</th>
<th>DATA BASE CALLS</th>
<th>MSG CALLS</th>
<th>SYSTEM SERVICE CALLS</th>
<th>CALL COMPLETED</th>
<th>ERROR IN CALL</th>
<th>I/O OR SYST ERROR</th>
<th>CATEGORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>NI</td>
<td></td>
<td></td>
<td></td>
<td>NO</td>
<td>X</td>
<td></td>
<td>1</td>
<td>INDEX MAINTENANCE FOUND DUPLICATE SEGMENT IN INDEX.</td>
</tr>
<tr>
<td>NO</td>
<td></td>
<td></td>
<td></td>
<td>NO</td>
<td>X</td>
<td></td>
<td>4</td>
<td>I/O ERROR ISAM, OSAM, BSAM, OR VSAM.</td>
</tr>
<tr>
<td>QC</td>
<td></td>
<td>X</td>
<td></td>
<td>NO</td>
<td>X</td>
<td></td>
<td>2</td>
<td>NO MORE INPUT MESSAGES.</td>
</tr>
<tr>
<td>QD</td>
<td></td>
<td>X</td>
<td></td>
<td>NO</td>
<td>X</td>
<td></td>
<td>1</td>
<td>NO MORE SEGMENTS FOR THIS MESSAGE.</td>
</tr>
<tr>
<td>QG</td>
<td></td>
<td></td>
<td></td>
<td>NO</td>
<td>X</td>
<td></td>
<td>3</td>
<td>GET NEXT REQUEST BEFORE GET UNIQUE.</td>
</tr>
<tr>
<td>OF</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>SEGMENT LESS THAN FIVE CHARACTERS (SEG LENGTH IS MSG TEXT LENGTH PLUS FOUR CONTROL CHARACTERS).</td>
</tr>
<tr>
<td>QH</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>TERMINAL SYMBOLIC ERROR - OUTPUT DESIGNATION UNKNOWN TO IMS/VS (LOGICAL TERMINALS OR TRAN CODE).</td>
</tr>
<tr>
<td>RX</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>VIOLATED REPLACE RULE.</td>
</tr>
<tr>
<td>UC</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>CHECKPOINT * TAKEN.</td>
</tr>
<tr>
<td>UR</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>RESTART*.</td>
</tr>
<tr>
<td>US</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>STOP*.</td>
</tr>
<tr>
<td>UX</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>CHECKPOINT AND STOP*.</td>
</tr>
<tr>
<td>V1</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>3</td>
<td>INVALID LENGTH FOR VARIABLE LENGTH SEGMENT.</td>
</tr>
<tr>
<td>X1</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>4</td>
<td>I/O ERROR WRITING SPA.</td>
</tr>
<tr>
<td>X2</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>1ST INSERT TO TRAHCODE PCB THAT IS CONVERSATIONAL, IS NOT AN SPA.</td>
</tr>
<tr>
<td>X3</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>INVALID SPA.</td>
</tr>
<tr>
<td>X4</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>INSERT TO A TRAN CODE PCB THAT IS NOT CONVERSATIONAL AND THE SEGMENT IS AN SPA.</td>
</tr>
<tr>
<td>X5</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>INSERT OF MULTIPLE SPAs TO TRAN CODE PCB.</td>
</tr>
<tr>
<td>X6</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>INVALID TRAN CODE NAME INSERTED INTO SPA.</td>
</tr>
<tr>
<td>X7</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>LENGTH OF SPA IS INCORRECT (USER MODIFIED FIRST SIX BYTES).</td>
</tr>
<tr>
<td>X8</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>4</td>
<td>ERROR ATTEMPTING TO QUEUE AN SPA ON A TRAN CODE PCB.</td>
</tr>
<tr>
<td>X9</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>INCOMPATIBLE CONVERSATIONAL PROGRAM CALL PATH.</td>
</tr>
<tr>
<td>XA</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>3</td>
<td>ATTEMPT TO CONT. PROC. CONV. BY PASSING SPA VIA PGM-TO-PGM SW AFTER ANSWERING TERMINAL.</td>
</tr>
</tbody>
</table>

*Utility Control Facility Status Codes*
<table>
<thead>
<tr>
<th>STATUS CODE</th>
<th>DATA BASE CALLS</th>
<th>MSC CALLS</th>
<th>SYSTEM SERVICE CALLS</th>
<th>CALL COMPLETED</th>
<th>ERROR IN CALL</th>
<th>I/O OR SYS ERROR</th>
<th>CATEGORY</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>XB</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>3</td>
<td>PGM PASSED SPA TO OTHER PGM BUT TRYING TO RESPOND.</td>
</tr>
<tr>
<td>XC</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>3</td>
<td>PGM INSERTED MSG WITH 21 FLD BITS SET RESERVED FOR SYSTEM USE.</td>
</tr>
<tr>
<td>XD</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>IMS IS TERMINATING. FURTHER DL/I CALLS MUST NOT BE ISSUED. NO MESSAGE RETURNED.</td>
</tr>
<tr>
<td>XE</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>3</td>
<td>TRIED TO ISRT SPA TO EXPRESS PCB.</td>
</tr>
<tr>
<td>XF</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>3</td>
<td>ALTERNATE PCB REFERENCED IN ISRT CALL FOR SPA HAD DESTINATION SET TO A LOGICAL TERMINAL, BUT WAS NOT DEFINED AS ALTRESP = YES.</td>
</tr>
<tr>
<td>XG</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>3</td>
<td>CURRENT CONVERSATION REQUIRES FIXED-LENGTH SPAS. ATTEMPT WAS MADE TO INSERT SPA TO TRANSACTION WITH A DIFFERENT OR NON-FIXED LENGTH SPA.</td>
</tr>
<tr>
<td>BB</td>
<td>X X X X X X</td>
<td>X</td>
<td>X X X X X X X X</td>
<td>X X X X X X</td>
<td>X X X X X X</td>
<td>YES</td>
<td></td>
<td>GOOD. NO STATUS CODE RETURNED, PROCEED.</td>
</tr>
</tbody>
</table>

* bb indicates blanks*
CUSTOMER INFORMATION CONTROL SYSTEM (CICS)
CICS MAP DEFINITION EXAMPLE

GRADE RECORDING SYSTEM

SSN: ___________X
CLASS: _____X
GRADE: _X

INVALID INPUT -- PLEASE REENTER
CICS MAP DEFINITION EXAMPLE

GRSET DFHMSD TYPE=DSECT,MODE=INOUT,CTRL=FREEKB, (col. 72)
LANG=COBOL,TIOAPFX=YES

* GRMAP DFHMDI SIZE=(24,80)
*
DFHMDF POS=(1,30),LENGTH=22,ATTRB=(ASKIP,NORM), *
INITIAL='GRADE RECORDING SYSTEM'
*
DFHMDF POS=(6,35),LENGTH=4,ATTRB=(ASKIP,NORM), *
INITIAL='SSN:
*
STSSN DFHMDF POS=(6,40),LENGTH=9,ATTRB=(UNPROT,NORM,IC)
DFHMDF POS=(6,50),LENGTH=1,ATTRB=(ASKIP,NORM), *
INITIAL='
*
DFHMDF POS=(7,33),LENGTH=6,ATTRB=(ASKIP,NORM), *
INITIAL='CLASS:
*
STCLASS DFHMDF POS=(7,40),LENGTH=4,ATTRB=(UNPROT,NUM,NORM)
DFHMDF POS=(7,45),LENGTH=1,ATTRB=(ASKIP,NORM), *
INITIAL='
*
DFHMDF POS=(8,33),LENGTH=6,ATTRB=(ASKIP,NORM), *
INITIAL='GRADE:
*
STGRADE DFHMDF POS=(8,40),LENGTH=1,ATTRB=(UNPROT,NORM)
DFHMDF POS=(8,42),LENGTH=1,ATTRB=(ASKIP,DRK)
*
ERRMSG1 DFHMDF POS=(14,24),LENGTH=31,ATTRB=(ASKIP,DRK), *
INITIAL='INVALID INPUT -- PLEASE REENTER'
*
DFHMSD TYPE=FINAL
*
END

Note: If the INITIAL='...' is too long to fit on one line
 type up to column 71, place a continuation "*" or
"X" in column 72, and begin typing on the next line
 in column 16.
CICS MAP DEFINITION EXAMPLE

01 GRMAPI.
  05 FILLER PIC X(12).
  05 STSSNL COMP PIC X9(4).
  05 STSSNF PIC X.
  05 FILLER REDEFINES STSSNF.
    10 STSSNA PIC X.
  05 STSSNI PIC X(9).
  05 STCLASSL COMP PIC S9(4).
  05 STCLASSF PIC X.
  05 FILLER REDEFINES STCLASSF.
    10 STCLASSA PIC X.
  05 STCLASSI PIC X(4).
  05 STGRADEL COMP PIC S9(4).
  05 STGRADEF PIC X.
  05 FILLER REDEFINES STGRADEF.
    10 STGRADEA PIC X.
  05 STGRADEI PIC X(1).
  05 ERRMSGIL COMP PIC S9(4).
  05 ERRMSG1F PIC X.
  05 FILLER REDEFINES ERRMSG1F.
    10 ERRMSG1A PIC X.
  05 ERRMSG1I PIC X(31).

01 GRMAPO REDEFINES GRMAPI.
  05 FILLER PIC X(12).
  05 FILLER PIC X(3).
  05 STSSNO PIC X(9).
  05 FILLER PIC X(3).
  05 STCLASSO PIC X(4).
  05 FILLER PIC X(3).
  05 STGRADEO PIC X(1).
  05 FILLER PIC X(3).
  05 ERRMSG1O PIC X(31).
CICS PROCEDURES

GENSMAP: in CICS.BATCH.PROCLIB

GENSMAP is a one-step procedure which executes the Assembler and may be used to create a symbolic map. The only required data set is the SYSIN data set on the ASSEM step. The symbolic map which is created in this step is routed to the printer. You may have this routed to a copy library by overriding the SYSLIN DD statement on the ASSEM step.

Example:

//PROCLIB DD DSN=CICS.BATCH.PROCLIB,DISP=SHR
// EXEC GENSMAP
// ASSEM.SYSLIN DD DSN=[your copy library]
// ASSEM.SYSIN DD *
// [your macros TYPE=DSECT]
//

GENPMAP: in CICS.BATCH.PROCLIB

GENPMAP is a two-step procedure which may be used to create a physical map. The first step (ASSEM) executes the Assembler. The only additional data set which you must provide is the SYSIN data set. The second step (LKED) executes the linkage editor. The only required data set on this step is the SYSLMOD data set.

Example:

//PROCLIB DD DSN=CICS.BATCH.PROCLIB,DISP=SHR
// EXEC GENPMAP
// ASSEM.SYSLIN DD *
// [your macros TYPE=MAP]
// LKED.SYSLMOD DD DSN=[a load module library]
//

GENPGM: in CICS.BATCH.PROCLIB

GENPGM is a three-step procedure which may be used to create a CICS program load module from COBOL source code. The first step (TRN) executes the CICS precompiler processor, which translates CICS commands into the appropriate CALL statements. The only data set which you must provide is the SYSIN data set which contains your COBOL source program. The second step (COB) executes the COBOL compiler. There are no data sets for you to provide on this step. The last step (LKED) executes the linkage editor. You must supply a SYSLMOD DD statement on this step. An example is on the following page.
CSCI466 is a procedure used to execute CICS. There are several symbolic parameters which you must provide when using this procedure:

- **LID=** your MVS logonid.
- **VOL=** the volume on which your load module library resides.
- **INPUT=** the name of the data set containing the simulated terminal input. This will change for each assignment and will be provided.
- **EXPARM=** file processing options. Once again, this will change for each assignment and will be provided.

The only required data sets are those processed by your program; generally you will provide a VSAM data set under your own account; the DDname required will be provided.

Example:

```
//PROCLIB DD DSN=CICS.BATCH.PROCLIB,DISP=SHR
// EXEC CSCI466,LID=T901234,VOL=ACA008,
// INPUT='T90MSTR.SOURCE,EXPARM=',FCT=91'
//GO.ddname DD DSN=T90MSTR.VSAM.FILE,DISP=SHR
```

**NOTE:** There are naming conventions which are assumed by this procedure: it is assumed that your load module library is named T90____.CICS.LOADLIB, where T90____ is your MVS account. Therefore, you must use this convention when creating your load module library.

**NOTE:** There are REGION parameters specified on each EXEC statement within this procedure; therefore, you must **NOT** include a REGION parameter on your JOB card, but you **must** include a REGION parameter on any EXEC statements you include in the job used to run CICS (ie. - IDCAMS DEFINE CLUSTER, etc.).
The following information should be considered when working on CICS Assignments 10, 11, and 12:

The following compiler SYSLIB DD statements are generated by the GENPGM procedure:

```
//SYSLIB  DD  DSN=T90MSTR.CICS.SORCLIB,DISP=SHR
//  DD  DSN=&INDEX..COBLIB,DISP=SHR
```

The T90MSTR.CICS.SORCLIB data set is not needed for assignments 10, 11, and 12. However, your own COPYLIB is needed since the symbolic map must be copied into the programs.

1. If the member name of your symbolic map is not 'BCC0SET' (there is a 'BCC0SET' member in the T90MSTR.... data set), the following concatenation can be used:

```
//COB.SYSLIB  DD
//  DD
//    DD  DSN=...(Your copy library)...
```

2. If the member name of your symbolic map is 'BCC0SET', the following override can be used:

```
//COB.SYSLIB  DD  DSN=&INDEX..COBLIB,DISP=SHR
//  DD  DSN=...(Your copy library)...
```
CICS ABEND CODES

ABMO - The map specified in a EXEC CICS command could not be located.

AEIx - An exceptional condition has occurred for which there was not an EXEC CICS HANDLE CONDITION command. Some of the abend codes and their corresponding exceptional conditions are as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AEIA</td>
<td>ERROR</td>
</tr>
<tr>
<td>AEIM</td>
<td>NOTFND</td>
</tr>
<tr>
<td>AEIP</td>
<td>INVREQ</td>
</tr>
<tr>
<td>AEIS</td>
<td>NOTOPEN</td>
</tr>
<tr>
<td>AEIV</td>
<td>LENGERR</td>
</tr>
<tr>
<td>AEIO</td>
<td>PGMIDERR</td>
</tr>
<tr>
<td>AEI9</td>
<td>MAPFAIL</td>
</tr>
</tbody>
</table>

AFCA - The transaction has attempted to access a data base file that does not exist or has been disabled.

APCT - A requested module cannot be located in the PPT, or the entry was disabled, or the module has a length of zero.

ASRA - The task has been abnormally terminated because of a program interrupt.

ATNI - The task has been abnormally terminated because of a terminal error.
### Attribute Byte Definitions

**01 FIELD-ATTRIBUTE-DEFINITIONS.**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Value Intensity</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>05 UNPROT</td>
<td>PIC X</td>
<td>VALUE ' '</td>
<td>0100 0000</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-MDT</td>
<td>PIC X</td>
<td>VALUE 'A'</td>
<td>1100 0001</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-BRT</td>
<td>PIC X</td>
<td>VALUE 'H'</td>
<td>1100 1000</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-BRT-MDT</td>
<td>PIC X</td>
<td>VALUE 'I'</td>
<td>1100 1001</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-DARK</td>
<td>PIC X</td>
<td>VALUE '&lt;'</td>
<td>0100 1100</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-DARK-MDT</td>
<td>PIC X</td>
<td>VALUE '('</td>
<td>0100 1101</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Value Intensity</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>05 UNPROT-NUM</td>
<td>PIC X</td>
<td>VALUE '&amp;</td>
<td>0101 0000</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-NUM-MDT</td>
<td>PIC X</td>
<td>VALUE 'J'</td>
<td>1101 0001</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-NUM-BRT</td>
<td>PIC X</td>
<td>VALUE 'Q'</td>
<td>1101 1000</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-NUM-BRT-MDT</td>
<td>PIC X</td>
<td>VALUE 'R'</td>
<td>1101 1001</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-NUM-DARK</td>
<td>PIC X</td>
<td>VALUE '*'</td>
<td>0101 1100</td>
<td></td>
</tr>
<tr>
<td>05 UNPROT-NUM-DARK-MDT</td>
<td>PIC X</td>
<td>VALUE ')'</td>
<td>0101 1101</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Value Intensity</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>05 PROT</td>
<td>PIC X</td>
<td>VALUE '-</td>
<td>0110 0000</td>
<td></td>
</tr>
<tr>
<td>05 PROT-MDT</td>
<td>PIC X</td>
<td>VALUE '/'</td>
<td>0110 0001</td>
<td></td>
</tr>
<tr>
<td>05 PROT-BRT</td>
<td>PIC X</td>
<td>VALUE 'Y'</td>
<td>1110 1000</td>
<td></td>
</tr>
<tr>
<td>05 PROT-BRT-MDT</td>
<td>PIC X</td>
<td>VALUE 'Z'</td>
<td>1110 1001</td>
<td></td>
</tr>
<tr>
<td>05 PROT-DARK</td>
<td>PIC X</td>
<td>VALUE '%'</td>
<td>0110 1100</td>
<td></td>
</tr>
<tr>
<td>05 PROT-DARK-MDT</td>
<td>PIC X</td>
<td>VALUE '_'</td>
<td>0110 1101</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
<th>Value Intensity</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>05 PROT-SKIP</td>
<td>PIC X</td>
<td>VALUE '0'</td>
<td>1111 0000</td>
<td></td>
</tr>
<tr>
<td>05 PROT-SKIP-MDT</td>
<td>PIC X</td>
<td>VALUE '1'</td>
<td>1111 0001</td>
<td></td>
</tr>
<tr>
<td>05 PROT-SKIP-BRT</td>
<td>PIC X</td>
<td>VALUE '8'</td>
<td>1111 1000</td>
<td></td>
</tr>
<tr>
<td>05 PROT-SKIP-BRT-MDT</td>
<td>PIC X</td>
<td>VALUE '9'</td>
<td>1111 1001</td>
<td></td>
</tr>
<tr>
<td>05 PROT-SKIP-DARK</td>
<td>PIC X</td>
<td>VALUE '@'</td>
<td>0111 1100</td>
<td></td>
</tr>
<tr>
<td>05 PROT-SKIP-DARK-MDT</td>
<td>PIC X</td>
<td>VALUE '&quot;'</td>
<td>0111 1101</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Normal intensity and alphanumeric are assumed unless otherwise stated.
**ATTRIBUTE BYTE 8-BIT FORMAT**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,1</td>
<td>Value is determined from value of bits 2-7</td>
</tr>
<tr>
<td>2,3</td>
<td>0,0 = unprotected, alphanumeric</td>
</tr>
<tr>
<td></td>
<td>0,1 = unprotected, numeric</td>
</tr>
<tr>
<td></td>
<td>1,0 = protected, alphanumeric</td>
</tr>
<tr>
<td></td>
<td>1,1 = autoskip</td>
</tr>
<tr>
<td>4,5</td>
<td>0,0 = normal intensity</td>
</tr>
<tr>
<td></td>
<td>0,1 = normal intensity</td>
</tr>
<tr>
<td></td>
<td>1,0 = high intensity</td>
</tr>
<tr>
<td></td>
<td>1,1 = dark intensity</td>
</tr>
<tr>
<td>6</td>
<td>must be 0</td>
</tr>
<tr>
<td>7</td>
<td>0 = not modified (MDT)</td>
</tr>
<tr>
<td></td>
<td>1 = modified (MDT)</td>
</tr>
</tbody>
</table>
CICS EXAMPLE PROGRAMS

BCC5PGM - is written in conversational mode. It sums user entered four-digit numbers until the user types the word 'DONE'. It then sends the user a map containing the sum of the numbers entered.

BCC6PGM - is written in pseudoconversational mode and contains a one-byte communication area. It sums two user entered four-digit numbers and sends the user a map containing the sum of the two numbers.

BCC7PGM - is written in pseudoconversational mode and contains a five-byte communication area. It is a pseudoconversational version of BCC5PGM.

BCC8PGM - is written in pseudoconversational mode and contains a five-byte communication area. While this program performs the same function as BCC7PGM, it handles user entered non-numeric data and also includes routines to handle unexpected processing errors, user failure to enter requested data, and user termination by CLEAR key. Printed output, simulating user input and output screens, is included.
CICS EXAMPLE PROGRAM - BCC5PGM
IDENTIFICATION DIVISION.
PROGRAM-ID. BCCSPGM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY BCCBSET.

PROCEDURE DIVISION.
MOVE 'N' TO WS-END-FLAG.
PERFORM OO1-ADD-LOOP-RTN
UNTIL WS-END-FLAG IS EQUAL TO 'Y'.
MOVE WS-SUM TO TOTALL.
EXEC CICS SEND
MAP('SCREEN3')
MAPSET('BCCBSET')
ERASE
END-EXEC.
EXEC CICS RETURN
END-EXEC.
GOBACK.

OO1-ADD-LOOP-RTN.
EXEC CICS SEND
MAP('SCREEN2')
MAPSET('BCCBSET')
MAPONLY
ERASE
END-EXEC.
EXEC CICS RECEIVE
MAP('SCREEN2')
MAPSET('BCCBSET')
END-EXEC.
IF ENNUMI IS EQUAL TO 'DONE'

<table>
<thead>
<tr>
<th>LINE</th>
<th>SOURCE LISTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>00052</td>
<td>MOVE 'Y' TO WS-END-FLAG</td>
</tr>
<tr>
<td>00053</td>
<td>ELSE</td>
</tr>
<tr>
<td>00056</td>
<td>MOVE ENTRNUM TO WS-NUM</td>
</tr>
<tr>
<td>00057</td>
<td>ADD WS-NUM TO WS-SUM</td>
</tr>
<tr>
<td>00060</td>
<td>001-ADD-LOOP-EXIT. EXIT.</td>
</tr>
</tbody>
</table>
IDENTIFICATION DIVISION.

PROGRAM-ID. BCC5PGM.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 WS-END-FLAG PIC X VALUE 'N'.

01 WS-SUM-FIELDS.

05 WS-SUM PIC 9(5) VALUE 0.

05 WS-NUM PIC 9(4).

COPY BCC5SET.

01 SCREEN1.

02 FILLER PIC X(12).

02 NUM1 COMP PIC S9(4).

02 NUM1F PIC X.

02 FILLER REDEFINES NUM1F.

03 NUM1A PIC X.

02 NUM1 PIC X(4).

02 NUM2L COMP PIC S9(4).

02 NUM2F PIC X.

02 FILLER REDEFINES NUM2F.

03 NUM2A PIC X.

02 NUM2 PIC X(4).

02 ERRMSG1L COMP PIC S9(4).

02 ERRMSG1F PIC X.

02 FILLER REDEFINES ERRMSG1F.

03 ERRMSG1A PIC X.

02 ERRMSG1I PIC X(50).

01 SCREEN10 REDEFINES SCREEN1.

02 FILLER PIC X(12).

02 FILLER PIC X(3).

02 NUM10 PIC X(4).

02 FILLER PIC X(9).

02 NUM20 PIC X(4).

02 FILLER PIC X(3).

02 ERRMSG10 PIC X(50).

01 SCREEN2.

02 FILLER PIC X(12).

02 ENTRNUML COMP PIC S9(4).

02 ENTRNUMF PIC X.

02 FILLER REDEFINES ENTRNUMF.

03 ENTRNUMA PIC X.

02 ENTRNUMI PIC X(4).

02 ERRMSG2L COMP PIC S9(4).

02 ERRMSG2F PIC X.

02 FILLER REDEFINES ERRMSG2F.

03 ERRMSG2A PIC X.

02 ERRMSG2I PIC X(50).

01 SCREEN20 REDEFINES SCREEN2.

02 FILLER PIC X(12).
I
00112 C      *      EIBCPDSN      CURSOR POSITION      30000000
00113 C      02      EIBCPDSN      PICTURE S9(4) USAGE COMPUTATIONAL.      32000000
00114 C      *      EIBCALN      COMMAREA LENGTH      34000000
00115 C      02      EIBCALN      PICTURE S9(4) USAGE COMPUTATIONAL.      36000000
00116 C      *      EIBAID      ATTENTION IDENTIFIER      38000000
00117 C      02      EIBAID      PICTURE X(4)      40000000
00118 C      *      EIBFN      FUNCTION CODE      42000000
00119 C      02      EIBFN      PICTURE X(2)      44000000
00120 C      *      EIBRCDSE      RESPONSE CODE      46000000
00121 C      02      EIBRCDSE      PICTURE X(6)      48000000
00122 C      *      EIBDS      DATASET NAME      50000000
00123 C      02      EIBDS      PICTURE X(8)      52000000
00124 C      *      EIBREQID      REQUEST IDENTIFIER      54000000
00125 C      02      EIBREQID      PICTURE X(8)      56000000
00126 C      *      EIBRSRCE      RESOURCE NAME      58000000
00127 C      02      EIBRSRCE      PICTURE X(8)      60000000
00128 C      *      EIBSYNC      SYNCPOINT REQUIRED      62000000
00129 C      02      EIBSYNC      PICTURE X.      64000000
00130 C      *      EIBFREE      TERMINAL FREE REQUIRED      66000000
00131 C      02      EIBFREE      PICTURE X.      68000000
00132 C      *      EIBRECV      DATA RECEIVE REQUIRED      70000000
00133 C      02      EIBRECV      PICTURE X.      72000000
00134 C      *      EIBSEND      RESERVED      74000000
00135 C      02      EIBSEND      PICTURE X.      76000000
00136 C      *      EIBATT      ATTACH DATA EXISTS      78000000
00137 C      02      EIBATT      PICTURE X.      80000000
00138 C      *      EIBEDC      GOTTEN DATA IS COMPLETE      82000000
00139 C      02      EIBEDC      PICTURE X.      84000000
00140 C      *      EIBFMH      GOTTEN DATA CONTAINS FMH      86000000
00141 C      02      EIBFMH      PICTURE X.      88000000
00142      01      DFHCOMMAREA      PICTURE X(1).      90000000
00143      01      DFHBLCSLOT1      PICTURE X(1).      92000000
00144      01      DFHBLCSLOT2      PICTURE X(1).      94000000
00145      PROCEDURE DIVISION USING DFHEILBLK DFHCOMMAREA.
00146      SERVICE RELOAD DFHEILBLK.
00147      MOVE 'N' TO WS-EN-FLAG.
00148      MOVE 'N' TO WS-EN-FLAG.
00149      PERFORM 001-ADD-LOOP-RTN
00150      UNTIL WS-EN-FLAG IS EQUAL TO 'Y'.
00151      MOVE WS-SUM TO TOTALO.
00152      EXEC CICS SEND
00153      MAP('SCREEN3')
00154      MAPSET('BCCBSBSET')
00155      ERASE
00156      END-EXEC.
00157      MOVE 'SCREEN3' TO DFHEIV1 MOVE 'BCCBSBSET' TO DFHEIV2 MOVE
00158      DFHEIV1 SCREEN30 DFHEIV89 DFHEIV2.
00159      EXEC CICS RETURN
00160      EXEC CICS RETURN
00161      *      TO DFHEIVO CALL 'DFHEIL' USING DFHEIVO
00162      TO DFHEIVO CALL 'DFHEIL' USING DFHEIVO.
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208

GOBACK.

EXEC CICS SEND
* MAP('SCREEN2')
* MAPSET('BCC8SET')
* MAPONLY
* ERASE
END-EXEC.

MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE '
) TO DFHEIVO CALL 'DFHEI' USING DFHEIVO
DFHEIV1 DFHEIV99 DFHEIV8 DFHEIV2.

EXEC CICS RECEIVE
* MAP('SCREEN2')
* MAPSET('BCC8SET')
END-EXEC.

MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE '
) TO DFHEIVO CALL 'DFHEI' USING DFHEIVO
DFHEIV1 DFHEIV98 DFHEIV2.

IF ENNUMI IS EQUAL TO 'DONE'
MOVE 'Y' TO WS-END-FLAG
ELSE
MOVE ENNUMI TO WS-NUM
ADD WS-NUM TO WS-SUM.

OO1-ADD-LOOP-EXIT. EXIT.
LINE      SOURCE LISTING
00001     IDENTIFICATION DIVISION.
00002     PROGRAM-ID. BCCBPGM.
00003     ENVIRONMENT DIVISION.
00004     DATA DIVISION.
00005     WORKING-STORAGE SECTION.
00006     COPY BCCBSSET.
00007     COPY DFHAID.
00008     LINKAGE SECTION.
00009     01 WS-SUM-FIELDS.
00010     05 WS-SUM                   PIC 9(5) VALUE 0.
00011     05 WS-NUM1                 PIC 9(4).
00012     05 WS-NUM2                 PIC 9(4).
00013     01 DFHCOMMAREA             PIC X.
00014     PROCEDURE DIVISION.
00015     IF EIBCALEN IS EQUAL TO ZEROS
00016     PERFORM 001-INITIAL-RTN
00017     ELSE
00018     PERFORM 002-RECEIVE-RTN.
00019     GOBACK.
00020     001-INITIAL-RTN.
00021     EXEC CICS SEND
00022     MAP('SCREEN1')
00023     MAPSET('BCCBSSET')
00024     MAPONLY
00025     ERASE
00026     END-EXEC.
00027     EXEC CICS RETURN
00028     TRANSID('BCC6')
00029     COMMAREA(DFHCOMMAREA)
00030     LENGTH(1)
00031     END-EXEC.
00032     002-RECEIVE-RTN.
00033     EXEC CICS RECEIVE
CICS/VSE COMMAND LANGUAGE TRANSLATOR VERSION 1.5

LINE SOURCE LISTING

00052   MAP('SCREEN1')
00053   MAPSET('BCC8SET')
00054   END-EXEC.
00055
00056   MOVE NUM1 TO WS-NUM1.
00057   MOVE NUM2 TO WS-NUM2.
00058
00059   ADD WS-NUM1 WS-NUM2 TO WS-SUM.
00060
00061   MOVE WS-SUM TO TOTALO.
00062
00063   EXEC CICS SEND
00064   MAP('SCREEN3')
00065   MAPSET('BCC8SET')
00066   ERASE
00067   END-EXEC.
00068
00069   EXEC CICS RETURN
00070   END-EXEC.
00071
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00055</td>
<td>O2 ENTRNUMO PIC X(4).</td>
<td></td>
</tr>
<tr>
<td>00056</td>
<td>O2 FILLER PICTURE X(3).</td>
<td></td>
</tr>
<tr>
<td>00057</td>
<td>O2 ERRMSG20 PIC X(50).</td>
<td></td>
</tr>
<tr>
<td>00058</td>
<td>O1 SCREEN3.</td>
<td></td>
</tr>
<tr>
<td>00059</td>
<td>O2 FILLER PIC X(12).</td>
<td></td>
</tr>
<tr>
<td>00060</td>
<td>O2 TOTALF PIC 38(4).</td>
<td></td>
</tr>
<tr>
<td>00061</td>
<td>O2 TOTALF PICTURE X.</td>
<td></td>
</tr>
<tr>
<td>00062</td>
<td>O2 FILLER REDEFINES TOTALF.</td>
<td></td>
</tr>
<tr>
<td>00063</td>
<td>O3 TOTALA PICTURE X.</td>
<td></td>
</tr>
<tr>
<td>00064</td>
<td>O2 TOTALI PIC X(5).</td>
<td></td>
</tr>
<tr>
<td>00065</td>
<td>O1 SCREEN30 REDEFINES SCREEN3I.</td>
<td></td>
</tr>
<tr>
<td>00066</td>
<td>O2 FILLER PIC X(12).</td>
<td></td>
</tr>
<tr>
<td>00067</td>
<td>O2 FILLER PICTURE X(3).</td>
<td></td>
</tr>
<tr>
<td>00068</td>
<td>O2 TOTALO PIC X(5).</td>
<td></td>
</tr>
</tbody>
</table>

**COPY DFHAILD.**

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00070</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00071</td>
<td>O1 DFHAILD.</td>
<td></td>
</tr>
<tr>
<td>00072</td>
<td>O2 DFHNULL PIC X VALUE IS ' '</td>
<td>04000000</td>
</tr>
<tr>
<td>00073</td>
<td>O2 DFHINTER PIC X VALUE IS QUOTE.</td>
<td>06000000</td>
</tr>
<tr>
<td>00074</td>
<td>O2 DFHCLEAR PIC X VALUE IS ' '</td>
<td>08000000</td>
</tr>
<tr>
<td>00075</td>
<td>O2 DFHPIC PIC X VALUE IS ' '</td>
<td>10000000</td>
</tr>
<tr>
<td>00076</td>
<td>O2 DFHOPID PIC X VALUE IS 'W'</td>
<td>12000000</td>
</tr>
<tr>
<td>00077</td>
<td>O2 DFHMSRE PIC X VALUE IS 'X'</td>
<td>14000000</td>
</tr>
<tr>
<td>00078</td>
<td>O2 DFHSTRF PIC X VALUE IS 'H'</td>
<td>16000000</td>
</tr>
<tr>
<td>00079</td>
<td>O2 DFHTRIG PIC X VALUE IS ' '</td>
<td>18000000</td>
</tr>
<tr>
<td>00080</td>
<td>O2 DFHP1 PIC X VALUE IS ' '</td>
<td>20000000</td>
</tr>
<tr>
<td>00081</td>
<td>O2 DFHP2 PIC X VALUE IS ' '</td>
<td>22000000</td>
</tr>
<tr>
<td>00082</td>
<td>O2 DFHP3 PIC X VALUE IS ' '</td>
<td>24000000</td>
</tr>
<tr>
<td>00083</td>
<td>O2 DFHP4 PIC X VALUE IS ' '</td>
<td>26000000</td>
</tr>
<tr>
<td>00084</td>
<td>O2 DFHP5 PIC X VALUE IS ' '</td>
<td>28000000</td>
</tr>
<tr>
<td>00085</td>
<td>O2 DFHP6 PIC X VALUE IS ' '</td>
<td>30000000</td>
</tr>
<tr>
<td>00086</td>
<td>O2 DFHP7 PIC X VALUE IS ' '</td>
<td>32000000</td>
</tr>
<tr>
<td>00087</td>
<td>O2 DFHP8 PIC X VALUE IS ' '</td>
<td>34000000</td>
</tr>
<tr>
<td>00088</td>
<td>O2 DFHP9 PIC X VALUE IS ' '</td>
<td>36000000</td>
</tr>
<tr>
<td>00089</td>
<td>O2 DFHPA PIC X VALUE IS ' '</td>
<td>38000000</td>
</tr>
<tr>
<td>00090</td>
<td>O2 DFHPB PIC X VALUE IS ' '</td>
<td>40000000</td>
</tr>
<tr>
<td>00091</td>
<td>O2 DFHPC PIC X VALUE IS ' '</td>
<td>42000000</td>
</tr>
<tr>
<td>00092</td>
<td>O2 DFHPD PIC X VALUE IS ' '</td>
<td>44000000</td>
</tr>
<tr>
<td>00093</td>
<td>O2 DFHEPICX VALUE IS ' '</td>
<td>46000000</td>
</tr>
<tr>
<td>00094</td>
<td>O2 DFHED PIC X VALUE IS ' '</td>
<td>48000000</td>
</tr>
<tr>
<td>00095</td>
<td>O2 DFHEF PIC X VALUE IS ' '</td>
<td>50000000</td>
</tr>
<tr>
<td>00096</td>
<td>O2 DFHEG PIC X VALUE IS ' '</td>
<td>52000000</td>
</tr>
<tr>
<td>00097</td>
<td>O2 DFHEH PIC X VALUE IS ' '</td>
<td>54000000</td>
</tr>
<tr>
<td>00098</td>
<td>O2 DFHEI PIC X VALUE IS ' '</td>
<td>56000000</td>
</tr>
<tr>
<td>00099</td>
<td>O2 DFHEJ PIC X VALUE IS ' '</td>
<td>58000000</td>
</tr>
<tr>
<td>00100</td>
<td>O2 DFHEK PIC X VALUE IS ' '</td>
<td>60000000</td>
</tr>
<tr>
<td>00101</td>
<td>O2 DFHEK PIC X VALUE IS ' '</td>
<td>62000000</td>
</tr>
<tr>
<td>00102</td>
<td>O2 DFHEO PIC X VALUE IS ' '</td>
<td>64000000</td>
</tr>
<tr>
<td>00103</td>
<td>O2 DFHEP PIC X VALUE IS ' '</td>
<td>66000000</td>
</tr>
<tr>
<td>00104</td>
<td>O2 DFHEQ PIC X VALUE IS ' '</td>
<td>68000000</td>
</tr>
<tr>
<td>00105</td>
<td>O2 DFHER PIC X VALUE IS ' '</td>
<td>70000000</td>
</tr>
<tr>
<td>00106</td>
<td>O2 DFHES PIC X VALUE IS ' '</td>
<td>72000000</td>
</tr>
<tr>
<td>00107</td>
<td>O2 DFHEV PIC X VALUE IS ' '</td>
<td>74000000</td>
</tr>
<tr>
<td>00108</td>
<td>O1 DFHEVAR COPY DFHEVAR.</td>
<td>76000000</td>
</tr>
<tr>
<td>00109</td>
<td>O1 DFHEVAR.</td>
<td>78000000</td>
</tr>
<tr>
<td>00110</td>
<td>O2 DFHEVAR PIC X(26).</td>
<td>80000000</td>
</tr>
<tr>
<td>00111</td>
<td>O2 DFHEVAR PIC X(8).</td>
<td>82000000</td>
</tr>
<tr>
<td>Offset</td>
<td>Contents</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>00112 C</td>
<td>02 DFHEI V2 PICTURE X(8).</td>
<td>16000000</td>
</tr>
<tr>
<td>00113 C</td>
<td>02 DFHEI V3 PICTURE X(8).</td>
<td>20000000</td>
</tr>
<tr>
<td>00114 C</td>
<td>02 DFHEI V4 PICTURE X(6).</td>
<td>24000000</td>
</tr>
<tr>
<td>00115 C</td>
<td>02 DFHEI V5 PICTURE X(4).</td>
<td>28000000</td>
</tr>
<tr>
<td>00116 C</td>
<td>02 DFHEI V6 PICTURE X(4).</td>
<td>32000000</td>
</tr>
<tr>
<td>00117 C</td>
<td>02 DFHEI V7 PICTURE X(2).</td>
<td>36000000</td>
</tr>
<tr>
<td>00118 C</td>
<td>02 DFHEI V8 PICTURE X(2).</td>
<td>40000000</td>
</tr>
<tr>
<td>00119 C</td>
<td>02 DFHEI V9 PICTURE X(1).</td>
<td>44000000</td>
</tr>
<tr>
<td>00120 C</td>
<td>02 DFHEI V10 PICTURE S9(7) USAGE COMPUTATIONAL-3.</td>
<td>48000000</td>
</tr>
<tr>
<td>00121 C</td>
<td>02 DFHEI V11 PICTURE S9(4) USAGE COMPUTATIONAL.</td>
<td>52000000</td>
</tr>
<tr>
<td>00122 C</td>
<td>02 DFHEI V12 PICTURE S9(4) USAGE COMPUTATIONAL.</td>
<td>56000000</td>
</tr>
<tr>
<td>00123 C</td>
<td>02 DFHEI V13 PICTURE S9(4) USAGE COMPUTATIONAL.</td>
<td>60000000</td>
</tr>
<tr>
<td>00124 C</td>
<td>02 DFHEI V14 PICTURE S9(4) USAGE COMPUTATIONAL.</td>
<td>64000000</td>
</tr>
<tr>
<td>00125 C</td>
<td>02 DFHEI V15 PICTURE S9(4) USAGE COMPUTATIONAL.</td>
<td>68000000</td>
</tr>
<tr>
<td>00126 C</td>
<td>02 DFHEI V16 PICTURE S9(9) USAGE COMPUTATIONAL.</td>
<td>72000000</td>
</tr>
<tr>
<td>00127 C</td>
<td>02 DFHEI V17 PICTURE X(4).</td>
<td>76000000</td>
</tr>
<tr>
<td>00128 C</td>
<td>02 DFHEI V18 PICTURE X(4).</td>
<td>80000000</td>
</tr>
<tr>
<td>00129 C</td>
<td>02 DFHEI V19 PICTURE X(4).</td>
<td>84000000</td>
</tr>
<tr>
<td>00130 C</td>
<td>02 DFHEI V97 PICTURE S9(7) USAGE COMPUTATIONAL-3 VALUE ZERO.88000000</td>
<td></td>
</tr>
<tr>
<td>00131 C</td>
<td>02 DFHEI V98 PICTURE S9(4) USAGE COMPUTATIONAL VALUE ZERO. 92000000</td>
<td></td>
</tr>
<tr>
<td>00132 C</td>
<td>02 DFHEI V99 PICTURE X(1) VALUE SPACE.</td>
<td>96000000</td>
</tr>
</tbody>
</table>

**LINKAGE SECTION.**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Contents</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00133 C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00134 C</td>
<td>01 DFHEI BLK COPY DFHEI BLK.</td>
<td></td>
</tr>
<tr>
<td>00136 C</td>
<td>* EIBLK EXEC INTERFACE BLOCK</td>
<td>02000000</td>
</tr>
<tr>
<td>00137 C</td>
<td>01 DFHEI BLK</td>
<td></td>
</tr>
<tr>
<td>00138 C</td>
<td>* EIBTIME TIME IN OH-MMSS FORMAT</td>
<td>06000000</td>
</tr>
<tr>
<td>00139 C</td>
<td>02 EIBTIME PICTURE S9(7) USAGE COMPUTATIONAL-3.</td>
<td>08000000</td>
</tr>
<tr>
<td>00140 C</td>
<td>* EIBDATE DATE IN O0YDDD FORMAT</td>
<td>10000000</td>
</tr>
<tr>
<td>00141 C</td>
<td>02 EIBDATE PICTURE S9(7) USAGE COMPUTATIONAL-3.</td>
<td>12000000</td>
</tr>
<tr>
<td>00142 C</td>
<td>* EIBTRNID TRANSACTION IDENTIFIER</td>
<td>14000000</td>
</tr>
<tr>
<td>00143 C</td>
<td>02 EIBTRNID PICTURE X(4).</td>
<td>16000000</td>
</tr>
<tr>
<td>00144 C</td>
<td>* EIBTASKN TASK NUMBER.</td>
<td>18000000</td>
</tr>
<tr>
<td>00145 C</td>
<td>02 EIBTASKN PICTURE S9(7) USAGE COMPUTATIONAL-3.</td>
<td>20000000</td>
</tr>
<tr>
<td>00146 C</td>
<td>* EIBTRMID TERMINAL IDENTIFIER</td>
<td>22000000</td>
</tr>
<tr>
<td>00147 C</td>
<td>02 EIBTRMID PICTURE X(4).</td>
<td>24000000</td>
</tr>
<tr>
<td>00148 C</td>
<td>* DFHEIGDI RESERVE (2)</td>
<td>26000000</td>
</tr>
<tr>
<td>00149 C</td>
<td>02 DFHEIGDI PICTURE S9(4) USAGE COMPUTATIONAL.</td>
<td>28000000</td>
</tr>
<tr>
<td>00150 C</td>
<td>* EIBCPDSON CURSOR POSITION</td>
<td>30000000</td>
</tr>
<tr>
<td>00151 C</td>
<td>02 EIBCPDSON PICTURE S9(4) USAGE COMPUTATIONAL.</td>
<td>32000000</td>
</tr>
<tr>
<td>00152 C</td>
<td>* EIBCALEN COMMAREA LENGTH</td>
<td>34000000</td>
</tr>
<tr>
<td>00153 C</td>
<td>02 EIBCALEN PICTURE S9(4) USAGE COMPUTATIONAL.</td>
<td>36000000</td>
</tr>
<tr>
<td>00154 C</td>
<td>* EIBAI D ATTENTION IDENTIFIER</td>
<td>38000000</td>
</tr>
<tr>
<td>00155 C</td>
<td>02 EIBAID PICTURE X(1).</td>
<td>40000000</td>
</tr>
<tr>
<td>00156 C</td>
<td>* EIBFN FUNCTION CODE</td>
<td>42000000</td>
</tr>
<tr>
<td>00157 C</td>
<td>02 EIBFN PICTURE X(2).</td>
<td>44000000</td>
</tr>
<tr>
<td>00158 C</td>
<td>* EIBRCODE RESPONSE CODE</td>
<td>46000000</td>
</tr>
<tr>
<td>00159 C</td>
<td>02 EIBRCODE PICTURE X(6).</td>
<td>48000000</td>
</tr>
<tr>
<td>00160 C</td>
<td>* EIBDS DATASET NAME</td>
<td>50000000</td>
</tr>
<tr>
<td>00161 C</td>
<td>02 EIBDS PICTURE X(8).</td>
<td>52000000</td>
</tr>
<tr>
<td>00162 C</td>
<td>* EIBREQD REQUEST IDENTIFIER</td>
<td>54000000</td>
</tr>
<tr>
<td>00163 C</td>
<td>02 EIBREQD PICTURE X(8).</td>
<td>56000000</td>
</tr>
<tr>
<td>00164 C</td>
<td>* EIBRSRC RESOURCE NAME</td>
<td>58000000</td>
</tr>
<tr>
<td>00165 C</td>
<td>02 EIBRSRC PICTURE X(8).</td>
<td>60000000</td>
</tr>
<tr>
<td>00166 C</td>
<td>* EIBSYNC SYNCPOINT REQUIRED</td>
<td>62000000</td>
</tr>
<tr>
<td>00167 C</td>
<td>02 EIBSYNC PICTURE X.</td>
<td>64000000</td>
</tr>
<tr>
<td>00168 C</td>
<td>* EIBFREE TERMINAL FREE REQUIRED</td>
<td>66000000</td>
</tr>
</tbody>
</table>

**BCCPGM 15.42.41 MAY 11, 1987**
00169 C 02 EIBFREE  PICTURE X. 68000000
00170 C  * EIBRECV DATA RECEIVE REQUIRED 70000000
00171 C  02 EIBRECV PICTURE X. 73000000
00172 C  * EIBSEND RESERVED 76000000
00173 C  02 EIBSEND PICTURE X. 79000000
00174 C  * EIBATT ATTACH DATA EXISTS 82000000
00175 C  02 EIBATT PICTURE X. 85000000
00176 C  * EIBECC GOTTEN DATA IS COMPLETE 88000000
00177 C  02 EIBECC PICTURE X. 91000000
00178 C  * EIBFHM GOTTEN DATA CONTAINS FMH 94000000
00179 C  02 EIBFHM PICTURE X. 97000000
00180 C 01 DFHCOMMAREA PIC X.  
00181 01 DFHBLSSLT1 PICTURE X(1). 
00182 01 DFHBLSSLT2 PICTURE X(1). 
00183 01 DFHEIV1 PICTURE X(1). 
00184 PROCEDURE DIVISION USING DFHEIBLK DFHCOMMAREA. 
00185 SERVICE RELOAD DFHEIBLK. 
00186 SERVICE RELOAD DFHCOMMAREA. 
00187 IF EIBCALCN IS EQUAL TO ZEROS 
00188 ELSE 
00189 PERFORM 001-INITIAL-RTN 
00190 00191 ELSE 
00192 PERFORM 002-RECEIVE-RTN. 
00193 00194 00195 GOBACK. 
00196 00197 00198 001-INITIAL-RTN. 
00199 00200  * EXEC CICS SEND 
00201  * MAP('SCREEN1') 
00202  * MAPSET('BCCBSET') 
00203  * MAPONLY 
00204  * ERASE 
00205  * END-EXEC. 
00206  MOVE 'SCREEN1' TO DFHEIV1 MOVE 'BCCBSET' TO DFHEIV2 MOVE ' 
00207  ') ' TO DFHIECO CALL 'DFHEIV1' USING DFHEIV0 
00208  DFHEIV1 DFHEIV99 DFHEIV98 DFHEIV2. 
00209 
00210 00211 
00212  * EXEC CICS RETURN 
00213  * TRANSID('BCC6') 
00214  * COMMAREA(DFHCOMMAREA) 
00215  * LENGTH(1) 
00216  * END-EXEC. 
00217  MOVE 'BCC6' TO DFHEIV5 MOVE 1 TO DFHEIV11 MOVE ' 
00218  TO DFHIECO CALL 'DFHEIV1' USING DFHEIV5 DFHCOMMAREA 
00219  DFHEIV11. 
00220 00221 00222 00223 00224 002-RECEIVE-RTN. 
00225
00226  * EXEC CICS RECEIVE
00227  + MAP('SCREEN1')
00228  + MAPSET('BCC8SET')
00229  + END-EXEC.
00230  MOVE 'SCREEN1' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE
00231  ' ' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO
00232  DFHEIV1 SCREEN11 DFHEIV98 DFHEIV2.
00233
00234  MOVE NUM11 TO WS-NUM1.
00235  MOVE NUM2I TO WS-NUM2.
00236
00237  ADD WS-NUM1 WS-NUM2 TO WS-SUM.
00238
00239  MOVE WS-SUM TO TOTALO.
00240
00241  * EXEC CICS SEND
00242  + MAP('SCREEN3')
00243  + MAPSET('BCC8SET')
00244  + ERASE
00245  + END-EXEC.
00246  MOVE 'SCREEN3' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE
00247  ' ' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO
00248  DFHEIV1 SCREEN30 DFHEIV98 DFHEIV2.
00249
00250  1
00251
00252
00253  * EXEC CICS RETURN
00254  + END-EXEC.
00255  MOVE ' ' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO.
00256
00257
CICS EXAMPLE PROGRAM - BCC7PGM
EXEC CICS RECEIVE
   MAP('SCREEN2')
   MAPSET('BCC8SET')
END-EXEC.

IF ENTRNUM IS NOT EQUAL TO 'DONE'

MOVE ENTRNUM TO WS-NUM
ADD WS-NUM TO WS-SUM

EXEC CICS SEND
   MAP('SCREEN2')
   MAPSET('BCC8SET')
END-EXEC

EXEC CICS RETURN

EXEC CICS RETURN
<table>
<thead>
<tr>
<th>Line</th>
<th>Source Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>IDENTIFICATION DIVISION.</td>
</tr>
<tr>
<td>00002</td>
<td>PROGRAM-ID. BCC7PGM.</td>
</tr>
<tr>
<td>00003</td>
<td>ENVIRONMENT DIVISION.</td>
</tr>
<tr>
<td>00004</td>
<td>DATA DIVISION.</td>
</tr>
<tr>
<td>00005</td>
<td>WORKING-STORAGE SECTION.</td>
</tr>
<tr>
<td>00006</td>
<td>01 WS-NUM PIC 9(4).</td>
</tr>
<tr>
<td>00007</td>
<td>01 ORIGINAL-COMMAREA.</td>
</tr>
<tr>
<td>00008</td>
<td>05 ORG-SUM PIC 9(5) VALUE 0.</td>
</tr>
<tr>
<td>00009</td>
<td>COPY BCCBSET.</td>
</tr>
<tr>
<td>00010</td>
<td>01 SCREENII.</td>
</tr>
<tr>
<td>00011</td>
<td>02 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00012</td>
<td>02 NUM1L COMP PIC S9(4).</td>
</tr>
<tr>
<td>00013</td>
<td>02 NUM1F PICTURE X.</td>
</tr>
<tr>
<td>00014</td>
<td>02 FILLER REDEFINES NUM1F.</td>
</tr>
<tr>
<td>00015</td>
<td>03 NUM1L PICTURE X.</td>
</tr>
<tr>
<td>00016</td>
<td>02 NUM11 PIC X(4).</td>
</tr>
<tr>
<td>00017</td>
<td>02 NUM2 COMP PIC S9(4).</td>
</tr>
<tr>
<td>00018</td>
<td>02 NUM2F PICTURE X.</td>
</tr>
<tr>
<td>00019</td>
<td>02 FILLER REDEFINES NUM2F.</td>
</tr>
<tr>
<td>00020</td>
<td>03 NUM2L PICTURE X.</td>
</tr>
<tr>
<td>00021</td>
<td>02 NUM21 PIC X(4).</td>
</tr>
<tr>
<td>00022</td>
<td>02 ERRMSG1L COMP PIC S9(4).</td>
</tr>
<tr>
<td>00023</td>
<td>02 ERRMSG1F PICTURE X.</td>
</tr>
<tr>
<td>00024</td>
<td>02 FILLER REDEFINES ERRMSG1F.</td>
</tr>
<tr>
<td>00025</td>
<td>03 ERRMSG1A PICTURE X.</td>
</tr>
<tr>
<td>00026</td>
<td>02 ERRMSG2L PIC X(50).</td>
</tr>
<tr>
<td>00027</td>
<td>02 ERRMSG2F PICTURE X.</td>
</tr>
<tr>
<td>00028</td>
<td>02 FILLER REDEFINES ERRMSG2F.</td>
</tr>
<tr>
<td>00029</td>
<td>03 ERRMSG2A PICTURE X.</td>
</tr>
<tr>
<td>00030</td>
<td>02 SCREEN10 REDEFINES SCREENII.</td>
</tr>
<tr>
<td>00031</td>
<td>02 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00032</td>
<td>02 FILLER PICTURE X(3).</td>
</tr>
<tr>
<td>00033</td>
<td>02 NUM10 PIC X(4).</td>
</tr>
<tr>
<td>00034</td>
<td>02 FILLER PICTURE X(3).</td>
</tr>
<tr>
<td>00035</td>
<td>02 NUM20 PIC X(4).</td>
</tr>
<tr>
<td>00036</td>
<td>02 FILLER PICTURE X(3).</td>
</tr>
<tr>
<td>00037</td>
<td>02 ERRMSG10 PIC X(50).</td>
</tr>
<tr>
<td>00038</td>
<td>02 SCREENI.</td>
</tr>
<tr>
<td>00039</td>
<td>02 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00040</td>
<td>02 ENTNUM1L COMP PIC S9(4).</td>
</tr>
<tr>
<td>00041</td>
<td>02 ENTNUM1F PICTURE X.</td>
</tr>
<tr>
<td>00042</td>
<td>02 FILLER REDEFINES ENTNUM1F.</td>
</tr>
<tr>
<td>00043</td>
<td>03 ENTNUM1A PICTURE X.</td>
</tr>
<tr>
<td>00044</td>
<td>02 ENTNUM2L PIC X(4).</td>
</tr>
<tr>
<td>00045</td>
<td>02 ERRMSG2L COMP PIC S9(4).</td>
</tr>
<tr>
<td>00046</td>
<td>02 ERRMSG2F PICTURE X.</td>
</tr>
<tr>
<td>00047</td>
<td>02 FILLER REDEFINES ERRMSG2F.</td>
</tr>
<tr>
<td>00048</td>
<td>03 ERRMSG2A PICTURE X.</td>
</tr>
<tr>
<td>00049</td>
<td>02 ERRMSG21 PIC X(50).</td>
</tr>
<tr>
<td>00050</td>
<td>02 SCREEN20 REDEFINES SCREEN21.</td>
</tr>
<tr>
<td>00051</td>
<td>02 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00052</td>
<td>02 FILLER PICTURE X(3).</td>
</tr>
</tbody>
</table>
00055 C 02 ENNUMO PIC X(4).
00056 C 02 FILLER PICTURE X(3).
00057 C 02 ERRMSG2D PIC X(50).
00058 C 01 SCREEN31.
00059 C 02 FILLER PIC X(12).
00060 C 02 TOTALL COMP PIC 59(4).
00061 C 02 TOTALL PICTURE X.
00062 C 02 FILLER REDEFINES TOTALL.
00063 C 03 TOTALA PICTURE X.
00064 C 02 TOTAL PIC X(5).
00065 C 01 SCREEN30 REDEFINES SCREEN31.
00066 C 02 FILLER PIC X(12).
00067 C 02 FILLER PICTURE X(3).
00068 C 02 TOTALO PIC X(5).
00069 C
00070 C
00071 C 01 DFHAID.
00072 C 02 DFHNULL PIC X VALUE IS '').
00073 C 02 DFHENTER PIC X VALUE IS 'QUOTE.
00074 C 02 DFHCLEAR PIC X VALUE IS '}'.
00075 C 02 DFHPEN PIC X VALUE IS ' &'.
00076 C 02 DFHSEND PIC X VALUE IS 'X'.
00077 C 02 DFHMSRE PIC X VALUE IS 'X'.
00078 C 02 DFHSTRF PIC X VALUE IS 'h'.
00079 C 02 DFHTRIG PIC X VALUE IS 'h'.
00080 C 02 DFHFA1 PIC X VALUE IS '%'.
00081 C 02 DFHFA2 PIC X VALUE IS '>'.
00082 C 02 DFHFA3 PIC X VALUE IS '.
00083 C 02 DFHFA1 PIC X VALUE IS '1'.
00084 C 02 DFHFA2 PIC X VALUE IS '2'.
00085 C 02 DFHFA3 PIC X VALUE IS '3'.
00086 C 02 DFHFA4 PIC X VALUE IS '4'.
00087 C 02 DFHFA5 PIC X VALUE IS '5'.
00088 C 02 DFHFA6 PIC X VALUE IS '6'.
00089 C 02 DFHFA7 PIC X VALUE IS '7'.
00090 C 02 DFHFA8 PIC X VALUE IS '8'.
00091 C 02 DFHFA9 PIC X VALUE IS '9'.
00092 C 02 DFHFA10 PIC X VALUE IS '.',
00093 C 02 DFHFA11 PIC X VALUE IS '#'.
00094 C 02 DFHFA12 PIC X VALUE IS '@'.
00095 C 02 DFHFA13 PIC X VALUE IS '$'.
00096 C 02 DFHFA14 PIC X VALUE IS '$'.
00097 C 02 DFHFA15 PIC X VALUE IS 'C'.
00098 C 02 DFHFA16 PIC X VALUE IS 'Y'.
00099 C 02 DFHFA17 PIC X VALUE IS 'E'.
00100 C 02 DFHFA18 PIC X VALUE IS 'f'.
00101 C 02 DFHFA19 PIC X VALUE IS 'G'.
00102 C 02 DFHFA20 PIC X VALUE IS 'H'.
00103 C 02 DFHFA21 PIC X VALUE IS 't'.
00104 C 02 DFHFA22 PIC X VALUE IS '4'.
00105 C 02 DFHFA23 PIC X VALUE IS '.
00106 C 02 DFHFA24 PIC X VALUE IS '<'.
00107 C
00108 C 01 DFHEIVAR COPY DFHEIVAR.
00109 C 01 DFHEIVAR.
00110 C 02 DFHEIVO PICTURE X(26).
00111 C 02 DFHEIV1 PICTURE X(8).
00112 C  02 DFHEIV2 PICTURE X(8).               16000000
00113 C  02 DFHEIV3 PICTURE X(8).               20000000
00114 C  02 DFHEIV4 PICTURE X(6).               24000000
00115 C  02 DFHEIV5 PICTURE X(4).               28000000
00116 C  02 DFHEIV6 PICTURE X(4).               32000000
00117 C  02 DFHEIV7 PICTURE X(2).               36000000
00118 C  02 DFHEIV8 PICTURE X(2).               40000000
00119 C  02 DFHEIV9 PICTURE X(1).               44000000
00120 C  02 DFHEIV10 PICTURE S9(7) USAGE COMPUTATIONAL-3.  48000000
00121 C  02 DFHEIV11 PICTURE S9(4) USAGE COMPUTATIONAL.  52000000
00122 C  02 DFHEIV12 PICTURE S9(4) USAGE COMPUTATIONAL.  56000000
00123 C  02 DFHEIV13 PICTURE S9(4) USAGE COMPUTATIONAL.  60000000
00124 C  02 DFHEIV14 PICTURE S9(4) USAGE COMPUTATIONAL.  64000000
00125 C  02 DFHEIV15 PICTURE S9(4) USAGE COMPUTATIONAL.  68000000
00126 C  02 DFHEIV16 PICTURE S9(9) USAGE COMPUTATIONAL.  72000000
00127 C  02 DFHEIV17 PICTURE X(4).               76000000
00128 C  02 DFHEIV18 PICTURE X(2).               80000000
00129 C  02 DFHEIV19 PICTURE X(4).               84000000
00130 C  02 DFHEIV97 PICTURE S9(7) USAGE COMPUTATIONAL-3 VALUE ZERO.  88000000
00131 C  02 DFHEIV98 PICTURE S9(4) USAGE COMPUTATIONAL VALUE ZERO.  92000000
00132 C  02 DFHEIV99 PICTURE X(1) VALUE SPACE.      96000000

00133 C
00134
00135       01 DFHEIBLK COPY DFHEIBLK.
00136 C  *   EIBLK EXEC INTERFACE BLOCK
00137 C       01 DFHEIBLK
00138 C       *   EIBTIME TIME IN OH&MSS FORMAT
00139 C       02 EIBTIME PICTURE S9(7) USAGE COMPUTATIONAL-3.  06000000
00140 C       *   EIBDATE DATE IN OCYMYDDD FORMAT
00141 C       02 EIBDATE PICTURE S9(7) USAGE COMPUTATIONAL-3.  12000000
00142 C       *   EIBTRNID TRANSACTION IDENTIFIER
00143 C       02 EIBTRNID PICTURE X(4).               14000000
00144 C       *   EIBTASKN TASK NUMBER
00145 C       02 EIBTASKN PICTURE S9(7) USAGE COMPUTATIONAL-3.  18000000
00146 C       *   EIBTRNID TERMINAL IDENTIFIER
00147 C       02 EIBTRNID PICTURE X(4).               22000000
00148 C       *   DFHEIDG RESERVE
00149 C       02 DFHEIDG PICTURE S9(4) USAGE COMPUTATIONAL.  26000000
00150 C       *   EIBCPDSN CURSOR POSITION
00151 C       02 EIBCPDSN PICTURE S9(4) USAGE COMPUTATIONAL.  30000000
00152 C       *   EIBCALEN COMAREA LENGTH
00153 C       02 EIBCALEN PICTURE S9(4) USAGE COMPUTATIONAL.  34000000
00154 C       *   EIBAID ATTENTION IDENTIFIER
00155 C       02 EIBAID PICTURE X(1).               38000000
00156 C       *   EIBFN FUNCTION CODE
00157 C       02 EIBFN PICTURE X(2).                42000000
00158 C       *   EIBRCODE RESPONSE CODE
00159 C       02 EIBRCODE PICTURE X(6).            46000000
00160 C       *   EIBDS DATASET NAME
00161 C       02 EIBDS PICTURE X(8).                50000000
00162 C       *   EIBREQD REQUEST IDENTIFIER
00163 C       02 EIBREQD PICTURE X(8).              54000000
00164 C       *   EIBRSRC RESouce NAME
00165 C       02 EIBRSRC PICTURE X(8).              58000000
00166 C       *   EIBSYNC SYNCPOINT REQUIRED
00167 C       02 EIBSYNC PICTURE X.                62000000
00168 C       *   EIBFRE FREE TERMINAL FREE REQUIRED

00168 C       02 EIBFREE PICTURE X.           68000000
00170 C       * EIBRECQ DATA RECEIVE REQUIRED    70000000
00171 C       * 02 EIBRECQ PICTURE X.            73000000
00172 C       * EIBSEND RESERVED                  76000000
00173 C       * 02 EIBSEND PICTURE X.             79000000
00174 C       * EIBATT ATTACH DATA EXISTS         82000000
00175 C       * 02 EIBATT PICTURE X.              85000000
00176 C       * EIBEOC GOTTEN DATA IS COMPLETE     88000000
00177 C       * 02 EIBEOC PICTURE X.              91000000
00178 C       * EIBFMH GOTTEN DATA CONTAINS FMH     94000000
00179 C       * 02 EIBFMH PICTURE X.              97000000
00180 01 DFHCMMAREA PIC X(5).
00181 01 PASSED-CMAREA REDEFINES DFHCMMAREA.
00182 05 WS-SUM PIC 9(5).
00183
00184 01 DFHBLSSLT1 PICTURE X(1).
00185 PROCEDURE DIVISION USING DFHEIBLK DFHCMMAREA.
00186 SERVICE RELOAD DFHEIBLK.
00187 SERVICE RELOAD DFHCMMAREA.
00188 IF EIBCALN IS EQUAL TO ZEROS
00189
00190 PERFORM 001-INITIAL-RTN
00191
00192 ELSE
00193
00194 PERFORM 002-RECEIVE-RTN.
00195
00196 GBACK.
00197
00198 001-INITIAL-RTN.
00199
00200 EXEC CICS SEND
00201 *
00202 MAP('SCREEN2')
00203 MAPSET('BCCSET')
00204 MAPONLY
00205 ERASE
00206 END-EXEC.
00207 MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCCSET' TO DFHEIV2 MOVE ')
00208 TO DFHEIVO CALL 'DFHEIV' USING DFHEIVO
00209 DFHEIV1 DFHEIV99 DFHEIV98 DFHEIV2.
00210
00211
00212
00213
00214 EXEC CICS RETURN
00215 TRANSID('BCC7')
00216 COMMAREA(ORIGINAL-CMAREA)
00217 LENGTH(S)
00218 END-EXEC.
00219 MOVE 'BCC7' TO DFHEIV5 MOVE 5 TO DFHEIV11 MOVE ' \n00220 TO DFHEIVO CALL 'DFHEIV' USING DFHEIVO DFHEIV5
00221 ORIGINAL-CMAREA DFHEIV11.
00222
00223
00224
00225 002-RECEIVE-RTN.
00226 * EXEC CICS RECEIVE
00228 * MAP('SCREEN2')
00230 * MAPSET('BCC8SET')
00230 * END-EXEC.
00231 MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE '
00232 - ' TO DFHEIVO CALL 'DFHEIV' USING DFHEIVO
00233 DFHEIV1 SCREEN2I DFHEIV98 DFHEIV2.
00234
00235 IF ENNUMI IS NOT EQUAL TO 'DONE'
00236 MOVE ENNUMI TO WS-NUM
00239 ADD WS-NUM TO WS-SUM
00241 * EXEC CICS SEND
00242 * MAP('SCREEN2')
00244 * MAPSET('BCC8SET')
00246 + ERASE
00247 * END-EXEC
00248 MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE '
00249 - ' TO DFHEIVO CALL 'DFHEIV' USING DFHEIVO
00250 DFHEIV1 DFHEIV99 DFHEIV98 DFHEIV2
00251
00252 + EXEC CICS RETURN
00256 + TRANSID('BCC7')
00257 + COMMAREA(PASS-COMMAREA)
00258 + LENGTH(5)
00259 + END-EXEC
00260 MOVE 'BCC7' TO DFHEIV5 MOVE 5 TO DFHEIV11 MOVE '
00261 - ' TO DFHEIVO CALL 'DFHEIV' USING DFHEIVO DFHEIV5
00262 PASSED-COMMAREA DFHEIV11
00263 ELSE
00266 MOVE WS-SUM TO TOTAI0
00269 * EXEC CICS SEND
00271 + MAP('SCREEN3')
00272 + MAPSET('BCC8SET')
00273 + ERASE
00274 + END-EXEC
00275 MOVE 'SCREEN3' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE '
00276 - ' TO DFHEIVO CALL 'DFHEIV' USING DFHEIVO
00277 DFHEIV1 SCREEN30 DFHEIV98 DFHEIV2
00278
00280 * EXEC CICS RETURN
00282 + END-EXEC.
MOVE ' ' TO DFHEIVO
CALL 'DFHEI1' USING DFHEIVO.
IDENTIFICATION DIVISION.
PROGRAM-ID. BCCBPGM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-NUM PIC 9(4).
01 ORIGINAL-COMMArea.
05 ORIG-SUM PIC 9(5) VALUE 0.
01 MESSAGE-AREA PIC X(50).
01 ATTRIBUTES.
05 ATT-NUMERIC-BRT-FSET PIC X VALUE 'R'.
COPY BCCBSET.
COPY DFHAID.
LINKAGE SECTION.
01 DFHCOMMAREA PIC X(5).
01 PASSED-COMMArea REDEFINES DFHCOMMAREA.
05 WS-SUM PIC 9(5).
PROCEDURE DIVISION.
EXEC CICS HANDLE CONDITION
ERROR (COB-ERROR-RTN)
END-EXEC.
IF EIBCALLEN IS EQUAL TO ZEROS
PERFORM OO1-INITIAL-RTN
ELSE
PERFORM OO2-RECEIVE-RTN.
GOBACK.
OO1-INITIAL-RTN.
EXEC CICS SEND
MAP('SCREEN2')
MAPS('BCCBSET')
MAPONLY
ERASE
CICS/VS COMMAND LANGUAGE TRANSLATOR VERSION 1.5

LINE SOURCE LISTING

00052       END-EXEC.
00053       EXEC CICS RETURN
00054       TRANSID('BCC8')
00056       COMMAREA(ORIGINAL-COMMAREA)
00057       LENGTH(5)
00058       END-EXEC.
00060       002-RECEIVE-RTN.
00061       EXEC CICS HANDLE CONDITION
00063       MAPFAIL (003-NO-DATA-RTN)
00064       END-EXEC.
00065       EXEC CICS HANDLE AID
00066       CLEAR (004-CLEAR-KEY-RTN)
00067       END-EXEC.
00069       EXEC CICS RECEIVE
00071       MAP('SCREEN2')
00072       MAPSET('BCC8SET')
00073       END-EXEC.
00075       IF ENTNUMI IS NOT NUMERIC
00076       MOVE ATTNUMERIC-BRT-FSET TO ENTNUMA ERRMSG2A
00077       MOVE 'NUMBER NOT NUMERIC - CORRECT AND REENTER'
00078       TO ERRMSG20
00080       EXEC CICS SEND
00082       MAP('SCREEN2')
00084       MAPSET('BCC8SET')
00085       DATANLY
00086       END-EXEC
00087       EXEC CICS RETURN
00090       COMMAREA(PASSED-COMMAREA)
00091       LENGTH(5)
00092       END-EXEC
00093       ELSE
00095       MOVE ENTNUMI TO WS-NUM
00097       ADD WS-NUM TO WS-SUM
00098       EXEC CICS SEND
00101       MAP('SCREEN2')
00102       MAPSET('BCC8SET')
ERASE
MAPONLY
EXEC CICS RETURN
EXEC CICS SEND
MAP('SCREEN3')
EXEC CICS RETURN
EXEC CICS SEND
MAP('MESSAGE-AREA')
<table>
<thead>
<tr>
<th>LINE</th>
<th>SOURCE LISTING</th>
</tr>
</thead>
<tbody>
<tr>
<td>00154</td>
<td>END-EXEC.</td>
</tr>
<tr>
<td>00155</td>
<td></td>
</tr>
<tr>
<td>00156</td>
<td>EXEC CICS RETURN</td>
</tr>
<tr>
<td>00157</td>
<td>END-EXEC.</td>
</tr>
<tr>
<td>00158</td>
<td></td>
</tr>
</tbody>
</table>
IDENTIFICATION DIVISION.

PROGRAM-ID. BCCBPGM.

ENVIRONMENT DIVISION.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 WS-NUM PIC 9(4).

01 ORIGINAL-COMMAREA.

05 ORIG-SUM PIC 9(5) VALUE 0.

01 MESSAGE-AREA PIC X(50).

01 ATTRIBUTES.

05 ATT-NUMERIC-BRT-FSET PIC X VALUE 'R'.

COPY BCCBSET.

01 SCREEN11.

02 FILLER PIC X(12).

02 NUM1L COMP PIC S9(4).

02 NUM1F PICTURE X.

02 FILLER REDEFINES NUM1F.

03 NUM1A PICTURE X.

02 NUM1 PIC X(4).

02 NUM2L COMP PIC S9(4).

02 NUM2F PICTURE X.

02 FILLER REDEFINES NUM2F.

03 NUM2A PICTURE X.

02 NUM2 PIC X(4).

02 ERRMSG1L COMP PIC S9(4).

02 ERRMSG1F PICTURE X.

02 FILLER REDEFINES ERRMSG1F.

03 ERRMSG1A PICTURE X.

02 ERRMSG11 PIC X(50).

01 SCREEN10 REDEFINES SCREEN11.

02 FILLER PIC X(12).

02 FILLER PICTURE X(3).

02 NUM1D PIC X(4).

02 FILLER PICTURE X(3).

02 NUM2D PIC X(4).

02 FILLER PICTURE X(3).

02 ERRMSG10 PIC X(50).

01 SCREEN21.

02 FILLER PIC X(12).

02 ENTNUML COMP PIC S9(4).

02 ENTNUMF PICTURE X.

02 FILLER REDEFINES ENTNUMF.

03 ENTNUMA PICTURE X.

02 ENTNUMI PIC X(4).

02 ERRMSG2L COMP PIC S9(4).

02 ERRMSG2F PICTURE X.

02 FILLER REDEFINES ERRMSG2F.
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00055</td>
<td>O3 ERRMSG2A PICTURE X.</td>
</tr>
<tr>
<td>00056</td>
<td>O2 ERRMSG21 PIC X(80).</td>
</tr>
<tr>
<td>00057</td>
<td>O1 SCREEN20 REDEFINES SCREEN21.</td>
</tr>
<tr>
<td>00058</td>
<td>O2 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00059</td>
<td>O2 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00060</td>
<td>O1 ENTNUMO PIC X(4).</td>
</tr>
<tr>
<td>00061</td>
<td>O2 FILLER PICTURE X(3).</td>
</tr>
<tr>
<td>00062</td>
<td>O2 ERRMSG20 PIC X(80).</td>
</tr>
<tr>
<td>00063</td>
<td>O1 SCREEN31.</td>
</tr>
<tr>
<td>00064</td>
<td>O2 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00065</td>
<td>O2 TOTALC PIC X(90).</td>
</tr>
<tr>
<td>00066</td>
<td>O2 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00067</td>
<td>O2 TOTALF PIC X.</td>
</tr>
<tr>
<td>00068</td>
<td>O3 TOTALA PIC X.</td>
</tr>
<tr>
<td>00069</td>
<td>O2 TOTALI PIC X(5).</td>
</tr>
<tr>
<td>00070</td>
<td>O1 SCREEN30 REDEFINES SCREEN31.</td>
</tr>
<tr>
<td>00071</td>
<td>O2 FILLER PIC X(12).</td>
</tr>
<tr>
<td>00072</td>
<td>O2 FILLER PICTURE X(3).</td>
</tr>
<tr>
<td>00073</td>
<td>O2 TOTALO PIC X(5).</td>
</tr>
<tr>
<td>00074</td>
<td>COPY DFHAID.</td>
</tr>
</tbody>
</table>
| 00075 | |}

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00076</td>
<td>O1 DFHAID.</td>
</tr>
<tr>
<td>00077</td>
<td>O2 DFHNULL PIC X VALUE IS ' '.</td>
</tr>
<tr>
<td>00078</td>
<td>O2 DFHENTER PIC X VALUE IS QUOTE.</td>
</tr>
<tr>
<td>00079</td>
<td>O2 DFHCLEAR PIC X VALUE IS ' '.</td>
</tr>
<tr>
<td>00080</td>
<td>O2 DFHPEN PIC X VALUE IS '2'.</td>
</tr>
<tr>
<td>00081</td>
<td>O2 DFHSPORT PIC X VALUE IS 'W'.</td>
</tr>
<tr>
<td>00082</td>
<td>O2 DFHMSRE PIC X VALUE IS 'X'.</td>
</tr>
<tr>
<td>00083</td>
<td>O2 DFHSTRF PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>00084</td>
<td>O2 DFHTRIG PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>00085</td>
<td>O2 DFHPA1 PIC X VALUE IS '5'.</td>
</tr>
<tr>
<td>00086</td>
<td>O2 DFHPA2 PIC X VALUE IS '5'.</td>
</tr>
<tr>
<td>00087</td>
<td>O2 DFHPA3 PIC X VALUE IS '5'.</td>
</tr>
<tr>
<td>00088</td>
<td>O2 DFHPF1 PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>00089</td>
<td>O2 DFHPF2 PIC X VALUE IS '2'.</td>
</tr>
<tr>
<td>00090</td>
<td>O2 DFHPF3 PIC X VALUE IS '3'.</td>
</tr>
<tr>
<td>00091</td>
<td>O2 DFHPF4 PIC X VALUE IS '4'.</td>
</tr>
<tr>
<td>00092</td>
<td>O2 DFHPF5 PIC X VALUE IS '5'.</td>
</tr>
<tr>
<td>00093</td>
<td>O2 DFHPF6 PIC X VALUE IS '6'.</td>
</tr>
<tr>
<td>00094</td>
<td>O2 DFHPF7 PIC X VALUE IS '7'.</td>
</tr>
<tr>
<td>00095</td>
<td>O2 DFHPF8 PIC X VALUE IS '8'.</td>
</tr>
<tr>
<td>00096</td>
<td>O2 DFHPF9 PIC X VALUE IS '9'.</td>
</tr>
<tr>
<td>00097</td>
<td>O2 DFHPF10 PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>00098</td>
<td>O2 DFHPF11 PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>00099</td>
<td>O2 DFHPF12 PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>01000</td>
<td>O2 DFHPF13 PIC X VALUE IS 'A'.</td>
</tr>
<tr>
<td>01010</td>
<td>O2 DFHPF14 PIC X VALUE IS 'B'.</td>
</tr>
<tr>
<td>01011</td>
<td>O2 DFHPF15 PIC X VALUE IS 'C'.</td>
</tr>
<tr>
<td>01012</td>
<td>O2 DFHPF16 PIC X VALUE IS 'D'.</td>
</tr>
<tr>
<td>01013</td>
<td>O2 DFHPF17 PIC X VALUE IS 'E'.</td>
</tr>
<tr>
<td>01014</td>
<td>O2 DFHPF18 PIC X VALUE IS 'F'.</td>
</tr>
<tr>
<td>01015</td>
<td>O2 DFHPF19 PIC X VALUE IS 'G'.</td>
</tr>
<tr>
<td>01016</td>
<td>O2 DFHPF20 PIC X VALUE IS 'H'.</td>
</tr>
<tr>
<td>01017</td>
<td>O2 DFHPF21 PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>01018</td>
<td>O2 DFHPF22 PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>01019</td>
<td>O2 DFHPF23 PIC X VALUE IS 'I'.</td>
</tr>
<tr>
<td>01020</td>
<td>O2 DFHPF24 PIC X VALUE IS 'I'.</td>
</tr>
</tbody>
</table>
00169 C * EIBRSRCE RESOURCE_NAME 58000000
00170 C * 02 EIBRSRCE PICTURE X(8).
00171 C * EIBSYNC SYNCPONT REQUIRED 62000000
00172 C * 02 EIBSYNC PICTURE X.
00174 C * EIBFREE TERMINAL FREE REQUIRED 66000000
00175 C * EIBRECV DATA RECEIVE REQUIRED 70000000
00176 C * 02 EIBRECV PICTURE X.
00177 C * EIBSEND RESERVED 76000000
00178 C * 02 EIBSEND PICTURE X.
00179 C * EIBATT ATTACH DATA EXISTS 82000000
00180 C * 02 EIBATT PICTURE X.
00181 C * EIBEOC GOTTEN DATA IS COMPLETE 88000000
00182 C * 02 EIBEOC PICTURE X.
00183 C * EIBFMH GOTTEN DATA CONTAINS FMH 94000000
00184 C * 02 EIBFMH PICTURE X.
00185 01 DFHCMMAREA PIC X(5).
00186 01 PASSED-CMAREA REDEFINES DFHCMMAREA.
00187 05 WS-SUM PIC 9(5).
00188
00189 01 DFHEIVSL1 PICTURE X(1).
00190 PROCEDURE DIVISION USING DFHEIVBLK DFHCMMAREA.
00191 SERVICE RELAOD DFHEIVBLK.
00192 SERVICE RELAOD DFHCMMAREA.
00193
00194 + EXEC CICS HANDLE CONDITION
00195 + ERROR (005-ERROR-RTN)
00196 + END-EXEC.
00197 MOVE ' ' TO DFHEIV CALL 'DFHEI1' USING
00198 DFHEIVO GO TO 005-ERROR-RTN DEPENDING ON DFHEIGDI.
00199
00200 IF EIBCALLEN IS EQUAL TO ZEROS
00201 PERFORM 001-INITIAL-RTN
00202 ELSE
00203 PERFORM 002-RECEIVE-RTN.
00204 G0BACK.
00205
00211 001-INITIAL-RTN.
00212 + EXEC CICS SEND
00213 + MAP('SCREEN2')
00215 + MAPSET('BCC8SET')
00216 + MAPONLY
00217 + ERASE
00218 + END-EXEC.
00219 MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE '
00220 ' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO
00221 DFHEIV1 DFHEIV99 DFHEIV98 DFHEIV2
00226  *  EXEC CICS RETURN
00227  *      TRANSID('BCC8')
00228  *      COMMAREA(ORIGINAL-COMMAREA)
00229  *      LENGTH(S)
00230  *      END-EXEC.
00231  MOVE 'BCC8' TO DFHEIV5 MOVE 5 TO DFHEIV11 MOVE '  
00232  TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO
00233  ORIGINAL-COMMAREA DFHEIV11.
00234
00235
00236  002-RECEIVE-RTN.
00237
00238
00239  *  EXEC CICS HANDLE CONDITION
00240  *      MAPFAIL (003-NO-DATA-RTN)
00241  *      END-EXEC.
00242  MOVE '  ' TO DFHEIVO CALL 'DFHEI1' USING
00243  DFHEIVO GO TO 003-NO-DATA-RTN DEPENDING ON DFHEIGDI.
00244
00245
00246  *  EXEC CICS HANDLE AID
00247  *      CLEAR (004-CLEAR-KEY-RTN)
00248  *      END-EXEC.
00249  MOVE '  ' TO DFHEIVO CALL 'DFHEI1' USING
00250  DFHEIVO GO TO 004-CLEAR-KEY-RTN DEPENDING ON DFHEIGDI.
00251
00252
00253  * EXEC CICS RECEIVE
00254  *      MAP('SCREEN2')
00255  *      MAPSET('BCC8SET')
00256  *      END-EXEC.
00257  MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE '  
00258  ' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO
00259  DFHEIV1 SCREEN2 DFHEIV98 DFHEIV2.
00260
00261  IF ENTRNUMI IS NOT NUMERIC
00262
00263  MOVE ATT-NUMERIC-BRT-FSET TO ENTRUMA ERRMSG2A
00264
00265  MOVE 'NUMBER NOT NUMERIC - CORRECT AND REENTER'
00266  TO ERRMSG20
00267
00268  * EXEC CICS SEND
00269  *      MAP('SCREEN2')
00270  *      MAPSET('BCC8SET')
00271  *      DATAONLY
00272  *      END-EXEC.
00273  MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE '  
00274  ' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO
00275  DFHEIV1 SCREEN20 DFHEIV98 DFHEIV2.
00276
00277
00278
00279
00280  * EXEC CICS RETURN
00281  *      TRANSID('BCC8')
00282  *      COMMAREA(PASSED-COMMAREA)
* LENGTH(5)
00284  * END-EXEC
00285  MOVE 'BCC8' TO DFHEIV5 MOVE 5 TO DFHEIV11 MOVE ' \
00286  ' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO DFHEIV5
00287  PASSED-COMMAREA DFHEIV11
00288  
00289  
00290  ELSE
00291  MOVE ENTNUMI TO WS-NUM
00292  ADD WS-NUM TO WS-SUM
00293  
00294  EXEC CICS SEND
00295  MAP('SREEN2')
00296  MAPSET('BCC8SET')
00297  * ERASE
00298  * MAPONLY
00299  * END-EXEC
00300  MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE \ 
00301  '}' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO
00302  DFHEIV1 DFHEIV99 DFHEIV98 DFHEIV2
00303  
00304  
00305  EXEC CICS RETURN
00306  * TRANSID('BCC8')
00307  * COMMAREA(PASSED-COMMAREA)
00308  * LENGTH(5)
00309  * END-EXEC.
00310  MOVE 'BCC8' TO DFHEIV5 MOVE 5 TO DFHEIV11 MOVE ' \
00311  ' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO DFHEIV5
00312  PASSED-COMMAREA DFHEIV11.
00313  
00314  
00315  003-NO-DATA-RTN.
00316  
00317  MOVE ATT-NUMERIC-BRT-FSET TO ERRMSG2A.
00318  
00319  MOVE 'ENTER A NUMBER OR PRESS CLEAR TO DISPLAY TOTAL' \ 
00320  TO ERRMSG22.
00321  
00322  EXEC CICS SEND
00323  MAP('SCREEN2')
00324  MAPSET('BCC8SET')
00325  * DATAONLY
00326  * END-EXEC.
00327  MOVE 'SCREEN2' TO DFHEIV1 MOVE 'BCC8SET' TO DFHEIV2 MOVE \ 
00328  '}' TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO
00329  DFHEIV1 SCREEN20 DFHEIV99 DFHEIV2.
00330  
00331  
00332  
00333  EXEC CICS RETURN
TRANSID('BCC8')
COMMAREA(PASS=COMAREA)
LENGTH(5)

MOVE 'BCC8' TO DFHEIVS MOVE 5 TO DFHEIV11 MOVE '/
TO DFHEIVO CALL 'DFHEI1' USING DFHEIVO DFHEIVS
PASSED-COMAREA DFHEIV11.

OO4-CLEAR-KEY-RTN.

MOVE WS-SUM TO TOTALO.
EXEC CICS SEND
MAP('SCREEN3')
EXEC CICS SEND
ERASE('BCC8SET')
EXEC CICS RETURN.

EXEC CICS RETURN.

EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RETURN.
EXEC CICS RET
OPTIONS IN EFFECT: MOD2.HEX

BCC8:  *

12' DBC8
CF74CCCCFE4444444
92D04233800000000

A=ENTER:
(6,46)1111:

12'GB F=1111
CF7CC1C7FFFFE4444
92D7216E111100000

A=ENTER:
(6,46)ABCD:

12'GB F=ABCD
CF7CC1C7CCCCC4444
92D7216E123400000

A=ENTER:
(6,46)2222:

12'GB F=2222
CF7CC1C7FFFFE4444
92D7216E222200000

A=ENTER:

12'
CF74E44444444444
92D00000000000000

A=ENTER:
I2'GB F=1000
CF7CC1C7FFFFFE44444
92D7216E1000000000

A=CLEAR:

I2
CF6FFFE444444444444
92DFF000000000000

CSMT SHUTDN:

I2'CSMT SHUTDN
CF744CDE4ECEECDE4
92D0B3243028434500

YES:

I2'YES
CF74CECEEE4444444444
92D038520000000000
## CICS/3270 - OUTPUT SIMULATOR

### INPUT STREAM:

```
12' DBCC8
CF74C8CCE0000000
S2D0423380000000
```

### INPUT STREAM: AID=ENTER

```
00000000111111111222222223333333444444455555555566666666777777778
1234567890123456789012345678901234567890123456789012345678901234567890
```

### ATTRIBUTE DETAIL SECTION

| 0000/01 | BCC8 | 0080/02 | 0160/03 | 0240/04 | 0320/05 | 0340/06 | 0480/07 | 0560/08 | 0640/09 | 0720/10 | 0800/11 | 0880/12 | 0960/13 | 1040/14 | 1120/15 | 1200/16 | 1280/17 | 1360/18 | 1440/19 | 1520/20 | 1600/21 | 1680/22 | 1760/23 | 1840/24 |
|---------|------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
|         |      |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
CICS/3270 - OUTPUT SIMULATOR

OUTPUT STREAM:

```
02T B F= Q OCOMPUTE SUM OF SEVERAL NUMBERS, $ ENTER A NUMBER: & GB @ PO @
DFE0C1C711401FDD0E4E14EDA4ECE0CDD4DEDCDE161F0E04D4E0D0D7151CC1710F17
623E216E310B03647435024406025559130544259216DD0553590105442599AD0172DC170DC
```
INPUT STREAM:

12 'GB F-1111
CFFC1C7FFFE00000
92D7216E111100000

CICS/3270 - OUTPUT SIMULATOR
05/11/87 14:46 PAGE 007

INPUT STREAM: AID=ENTER

00000000001111111111222222233333333444444445555555555566666666666666666666677777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

******************************************************************************
* $COMPUTE SUM OF SEVERAL NUMBERS                                         *
******************************************************************************

0000/01   * 0080/02   * 0160/03   * 0240/04   * 0320/05   * 0400/06   * 0480/07   * 0560/08   * 0640/09   * 0720/10   *
0800/11   * 0880/12   * 0960/13   * 1040/14   * 1120/15   * 1200/16   * 1280/17   * 1360/18   * 1440/19   *
1520/20   * 1600/21   * 1680/22   * 1760/23   * 1840/24   *

******************************************************************************
* ENTER A NUMBER:$1111$                                                    *
******************************************************************************

ATTRIBUTE DETAIL SECTION

Y

ATTRIBUTE EXPLANATION SECTION

J - UNPROTECTED NUMERIC  DISPLAY  NON-DETECTABLE  MDT-ON
Y - PROTECTED NUMERIC (ASKIP)  DISPLAY  NON-DETECTABLE  MDT-OFF
5 - PROTECTED NUMERIC (ASKIP)  DARK  NON-DETECTABLE  MDT-OFF
INPUT STREAM:

12'GB  F=ABCD
CF7CC1C7CCCE00000
92D7216E123400000

CICS/3270 - OUTPUT SIMULATOR

05/11/87  14:46  PAGE 010

0000000001111111111222222222333333334444444444445555
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

INPUT STREAM: AID=ENTER

000000000111111111122222222233333333344444444444455
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

********************************************************************************
| ATTRIBUTE DETAIL SECTION |
| $COMPUTE SUM OF SEVERAL NUMBERS | Y |
| $ENTER A NUMBER:$ABCD$ | YJ5 |
********************************************************************************

CURSOR POSITION: ROW-06 COL-51

ATTRIBUTE EXPLANATION SECTION

J - UNPROTECTED  NUMERIC  DISPLAY  NON-DETECTABLE  MDT-ON
Y - PROTECTED  NUMERIC  (ASKIP)  DISPLAY  NON-DETECTABLE  MDT-OFF
5 - PROTECTED  NUMERIC  (ASKIP)  DARK  NON-DETECTABLE  MDT-OFF
**OUTPUT STREAM:**

```
$ $  
02T B F = F' RABCD PO RNUMBER NOT NUMERIC - CORRECT AND REENTER  
DFEOC1C711C71DCCCC1DF1DEDCDDE4DEDCDCC464CDDDCCE4DCDCEDCD444444444 
6232216E316D91234170D05442590563054459930036695301540955559000000000 
```

---

**OUTPUT STREAM: WRITE-INITIAL,FREEKB**

```
0000000001111111122222222333333344444444455555555555666666666777777777778 
12345678901234567890123456789012345678901234567890123456789012345678901234567890 
```

---

```
| 0000/01 | Y      | 0080/02 | Y      | 0160/03 | N      | 0240/04 | YN5    | 0420/05 | N      | 0400/06 | YN5    | 0480/07 | YN5    | 0560/08 | YN5    | 0640/09 | YN5    | 0720/10 | YN5    | 0800/11 | YN5    | 0880/12 | YN5    | 0960/13 | YN5    | 1040/14 | YN5    | 1120/15 | YN5    | 1200/16 | YN5    | 1280/17 | YN5    | 1360/18 | YN5    | 1440/19 | YN5    | 1520/20 | YN5    | 1600/21 | YN5    | 1680/22 | YN5    | 1760/23 | YN5    | 1840/24 | YN5    | 1920/25 | YN5    | 1980/26 | YN5    |
|---------|--------|---------|--------|---------|-------|---------|--------|--------|-------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|---------|--------|
```

---

```
*ATTRIBUTE DETAIL SECTION*  

$COMPUTE SUM OF SEVERAL NUMBERS$  

$ENTER A NUMBER:$ABCD$  

*NUMER NOT NUMERIC - CORRECT AND REENTER*  

CURSOR POSITION: ROW-06 COL-47  

**ATTRIBUTE EXPLANATION SECTION**  

<table>
<thead>
<tr>
<th>N</th>
<th>UNPROTECTED NUMERIC</th>
<th>BRIGHT</th>
<th>PEN-DETECTABLE</th>
<th>MDT-ON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>PROTECTED NUMERIC</td>
<td>(ASKIP)</td>
<td>DISPLAY</td>
<td>NON-DETECTABLE</td>
</tr>
<tr>
<td>S</td>
<td>PROTECTED NUMERIC</td>
<td>(ASKIP)</td>
<td>DARK</td>
<td>NON-DETECTABLE</td>
</tr>
</tbody>
</table>
INPUT STREAM:

12'GB F=2222
CFTCC1C7FFFFEO0000
92D7216E2222000000

000000001111111111222222223333333344444444445555555555555566666666667777777777788888888999999999990
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

CICS/3270 - OUTPUT SIMULATOR

05/11/87 14:46 PAGE 014

CICS/3270 - OUTPUT SIMULATOR

05/11/87 14:46 PAGE 015

INPUT STREAM: AID=ENTER

000000001111111111222222223333333344444444445555555555555566666666667777777777788888888999999999990
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

******************************

ATTRIBUTE DETAIL SECTION

$COMPUTE SUM OF SEVERAL NUMBERS * Y

$ENTER A NUMBER: $2222$ *

ATTRIBUTE EXPLANATION SECTION

N - UNPROTECTED NUMERIC  BRIGHT  PEN-DETECTABLE MDT-ON

Y - PROTECTED NUMERIC (ASKIP) DISPLAY  NON-DETECTABLE MDT-OFF

5 - PROTECTED NUMERIC (ASKIP) DARK  NON-DETECTABLE MDT-OFF

CURSOR POSITION: ROW-06 COL-51

******************************
### INPUT STREAM:  

`I2' CF744E00000000000 92D0000000000000`
CICS/3270 - OUTPUT SIMULATOR
05/11/77 14:46 PAGE 020

0000000001111111112222222223333333333334444444455555555555666666666677777777777888888888899999999990
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

OUTPUT STREAM:

$02T B F= PO ENTER A NUMBER OR PRESS CLEAR TO DISPLAY TOTAL
DF80C1C711DF1DDC8CD4C4DCCCD4DADDCE4DCCDC4EDCDCE4DCE4444
6232216E3170955359010544259069079522033519036049273180363130000

CICS/3270 - OUTPUT SIMULATOR
05/11/77 14:46 PAGE 021

OUTPUT STREAM: WRITE-INITIAL,FREEKB

0000000001111111112222222223333333333334444444455555555555666666666677777777777888888888899999999990
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

****************************************************************************** ATTRIBUTE DETAIL SECTION
******************************************************************************

$COMPUTE SUM OF SEVERAL NUMBERS

0000/01 *
0080/02 *
0160/03 *
0240/04 *
0320/05 *
0400/06 *
0480/07 *
0560/08 *
0640/09 *
0720/10 *
0800/11 *
0880/12 *
0960/13 *
1040/14 *
1120/15 *
1200/16 *
1280/17 *
1360/18 *
1440/19 *
1520/20 *
1600/21 *
1680/22 *
1760/23 *
1840/24 *

******************************************************************************

CURSOR POSITION: ROW-06 COL-47

ATTRIBUTES EXPLANATION SECTION

I - UNPROTECTED NUMERIC DISPLAY NON-DETECTABLE MDT-OFF
N - UNPROTECTED NUMERIC BRIGHT PEN-DETECTABLE MDT-ON
Y - PROTECTED NUMERIC (ASKIP) DISPLAY NON-DETECTABLE MDT-OFF
S - PROTECTED NUMERIC (ASKIP) DARK NON-DETECTABLE MDT-OFF
I2'QB F+1000
CF7CC1C7FFFFE00000
92D7216E100000000

**INPUT STREAM: AID ENTER**

| 0000/01 | *  |
| 0080/02 | *  |
| 0160/03 | *  |
| 0240/04 | *  |
| 0320/05 | *  |
| 0400/06 | *  |
| 0480/07 | *  |
| 0560/08 | *  |
| 0640/09 | *  |
| 0720/10 | *  |
| 0800/11 | *  |
| 0880/12 | *  |
| 0960/13 | *  |
| 1040/14 | *  |
| 1120/15 | *  |
| 1200/16 | *  |
| 1280/17 | *  |
| 1360/18 | *  |
| 1440/19 | *  |
| 1520/20 | *  |
| 1600/21 | *  |
| 1680/22 | *  |
| 1760/23 | *  |
| 1840/24 | *  |

**CURSOR POSITION: ROW-06 COL-51**

**ATTRIBUTE EXPLANATION SECTION**

| J  | UNPROTECTED NUMERIC | DISPLAY | NON-DETECTABLE | MDT-ON |
| N  | UNPROTECTED NUMERIC | BRIGHT  | PEN-DETECTABLE | MDT-ON |
| Y  | PROTECTED NUMERIC  | (ASKIP) | NON-DETECTABLE | MDT-OFF |
| 5  | PROTECTED NUMERIC  | (ASKIP) | DARK          | NON-DETECTABLE | MDT-OFF |
OUTPUT STREAM: WRITE-ERASE.FREEKB

<table>
<thead>
<tr>
<th>$</th>
<th>$</th>
<th>$</th>
<th>$</th>
<th>$</th>
</tr>
</thead>
</table>
| 02T B F= Q ODCOMPUTE SUM OF SEVERAL NUMBERS F_OENTER A NUMBER: & GB @ PO @
| DFC0CC1E1141FCDDEEC4EED4C4ECECD4EDC4DCECDE1C61FDECD4C0EDCCD7151CC71DF17
| 623E216E31G03647435024406602555913054429216DD05359010544259ADD172DC170DC

OUTPUT STREAM: WRITE-ERASE.FREEKB

| 0000/01 * | 0080/02 * | 0160/03 * | 0240/04 * | 0320/05 * |
| 0400/06 * | 0480/07 * | 0560/08 * | 0640/09 * | 0720/10 * |
| 0800/11 * | 0880/12 * | 0960/13 * | 1040/14 * | 1120/15 * |
| 1200/16 * | 1280/17 * | 1360/18 * | 1440/19 * | 1520/20 * |
| 1600/21 * | 1680/22 * | 1760/23 * | 1840/24 * |

************** ATTRIBUTE DETAIL SECTION ******

$COMPUTE SUM OF SEVERAL NUMBERS

$ENTER A NUMBER:$ $ $ $ Y15

CURSOR POSITION: RDW-06 COL-47

ATTRIBUTE EXPLANATION SECTION

I - UNPROTECTED NUMERIC DISPLAY NON-DETECTABLE MDT-OFF

Y - PROTECTED NUMERIC (ASKIP) DISPLAY NON-DETECTABLE MDT-OFF

S - PROTECTED NUMERIC (ASKIP) DARK NON-DETECTABLE MDT-OFF
### INPUT STREAM:

```
I2_
CF6FEOOOO00000000
92DFF00000000000
```

### INPUT STREAM: AID=CLEAR

```
000000001111111122222222333333334444444555555556666666666667777777777
1234567890123456789012345678901234567890123456789012345678901234567890
```

```
+---------------------------------+--------+
| 0000/01 | 0080/02 | 0160/03 | 0240/04 | 0320/05 | 0400/06 | 0480/07 | 0560/08 | 0640/09 | 0720/10 | 0800/11 | 0880/12 | 0960/13 | 1040/14 | 1120/15 | 1200/16 | 1280/17 | 1360/18 | 1440/19 | 1520/20 | 1600/21 | 1680/22 | 1760/23 | 1840/24 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
|        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |
```

**ATTRIBUTE DETAIL SECTION**

**CURSOR POSITION: ROW-01 COL-01**

**ATTRIBUTE EXPLANATION SECTION**
CICS/3270 - OUTPUT SIMULATOR

05/11/87 14:46 PAGE 028

OUTPUT STREAM:

02T B ) TOTAL SUM OF NUMBERS F_ OCOMPUTED TOTAL: -04333
DFEE0C14411451FEDEC04EE04E061CECF0DDE00C14EDC0716FFFF

623E21003100D36313024406605442592166D036474354036313AD004333

CICS/3270 - OUTPUT SIMULATOR

05/11/87 14:46 PAGE 029

OUTPUT STREAM: WRITE-ERASE,FREEKB

00000000111111112222222233333333444444445555555555666666666666777777777777777777778
12345678901234567890123456789012345678901234567890123456789012345678901234567890

*************** ATTRIBUTE DETAIL SECTION ***************

$TOTAL SUM OF NUMBERS

Y

$COMPUTED TOTAL:$04333

YQ

*************** ATTRIBUTE EXPLANATION SECTION ***************

0000/01

0080/02

0160/03

0240/04

0320/05

0400/06

0480/07

0560/08

0640/09

0720/10

0800/11

0880/12

0960/13

1040/14

1120/15

1200/16

1280/17

1360/18

1440/19

1520/20

1600/21

1680/22

1760/23

1840/24

CURSOR POSITION: ROW-01 COL-01

O - PROTECTED ALPHAMERIC DISPLAY NON-DETECTABLE MDT-OFF

Y - PROTECTED NUMERIC (ASKIP) DISPLAY NON-DETECTABLE MDT-OFF
INPUT STREAM:

I2'. CSMT SHUTDN
CF74CEDE4EC4BECDE0
92D0B3243028434500

**** PO08 DO005/BO000 WARNING-UNFORMATTED STREAM, BUFFER CLEARED

INPUT STREAM: AID=ENTER

<table>
<thead>
<tr>
<th>0000/01</th>
<th>CSMT SHUTDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0080/02</td>
<td>0160/03</td>
</tr>
<tr>
<td>0240/04</td>
<td>0320/05</td>
</tr>
<tr>
<td>0400/06</td>
<td>0480/07</td>
</tr>
<tr>
<td>0560/08</td>
<td>0640/09</td>
</tr>
<tr>
<td>0720/10</td>
<td>0800/11</td>
</tr>
<tr>
<td>0880/12</td>
<td>0960/13</td>
</tr>
<tr>
<td>1040/14</td>
<td>1120/15</td>
</tr>
<tr>
<td>1200/16</td>
<td>1280/17</td>
</tr>
<tr>
<td>1360/18</td>
<td>1440/19</td>
</tr>
<tr>
<td>1520/20</td>
<td>1600/21</td>
</tr>
<tr>
<td>1680/22</td>
<td>1760/23</td>
</tr>
<tr>
<td>1840/24</td>
<td></td>
</tr>
</tbody>
</table>

ATTRIBUTE DETAIL SECTION

CURSOR POSITION: ROW-01 COL-12

ATTRIBUTE EXPLANATION SECTION
COMMUNICATION AREA

1. Data can be passed between programs when control is passed to another program by means of a program control command: LINK, XCTL, or RETURN.

   a. EXEC CICS LINK
      
          PROGRAM(name)
          [COMMAREA(data-area)
           LENGTH(length-in-bytes)]
      
      END-EXEC.

   b. EXEC CICS XCTL

          PROGRAM(name)
          [COMMAREA(data-area)
           LENGTH(length-in-bytes)]

      END-EXEC.

   c. EXEC CICS RETURN

          [TRANSID(name)]
          [COMMAREA(data-area)
           LENGTH(length-in-bytes)]

      END-EXEC.

Note: Data can also be passed between application programs and transactions in other ways. For example, the data can be stored in a CICS storage area outside the local environment of the application program, such as the Transaction Work Area (TWA).

2. The COMMAREA option of the LINK and XCTL commands specifies the name of a data area (known as a Communication Area) in which data can be passed to the program being invoked.

3. In a similar manner, the COMMAREA of the RETURN command specifies the name of a data area in which data can be passed to the Transaction identified in the TRANSID option. The LENGTH option specifies the length (in bytes) of the communication area.

   The TRANSID option specifies a transaction that will be initiated when input is received from the terminal (user) associated with the task.

4. The invoked program receives the data as a parameter. The program must contain the definition of a data area to allow access to the passed data. In a COBOL program, the data must be called DFHCOMMAREA. The data area need not be of the same length as the original communication area; if access is required only to the first part of the data, the new data area can be shorter.
5. The invoked program can determine the length of any communication area that has been passed to it by accessing the EIBCALEN field in the EIB of the task.

If no communication area has been passed, the value of EIBCALEN will be zero. Otherwise, EIBCALEN will contain the value specified in the LENGTH of the LINK, XCTL, or RETURN command, regardless of the size of the data area in the invoked program.

6. When a communication area is passed by a LINK command, the invoked program is passed a pointer to the communication area itself. Any changes made to the contents of the data area in the invoked program are available to the invoking program, when control returns to it.

To access any such changes, the program names the data area specified in the original COMMAREA option.

7. When a communication area is passed by an XCTL command, a copy of that area is made. For example:

If Program A issues a LINK command to Program B, which in turn issues an XCTL to Program C, and if Program B passes to Program C the same communication area that Program A passed to Program B, then Program C will be passed addressability to the same communication area that belongs to Program A (not a copy of it) and any changes made by Program C will be made available to Program A when control returns to it.

8. A communication area can be passed by a RETURN command issued at the highest logical level (Level 1; CICS is considered to be at Level 0) when control returns to CICS.

In this case, a copy of the communication area is made, and addressability to the copy is passed to the first program of the next transaction.

At this level (Level 1), the task is associated with a terminal and the TRANSID option can be used to specify the transaction identifier for the next program to be associated with that terminal; this causes the subsequent input entered from the terminal (by the user) to be interpreted wholly as data.

In addition, the COMMAREA option can be used to pass data to the new task that will be started. The COMMAREA and LENGTH options can be used only when the RETURN command returns control to CICS; otherwise, the INVREQ exceptional condition will occur.
Example 1:

IDENTIFICATION DIVISION.
PROGRAM-ID. PROG1.

WORKING-STORAGE SECTION.
01 COM-REGION.
  05 ORIGINAL-COMMAREA PIC X(3).

PROCEDURE DIVISION.
  MOVE 'ABC' TO ORIGINAL-COMMAREA.
  EXEC CICS LINK
    PROGRAM('PROG2')
    COMMAREA(COM-REGION)
    LENGTH(3)
  END-EXEC.

IDENTIFICATION DIVISION.
PROGRAM-ID. PROG2.

LINKAGE SECTION.
01 DFHCOMMAREA.
  05 PASSED-COMMAREA PIC X(3).

PROCEDURE DIVISION.
  IF EIBCALEN GREATER THAN ZERO
    IF PASSED-COMMAREA EQUALS 'ABC'
      ...
Example 2:

IDENTIFICATION DIVISION.
PROGRAM-ID. PROG1.

. .

WORKING-STORAGE SECTION.
01 TERMINAL-STORAGE.
  05 ORIGINAL-FIELD PIC X(3).
  05 DATA-FIELD PIC X(17).

. .

PROCEDURE DIVISION.
MOVE 'ABC' TO ORIGINAL-FIELD.
EXEC CICS RETURN
  TRANSID('TRN2')
  COMMAREA(TERMINAL-STORAGE)
  LENGTH(20)
END-EXEC.

. .

IDENTIFICATION DIVISION. (COBOL program to be invoked)
PROGRAM-ID. PROG2.

. .

LINKAGE SECTION.
01 DFHCOMMAREA.
  05 PASSED-FIELD PIC X(3).
  05 DATA-FIELD PIC X(17).

. .

PROCEDURE DIVISION.
  IF EIBCALEN GREATER THAN ZERO
  IF PASSED-FIELD EQUALS 'ABC'
  MOVE 'XYZ' TO PASSED-FIELD'
  . .

EXEC CICS RETURN
END-EXEC.
CICS/VS FILE CONTROL BROWSE COMMANDS

To begin a browse, you must establish position for the first record to be read.

STARTBR
  DATASET (NAME)
  RIDFLD (DATA AREA)
  GENERIC
  KEYLENGTH (DATA VALUE)
  GTEQ / EQUAL

Exceptional Conditions:
- NOTOPEN
- NOTFND
- INVREQ

To read the next record during a browse operation

READNEXT
  DATASET (NAME)
  INTO (DATA AREA) / SET (POINTER)
  LENGTH (DATA AREA)
  RIDFLD (DATA AREA)
  KEYLENGTH (DATA VALUE)

Exceptional Conditions:
- NOTFND - Only if a new key is specified for browse during a generic search.
- ENDFILE
- INVREQ
- NOTOPEN

To read the previous record during a browse (VSAM only)

READPREV
  DATASET (NAME)
  INTO (DATA AREA) / SET (POINTER)
  LENGTH (DATA AREA)
  RIDFLD (DATA AREA)
  KEYLENGTH (DATA VALUE)

Exceptional Conditions:
- Same as those for READNEXT.
CICS/VS FILE CONTROL BROWSE COMMANDS

To reset the starting position of a browse

RESETBR
   DATASET (NAME)
   RIDFLD (DATA AREA)
   GENERIC
   KEYLENGTH (DATA VALUE)
   GTEQ / EQUAL

   Exceptional Conditions:
   - NOTOPEN
   - NOTFND
   - INVREQ

To terminate a browse operation

ENDBR
   DATASET (NAME)

   Exceptional Conditions:
   - NOTOPEN
   - INVREQ
CICS/VS STORAGE CONTROL COMMANDS

To obtain and initialize a main storage area

GETMAIN
  SET (POINTERS)
  LENGTH (VALUE)
  INITIMG (VALUE) OPTIONAL

To release main storage acquired with a GETMAIN request

FREEMAIN
  DATA (DATA AREA)

The following is an example of the use of Storage Control commands:

DATA DIVISION.
WORKING-STORAGE SECTION.
  .
  .
LINKAGE SECTION.
  01 DFHCOMMAREA           PIC X(10).   (VARIATES WITH PROGRAM)
  01 PARM-LIST.
      05 FILLER          PIC S9(8) COMP. (USED BY CICS)
      05 STORAGE-PTR     PIC S9(8) COMP.
  01 STORAGE-AREA         PIC X(100).
PROCEDURE DIVISION.
  .
  EXEC CICS GETMAIN
      SET (STORAGE-PTR)
      LENGTH (100)
END-EXEC.
  .
  . PROGRAM CAN NOW USE STORAGE AREA AS NEEDED TO PERFORM ITS
  . PROCESSING.
  .
  EXEC CICS FREEMAIN
      DATA (STORAGE-AREA)
END-EXEC.
  .
  . ANY REFERENCE TO STORAGE-AREA AFTER FREEMAIN WILL CAUSE THE
  . PROGRAM TO ABEND.
The following example initializes the contents of the acquired storage to LOW-VALUES. However, LOW-VALUES can not be specified in the INITIMG option of the GETMAIN command, so a working storage field named BINARY-ZEROS is defined and given a value of LOW-VALUES.

DATA DIVISION.

WORKING-STORAGE SECTION.

77 BINARY-ZEROS PIC X VALUE LOW-VALUES.

LINKAGE SECTION.

01 DFHCOMMAREA PIC X(...).

01 PARM-LIST.
  05 FILLER PIC S9(8) COMP.
  05 STORAGE-PTR PIC S9(8) COMP.

01 STORAGE-AREA PIC X(100).

PROCEDURE DIVISION.

EXEC CICS GETMAIN
  SET (STORAGE-PTR)
  LENGTH (100)
  INITIMG (BINARY-ZEROS)
END-EXEC.

ACQUIRED STORAGE AREA IS INITIALIZED TO LOW-VALUES.
CICS/VS TRANSIENT DATA CONTROL COMMANDS

To write data to a transient data queue

WRITEQ TD
  QUEUE (NAME)
  FROM (DATA AREA)
  LENGTH (VALUE)

Note: LENGTH is optional if the WRITEQ command is being used with an extrapartition queue that has fixed length records.

Exceptional Conditions:
- QUIDERR - The queue name specified is not defined in the destination control table.
- NOTOPEN - The specified queue is closed.
- NOSPACE - No more space on the queue (intrapartition).
- LENGERR - Length of record is not specified for a variable length queue, length specified is greater than maximum for queue, of incorrect length is specified for fixed length queue.

To read a record from a transient data queue

READQ TD
  QUEUE (NAME)
  INTO (DATA AREA) / SET (POINTER)
  LENGTH (DATA AREA) OPTIONAL WITH SET

Exceptional Conditions:
- QIDERR - The queue name specified is not defined in the destination control table.
- NOTOPEN - The specified queue is closed.
- LENGERR - Length of INTO area is smaller than the transient data record being read.
- QZERO - This error occurs when the queue being read is empty (analogous to an end of file condition).
- QBUSY - Occurs if queue being read is being written to or deleted.

To delete an intrapartition transient data queue

DELETEQ TD
  QUEUE (NAME)

Exceptional Conditions:
- QIDERR - The queue name specified is not defined in the destination control table.