CICS update

187

June 2001

In this issue

CICS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office

Xephon
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

Subscriptions and back-issues

A year’s subscription to *CICS Update*, comprising twelve monthly issues, costs £175.00 in the UK; $270.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £16.00 ($23.50) each including postage.

CICS Update on-line

Code from *CICS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon. com/cicsupdate.html; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

Contributions

When Xephon is given copyright, articles published in *CICS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*, or you can download a copy from www.xephon.com/contnote.html.

Testing CICS-PL/I transactions

As you know, testing CICS programs is rather limited. We have the CEDF transaction, which allows us to carry out interactive testing, but we can see only the execution of CICS statements, and we can change some parameters before and after each of these statements. Unfortunately, we can't see the execution of PL/I statements, we can't stop the execution before each statement, see the values of certain variables, and compare them with their values after execution of the statement. As a replacement for this we have the statement:

```
ENTER TRACENUM(data_value) FROM(user_trace_entry) RESOURCE(name)  [EXCEPTION],
```

where:

- *data_value* must be a BIN FIXED(15) variable and represents the identification of a record in the user trace table, and must be a number from 1 to 199.
- *user_trace_entry* specifies a data area whose contents are to be entered into the data field of the trace table entry.
- *name* is a CHAR(8) variable and will be entered in the resource field in the trace table.
- The *EXCEPTION* parameter specifies that CICS is to write a user exception trace entry. This parameter overrides the master user trace flag, and CICS writes the trace entry even if the user trace flag is off. Exception trace entries are identified by the characters '*EXCU' when the trace entries are formatted by the trace utility program.

The statement above makes a trace entry in the currently active trace destinations only if the master and user trace flags are **on** (you must call your systems programmers), unless you specify the EXCEPTION option, in which case a user trace entry is always written, even if the master and user trace flags are off. Exception trace entries are always written to the internal trace table (even if internal tracing is set **off**), but they are written to other destinations only if they are active.

We can use transaction CEDF in a test environment without problems and get the information about the execution of CICS commands, but we don't know anything about values of variables and can't follow the flow of PL/I commands because the execution is stopped just before CICS statements. In that case we have a rough but accurate solution when we send a Picture or Char variable to the screen. (Before that we equalize it with certain variables we want to see with the ERASE option.) We use:

```
EXEC CICS SEND FROM(Pic_variable) ;
```

We have a more elegant but less accurate alternative to the statement above:

```
ENTER TRACENUM FROM(variable),
```

This can write everywhere we want in the code and we can see accurate values for variables in the **FROM** option. This statement won't write an entry in the trace table if the flag is **off**, but we will see the values if we are using CEDF.

But how can we test program flow, which depends on user name, OPID, terminal ID, or time of execution? Of course, many of these parameters can be changed when we use CEDF, but not all that we want. It may be very difficult to test every possible flow through the program, and, besides that, we need a lot of support from system programmers, and some tools, which may not be installed at our site, or perhaps have bugs or are too complicated.

We can overcome these problems in a test environment. However, the word 'test' means that we have test data which won't be exactly the same as production data. But what will happen when we have a problem in production? Developers and project developers in most cases are not authorized for production transactions as well as CEDF so we can't follow any statements, even CICS statement. If we have an abend in a program we can see this in the CICS log, but we can see only the name of the procedure and offset, which is probably not enough information. We can see the number of the statement if we used the option **GOSTMT** in the compiler, but this increases the load module and for that reason it is not recommended for production. In that case we should write a record in the trace table anyway and write the data which leads us to the abend in the data field.

From the CICS log we can see where in the program the abend occurred, and from the trace table we can see specific data (account number, amount of money, etc). When we haven't abended, we can't see anything, but our users can tell us about unusual situations. When dealing with financial transactions, all we can do is test it again in the test environment. I have a better solution.

As I said, we can solve most of the problems in production if we have active trace tables and write the records into them, but only when we want to. We also need very good tools for reading formatted versions of these records. Setting trace flags in production may be very risky for the system and we must be very careful and monitor this. We must also work very closely with our system programmers (who may be thousands of miles away).

For this reason I developed my own system to trace CICS programs, and we can use it in the test as well as the production environment. It consists of two programs. The first program writes an entry into the log file (or DB2 table) depending on the value of a field from another file (or DB2 table) and some other fields which indicate whether the program is for testing or not. We can write the entry in error routines. We call this program with:

```
EXEC CICS LINK PROGRAM('WRI_TRA') COMMAREA(COMA_WRI_TRA) RESP(S);
```

It doesn't matter whether this command executes correctly or not. If it doesn't work, we will not have an entry in the log. This isn't very important.

In the source of program we want to test, we must have a declaration of Common Area (COMA_WRI_TRA) and fill in some fields. We also have one variable, Char (137) (from COMA_WRI_TRA, you can set more or less), where you can write the names and values for all relevant variables that you want to watch and trace.

The first call of the program 'WRI_TRA' determines whether the program is to be tested or not. If it is, we write an entry and set the specific variable from COMA_WRI_TRA, and continue writing if we have more control points. If the program isn't for testing, we will not write an entry and we will not go to the next control point. We have control of logging only on specific terminals, users, or accounts.

We must enlarge the source with a declaration of the Common Area, which can be included with the %INCLUDE statement, and with one internal or external procedure (this is because of simplicity of the source). If the program doesn't need to be tested, its speed is reduced only by one additional READ statement.

A second program writes an entry about program testing characteristics and at the same time a transaction to view the log of a particular program. Here we write the flag which determines whether a program is for testing or not. We can specify only specific terminals, users, or accounts for logging and for viewing. We can change all of that very quickly and easily. We can also specify a starting date and time for viewing the log. As I mentioned earlier, data can be stored in a VSAM file or in a DB2 table.

Nebojsa Cosic
Project Developer
Postal Savings Bank of Yugoslavia (Yugoslavia)

A simple interface to DFHCSDUP to extract CICS objects' cross-reference data

The CICS system definition utility program DFHCSDUP is a component of Resource Definition Online. It's an offline utility program that allows users to read from and/or write to a CICS System Definition file, either while CICS is running or while it is inactive.

Using DFHCSDUP you can extract the requested data from the CSD file and pass it to a named user program for processing. The DFHCSDUP EXTRACT command causes the CSD data you select to be passed unformatted to the user program.

The user program can then create reports of CSD data that meet local requirements. For example, you could cross-reference related definitions (such as programs, transactions, sessions, etc).

There is a CSD cross-referencing utility program to produce a cross-reference listing of objects and keywords on the CSD. The program is DFH0CRFC and it is in VS COBOL II.

The data gathered by the EXTRACT command can be passed to the DFH0CRFC program, where it is saved in a file (cross-reference table). The file produced from the utility can be consulted or can be processed to give a different interface to the user.

The tool is composed of two parts:

- Extraction of the cross-reference data from the CSD file using the EXTRACT command and the DFH0CRFC program.
- Processing the output from the DFH0CRFC program for display to the user.

The main purpose of this tool is to supply a simple interface to the users so they can cross-reference objects in a CSD group.

The execution of program DFH0CRFC and the display for the user are carried out by CSDXREF EXEC. The program DFH0CRFC must be run against an EXTRACT command using the following syntax:

```
EXTRACT GROUP(group name) OBJECTS USERPROGRAM(DFHØCRFC)
```

For program DFH0CRFC, you must define, in a sequential dataset, the objects and keywords for which you want a cross-reference listing. The dataset is read by the program using the DDname CRFINPT. CRFINPT is a sequential file containing 80-byte records, and each record contains one object or keyword to be cross-referenced. You can cross-reference any valid resource type or attribute known to CEDA.

For each record in the file, a report is produced detailing the different values assigned to the keyword, where they are defined, and where they are used. You should define the dataset control block subparameters for CRFINPT as DSORG=PS, RECFM=F, LRECL=80, and BLKSIZE=80. You must specify to CSDXREF EXEC the name of the input file (CRFINPT), the name of the CSD group, and the name of DFHCSD file. After the execution of the first part of the tool (DFH0CRFC), you can see, in simple form, the CICS objects cross-reference table.

CSDXREF has been developed in REXX and uses ISPF functions. The utility was developed and tested on OS/390 2.6 and OS/390 2.8, and CICS/ESA 4.1.0 and CICS TS 1.3.

COMPILE AND LINK-EDIT EXTRACT PROGRAM

You must compile and link-edit the DFH0CRFC user programs as batch programs, not as CICS applications. When you link-edit the programs, you must specify the following link-edit control statements:

- An ENTRY statement that defines the entry name as DFHEXTRA. DFHEXTRA is the entry name in the CICS-supplied stub, DFHEXCI.
- An INCLUDE statement for a CICS-supplied stub that must be included in your user program. Include DFHEXCI in any COBOL user program that you write for use with the DFHCSDUP EXTRACT command. DFHEXCI is the interface stub between DFHCSDUP and the COBOL user program.
- Specify the COBOL routines on the INCLUDE statements.
- A CHANGE statement to change the dummy CSECT name in the CICS-supplied stub from EXITEP to the name of your user program. The CICS-supplied stub, DFHEXCI, is generated with a link to the user program using a dummy CSECT name (EXITEP). Use the link-edit CHANGE statement to change the CSECT name from EXITEP to the name specified on the PROGRAM-ID statement in the user program. The CSD user program DFH0CRFC uses the program-id 'CREFCSD'.

SAMPLE JOB TO COMPILE AND LINK-EDIT THE USER PROGRAM

```
//CICSUSER JOB (CICSØØØØ),
//         CLASS=S,
//         MSGCLASS=X,
//         MSGLEVEL=(1,1),
//         NOTIFY=&SYSUID
//*-------------------------------------------------------------*/
//*--------   Step to compile COBOL II  user program     ---------*/
//*-------------------------------------------------------------*/
//COB2    EXEC PGM=IGYCRCTL,
//   PARM='APOST,LIB,OFFSET,MAP,NOSEQ,BUFSIZE(3ØK),NORENT,NORES'
//*
//STEPLIB  DD DISP=SHR,DSN=COBII.COB2COMP
//SYSIN    DD DISP=SHR,DSN=CICS.USERLIB.SOURCE(DFHØCRFC)
//SYSLIB   DD DISP=SHR,DSN=CICS.USERLIB.SOURCE
//SYSLIN   DD DSN=&&LOADSET,DISP=(MOD,PASS),UNIT=VIO,
//        SPACE=(8ØØ,(1ØØØ,1ØØØ))
//SYSUDUMP DD SYSOUT=*
```

```
//SYSUT1   DD SPACE=(800,(1000,1000),,,ROUND),UNIT=VIO
//*
//MODOBJ   EXEC PGM=IEBGENER
//SYSUT1    DD DSN=&&LOADSET,DISP=(OLD,PASS)
//SYSUT2    DD DSN=CICS.LIBRARY.USER.OBJ(DFH0CRFC),DISP=SHR
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD DUMMY
//*-------------------------------------------------------------*/
//*--------   Step to link-edit COBOL II  user program    --------*/
//*-------------------------------------------------------------*/
//LKED    EXEC PGM=IEWL,
//   PARM='LIST,XREF,LET'
//*
//SYSLIB   DD DISP=SHR,DSN=COBII.COB2LIB
//CICSLIB  DD DISP=SHR,DSN=CICS.SDFHLOAD
//OBJLIB   DD DISP=SHR,DSN=CICS.LIBRARY.USER.OBJ
//SYSLMOD  DD DISP=SHR,DSN=CICS.LIBRARY.USER.LOAD
//SYSUT1   DD SPACE=(1024,(50,50)),UNIT=VIO
//SYSPRINT DD SYSOUT=*
//COBLIB   DD DISP=SHR,DSN=COBII.COB2LIB
//SYSUDUMP DD SYSOUT=*
//SYSLIN   DD *
 ENTRY     DFHEXTRA
 CHANGE    EXITEP(CREFCSD)
 INCLUDE   CICSLIB(DFHEXCI)
 INCLUDE   SYSLIB(ILBOSRV)
 INCLUDE   SYSLIB(ILBOCMM)
 INCLUDE   SYSLIB(ILBOBEG)
 INCLUDE   OBJLIB(DFH0CRFC)
 NAME      DFH0CRFC(R)
/*
/*
//
```

PROGRAM DFH0CRFC

```
****************************************************************
* MODULE NAME = DFH0CRFC                                       *
* DESCRIPTIVE NAME = CSD Cross-referencing Sample Program      *
*--------------------------------------------------------------*
* The function of this program is to produce a cross-reference  *
* listing of objects or keywords on the CICS system definition  *
* file (DFHCSD). The program is driven by a user exit in the CSD*
* off-line utility DFHCSDUP. Data describing resource definition*
* on the CSD is passed to this exit in a standard parameter list*
* format, and the pertinent data saved in a cross-reference     *
* table which is built in store.                               *
*                                                              *
* After all the data passed to the exit has been processed,     *
* that is on the FINAL call to the exit, the contents of the    *
* cross-reference table are printed out in collating sequence.  *
*                                                               *
* This program  assumes that it is run via a CSD utility        *
* EXTRACT command of the form:                                  *
*        EXTRACT GROUP(<group name>) OBJECTS                    *
*          where <group name> may be generic                    *
*     or                                                        *
*        EXTRACT LIST(<list name>)   OBJECTS                    *
*          where <list name> is NOT generic                     *
*                                                               *
* NB  This program must be link edited with the stub DFHEXCI    *
*     and the VS COBOL II library subroutines ILBOSRV, ILBOBEG, *
*     and ILBOCMM.                                              *
*****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. CREFCSD.
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT DATA-IN
            ASSIGN TO SYS002-DA-3380-S-CRFINPT
            ORGANIZATION IS SEQUENTIAL
            FILE STATUS IS INPUT-FILE-STATUS
            ACCESS IS SEQUENTIAL.
     SELECT DATA-OUT
            ASSIGN TO SYS002-DA-3380-S-CRFOUT
            ORGANIZATION IS SEQUENTIAL
            FILE STATUS IS OUTPUT-FILE-STATUS
            ACCESS IS SEQUENTIAL.
 EJECT.
* ************************************************************* *
*  D E F I N E  T H E  I N P U T  A N D  O U T P U T  F I L E S *
* This program requires two QSAM files to be defined for its     *
* use:                                                           *
*    1) DATA-IN: This is the input file to the cross referencing *
*                program and its contents define the objects and *
*                keywords to be cross-referenced (the program    *
*                allows for up to a maximum of 10 in any one run)*
*                The file should be defined as a file of fixed-  *
*                length records of 80 characters each, with one  *
*                record for each object or keyword to be cross-  *
*                referenced.                                     *
*                Only the characters in columns 1-12 of a record *
*                will be considered to be significant.           *
*              eg To cross-reference PROGRAMs with TRANSACTIONs *
*                  TYPETERMs with TERMINALs the input would be:  *
*                   PROGRAM                                      *
*                   TYPETERM                                     *
*                                                                *
*    2) DATA_OUT:This is the output file to contain the cross-   *
*                reference report. It should be defined as a     *
*                file of fixed-length records of length 132      *
*                characters.                                      *
* ************************************************************** *
 DATA DIVISION.
 FILE SECTION.
 FD  DATA-IN
     RECORD CONTAINS 8Ø CHARACTERS
     BLOCK CONTAINS Ø RECORDS
     RECORDING MODE IS F
     LABEL RECORDS ARE OMITTED.
* ************************************************************** *
* Define the input record. Only the first 12 characters are      *
* significant, the maximum length of a keyword name or object    *
* type with RDO.                                                 *
* ************************************************************** *
 Ø1  XREF-INPUT-REC.
     Ø5  XREF-TYPE-NAME              PIC X(12).
     Ø5  FILLER                      PIC X(68).
 FD  DATA-OUT
     RECORD CONTAINS 132 CHARACTERS
     BLOCK CONTAINS Ø RECORDS
     RECORDING MODE IS F
     LABEL RECORDS ARE OMITTED.
 Ø1  PRINT-REC                       PIC X(132).
 EJECT.
* ************************************************************** *
*  Define the flags and constants used by this program           *
* ************************************************************** *
 WORKING-STORAGE SECTION.
 77  CMD-PART-1                      PIC X(1ØØ).
 77  OBJECTS-OPTION                  PIC X(7).
     88  OBJECTS-NOT-SPECIFIED       VALUE IS SPACES.
 77  INPUT-FILE-FLAG                 PIC X.
     88  INPUT-FILE-OPEN             VALUE IS '1'.
 77  OUTPUT-FILE-FLAG                PIC X.
     88  OUTPUT-FILE-OPEN            VALUE IS '1'.
 77  FILE-OPEN                       PIC X VALUE '1'.
 77  RESOURCE-DEFINITION-PTR         PIC 999.
 77  INPUT-FILE-STATUS               PIC XX.
     88 END-OF-INPUT-FILE            VALUE IS '1Ø'.
     88 ERROR-ON-INPUT-FILE          VALUES ARE '11' THRU '99'.
 77  OUTPUT-FILE-STATUS              PIC XX.
     88 ERROR-ON-OUTPUT-FILE         VALUES ARE 'Ø1' THRU '99'.
 77  INSUFFICIENT-STORAGE-FLAG       PIC X VALUE IS 'Ø'.
     88  INSUFFICIENT-STORAGE        VALUE IS '1'.
     88  SUFFICIENT-STORAGE          VALUE IS 'Ø'.
 77  OBJECTS-OPTION-MISSING          PIC S9999 COMP VALUE IS 1.
 77  INPUT-FILE-OPEN-ERROR           PIC S9999 COMP VALUE IS 2.
 77  INPUT-FILE-CLOSE-ERROR          PIC S9999 COMP VALUE IS 3.
 77  INPUT-FILE-READ-ERROR           PIC S9999 COMP VALUE IS 4.
 77  OUTPUT-FILE-OPEN-ERROR          PIC S9999 COMP VALUE IS 5.
 77  OUTPUT-FILE-CLOSE-ERROR        PIC S9999 COMP VALUE IS 6.
 77  OUTPUT-FILE-WRITE-ERROR        PIC S9999 COMP VALUE IS 7.
 77  INSUFFICIENT-TYPE-ELEMENTS     PIC S9999 COMP VALUE IS 8.
 77  INSUFFICIENT-NAME-ELEMENTS     PIC S9999 COMP VALUE IS 9.
 77  INSUFFICIENT-DEFINED-ELEMENTS  PIC S9999 COMP VALUE IS 1Ø.
 77  INSUFFICIENT-USED-ELEMENTS     PIC S9999 COMP VALUE IS 11.
 77  SPACE-REQUIREMENTS             PIC X.
     88  NO-SPACE                   VALUE IS 'N'.
     88  SPACE-LINES                VALUE IS 'S'.
     88  PAGE-EJECT                 VALUE IS 'P'.

 EJECT.
* ************************************************************ *
*  D E F I N E  T H E  C R O S S - R E F E R E N C E  T A B L E *
*  The table consists of 4 different elements. They are:        *
*   1) Type Elements   -  These define the objects or keywords   *
*                         which are to be cross-referenced.      *
*                         The values are those read from the     *
*                         input file. Each type element has a    *
*                         chain of 'name' elements associated with
*                         it.                                    *
*   2) Name Elements   -  These define the different values found*
*                         on the CSD for a particular object or  *
*                         keyword name. Each name element has a  *
*                         chain of 'defined' elements and a chain*
*                         of 'used' elements associated with it. *
*   3) Defined Elements-  These define where an object of a given*
*                         name is defined (keywords cannot be    *
*                         defined). As no two objects in a given *
*                         group can have the same name, all that *
*                         it is necessary to record is the RDO   *
*                         group name.                            *
*   4) Used Elements   -  These define in which resource         *
*                         definitions a particular value of an   *
*                         object name or a keyword name is used. *
*                         The object type, name, and group must  *
*                         recorded to uniquely define a use.     *
* ************************************************************ *
 Ø1  TYPE-ELEMENT-POOL.
     Ø5  TYPE-ELEMENT            OCCURS 1Ø TIMES INDEXED BY
                                 NEXT-AVAILABLE-TYPE-ELEMENT
                                 TYPE-ELEMENT-PTR
                                 CURRENT-TYPE-ELEMENT-PTR.
         1Ø  TYPE-KEY.
             15  XREF-TYPE          PIC X(12).
         1Ø  NAME-CHAIN-START       USAGE IS INDEX.
         1Ø  NAME-CHAIN-COUNT       PIC 999.
 77  TYPE-ELEMENTS-ALLOCATED        PIC 99 VALUE IS Ø.
     88  TYPE-ELEMENT-FREE          VALUES ARE Ø THRU 9.
 77  TYPE-NUMBER                    PIC 99.
 77  TYPE-CHAIN-SEARCH-FLAG         PIC X VALUE 'Ø'.
```

```
    88 TYPE-ELEMENT-FOUND          VALUE IS '1'.
    88 TYPE-ELEMENT-NOT-FOUND      VALUE IS 'Ø'.
Ø1  NAME-ELEMENT-POOL.
    Ø5  NAME-ELEMENT            OCCURS 1ØØØ TIMES INDEXED BY
                                NEXT-AVAILABLE-NAME-ELEMENT
                                NAME-ELEMENT-PTR
                                CURRENT-NAME-ELEMENT-PTR
                                PREVIOUS-NAME-ELEMENT-PTR.
        1Ø  NAME-KEY.
            15  XREF-NAME          PIC X(44).
        1Ø  NEXT-NAME-PTR          USAGE IS INDEX.
        1Ø  DEFINED-CHAIN-START    USAGE IS INDEX.
        1Ø  DEFINED-CHAIN-COUNT    PIC 999.
        1Ø  USED-CHAIN-START       USAGE IS INDEX.
        1Ø  USED-CHAIN-COUNT       PIC 999.
77  NAME-ELEMENTS-ALLOCATED        PIC 9999 VALUE IS Ø.
    88  NAME-ELEMENT-FREE          VALUES ARE Ø THRU 999.
Ø1  DEFINED-ELEMENT-POOL.
    Ø5  DEFINED-ELEMENT         OCCURS 1ØØØ TIMES INDEXED BY
                                NEXT-AVAILABLE-DEFINED-ELEMENT
                                DEFINED-ELEMENT-PTR
                                CURRENT-DEFINED-ELEMENT-PTR
                                PREVIOUS-DEFINED-ELEMENT-PTR.
        1Ø  DEFINED-KEY.
            15  RDO-GROUP-NAME     PIC X(8).
        1Ø  NEXT-DEFINED-PTR       USAGE IS INDEX.
77  DEFINED-ELEMENTS-ALLOCATED     PIC 9999 VALUE IS Ø.
    88  DEFINED-ELEMENT-FREE       VALUES ARE Ø THRU 999.
Ø1  USED-ELEMENT-POOL.
    Ø5  USED-ELEMENT            OCCURS 1ØØØ TIMES INDEXED BY
                                NEXT-AVAILABLE-USED-ELEMENT
                                USED-ELEMENT-PTR
                                CURRENT-USED-ELEMENT-PTR
                                PREVIOUS-USED-ELEMENT-PTR.
        1Ø  USED-KEY.
            15  OBJECT-TYPE        PIC X(12).
            15  OBJECT-NAME        PIC X(8).
            15  RDO-GROUP-NAME     PIC X(8).
        1Ø  NEXT-USED-PTR          USAGE IS INDEX.
77  USED-ELEMENTS-ALLOCATED        PIC 9999 VALUE IS Ø.
    88  USED-ELEMENT-FREE          VALUES ARE Ø THRU 999.
EJECT.
* ************************************************************* *
* Define the temporary variables used in this program.          *
* ************************************************************* *
Ø1  EXIT-KEYWORD-VALUE-W.
    Ø5 KEYWORD-VALUE-CHAR          PIC X OCCURS 8Ø TIMES.
Ø1  WORK-TYPE.
    Ø5  OBJECT-TYPE                PIC X(12).
    Ø5  KEYWORD-NAME               REDEFINES
        OBJECT-TYPE                PIC X(12).
```

```
 Ø1  WORK-NAME.
     Ø5  KEYWORD-VALUE              PIC X(44).
     Ø5  OBJECT-NAME-DEFINITION    REDEFINES
         KEYWORD-VALUE.
         1Ø  OBJECT-NAME.
             15  OBJECT-NAME-VALUE  PIC X(12).
             15  FILLER             PIC X(32).
 77  MAX-KEYWORD-VALUE-LENGTH       PIC 9(2) VALUE 44.
 Ø1  WORK-DEFINED.
     Ø5  RDO-GROUP-NAME             PIC X(8).
 Ø1  WORK-USED.
     Ø5  OBJECT-TYPE                PIC X(12).
     Ø5  OBJECT-NAME                PIC X(8).
     Ø5  RDO-GROUP-NAME             PIC X(8).
 EJECT.
* ************************************************************ *
* Define the output record structures used by this program.      *
* ************************************************************ *
 Ø1  NEW-XREF-OBJECT-HEADER-LINE    PIC X(75).
 Ø1  CROSS-REFERENCE-OBJECT-LINE.
     Ø5  FILLER                     PIC X(8)
         VALUE IS '* * * * '.
     Ø5  FILLER                     PIC X(19)
         VALUE IS 'CROSS REFERENCE OF '.
     Ø5  TYPE-NAME                  PIC X(12).
     Ø5  FILLER                     PIC X(8)
         VALUE IS ' * * * *'.
 Ø1  EMPTY-NAME-CHAIN-LINE.
     Ø5  FILLER                     PIC X(33)
         VALUE IS 'THERE ARE NO DEFINITIONS/USES OF'.
     Ø5  EMPTY-NAME                 PIC X(12).
     Ø5  FILLER                     PIC X(3Ø)
         VALUE IS ' IN THE SPECIFIED GROUP/LISTS.'.
 Ø1  DEFINED-MSG-LINE.
     Ø5  NUMBER-TIMES-DEFINED       PIC ZZZ9.
     Ø5  FILLER                     PIC X(33)
         VALUE IS ' GROUP(S) CONTAIN DEFINITIONS OF '.
     Ø5  DEFINED-NAME               PIC X(12).
 Ø1  WHERE-DEFINED-LINE.
     Ø5  FILLER                     PIC X(14) VALUE SPACES.
     Ø5  RDO-GROUP-NAME             PIC X(8).
 Ø1  USED-MSG-LINE.
     Ø5  NUMBER-TIMES-USED          PIC ZZZ9.
     Ø5  FILLER                     PIC X(18)
         VALUE IS ' USES ARE MADE OF '.
     Ø5  USED-NAME                  PIC X(44).
 Ø1  WHERE-USED-LINE.
     Ø5  FILLER                     PIC X(14) VALUE SPACES.
     Ø5  OBJECT-TYPE                PIC X(12).
     Ø5  FILLER                     PIC X VALUE SPACE.
     Ø5  OBJECT-NAME                PIC X(8).
```

```
    Ø5  FILLER                     PIC X(4)
        VALUE IS ' IN '.
    Ø5  RDO-GROUP-NAME             PIC X(8).
Ø1  THEY-ARE-LINE.
    Ø5  FILLER                     PIC X(5) VALUE SPACES.
    Ø5  FILLER                     PIC X(9) VALUE 'THEY ARE:'.
EJECT.
* ************************************************************ *
* Define the linkage between this program and the CSD off-line   *
* utility program. The addressability of the values addressed by *
* the parameter list passed from DFHCSDUP is established         *
* automatically by the COBOL compiler so all we  need to define  *
* here are the actual formats of the values themselves.          *
* ************************************************************ *
LINKAGE SECTION.
Ø1  EXIT-FUNCTION-CODE      PIC 99 COMP.
      88 INITIAL-CALL         VALUE IS Ø.
      88 LIST-START-CALL      VALUE IS 2.
      88 GROUP-START-CALL     VALUE IS 4.
      88 OBJECT-START-CALL    VALUE IS 6.
      88 DETAIL-CALL          VALUE IS 8.
      88 OBJECT-END-CALL      VALUE IS 1Ø.
      88 GROUP-END-CALL       VALUE IS 12.
      88 LIST-END-CALL        VALUE IS 14.
      88 FINAL-CALL           VALUE IS 16.
Ø1  EXIT-WORK-AREA-PTR      POINTER.
Ø1  EXIT-BACK-TRANS-CMD-PTR POINTER.
Ø1  EXIT-LIST-NAME          PIC X(8).
Ø1  EXIT-GROUP-NAME         PIC X(8).
Ø1  EXIT-OBJECT-TYPE        PIC X(12).
Ø1  EXIT-OBJECT-NAME        PIC X(8).
Ø1  EXIT-KEYWORD-NAME       PIC X(12).
Ø1  EXIT-KEYWORD-LENGTH     PIC 999 COMP.
Ø1  EXIT-KEYWORD-VALUE.
    Ø3 EXIT-KEYWORD-CHAR    PIC X OCCURS 1 TO 183
                              DEPENDING ON EXIT-KEYWORD-LENGTH.
Ø1  BACK-TRANSLATED-COMMAND        PIC X(1ØØ).
EJECT.
* ************************************************************ *
*         M A I N L I N E  C O D E  S T A R T S  H E R E       *
* ************************************************************ *
PROCEDURE DIVISION USING EXIT-FUNCTION-CODE
                         EXIT-WORK-AREA-PTR
                         EXIT-BACK-TRANS-CMD-PTR
                         EXIT-LIST-NAME
                         EXIT-GROUP-NAME
                         EXIT-OBJECT-TYPE
                         EXIT-OBJECT-NAME
                         EXIT-KEYWORD-NAME
                         EXIT-KEYWORD-LENGTH
                         EXIT-KEYWORD-VALUE.
```

```
* ********************************************************** *
* Determine the type of call being made to this program and      *
* process it accordingly. The only types of call of any interest *
* in this instance are                                           *
*                     INITIAL call                               *
*                     OBJECT START calls                         *
*                     DETAIL calls                               *
*                     FINAL call                                 *
* All other call types are ignored.                              *
* ********************************************************** *
 MAIN-CONTROL.
     IF INITIAL-CALL THEN
       PERFORM PROCESS-INITIAL-CALL
     ELSE
          IF OBJECT-START-CALL THEN
            PERFORM PROCESS-OBJECT-START-CALL
          ELSE
               IF DETAIL-CALL THEN
                 PERFORM PROCESS-KEYWORD-DETAIL-CALL
               ELSE
                    IF FINAL-CALL THEN
                      PERFORM PROCESS-FINAL-CALL
                      PERFORM CHECK-RETURN-CODE
                      STOP RUN.
* ********************************************************** *
* If the value in the 'RETURN-CODE' special register at this     *
* point is non-zero then the current EXTRACT command will        *
* be terminated. Return must be made via a GOBACK in order to    *
* prevent STOP RUN processing.                                   *
* ********************************************************** *
     PERFORM CHECK-RETURN-CODE.
     GOBACK.
 EJECT.
* ********************************************************** *
*         P R O C E S S  A N I N I T I A L  C A L L              *
*  This procedure:                                               *
*      1) Checks specified command to see if OBJECTS option was  *
*         specified.
*      2) Controls the opening of the input and output files.    *
*      3) Initializes the pointers to the free element pools.    *
*      4) Controls the reading of the input records into type    *
*         elements of the cross-reference table.                 *
*      5) Saves a copy of the back-translated command issued,    *
*         which caused this program to be loaded and invoked.    *
* ********************************************************** *
 PROCESS-INITIAL-CALL.
     MOVE SPACES TO OBJECTS-OPTION.
     MOVE ZERO TO INPUT-FILE-FLAG.
     MOVE ZERO TO OUTPUT-FILE-FLAG.
     SET ADDRESS OF BACK-TRANSLATED-COMMAND TO
                    EXIT-BACK-TRANS-CMD-PTR.
     UNSTRING BACK-TRANSLATED-COMMAND DELIMITED BY 'OBJECTS'
              INTO CMD-PART-1 DELIMITER IN OBJECTS-OPTION.
     IF OBJECTS-NOT-SPECIFIED THEN
       MOVE OBJECTS-OPTION-MISSING TO RETURN-CODE
     ELSE
         PERFORM OPEN-FILES
         IF NOT ERROR-ON-INPUT-FILE AND
            NOT ERROR-ON-OUTPUT-FILE THEN
              SET NEXT-AVAILABLE-TYPE-ELEMENT TO 1
              SET NEXT-AVAILABLE-NAME-ELEMENT TO 1
              SET NEXT-AVAILABLE-DEFINED-ELEMENT TO 1
              SET NEXT-AVAILABLE-USED-ELEMENT TO 1
              PERFORM WITH TEST AFTER
                      UNTIL END-OF-INPUT-FILE   OR
                            ERROR-ON-INPUT-FILE OR
                            INSUFFICIENT-STORAGE
                      READ DATA-IN
                      IF NOT ERROR-ON-INPUT-FILE AND
                         NOT END-OF-INPUT-FILE   THEN
                           PERFORM ADD-TYPE-TO-TYPE-CHAIN
                      ELSE
                           IF ERROR-ON-INPUT-FILE THEN
                                MOVE INPUT-FILE-READ-ERROR TO
                                     RETURN-CODE
                           END-IF
                      END-IF
              END-PERFORM
              IF NOT ERROR-ON-INPUT-FILE THEN
                   SET ADDRESS OF BACK-TRANSLATED-COMMAND TO
                       EXIT-BACK-TRANS-CMD-PTR
                   MOVE SPACES TO NEW-XREF-OBJECT-HEADER-LINE
                   MOVE BACK-TRANSLATED-COMMAND TO
                       NEW-XREF-OBJECT-HEADER-LINE
              END-IF
         END-IF
     END-IF.
* ************************************************************* *
*                 O P E N   T H E   F I L E S                   *
*   This subroutine opens the input and output files to be used *
*   by this program. An open failure will result in the setting *
*   of the appropriate file status flag and consequently the    *
*   'RETURN-CODE' special register will be set to identify the  *
*   error.                                                      *
* ************************************************************* *
 OPEN-FILES.
     OPEN INPUT DATA-IN.
     IF ERROR-ON-INPUT-FILE THEN
          MOVE INPUT-FILE-OPEN-ERROR TO RETURN-CODE
     ELSE
          MOVE FILE-OPEN TO INPUT-FILE-FLAG
          OPEN OUTPUT DATA-OUT
          IF ERROR-ON-OUTPUT-FILE THEN
               MOVE OUTPUT-FILE-OPEN-ERROR TO RETURN-CODE
          ELSE
               MOVE FILE-OPEN TO OUTPUT-FILE-FLAG
          END-IF
     END-IF.
 EJECT.
* ************************************************************* *
*       A D D  A  T Y P E  T O  T H E  T Y P E  C H A I N       *
*  This procedure adds a new element to the type chain in the    *
*  order in which they occur in the input file XREFIN. As a new  *
*  type element is defined, the associated name chain is         *
*  initialized, dummy head and tail elements area are added so as*
*  to make the algorithm for adding a new name element much      *
*  easier.                                                       *
* ************************************************************* *
 ADD-TYPE-TO-TYPE-CHAIN.
     PERFORM ALLOCATE-A-TYPE-ELEMENT.
     IF SUFFICIENT-STORAGE THEN
          MOVE XREF-TYPE-NAME TO XREF-TYPE (TYPE-ELEMENT-PTR)
          MOVE ZEROES TO NAME-CHAIN-COUNT (TYPE-ELEMENT-PTR)
          PERFORM ALLOCATE-A-NAME-ELEMENT
     END-IF.
     IF SUFFICIENT-STORAGE THEN
          SET NAME-CHAIN-START (TYPE-ELEMENT-PTR) TO
              NAME-ELEMENT-PTR
*         * *************************************************** *
*         * Ensure no elements will need to be inserted in front *
*         * of the dummy 'head' element                          *
*         * *************************************************** *
          MOVE LOW-VALUES TO NAME-KEY (NAME-ELEMENT-PTR)
          SET PREVIOUS-NAME-ELEMENT-PTR TO
              NAME-ELEMENT-PTR
          PERFORM ALLOCATE-A-NAME-ELEMENT
     END-IF.
     IF SUFFICIENT-STORAGE THEN
*         * *************************************************** *
*         * Ensure no elements will need to be inserted after    *
*         * the dummy 'tail' element                             *
*         * *************************************************** *
          MOVE HIGH-VALUES TO
               NAME-KEY (NAME-ELEMENT-PTR)
          SET NEXT-NAME-PTR (PREVIOUS-NAME-ELEMENT-PTR)
              TO NAME-ELEMENT-PTR
     END-IF.
 EJECT.
* ************************************************************* *
*         P R O C E S S  A  O B J E C T  S T A R T  C A L L     *
*  This subroutine performs the following:                       *
*    1) Searches the 'type' chain to see if the new object is an *
*       object in which we are interested, ie one specified in   *
*       the input file XREFIN.                                *
*    2) If it is then:                                        *
*       a) Search the name chain associated with the appropriate *
*          type element to see if an object with the same name   *
*          has already been encountered. If not then a new name  *
*          element is added in collating sequence, and the       *
*          associated defined and used chains are initialized.   *
*       b) The definition of the name is then added to the       *
*          'defined' chain associated with the name.             *
* ************************************************************* *
 PROCESS-OBJECT-START-CALL.
     MOVE EXIT-OBJECT-TYPE TO OBJECT-TYPE OF WORK-TYPE.
     PERFORM FIND-TYPE-IN-CHAIN.
     IF TYPE-ELEMENT-FOUND THEN
           MOVE EXIT-OBJECT-NAME TO
                OBJECT-NAME OF WORK-NAME
           PERFORM FIND-NAME-IN-CHAIN
           IF SUFFICIENT-STORAGE THEN
                MOVE EXIT-GROUP-NAME TO
                     RDO-GROUP-NAME OF WORK-DEFINED
                PERFORM FIND-DEFINITION-IN-CHAIN
           END-IF
     END-IF.

 EJECT.
* ************************************************************* *
*      P R O C E S S  A  K E Y W O R D  D E T A I L  C A L L    *
*  This subroutine performs the following:                     *
*    1) Searches the 'type' chain to see if the keyword is one   *
*       in which we are interested, ie one specified in the input*
*       file XREFIN.                                            *
*    2) If it is then:                                          *
*       a) Search the name chain associated with the appropriate *
*          type element to see if a keyword or object with the   *
*          same name has already been encountered. If it is a    *
*          new name then a new element is added in collating     *
*          sequence, and the associated defined and used chains  *
*          initialized.                                         *
*       b) The use of the keyword is then added to the 'used'    *
*          chain associated with the name.                      *
* ************************************************************* *
 PROCESS-KEYWORD-DETAIL-CALL.
     MOVE EXIT-KEYWORD-NAME TO KEYWORD-NAME OF WORK-TYPE.
     PERFORM FIND-TYPE-IN-CHAIN.
     IF TYPE-ELEMENT-FOUND THEN
          MOVE EXIT-KEYWORD-VALUE TO KEYWORD-VALUE OF WORK-NAME
          PERFORM FIND-NAME-IN-CHAIN
          IF SUFFICIENT-STORAGE THEN
               MOVE EXIT-OBJECT-TYPE TO
                    OBJECT-TYPE OF WORK-USED
               MOVE EXIT-OBJECT-NAME TO
```

```
                   OBJECT-NAME OF WORK-USED
              MOVE EXIT-GROUP-NAME  TO
                   RDO-GROUP-NAME OF WORK-USED
              PERFORM FIND-USE-IN-CHAIN
         END-IF
    END-IF.
 EJECT.
* ************************************************************* *
*            P R O C E S S  A  F I N A L  C A L L               *
*  This subroutine controls the printing out of the contents of  *
*  the cross-reference table in the required format. It also     *
*  handles the closing of the input and output files.            *
* ************************************************************* *
 PROCESS-FINAL-CALL.
*    * ******************************************************** *
*    * Scan along the type chain and for each type element       *
*    * produce an analysis of the named occurrences of it, where *
*    * these occurrences are defined and in which resource       *
*    * definitions they are used.                                *
*    * ******************************************************** *
     SET TYPE-ELEMENT-PTR TO 1.
     PERFORM LIST-XREF-DETAILS-FOR-TYPE
             WITH TEST BEFORE
             VARYING TYPE-NUMBER FROM 1 BY 1
             UNTIL TYPE-NUMBER GREATER THAN
                   TYPE-ELEMENTS-ALLOCATED OR
                   ERROR-ON-OUTPUT-FILE.
     PERFORM CLOSE-FILES.
 EJECT.
* ************************************************************* *
*           C L O S E  T H E  F I L E S                         *
*   This subroutine closes the input and output files used by    *
*   this program. A close failure will result in the setting     *
*   of the appropriate file status flag and consequently the     *
*   'RETURN-CODE' special register will be set to identify the   *
*   error.                                                       *
* ************************************************************* *
 CLOSE-FILES.
     CLOSE DATA-IN.
     IF ERROR-ON-INPUT-FILE THEN
          MOVE INPUT-FILE-CLOSE-ERROR TO RETURN-CODE
     ELSE
          CLOSE DATA-OUT
          IF ERROR-ON-OUTPUT-FILE THEN
               MOVE OUTPUT-FILE-CLOSE-ERROR TO RETURN-CODE
          END-IF
     END-IF.
 EJECT.
* ************************************************************* *
*          C H E C K  R E T U R N  C O D E                      *
*  If return code is non-zero and any file is still open then    *
*  we must close that file to avoid any CØ3 abends.             *
*  Any errors on a CLOSE are ignored as we already have an error.*
* ************************************************************ *
 CHECK-RETURN-CODE.
     IF RETURN-CODE IS NOT ZERO AND INPUT-FILE-OPEN THEN
         CLOSE  DATA-IN.
     IF RETURN-CODE IS NOT ZERO AND OUTPUT-FILE-OPEN THEN
         CLOSE  DATA-OUT.
 EJECT.
* ************************************************************ *
*     L I S T  T H E  X R E F  T A B L E  C O N T E N T S      *
*  This subroutine is invoked once for each type element in the  *
*  type chain. Its function is to produce an analysis of the     *
*  definition and/or use of the objects and keywords specified   *
*  in the input file. Each report of a 'type' is started on a new*
*  page and the general format of each report is as follows:     *
*              < back translated command line>                   *
*              < type name line>                                 *
*          |~~ <number times name defined line>                  *
*          |           <where defined line>  ~~| once per def    *
* once per |           <where defined line>  __| of named type   *
* named    |                                                     *
*occurrence|   <number times name used>                          *
* of a type|           <where used>          ~~| once per use    *
*          |           <where used>          __| of named type   *
*          |__                                                   *
* ************************************************************ *
 LIST-XREF-DETAILS-FOR-TYPE.
     MOVE NEW-XREF-OBJECT-HEADER-LINE TO PRINT-REC.
     SET PAGE-EJECT TO TRUE.
     PERFORM WRITE-PRINT-REC.
     MOVE XREF-TYPE (TYPE-ELEMENT-PTR) TO
          TYPE-NAME OF CROSS-REFERENCE-OBJECT-LINE.
     MOVE CROSS-REFERENCE-OBJECT-LINE TO PRINT-REC.
     SET SPACE-LINES TO TRUE.
     PERFORM WRITE-PRINT-REC.
     SET NAME-ELEMENT-PTR TO
         NAME-CHAIN-START (TYPE-ELEMENT-PTR).
     SET NAME-ELEMENT-PTR TO
         NEXT-NAME-PTR (NAME-ELEMENT-PTR).
     IF NAME-CHAIN-COUNT (TYPE-ELEMENT-PTR) EQUAL TO ZEROES THEN
          MOVE XREF-TYPE (TYPE-ELEMENT-PTR) TO
               EMPTY-NAME OF EMPTY-NAME-CHAIN-LINE
          MOVE EMPTY-NAME-CHAIN-LINE TO PRINT-REC
          SET SPACE-LINES TO TRUE
          PERFORM WRITE-PRINT-REC
     ELSE
*    * ************************************************
*    * SCAN ALONG THE NAME CHAIN PRODUCING AN ANALYSIS OF   *
*    * THE DEFINITIONS AND USES OF THE NAME OCCURRENCES OF  *
*    * a type.                                              *
*    * **************************************************** *
          PERFORM WITH TEST BEFORE
                  UNTIL NAME-KEY (NAME-ELEMENT-PTR) EQUAL TO
                        HIGH-VALUES OR
                        ERROR-ON-OUTPUT-FILE
*              * ********************************************** *
*              * PRODUCE THE ANALYSIS OF THE DEFINITIONS OF A   *
*              * NAMED OCCURRENCE.                              *
*              * ********************************************** *
               IF DEFINED-CHAIN-COUNT (NAME-ELEMENT-PTR) > Ø THEN
                    MOVE NAME-KEY (NAME-ELEMENT-PTR) TO
                         DEFINED-NAME OF DEFINED-MSG-LINE
                    MOVE DEFINED-CHAIN-COUNT (NAME-ELEMENT-PTR) TO
                         NUMBER-TIMES-DEFINED OF DEFINED-MSG-LINE
                    MOVE DEFINED-MSG-LINE TO PRINT-REC
                    SET SPACE-LINES TO TRUE
                    PERFORM WRITE-PRINT-REC
                    MOVE THEY-ARE-LINE TO PRINT-REC
                    SET NO-SPACE TO TRUE
                    PERFORM WRITE-PRINT-REC
*                   * ****************************************** *
*                   * SET THE POINTER TO THE FIRST ELEMENT IN THE*
*                   * DEFINED CHAIN AFTER THE DUMMY HEAD ELEMENT *
*                   * ****************************************** *
                    SET DEFINED-ELEMENT-PTR TO
                        DEFINED-CHAIN-START (NAME-ELEMENT-PTR)
                    SET DEFINED-ELEMENT-PTR TO
                        NEXT-DEFINED-PTR (DEFINED-ELEMENT-PTR)
*                   * ****************************************** *
*                   * LIST THE DEFINITIONS OF A NAMED OCCURRENCE *
*                   * BY SCANNING THE DEFINED CHAIN ASSOCIATED   *
*                   * WITH THE CURRENT NAMED ELEMENT             *
*                   * ****************************************** *
                    PERFORM WITH TEST BEFORE
                            UNTIL DEFINED-KEY(DEFINED-ELEMENT-PTR)
                                  EQUAL TO HIGH-VALUES OR
                                  ERROR-ON-OUTPUT-FILE
                         MOVE DEFINED-KEY (DEFINED-ELEMENT-PTR) TO
                              RDO-GROUP-NAME OF WHERE-DEFINED-LINE
                         MOVE WHERE-DEFINED-LINE TO PRINT-REC
                         SET NO-SPACE TO TRUE
                         PERFORM WRITE-PRINT-REC
*                        * ************************************* *
*                        * SET THE PTR TO THE NEXT DEFINED       *
*                        * ELEMENT IN THE DEFINED CHAIN.         *
*                        * ************************************* *
                         SET DEFINED-ELEMENT-PTR TO
                            NEXT-DEFINED-PTR (DEFINED-ELEMENT-PTR)
                    END-PERFORM
               ELSE
                    MOVE SPACES TO PRINT-REC
```

```
                 SET SPACE-LINES TO TRUE
                 PERFORM WRITE-PRINT-REC
            END-IF
*           * ***************************************** *
*           * PRODUCE THE ANALYSIS OF THE USES OF A NAMED *
*           * OCCURRENCE                                  *
*           * ***************************************** *
            IF USED-CHAIN-COUNT (NAME-ELEMENT-PTR) > Ø AND
               NOT ERROR-ON-OUTPUT-FILE THEN
                 MOVE NAME-KEY (NAME-ELEMENT-PTR) TO
                      USED-NAME OF USED-MSG-LINE
                 MOVE USED-CHAIN-COUNT (NAME-ELEMENT-PTR) TO
                      NUMBER-TIMES-USED OF USED-MSG-LINE
                 MOVE USED-MSG-LINE TO PRINT-REC
                 SET NO-SPACE TO TRUE
                 PERFORM WRITE-PRINT-REC
                 MOVE THEY-ARE-LINE TO PRINT-REC
                 SET NO-SPACE TO TRUE
                 PERFORM WRITE-PRINT-REC
*                * *******************************************
*                * SET THE POINTER TO THE FIRST ELEMENT IN THE*
*                * USED CHAIN AFTER THE DUMMY HEAD ELEMENT.   *
*                * *******************************************
                 SET USED-ELEMENT-PTR TO
                     USED-CHAIN-START (NAME-ELEMENT-PTR)
                 SET USED-ELEMENT-PTR TO
                     NEXT-USED-PTR (USED-ELEMENT-PTR)
*                * ***************************************** *
*                * LIST THE USES OF A NAMED OCCURRENCE BY      *
*                * SCANNING THE USED CHAIN ASSOCIATED WITH     *
*                * THE CURRENT NAMED ELEMENT.                  *
*                * ***************************************** *
                 PERFORM WITH TEST BEFORE
                         UNTIL USED-KEY (USED-ELEMENT-PTR)
                               EQUAL TO HIGH-VALUES OR
                               ERROR-ON-OUTPUT-FILE
                      MOVE CORRESPONDING
                           USED-KEY (USED-ELEMENT-PTR)
                           TO WHERE-USED-LINE
                      SET NO-SPACE TO TRUE
                      MOVE WHERE-USED-LINE TO PRINT-REC
                      SET NO-SPACE TO TRUE
                      PERFORM WRITE-PRINT-REC
*                     * ************************************ *
*                     * SET THE PTR TO THE NEXT USED ELEMENT  *
*                     * USED CHAIN.                           *
*                     * ************************************ *
                      SET USED-ELEMENT-PTR TO
                          NEXT-USED-PTR (USED-ELEMENT-PTR)
                 END-PERFORM
            END-IF
```

```
*              * ********************************************** *
*              * SET PTR TO NEXT NAME ELEMENT IN THE NAME CHAIN  *
*              * ********************************************** *
               SET NAME-ELEMENT-PTR TO
                   NEXT-NAME-PTR (NAME-ELEMENT-PTR)
          END-PERFORM
     END-IF.
*    * ************************************************** *
*    * Set ptr to next type element in the type chain      *
*    * ************************************************** *
     SET TYPE-ELEMENT-PTR UP BY 1.
 EJECT.
* ************************************************************ *
*        W R I T E  O U T  A  P R I N T  R E C O R D           *
* This subroutine writes out the current contents of the output  *
* file buffer in the manner described by the current setting of  *
* the SPACE-REQUIREMENTS flag. If an I/O error is encountered    *
* then the RETURN-CODE special register is set.                  *
* ************************************************************ *
 WRITE-PRINT-REC.
     IF SPACE-LINES THEN
          WRITE PRINT-REC AFTER 3 LINES
     ELSE
          IF PAGE-EJECT THEN
               WRITE PRINT-REC AFTER PAGE
          ELSE
               WRITE PRINT-REC
          END-IF
     END-IF.
     IF ERROR-ON-OUTPUT-FILE THEN
           MOVE OUTPUT-FILE-WRITE-ERROR TO RETURN-CODE
     END-IF.
 EJECT.
* ************************************************************ *
*   F I N D  A  T Y P E  E L E M E N T  I N  I T S  C H A I N   *
*  This subroutine scans the type chain to see if a given type   *
*  (may be an object type or a keyword name) exists in the chain *
*  or not. On return the setting of TYPE-CHAIN-SEARCH-FLAG can   *
*  be used to determine the success of the search. If it is      *
*  equal to '1' (TRUE) the CURRENT-TYPE-ELEMENT-PTR addresses it.*
* ************************************************************ *
 FIND-TYPE-IN-CHAIN.
     MOVE ZERO  TO TYPE-CHAIN-SEARCH-FLAG.
     SET CURRENT-TYPE-ELEMENT-PTR TO 1.
*    * ******************************************************* *
*    * Scan the type chain for a given type. The search is       *
*    * terminated either when all elements have been examined or *
*    * when an element is found with the same name.              *
*    * ******************************************************* *
     PERFORM WITH TEST BEFORE
             VARYING TYPE-NUMBER FROM 1 BY 1
```

```
               UNTIL TYPE-NUMBER GREATER THAN
                     TYPE-ELEMENTS-ALLOCATED OR
                     WORK-TYPE IS EQUAL TO
                     TYPE-KEY (CURRENT-TYPE-ELEMENT-PTR)
            SET CURRENT-TYPE-ELEMENT-PTR UP BY 1
       END-PERFORM.
*      * ************************** *
*      * Was the search successful ? *
*      * ************************** *
       IF TYPE-NUMBER NOT GREATER THAN TYPE-ELEMENTS-ALLOCATED THEN
            SET TYPE-ELEMENT-FOUND TO TRUE
       END-IF.
 EJECT.
* ************************************************************ *
*   F I N D  A  N A M E  E L E M E N T  I N  I T S  C H A I N   *
*  This subroutine scans the name chain associated with the      *
*  currently addressed type element to see if the new named      *
*  occurrence of a type has already been recorded. If not then   *
*  a new 'name' element is added in collating sequence and       *
*  the 'defined' and 'used' chains associated with it are        *
*  initialized, ie the dummy head and tail elements are acquired *
*  and linked together with the name element.                    *
* ************************************************************ *
 FIND-NAME-IN-CHAIN.
       SET CURRENT-NAME-ELEMENT-PTR TO
           NAME-CHAIN-START (CURRENT-TYPE-ELEMENT-PTR)
*      * ******************************************************* *
*      * Scan the name chain for a particular name. The search is  *
*      * terminated  when a name with a value greater or equal     *
*      * to the specified one is found. This test will always      *
*      * be successful due to the dummy head element being set to  *
*      * HIGH values.                                              *
*      * ******************************************************* *
       PERFORM WITH TEST BEFORE
               UNTIL NAME-KEY (CURRENT-NAME-ELEMENT-PTR)
                     IS NOT LESS THAN WORK-NAME
            SET PREVIOUS-NAME-ELEMENT-PTR TO
                CURRENT-NAME-ELEMENT-PTR
            SET CURRENT-NAME-ELEMENT-PTR TO
                NEXT-NAME-PTR (CURRENT-NAME-ELEMENT-PTR)
       END-PERFORM.
*      * ******************************************************* *
*      * If the current element after the search does not have a   *
*      * value equal to the search argument then a new 'name'      *
*      * element must be added 'before' the current one.           *
*      * ******************************************************* *
       IF NAME-KEY (CURRENT-NAME-ELEMENT-PTR) IS GREATER THAN
          WORK-NAME THEN
            ADD 1 TO NAME-CHAIN-COUNT (CURRENT-TYPE-ELEMENT-PTR)
            PERFORM ALLOCATE-A-NAME-ELEMENT
            IF SUFFICIENT-STORAGE THEN
```

```
               MOVE WORK-NAME TO NAME-KEY (NAME-ELEMENT-PTR)
               MOVE ZEROES TO
                    DEFINED-CHAIN-COUNT (NAME-ELEMENT-PTR)
                    USED-CHAIN-COUNT (NAME-ELEMENT-PTR)
               SET NEXT-NAME-PTR (PREVIOUS-NAME-ELEMENT-PTR) TO
                   NAME-ELEMENT-PTR
               SET NEXT-NAME-PTR (NAME-ELEMENT-PTR) TO
                   CURRENT-NAME-ELEMENT-PTR
               SET CURRENT-NAME-ELEMENT-PTR TO NAME-ELEMENT-PTR
               PERFORM ALLOCATE-A-DEFINED-ELEMENT
          END-IF
*         * ************************************************** *
*         * If sufficient storage, initialize the head defined   *
*         * element to LOW values                                *
*         * ************************************************** *
          IF SUFFICIENT-STORAGE THEN
               SET DEFINED-CHAIN-START (CURRENT-NAME-ELEMENT-PTR)
                   TO DEFINED-ELEMENT-PTR
               MOVE LOW-VALUES TO
                    DEFINED-KEY (DEFINED-ELEMENT-PTR)
               SET PREVIOUS-DEFINED-ELEMENT-PTR TO
                   DEFINED-ELEMENT-PTR
               PERFORM ALLOCATE-A-DEFINED-ELEMENT
          END-IF
*         * ************************************************** *
*         * If sufficient storage, initialize the tail defined   *
*         * element to HIGH values. Set the next element pointer *
*         * in the head element to point to the tail and set the *
*         * pointer in the tail to NULL.                         *
*         * ************************************************** *
          IF SUFFICIENT-STORAGE THEN
               MOVE HIGH-VALUES TO
                    DEFINED-KEY (DEFINED-ELEMENT-PTR)
               SET NEXT-DEFINED-PTR (PREVIOUS-DEFINED-ELEMENT-PTR)789
                   TO DEFINED-ELEMENT-PTR
               PERFORM ALLOCATE-A-USED-ELEMENT
          END-IF
*         * ************************************************** *
*         * If sufficient storage, initialize the head used      *
*         * element to LOW values                                *
*         * ************************************************** *
          IF SUFFICIENT-STORAGE THEN
               SET USED-CHAIN-START (CURRENT-NAME-ELEMENT-PTR)
                   TO USED-ELEMENT-PTR
               MOVE LOW-VALUES TO USED-KEY (USED-ELEMENT-PTR)
               SET PREVIOUS-USED-ELEMENT-PTR TO
                   USED-ELEMENT-PTR
               PERFORM ALLOCATE-A-USED-ELEMENT
          END-IF
*         * ************************************************** *
*         * If sufficient storage, initialize the tail used      *
*         * element to HIGH values. Set the next element pointer *
*         * in the head element to point to the tail and set the *
*         * pointer in the tail to NULL.                         *
*         * ***************************************************** *
          IF SUFFICIENT-STORAGE THEN
               MOVE HIGH-VALUES TO
                    USED-KEY (USED-ELEMENT-PTR)
               SET NEXT-USED-PTR (PREVIOUS-USED-ELEMENT-PTR)
                   TO USED-ELEMENT-PTR
          END-IF
     END-IF.
 EJECT.
* ************************************************************* *
*          F I N D  A  D E F I N E D  E L E M E N T             *
*  This subroutine adds a new element to the 'defined' chain     *
*  at the correct place in the collating sequence.               *
* ************************************************************* *
 FIND-DEFINITION-IN-CHAIN.
     SET CURRENT-DEFINED-ELEMENT-PTR TO
         DEFINED-CHAIN-START (CURRENT-NAME-ELEMENT-PTR)
*    * ******************************************************** *
*    * Scan the name chain to determine the place in the         *
*    * collating sequence for the recording of a definition of   *
*    * a named type occurrence. The search is terminated when an *
*    * element is found with a value greater than the search     *
*    * argument. This search will always be successful due to the*
*    * dummy tail element being set to HIGH values.              *
*    * ******************************************************** *
     PERFORM WITH TEST BEFORE
             UNTIL DEFINED-KEY (CURRENT-DEFINED-ELEMENT-PTR)
                   IS NOT LESS THAN WORK-DEFINED
          SET PREVIOUS-DEFINED-ELEMENT-PTR TO
              CURRENT-DEFINED-ELEMENT-PTR
          SET CURRENT-DEFINED-ELEMENT-PTR TO
              NEXT-DEFINED-PTR (CURRENT-DEFINED-ELEMENT-PTR)
     END-PERFORM.
*    * ******************************************************** *
*    * This test should always be true as no two resources of    *
*    * the same type can have the same name in any one RDO group.*
*    * ******************************************************** *
     IF DEFINED-KEY (CURRENT-DEFINED-ELEMENT-PTR) IS GREATER THAN
        WORK-DEFINED THEN
          ADD 1 TO DEFINED-CHAIN-COUNT (CURRENT-NAME-ELEMENT-PTR)
          PERFORM ALLOCATE-A-DEFINED-ELEMENT
          IF SUFFICIENT-STORAGE THEN
*              * ***********************************************
*              * Link the new element into the defined chain by  *
*              * setting the 'next element' ptrs in the previous *
*              * element and the new element so that:            *
*              *       Previous-----> Current--->                *
*              *                 becomes                         *
*              *       Previous-----> New-------> Current----->  *
*              *                                                 *
*              * The new element then becomes the current element*
*              * ************************************************
               MOVE WORK-DEFINED TO
                    DEFINED-KEY (DEFINED-ELEMENT-PTR)
               SET NEXT-DEFINED-PTR (PREVIOUS-DEFINED-ELEMENT-PTR)
                   TO DEFINED-ELEMENT-PTR
               SET NEXT-DEFINED-PTR (DEFINED-ELEMENT-PTR) TO
                   CURRENT-DEFINED-ELEMENT-PTR
               SET CURRENT-DEFINED-ELEMENT-PTR TO
                   DEFINED-ELEMENT-PTR
          END-IF
     END-IF.
 EJECT.
* ************************************************************* *
*   F I N D  A  U S E D  E L E M E N T  I N  I T S  C H A I N   *
*  This subroutine adds a new element to the 'used' chain at    *
*  the correct place in the collating sequence.                 *
* ************************************************************* *
 FIND-USE-IN-CHAIN.
     SET CURRENT-USED-ELEMENT-PTR TO
         USED-CHAIN-START (CURRENT-NAME-ELEMENT-PTR)
*    * ******************************************************** *
*    * Scan the used chain to determine the place in the        *
*    * collating sequence for the recording of a use of a named  *
*    * type occurrence. The search is terminated when an element *
*    * is found with a value greater than the search argument.   *
*    * This search will always be successful due to the dummy    *
*    * tail element being set to HIGH values.                    *
*    * ******************************************************** *
     PERFORM WITH TEST BEFORE
             UNTIL USED-KEY (CURRENT-USED-ELEMENT-PTR)
                   IS NOT LESS THAN WORK-USED
          SET PREVIOUS-USED-ELEMENT-PTR TO
              CURRENT-USED-ELEMENT-PTR
          SET CURRENT-USED-ELEMENT-PTR TO
              NEXT-USED-PTR (CURRENT-USED-ELEMENT-PTR)
     END-PERFORM.
*    * ******************************************************** *
*    * This test should always be true as no two resources of    *
*    * the same type can have the same name in any one RDO group.*
*    * and so no combination of resource type, resource name, and*
*    * group should ever be the same.                            *
*    * ******************************************************** *
     IF USED-KEY (CURRENT-USED-ELEMENT-PTR) IS GREATER THAN
        WORK-USED THEN
          ADD 1 TO USED-CHAIN-COUNT (CURRENT-NAME-ELEMENT-PTR)
          PERFORM ALLOCATE-A-USED-ELEMENT
          IF SUFFICIENT-STORAGE THEN
*              * ************************************************
*              * LINK THE NEW ELEMENT INTO THE USED CHAIN BY     *
*              * setting the 'next element' ptrs in the previous *
*              * element and the new element so that:            *
*              *       Previous-----> Current--->                *
*              *                 becomes                         *
*              *       Previous-----> New-------> Current----->  *
*              * The new element then becomes the current element*
*              * ************************************************
               MOVE WORK-USED TO USED-KEY (USED-ELEMENT-PTR)
               SET NEXT-USED-PTR (PREVIOUS-USED-ELEMENT-PTR) TO
                   USED-ELEMENT-PTR
               SET NEXT-USED-PTR (USED-ELEMENT-PTR) TO
                   CURRENT-USED-ELEMENT-PTR
               SET CURRENT-USED-ELEMENT-PTR TO USED-ELEMENT-PTR
          END-IF
     END-IF.
 EJECT.
* ************************************************************* *
*         A L L O C A T E  A  T Y P E  E L E M E N T            *
*  This subroutine allocates the next available element of TYPE *
*  ELEMENT POOL, if one is available. If not the return code    *
*  special register is set. TYPE-ELEMENT-PTR points to the new  *
*  element on exit.                                             *
* ************************************************************* *
 ALLOCATE-A-TYPE-ELEMENT.
     IF NOT TYPE-ELEMENT-FREE THEN
          MOVE INSUFFICIENT-TYPE-ELEMENTS TO RETURN-CODE
          SET INSUFFICIENT-STORAGE TO TRUE
     ELSE
          SET TYPE-ELEMENT-PTR TO NEXT-AVAILABLE-TYPE-ELEMENT
*         * *************************************************** *
*         * Point to next available element in pool             *
*         * *************************************************** *
          SET NEXT-AVAILABLE-TYPE-ELEMENT UP BY 1
          ADD 1 TO TYPE-ELEMENTS-ALLOCATED
     END-IF.
 EJECT.
* ************************************************************* *
*         A L L O C A T E  A  N A M E  E L E M E N T            *
*  This subroutine allocates the next available element of NAME *
*  ELEMENT POOL, if one is available. If not the return code    *
*  special register is set. NAME-ELEMENT-PTR points to the new  *
*  element on exit.                                             *
* ************************************************************* *
 ALLOCATE-A-NAME-ELEMENT.
     IF NOT NAME-ELEMENT-FREE THEN
          MOVE INSUFFICIENT-NAME-ELEMENTS TO RETURN-CODE
          SET INSUFFICIENT-STORAGE TO TRUE
     ELSE
          SET NAME-ELEMENT-PTR TO NEXT-AVAILABLE-NAME-ELEMENT
*         * *************************************************** *
*         * Point to next available element in pool            *
*         * ************************************************** *
          SET NEXT-AVAILABLE-NAME-ELEMENT UP BY 1
          ADD 1 TO NAME-ELEMENTS-ALLOCATED
     END-IF.
 EJECT.
* ********************************************************** *
*      A L L O C A T E  A  D E F I N E D  E L E M E N T        *
*  This subroutine allocates the next available element of     *
*  DEFINED ELEMENT POOL, if one is available. If not the return *
*  code special register is set. DEFINED-ELEMENT-PTR points to   *
*  new element on exit.                                        *
* ********************************************************** *
 ALLOCATE-A-DEFINED-ELEMENT.
     IF NOT DEFINED-ELEMENT-FREE THEN
          MOVE INSUFFICIENT-DEFINED-ELEMENTS TO RETURN-CODE
          SET INSUFFICIENT-STORAGE TO TRUE
     ELSE
          SET DEFINED-ELEMENT-PTR TO
              NEXT-AVAILABLE-DEFINED-ELEMENT
*         * ************************************************** *
*         * Point to next available element in pool            *
*         * ************************************************** *
          SET NEXT-AVAILABLE-DEFINED-ELEMENT UP BY 1
          ADD 1 TO DEFINED-ELEMENTS-ALLOCATED
     END-IF.
 EJECT.
* ********************************************************** *
*         A L L O C A T E  A  U S E D  E L E M E N T           *
*  This subroutine allocates the next available element of USED *
*  ELEMENT POOL, if one is available. If not the return code    *
*  special register is set. USED-ELEMENT-PTR points to the new  *
*  element on exit.                                            *
* ********************************************************** *
 ALLOCATE-A-USED-ELEMENT.
     IF NOT USED-ELEMENT-FREE THEN
          MOVE INSUFFICIENT-USED-ELEMENTS TO RETURN-CODE
          SET INSUFFICIENT-STORAGE TO TRUE
     ELSE
          SET USED-ELEMENT-PTR TO NEXT-AVAILABLE-USED-ELEMENT
*         * ************************************************** *
*         * Point to next available element in pool            *
*         * ************************************************** *
          SET NEXT-AVAILABLE-USED-ELEMENT UP BY 1
          ADD 1 TO USED-ELEMENTS-ALLOCATED
     END-IF.
```

CSDXREF EXEC

```
/* REXX */
/* CICS CSD CROSS-REFERENCE utility
```

```
   C-List CSDXREF
   Called by tso user command.
   It executes:
the DFHØCRFC CICS utility program;
display cross reference table;                              */
Trace ?o
mess1 = ''
/*************************************************************/
/* Init proc and choose function                            */
/*************************************************************/
InitProc:
Do forever
ADDRESS ISPEXEC
'ADDPOP ROW(Ø) COLUMN(Ø)'
'VGET ZKEYS PROFILE'
ZWINTTL='CICS object Cross Reference'
'DISPLAY PANEL(cxcrpØØØ)'
if rc = 8 then exit
'REMPOP'
func1 = k
func2 = h
if func1 ¬= '' then Call Exec_Function1
if func2 ¬= '' then Call Exec_Function2
signal InitProc
Exec_Function1:
/*************************************************************/
/* EXEC CICS utility DFHØCRFC                               */
/*************************************************************/
func1 = ''
k = ''
'VPUT K PROFILE'
Do forever
ADDRESS ISPEXEC
'ADDPOP ROW(Ø) COLUMN(Ø)'
'VGET ZKEYS PROFILE'
ZWINTTL='CICS object Cross Reference: generate list objects'
mess1 = ''
'DISPLAY PANEL(cxcrpØØ1)'
if rc = 8 then return
'REMPOP'
/*************************************************************/
/* Alloc work file for utility job                          */
/*************************************************************/
jobdsn= userid()||'.CXCRFC.WORKJOB'
ADDRESS TSO
 xx = outtrap(trpØØ.,)
    address tso "delete '"jobdsn"'"
    "alloc da('"jobdsn"') dir(Ø) space(1,1) dsorg(ps)" ,
    "recfm(f,b) lrecl(8Ø) blksize(2792Ø) tracks ",
    "unit(worka) new catalog f(fjob)"
 xx = outtrap(off)
 if rc > Ø then do
    mess1 = 'Allocation WORKJOB failed.' ,
             ' Return Code 'rc
    Return
    end
/***************************************************************/
/* Define job CICS utility DFHØCRFC                           */
/***************************************************************/
 sk.1='//'userid()'# JOB (CICSØLT9),CLASS=S,MSGCLASS=X,          '
 sk.2='//   MSGLEVEL=(1,1),REGION=8M,NOTIFY=&SYSUID              '
 sk.3='//*---------------------------------------------          '
 sk.4='//DELETE EXEC  PGM=IDCAMS                                 '
 sk.5='//SYSPRINT DD  SYSOUT=*                                   '
 sk.6='//SYSIN    DD  *                                          '
 sk.7=' DELETE 'userid()'.CICSSVIL.DFHCSD.CRFC.OUT NONVSAM       '
 sk.8='/*'
 sk.9='/*'
sk.1Ø='//*'
sk.11='//DEFINE   EXEC PGM=IEFBR14                               '
sk.12='//SYSPRINT DD  SYSOUT=*                                   '
sk.13='//SYSOUT   DD  SYSOUT=*                                   '
sk.14='//CRFCOUT  DD  DISP=(NEW,CATLG,DELETE),                   '
sk.15='//             DSN='userid()'.CICSSVIL.DFHCSD.CRFC.OUT,   '
sk.16='//             UNIT=WORKA,                                '
sk.17='//             DCB=(DSORG=PS,BLKSIZE=133ØØ,RECFM=FB,LRECL=133),'
sk.18='//             SPACE=(CYL,(5,5))                          '
sk.19='//*                                                       '
sk.2Ø='//CSDUP   EXEC PGM=DFHCSDUP,REGION=4Ø96K,                 '
sk.21="//       PARM='CSD(READWRITE)'                            "
sk.22='//STEPLIB   DD DSN=USER.CICS.LOAD,DISP=SHR                '
sk.23='//          DD DSN=CICS.LIB.SDFHLOAD.USER,DISP=SHR        '
sk.24='//          DD DSN=CICS.LIB.SDFHLOAD,DISP=SHR             '
sk.25='//DFHCSD    DD DSN=CICS.SVIL.DFHCSD,DISP=SHR              '
sk.26='//CRFINPT   DD DSN='userid()'.CICSSVIL.DFHCSD.CRFC.INP,DISP=SHR'
sk.27='//CRFOUT    DD DSN='userid()'.CICSSVIL.DFHCSD.CRFC.OUT,DISP=SHR'
sk.28='//*RFOUT    DD SYSOUT=X                                   '
sk.29='//SYSOUT    DD SYSOUT=*                                   '
sk.3Ø='//SYSABOUT  DD SYSOUT=*                                   '
sk.31='//SYSPRINT  DD SYSOUT=*                                   '
sk.32='//SYSUDUMP  DD SYSOUT=*                                   '
sk.33='//SYSIN     DD *                                          '
sk.34=' EXTRACT GROUP('cxgrp') OBJECTS USERPROGRAM(DFHØCRFC)     '
sk.35='/*                                                        '
sk.36='/*                                                        '
sk.Ø=36
/***************************************************************/
/* Write job CICS utility DFHØCRFC                            */
/***************************************************************/
 "execio * diskw "fjob" (stem sk. finis"
 xx = outtrap(trpØ8.,)
```

```
    address tso "submit '"jobdsn"'"
 xx = outtrap(off)
 if rc > Ø then do
    mess1 = 'Submit function Generate failed.' ,
            ' Return Code 'rc
    end
           else do
    mess1 = 'Verify job Generate and then execute display function.'
    end
"free fi(fjob)"
 address tso "delete '"jobdsn"'"
Return
Exec_Function2:
func2 = ''
h = ''
'VPUT H PROFILE'
dsnrep = userid()||'.CICSSVIL.DFHCSD.CRFC.OUT'
ADDRESS TSO
  dd=OUTTRAP(dd.)
  "LISTDS '"dsnrep"' status"
  dd=OUTTRAP('OFF')
  zz = rc
  if zz =4 then do
                 typfunc = 'Noreport'
                 Call CRFC_EventLog
                 Return
                End
  if zz =8 then do
                 typfunc = 'Allocazione'
                 Call CRFC_EventLog
                 Return
                End
  if zz ¬= Ø & zz ¬= 4 & zz ¬= 8 then do
                 typfunc = 'Errore'
                 Call CRFC_EventLog
                 Return
                End
  dd=OUTTRAP(dd.)
  "ALLOC DA('"dsnrep"') F(CRFCRPT) SHR REUSE"
  dd=OUTTRAP('OFF')
   zz = rc
   if zz¬=Ø then do
                 typfunc = 'Allocazione'
                 Call CRFC_EventLog
                 Return
                End
            else do
                 ADDRESS TSO
                 dd=OUTTRAP(dd.)
                 "EXECIO * DISKR crfcrpt (STEM recr. FINIS"
                 dd=OUTTRAP('OFF')
```

```
                  zz = rc
                  if zz¬=Ø then do
                                 typfunc = 'Lettura'
                                 Call CRFC_EventLog
                                 Return
                                End
                 End
tabella = '@@CRFC@@'
ADDRESS ISPEXEC 'TBOPEN 'tabella' NOWRITE'
 if rc=8 then,
   ADDRESS ISPEXEC 'TBCREATE 'tabella ,
   'NAMES(GROUP TYPE CAMPO1) NOWRITE'
   group  = substr(recr.1,36,8)
   MESS1  = ''
   do i=1 to recr.Ø
      tobj  = substr(recr.i,2,8)
      hdr   = substr(recr.i,2,8)
      if tobj = '* * * *' then do
                                type = substr(recr.i,29,11)
                                campo1 = COPIES(' ',69)
                                ADDRESS ISPEXEC 'TBADD 'tabella
                                type = ''
                                iterate
                               end
      if hdr  = 'UTILITY' then do
                                iterate
                               end
      campo1 = substr(recr.i,2,7Ø)
      ADDRESS ISPEXEC 'TBADD 'tabella
   end
   ADDRESS ISPEXEC 'TBTOP 'tabella
   csrr=Ø
   DO forever
      ADDRESS ISPEXEC
     'ADDPOP ROW(Ø) COLUMN(Ø)'
     'VGET ZKEYS PROFILE'
      ZWINTTL='CICS object Cross Reference: display list objects'
      ADDRESS ISPEXEC 'TBDISPL 'tabella,
      ' PANEL(CXCRPØØ2)',
      ' CSRROW('csrr') AUTOSEL(NO)'
      if RC=8 then leave
      end
ADDRESS ISPEXEC 'TBDELETE 'tabella
ADDRESS ISPEXEC 'TBEND 'tabella
ADDRESS TSO
dd=OUTTRAP(dd.)
 "FREE F(CRFCRPT)"
dd=OUTTRAP('OFF')
Return
CRFC_EventLog:
Select
```

```
when typfunc = 'Allocazione' then
  mess1 = 'Allocation report failed. Advise system support.'
when typfunc = 'Lettura' then
  mess1 = 'Read report failed. Advise system support.'
when typfunc = 'Noreport' then
  mess1 = 'Report not available.'
when typfunc = 'Errore' then
  mess1 = 'Error on the report. Advise system support.'
Otherwise nop
End
ADDRESS TSO
dd=OUTTRAP(dd.)
 "FREE F(CRFCRPT)"
dd=OUTTRAP('OFF')
Return
```

ISPF PANELS

Panel CXCRP000

```
)ATTR
 %   TYPE(TEXT) INTENS(HIGH) SKIP(ON)
 }   TYPE(TEXT) HILITE(BLINK) COLOR(GREEN)
 !   TYPE(INPUT) CAPS(ON) JUST(LEFT) HILITE(REVERSE)
 {   TYPE(INPUT) CAPS(ON) JUST(LEFT) PAD(_)
 `   COLOR(turq) TYPE(TEXT)
 ?   COLOR(white) TYPE(TEXT)
 <   COLOR(blue) TYPE(TEXT) intens(high)
 >   COLOR(yellow) TYPE(TEXT) intens(high)
 @ TYPE(INPUT) INTENS(HIGH) PAD('_') CAPS(ON) hilite(reverse)
 # TYPE(INPUT) INTENS(HIGH) CAPS(ON) hilite(reverse) color(yellow)
 \ type(TEXT) intens(HIGH) COLOR(green) hilite(reverse)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
)BODY EXPAND(//) WINDOW(75 22) SMSG(VIDEO)
+$VIDEO
+
+                                        User.....: &ZUSER
<       *******+                         Date.....: &ZDAY &ZMONTH &ZYEAR
<     **+     *+                         Time.....: &ZTIME
<    **+      ?*******+                  Appl.....: CICS XREF
<   **+      ?**     *+
<  **+      ?**+     `********+
<  *      *?**+     `**     **+
<  *******?**+     `**     **+               \ SELECTION FUNCTION +
>  CICS+ ?*      *`********+
+        ?*******`**   **+
+        >Cross+`**     **+          Generate CICS Cross Reference.: #K+
+              `**       **+
+              >Reference+           Display CICS Cross Reference..: #H+
+
+
+ $mess1                                                                +
+
+
+ PF1=Help  PF3=Exit  ENTER=Continue
)INIT
 .CURSOR = K
 &ZCMD=' '
  VGET (K,H) PROFILE
  .HELP = CXCRPØØH
  &ZHTOP = CXCRPØØH
  &ZHINDEX = CXCRPØØH
)PROC
IF (&K = ' ' AND &H = ' ') .MSG = 'CXCRMØØØ'
IF (&K ¬= ' ' AND &H ¬= ' ') .MSG = 'CXCRMØØ1'
VPUT (K,H) PROFILE
)END
```

Panel CXCRP00H

```
)ATTR
  $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
  \ type(TEXT) intens(HIGH) COLOR(green)
  % type(TEXT) intens(HIGH) COLOR(white)
  # type(TEXT) intens(HIGH) COLOR(turq) HILITE(BLINK)
)BODY EXPAND(§§) WINDOW(62 22) SMSG(VIDEO)
+$VIDEO
\ **********************#H E L P\************************ +
+    With this procedure it's possible to carry out the     +
+    objects Cross Reference of CSD CICS group.             +
+                                                           +
+    The functions are:                                     +
+ -%Generate + CSD resource list with relative Cross        +
+              Reference;                                   +
+              EXEC CICS utility to produce output;         +
+                                                           +
+ -%Display  + Cross Reference table of the output of       +
+              generate option;                             +
+                                                           +
+                                                           +
+ PF3=Exit from Help                                        +
+                                                           +
)INIT
)PROC
)END
```

Panel CXCRP001

```
)ATTR
 `   COLOR(turq) TYPE(TEXT)
 ?   COLOR(white) TYPE(TEXT)
 <   COLOR(blue) TYPE(TEXT) intens(high)
 # TYPE(INPUT) INTENS(HIGH) CAPS(ON) hilite(reverse) color(yellow)
 \ type(TEXT) intens(HIGH) COLOR(green) hilite(reverse)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
)BODY EXPAND(//) WINDOW(68 18) SMSG(VIDEO)
+$VIDEO
+
<                                    User.....: &ZUSER
`  * * CICS CROSS REFERENCE * *     <Date.....: &ZDAY &ZMONTH &ZYEAR
?      Generate list objects        <Time.....: &ZTIME
<                                    Appl.....: CICS XREF
<
<
<
<   \ Input dataset + #crfcin                                 +
+
<   \ CICS group    + #cxgrp   +
+
<   \ CICS CSD      + #cxcsd                                  +
+
+
+ $mess1                                                      +
+ PF1=Help  PF3=Exit  ENTER=Continue
)INIT
 .CURSOR = crfcin
 &ZCMD=' '
  VGET (crfcin,cxgrp,cxcsd) PROFILE
  .HELP = CXCRPØ1H
  &ZHTOP = CXCRPØ1H
  &ZHINDEX = CXCRPØ1H
)PROC
IF (&crfcin = ' ' AND &cxgrp = ' ' AND &cxcsd = ' ') .MSG = 'CXCRMØØ2'
IF (&crfcin = ' ' AND &cxgrp ¬= ' ' AND &cxcsd ¬= ' ') .MSG = 'CXCRMØØ3'
IF (&crfcin ¬= ' ' AND &cxgrp ¬= ' ' AND &cxcsd = ' ') .MSG = 'CXCRMØØ4'
IF (&crfcin ¬= ' ' AND &cxgrp = ' ' AND &cxcsd ¬= ' ') .MSG = 'CXCRMØØ5'
IF (&crfcin ¬= ' ' AND &cxgrp = ' ' AND &cxcsd = ' ') .MSG = 'CXCRMØØ6'
IF (&crfcin = ' ' AND &cxgrp ¬= ' ' AND &cxcsd = ' ') .MSG = 'CXCRMØØ7'
IF (&crfcin = ' ' AND &cxgrp = ' ' AND &cxcsd ¬= ' ') .MSG = 'CXCRMØØ8'
VPUT (crfcin,cxgrp,cxcsd) PROFILE
)END
```

Panel CXCRP01H

```
)ATTR
  $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
  \ type(TEXT) intens(HIGH) COLOR(green)
  % type(TEXT) intens(HIGH) COLOR(white)
  # type(TEXT) intens(HIGH) COLOR(turq) HILITE(BLINK)
)BODY EXPAND(§§) WINDOW(62 22) SMSG(VIDEO)
+$VIDEO
```

```
\ **********************#H E L P\************************* +
+  Function of%Generate+CICS objects Cross Reference list    +
+                                                            +
+  To specify the following fields:                          +
+ -%Input        + it is the file with CSD resources and/or  +
+                  keywords to obtain the cross reference;   +
+                                                            +
+ -%Group        + it is the CSD group name;                 +
+                                                            +
+ -%CSD          + it is DFHCSD file name;                   +
+                                                            +
+                                                            +
+ PF3=Exit from Help                                         +
+                                                            +
)INIT
)PROC
)END
```

Panel CXCRP002

```
)ATTR
 @ TYPE(OUTPUT) INTENS(LOW)
 [ TYPE(OUTPUT) INTENS(high) color(green) hilite(reverse)
 # TYPE(OUTPUT) INTENS(high) color(red) hilite(blink)
 $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON) color(yellow)
 | TYPE(text) INTENS(HIGH) color(green)
)BODY
+
+ %Command ===>_ZCMD
+
+
%             CICS Cross Reference Objects - Group $group    +
+
+
+
+    #MESS1
+
+  ==================================================================
+
)MODEL
[type       @campo1
+
)INIT
  .CURSOR = ZCMD
 &ZCMD=' '
  .HELP = CXCRPØ2H
  &ZHTOP = CXCRPØ2H
  &ZHINDEX = CXCRPØ2H
)PROC
)END
```

Panel CXCRP02H

```
)ATTR
  $ TYPE(OUTPUT) INTENS(HIGH) SKIP(ON)
  \ type(TEXT) intens(HIGH) COLOR(green)
  % type(TEXT) intens(HIGH) COLOR(white)
  # type(TEXT) intens(HIGH) COLOR(turq) HILITE(BLINK)
)BODY EXPAND(§§) WINDOW(62 22) SMSG(VIDEO)
+$VIDEO
\ **********************#H E L P\************************* +
+    Function of%display+CICS Cross Reference.            +
+                                                          +
+    Read output of CICS utility (Generate function).      +
+                                                          +
+    The sequence of the cross reference is visualized     +
+    therefore as specified in the input file.             +
+                                                          +
+                                                          +
+ PF3=Exit from Help                                       +
+                                                          +
)INIT
)PROC
)END
```

ISPF MESSAGES

CXCRM00

```
CXCRMØØØ 'Selection function'                              .ALARM=YES
' To select one at least function.'
CXCRMØØ1 'Selection error'                                 .ALARM=YES
' To specify one single function.'
CXCRMØØ2 'CICS XRef Generate'                              .ALARM=YES
' To specify the fields demands.'
CXCRMØØ3 'CICS XRef Generate'                              .ALARM=YES
' To specify the input dataset with CSD resources/keywords.'
CXCRMØØ4 'CICS XRef Generate'                              .ALARM=YES
' To specify DFHCSD file.'
CXCRMØØ5 'CICS XRef Generate'                              .ALARM=YES
' To specify CICS GROUP name.'
CXCRMØØ6 'CICS XRef Generate'                              .ALARM=YES
' To specify DFHCSD file and the GROUP name.'
CXCRMØØ7 'CICS XRef Generate'                              .ALARM=YES
' To specify INPUT dataset and the DFHCSD file.'
CXCRMØØ8 'CICS XRef Generate'                              .ALARM=YES
' To specify INPUT dataset and the GROUP name.'
```

SAMPLE FILE TO INPUT TO DFH0CRFC PROGRAM (CRFINPT)

```
BROWSE    USER.CICSSVIL.DFHCSD.CRFC.INP       Line 00000000 Col 001 080
 Command ===>                                            Scroll ===> CSR
***************************** Top of Data ******************************
PROGRAM
TRANSACTION
TYPETERM
DSNAME
FILE
PROFILE
MAPSET
CONNECTION
LSRPOOL
SESSIONS
**************************** Bottom of Data ****************************
```

Espedito Morvillo
Systems Programmer (Italy)

CICS and LE/370 – a workshop

The following are brief notes designed to summarize LE/370 and how it works with CICS.

LE/370 is a common run-time library for all languages.

It is derived from the CODE/370-debugger (IBM-debugging tool) for batch and later for CICS.

LE/370 is available with OS/390.

New(?) terms to get used to include:

- Region = Address space.
- Process = 1 batch job in a CICS transaction.
- Enclave = Each main program creates an enclave, eg EXEC CICS LINK creates a new enclave.

CEEMSG-DCT replaces the PLIMSG-DCT.

Perhaps (eg with a defect register savearea) an abend ASRA will be set to an abend 4083.

Literature: Bookmanager CEEA205 and CEEA305 *Language Environment for OS/390&VM, Programmer's Guide and Reference*.

The abend of the program might look like:

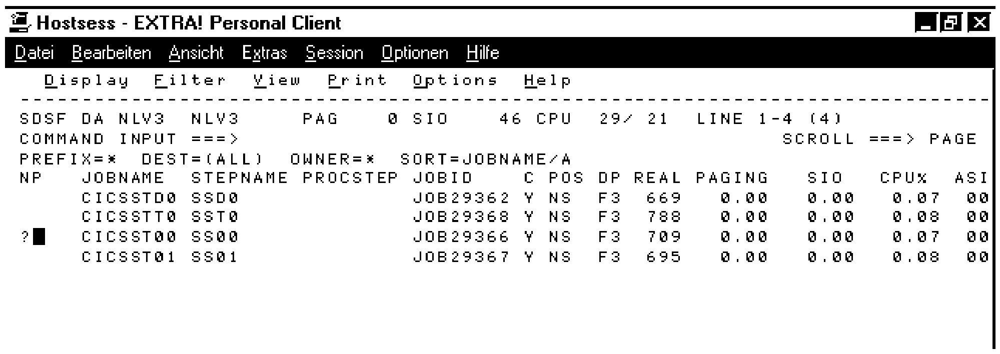
```
************************************************************************
************************************************************************
*****                                                              *****
*****  Welcome to the Computer Associates International            *****
*****            CA-InterTest Demo Session                         *****
*****                                                              *****
*****                                                              *****
*****  Before proceeding, please have on hand the                  *****
*****  guide which accompanies the Demo Session.                   *****
*****                                                              *****
*****                                                              *****
***** Please make sure that the program PL1DEMO is monitored by    *****
***** CA-InterTest. This program will abend if not being monitored.*****
*****                                                              *****
*****  To turn the monitor on, press CLEAR and follow the steps    *****
*****  outlined in the documentation.                              *****
*****                                                              *****
*****  If the monitor is already on, press ENTER to begin the      *****
*****  Basic Demo Session or PF2 to go to the Options Menu.        *****
*****                                                              *****
************************************************************************
************************************************************************
DFHAC2206 13:47:46 C41SS00 Transaction DEMP has failed with abend ASRA.
Resource backout was successful.
```

What happened? An abend 'ASRA': is that 0C1, 0C4, 0C7, etc?

Information about the abend of the program can be found from the following sources.

The CICSLOG, eg:

```
13.47.43 JOB29304  +DFHSR0001 C41SS00 An abend (code 0C7/AKEA) has
occurred at offset X'00001A74' in program PL1DEMO .
13.47.43 JOB29304  +DFHME0116 C41SS00
          (Module:DFHMEME) CICS symptom string for message DFHSR0001 is
          PIDS/565501800 LVLS/410 MS/DFHSR0001 RIDS/DFHSRP PTFS/UN94911
          AB/S00C7 AB/UAKEA RIDS/PL1DEMO ADRS/00001A74
```

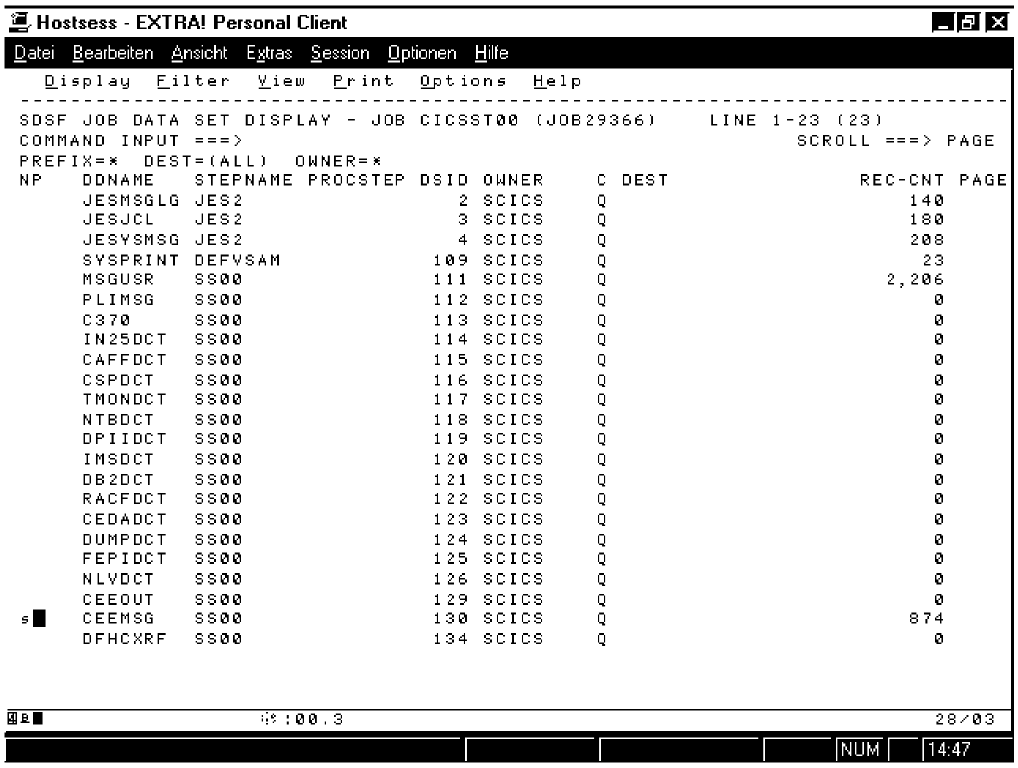

Hostsess - EXTRA! Personal Client
Datei Bearbeiten Ansicht Extras Session Optionen Hilfe
Display Filter View Print Options Help

```
SDSF DA NLV3  NLV3      PAG    0 SIO     46 CPU   29/ 21   LINE 1-4 (4)
COMMAND INPUT ===>                                          SCROLL ===> PAGE
PREFIX=*  DEST=(ALL)  OWNER=*  SORT=JOBNAME/A
NP   JOBNAME  STEPNAME PROCSTEP JOBID    C POS DP REAL PAGING   SIO   CPU%  ASI
     CICSSTD0 SSD0              JOB29362 Y NS  F3  669   0.00   0.00  0.07  00
     CICSSTT0 SST0              JOB29368 Y NS  F3  788   0.00   0.00  0.08  00
?█   CICSST00 SS00              JOB29366 Y NS  F3  709   0.00   0.00  0.07  00
     CICSST01 SS01              JOB29367 Y NS  F3  695   0.00   0.00  0.08  00
```

Figure 1: Example output

```
13.47.43 JOB29304  +DFHDU0205 C41SS00  A SYSTEM DUMP FOR DUMPCODE:
SR0001  , WAS SUPPRESSED BY THE DUMP TABLE OPTION
FOR THIS DUMPCODE
```

Browse CEEMSG-DCT. An example of what is seen is shown in Figure 1.


Hostsess - EXTRA! Personal Client
Datei Bearbeiten Ansicht Extras Session Optionen Hilfe
Display Filter View Print Options Help
NUM
14:47

```
SDSF JOB DATA SET DISPLAY - JOB CICSST00 (JOB29366)    LINE 1-23 (23)
COMMAND INPUT ===>                                          SCROLL ===> PAGE
PREFIX=*  DEST=(ALL)  OWNER=*
NP   DDNAME   STEPNAME PROCSTEP DSID OWNER    C DEST              REC-CNT PAGE
     JESMSGLG JES2                 2 SCICS    Q                       140
     JESJCL   JES2                 3 SCICS    Q                       180
     JESYSMSG JES2                 4 SCICS    Q                       208
     SYSPRINT DEFVSAM            109 SCICS    Q                        23
     MSGUSR   SS00               111 SCICS    Q                     2,206
     PLIMSG   SS00               112 SCICS    Q                         0
     C370     SS00               113 SCICS    Q                         0
     IN25DCT  SS00               114 SCICS    Q                         0
     CAFFDCT  SS00               115 SCICS    Q                         0
     CSPDCT   SS00               116 SCICS    Q                         0
     TMONDCT  SS00               117 SCICS    Q                         0
     NTBDCT   SS00               118 SCICS    Q                         0
     DPIIDCT  SS00               119 SCICS    Q                         0
     IMSDCT   SS00               120 SCICS    Q                         0
     DB2DCT   SS00               121 SCICS    Q                         0
     RACFDCT  SS00               122 SCICS    Q                         0
     CEDADCT  SS00               123 SCICS    Q                         0
     DUMPDCT  SS00               124 SCICS    Q                         0
     FEPIDCT  SS00               125 SCICS    Q                         0
     NLVDCT   SS00               126 SCICS    Q                         0
     CEEOUT   SS00               129 SCICS    Q                         0
s█   CEEMSG   SS00               130 SCICS    Q                       874
     DFHCXRF  SS00               134 SCICS    Q                         0

                 :00.3                                                28/03
```

Figure 2: Example output

Type a question-mark before the affected CICS in the SDSF.

The next page is displayed. From here ‘S’elect the DCT. This is illustrated in Figure 2.

The CEEMSG-DCT looks like:

```
 EJØBDEMP 1999Ø719134743 IBMØ537S ONCODE=8Ø97  Data exception
 EJØBDEMP 1999Ø719134743          From compile unit PL1DEMO at entry
point PL1DEMO at statement 94 at compile unit offset +ØØØØØ536 a
 EJØBDEMP 1999Ø719134743 CEE3DMP V1 R9.Ø: Condition processing resulted
in the unhandled condition.         19.Ø7.1999 13:47:43
 EJØBDEMP 1999Ø719134743
 EJØBDEMP 1999Ø719134743 Information for enclave PL1DEMO
 EJØBDEMP 1999Ø719134743
 EJØBDEMP 1999Ø719134743   Information for thread 8ØØØØØØØØØØØØØØØ
 EJØBDEMP 1999Ø719134743
 EJØBDEMP 1999Ø719134743   Traceback:
 EJØBDEMP 1999Ø719134743     DSA Addr  Program Unit  PU Addr   PU Offset
Entry         E Addr    E  Offset   Statement  Load Mod  Se
 EJØBDEMP 1999Ø719134743     ØØ2Ø75EØ  CEEHDSP       ØCEØ18DØ  +ØØØØ2526
CEEHDSP       ØCEØ18DØ  +ØØØØ2526                     UQ
 EJØBDEMP 1999Ø719134743     ØC9288EØ                ØØØØØØØØ  +ØØØØØØØØ
ØØØØØØØØ  +ØØØØØØØØ
 EJØBDEMP 1999Ø719134743     ØC9266FØ  PL1DEMO       ØCB96538  +ØØØØØ536
PL1DEMO       ØCB9654Ø  +ØØØØØ52E         94
 EJØBDEMP 1999Ø719134743     ØC9Ø431C  IBMRPMIA      ØØØA6AD8  +ØØØØØ4F6
IBMRPMIA      ØØØA6AD8  +ØØØØØ4F6
 EJØBDEMP 1999Ø719134743     ØC9Ø422C  CEEEVØ1Ø      ØCDB8EDØ  +ØØØØØ2D4
CEEEVØ1Ø      ØCDB8EDØ  +ØØØØØ2D4
 EJØBDEMP 1999Ø719134743     ØC9Ø4114  CEECRINV      ØCE3C768  +ØØØØØ424
CEECRINV      ØCE3C768  +ØØØØØ424
 EJØBDEMP 1999Ø719134743     ØC9Ø4Ø3C  CEECCICS      ØØØ33B9Ø  +ØØØØØ45E
CEECCICS      ØØØ33B9Ø  +ØØØØØ45E             CEECCICS  UQ
 EJØBDEMP 1999Ø719134743
_
```

The print job for the transaction dump looks like:

```
//B999999D JOB (ØØØØ,ØØØØØØØØ,ØØ,ØØØ,ØØØ),'MYNAME',NOTIFY=B999999,
//*      PRINT    CICS - DUMP
//             CLASS=H,MSGCLASS=X,REGION=ØM
/*JOBPARM TIME=1
//B999999E EXEC DFHDUMP,TASK=DEMP,CODE=ASRA,ID=SSØØ,DATE=992ØØ,
//             START='1Ø:3Ø:12',END='18:35:12',CICS=BOST
//*********************************************************
//*            VER=4,REL=1 --> CURRENT VERSION
//*            START='Ø7:ØØ:ØØ',END='18:ØØ:ØØ',CICS=PROD
//*            START='Ø7:ØØ:ØØ',END='18:ØØ:ØØ',CICS=VPRD
//*            START='Ø7:ØØ:ØØ',END='18:ØØ:ØØ',CICS=TEST
//*********************************************************
//*   Attention !!!                                       *
//*   Change ID if necessary  :  SPØØ,SP1Ø ... SVØØ ... STØØ  *
//*********************************************************
```

The specification can use:

- TASK = Name of the abended transaction.
- CODE = Abend code (eg ASRA).
- ID = SYSID from the CICS in which the abend occurred (eg SP00 for the PROD-CICS CICSPR00).
- DATE = Date from the abend in form of JJDDD.
- START/END = Range from the abend time.
- CICS = Name of the CICS-(MRO-)environment in which the abend occurred.

You can also use the IBM-standard utility!

The DSA in the transaction dump looks like:

```
 C41SSØØ    --- CICS TRANSACTION DUMP ---  CODE=ASRA   TRAN=DEMP   ID=1/
ØØØ2    DATE=99/Ø7/19 TIME=13:47:46  PAGE  147
 TRANSACTION STORAGE-USER31               ADDRESS ØC9266DØ TO ØC9288BF
LENGTH ØØØØ21FØ
 ØØØØØØØØ   E4FØFØFØ FØFØF5F5 E2E3D2E4 ØC9288C8  ØC9Ø4Ø24 ØØØØ21EØ
ØØØØØØØØ ØØØØØØØØ *UØØØØØ55STKU.khH.. .............* ØC9266DØ
 ØØØØØØ2Ø   8Ø25ØØØØ ØC9Ø431C ØØØØØØØØ 8CB96A74  ØØØØØØØØ ØC9287B8
ØC928758 8CB967BC *....................kg..kg.....* ØC9266FØ
 ØØØØØØ4Ø   ØCB95198 ØC9287ØØ ØØØØØØØØ ØØ2ØØØDØ  ØC9267BC ØC9267BC
ØC9267F8 ØC9267F8 *...q.kg..........k...k...k.8.k.8* ØC92671Ø
 ØØØØØØ6Ø   ØC9267BC ØØ2Ø4B38 ØØ2Ø688Ø ØC9287B8  ØC9287B8 91EØ91EØ
ØØØØØØØØ ØØØØØØØØ *.k...........kg..kg.j.j.........* ØC92673Ø
 ØØØØØØ8Ø   ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ  ØØØØØØØØ ØØØØØ2ØØ
ØØØØØØØØ ØØØØØØØØ *................................* ØC92675Ø
 ØØØØØØAØ   ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ  ØØØØØØØØ ØØØØØØØØ
ØØØØØØØØ ØØØØØØØØ *................................* ØC92677Ø
 ØØØØØØCØ   ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ  ØØØØØØØØ ØØØØØØØØ
ØC9276FØ ØC9286FØ *........................k.Ø.kfØ* ØC92679Ø
 ØØØØØØEØ   ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ ØCB95ØC8  ØØØØØØØØ ØØ1BØØØØ
ØØ35ØØ35 ØØ3ØØØ29 *...............H................* ØC9267BØ
 ØØØØØ1ØØ   ØØ1CØØ1B ØØØ1ØØ24 ØØØØØØØØ ØØØØFØFØ  D7D9D6E3 C8D3C64Ø
D7D9D6E3 C3D7C64Ø *..............ØØPROTHLF PROTCPF * ØC9267DØ
 ØØØØØ12Ø   C3D5E3D3 4Ø4Ø4Ø4Ø C4C5D4D7 C5D1FØC2  C4C1E3C1 E2C5E34Ø
D5D6E34Ø D6D7C5D5 *CNTL    DEMPEJØBDATASET NOT OPEN* ØC9267FØ
```

```
 ØØØØØ14Ø   4Ø6Ø4ØØØ ØØØØØØØØ ØØØØC4C1 E3C1E2C5  E34ØC9C4 4ØC5D9D9
D6D9406Ø 4ØØØØØØØ * - .......DATASET ID ERROR - ...* ØC92681Ø
 ØØØØØ16Ø   ØØØØØØØØ 4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø  4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø
4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø *....                            * ØC92683Ø
 ØØØØØ18Ø   4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø  4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø
4Ø4Ø4Ø4Ø 4Ø4Ø4Ø4Ø *                                *
```

Claus Reis
CICS Systems Programmer
Nuernberger Lebensversicherung AG (Germany)

Creating or modifying BMS sources

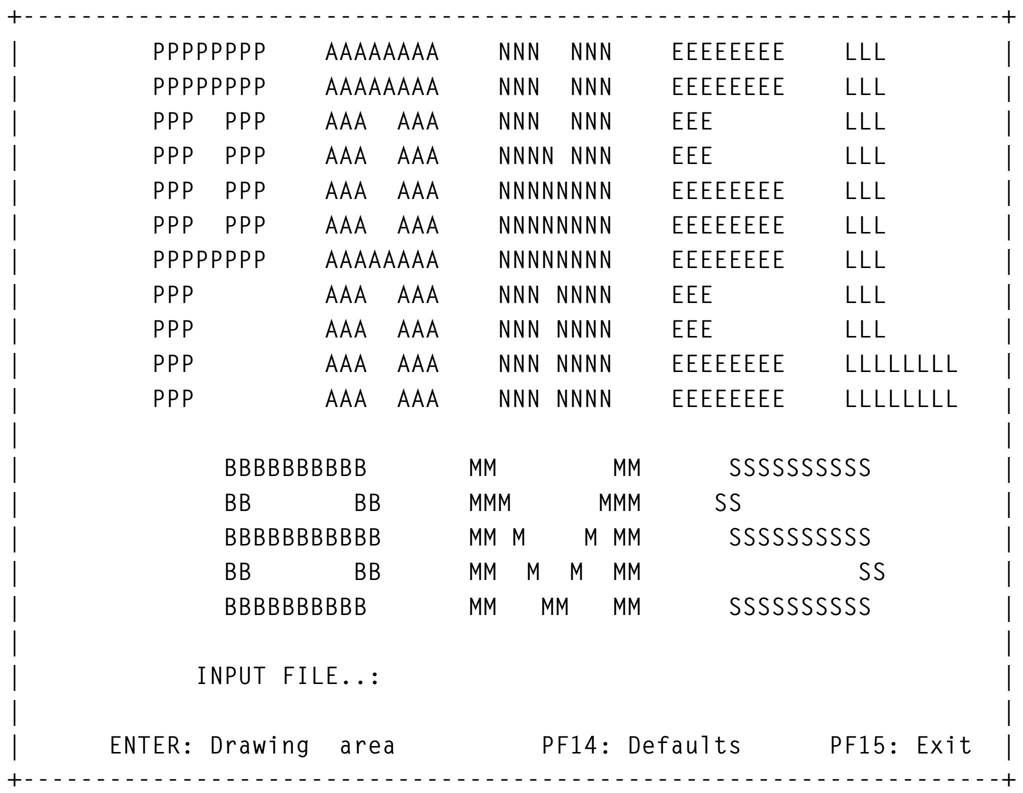
PANELBMS is a utility that was designed to facilitate the process of creating or modifying BMS sources and the associated copybooks. It allows you to draw a map on screen and to define the attributes, colours, and names of each field. It also automatically creates ‘stop fields’ whenever necessary at the end of data fields. It generates a BMS source ready to be assembled. It also creates a copybook in a way that is fully compatible with what IBM generates, but which is more elegant and easier to read (and modify, if necessary).

HOW TO WORK WITH PANELBMS

If you want to create a new BMS from scratch, simply invoke PANELBMS. If you want to work over an existing BMS source, then pass the full filename as an argument. When you finish your work, you can save it with the same or a different name, or simply cancel the operation. In either case, after calling PANELBMS, you will get the screen shown in Figure 1.

In this example, I did not pass any parameters, so the input file area is empty. If you did, the input filename would appear there. You can also type it at this stage, if you want to work over an existing BMS. Hit *Enter*, you will go to the next screen – shown in Figure 2. This screen has two parts. The upper part consists of a scale on the first line, followed by 15 lines where you can draw your map. The lower part contains the field attributes. There are two modes of operation here. When you enter this screen for the first time, you are in drawing mode. This means that you can freely move on the upper part of the screen to create your fields there. At this stage, the “Attributes of the current field” area in the lower left part of the screen is invisible. On the lower right side, there are symbols to initiate fields. These symbols are defined in my source code member named PANELTAB. However, you can change them here on the screen any time you like.

The symbols are used to define fields. In this example, the first field is a text field, so it starts with the ‘Field text’ symbol, followed by the actual text. Then, in the same line, there is a protected field without any initial text. To define this field, start with the ‘Field protect’



```
+---------------------------------------------------------------------+
|         PPPPPPPP    AAAAAAAA    NNN  NNN    EEEEEEEE    LLL         |
|         PPPPPPPP    AAAAAAAA    NNN  NNN    EEEEEEEE    LLL         |
|         PPP  PPP    AAA  AAA    NNN  NNN    EEE         LLL         |
|         PPP  PPP    AAA  AAA    NNNN NNN    EEE         LLL         |
|         PPP  PPP    AAA  AAA    NNNNNNNN    EEEEEEEE    LLL         |
|         PPP  PPP    AAA  AAA    NNNNNNNN    EEEEEEEE    LLL         |
|         PPPPPPPP    AAAAAAAA    NNNNNNNN    EEEEEEEE    LLL         |
|         PPP         AAA  AAA    NNN NNNN    EEE         LLL         |
|         PPP         AAA  AAA    NNN NNNN    EEE         LLL         |
|         PPP         AAA  AAA    NNN NNNN    EEEEEEEE    LLLLLLLL    |
|         PPP         AAA  AAA    NNN NNNN    EEEEEEEE    LLLLLLLL    |
|                                                                     |
|              BBBBBBBBBB       MM        MM      SSSSSSSSSS          |
|              BB       BB      MMM      MMM     SS                   |
|              BBBBBBBBBBB      MM M    M MM      SSSSSSSSSS          |
|              BB       BB      MM  M  M  MM               SS         |
|              BBBBBBBBBB       MM   MM   MM      SSSSSSSSSS          |
|                                                                     |
|            INPUT FILE..:                                            |
|                                                                     |
|      ENTER: Drawing  area          PF14: Defaults      PF15: Exit   |
+---------------------------------------------------------------------+
```

Figure 1: The initial screen

symbol, followed by the 'data fields filler' to define the area occupied by the field.

In the other lines there are more examples of text fields and data fields, unprotected and unprotected numeric. Since I only display 15 lines at a time in the drawing area, you must use PF8 and PF7 to display lines 1 to 15 or lines 10 to 24. The 'separation line' tells you, on the right side, what lines are being displayed. Also in that line, you can choose to have all your text automatically translated to uppercase or not. In this example, I have mixed case, which means, for example, that the text 'Address' on the screen remains as it is, otherwise it would be uppercased the first time you hit *Enter*.

As I explained above, the default field symbols that appear on the lower right side of the screen are accessible whenever you are in drawing mode, and you can change them if you need to. For example, the numeric field is defined by a question mark, but if you suddenly decide that you need to create text containing a question mark, you

```
+--------------------------------------------------------------------------------+
| 1...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....|
|                                                                                |
|    #THIS IS A TEST MAP                                         %++++++++++     |
|                                                                                |
|                                                                                |
|                                                                                |
|                                                                                |
|         #Name........:&+++++++++++++++++++++++++++++++++++++++++               |
|                                                                                |
|         #Address.....:&+++++++++++++++++++++++++++++++++++++++++               |
|                                                                                |
|         #Telephone...:?++++++++++++++                                          |
|                                                                                |
|                                                                                |
|                                                                                |
|                                                                                |
| ===== Case (Mixed,Up): M ========================= Display lines  1 to  15  ===|
| --- Attributes of the current field ----          Data fields Filler.....: +  |
| Name : DATA01    MDT on........: Y (Y, N)         Field Start: Numeric   : ?  |
| Line.: 02        Bright, Dark..: B (B, D)                     Unprotect : &  |
| Col..: 65        Color.........: R (G,Y,R,T,B,P,W)            Protect   : %  |
| Leng :           Extend Hilite.:   (Blink Rev Uline)          Text      : #  |
|                                                                                |
| F4:Draw area F5:Attrib prev field F6:Attrib next field F7:Top F8:Bot  F3:Exit  |
+--------------------------------------------------------------------------------+
```

Figure 2: The drawing area

must change the field symbol for any other character of your choice that is not used anywhere, for example by a '/'. If you make this type of change, two things happen. Firstly, all your current fields are scanned to see if the new symbol already exists somewhere. If it does, you get an error message (error messages appear at the first line of the screen, overlaying the scale) saying that the symbol already exists so you cannot use it. If it does not exist, then the old symbol is automatically changed to the new across all fields.

When you have finished drawing the screen, you must move to the second phase, to give names and attributes to the fields. For this, press PF6 (next field), the lower left side of the screen appears and the cursor moves there. For all data fields, you must type a name. Text fields have no name. The other attributes (Fset, Bright, Dark, colors, extended hilight) are all optional. Press F6 to circulate sequentially through all

```
+---------------------------------------------------------------------+
|         PPPPPPPP    AAAAAAAA    NNN  NNN    EEEEEEEE    LLL         |
|         PPPPPPPP    AAAAAAAA    NNN  NNN    EEEEEEEE    LLL         |
|         PPP  PPP    AAA  AAA    NNN  NNN    EEE         LLL         |
|         PPP  PPP    AAA  AAA    NNNN NNN    EEE         LLL         |
|         PPP  PPP    AAA  AAA    NNNNNNNN    EEEEEEEE    LLL         |
|         PPP  PPP    AAA  AAA    NNNNNNNN    EEEEEEEE    LLL         |
|         PPPPPPPP    AAAAAAAA    NNNNNNNN    EEEEEEEE    LLL         |
|         PPP         AAA  AAA    NNN NNNN    EEE         LLL         |
|         PPP         AAA  AAA    NNN NNNN    EEE         LLL         |
|         PPP         AAA  AAA    NNN NNNN    EEEEEEEE    LLLLLLLL    |
|         PPP         AAA  AAA    NNN NNNN    EEEEEEEE    LLLLLLLL    |
|                                                                     |
|              BBBBBBBBBB       MM        MM      SSSSSSSSSS          |
|              BB       BB      MMM      MMM     SS                   |
|              BBBBBBBBBBB      MM M    M MM      SSSSSSSSSS          |
|              BB       BB      MM  M  M  MM               SS         |
|              BBBBBBBBBB       MM   MM   MM      SSSSSSSSSS          |
|                                                                     |
|  OUTPUT  FILES                                                      |
|      BMS SOURCE: MY.BMS.SOURCES(TESTEØØ1)                           |
|      COBOL COPY: MY.COPY.SOURCES(TESTEØØ1)                          |
|      ENTER: SAVE                                PF3/15: CANCEL      |
+-------------------------------------------------------------------- +
```

Figure 3: The final screen

the fields, or F5 to circulate backwards. In this stage, the drawing area of the screen is not accessible. If you want to go back to it to change something, press PF4 and you are there again.

In my example, the current field is called DATA01, and is situated on line 2 column 65, the MDT is Yes (or FSET), Bright, and colour Red. To know what the current field is, you must look at the line and column indicators.

The length indicator is not accessible by the cursor and is merely for information. It works only when you load an already existing BMS. For a newly-created BMS, as in this example, I do not fill the length indicator.

When you are finished, press PF3 to go to the last screen, shown in Figure 3. Here you can decide where to save the BMS source and the associated copy. The output files must be existing 80-byte sequential or PDS files. For PDS files, you can specify either a new member or an existing one to be overwritten. If you decide to cancel your work and do not save anything, just leave with PF3.

HOW TO INSTALL PANELBMS

This utility is made up of the following components:

- PANELBMS is a REXX EXEC that starts the whole thing. It receives optionally as an argument the filename of an existing BMS, allocates two temporary output files (one for BMS, the other for COPY) for the COBOL programs to write on, then it calls PANELB0 module and, upon return, it copies from the temporary files to the defined locations. I do not allow COBOL to write directly to the real output files. You must modify the beginning of the EXEC to reflect the loadlib containing the PANELB0 module.
- PANELB0, the main COBOL program, is responsible for allocating the input file and the two ‘real’ output files. For this operation, it calls the TSO module IKJEFTSR, which normally resides at SYS1.LPALIB or equivalent. It is also responsible for displaying the start and ending screens, and for calling programs PANELB1, PANELB3, and PANELB4. If there is an input BMS, it also calls PANELB5.
- PANELB1 is the drawing program. It is responsible for creating an internal field table from your inputs. If you request uppercase translation, it also calls PANELB2.
- PANELB2 is an Assembler program that translates a string to uppercase.
- PANELB3 reads the internal field table created by PANELB1 and writes out the BMS source equivalent to a temporary file.
- PANELB4 does the same thing as PANELB3, only for the COBOL copybook.
- PANELB5 is called only when there is an input BMS. It reads the BMS source and creates the internal field table to be used by PANELB1.
- PANATRIB, PANELTAB, PANELZ0, and PANELZ1 are copybooks used by the above programs.

In addition to this you also need another program that is not included here, because it was already published in *MVS Update* in January 1997 under the title *A 3270 full screen utility* and is now freely downloadable from the Xephon Web site. I am talking about the PDISP Assembler program. It is a general-purpose routine I wrote several years ago that handles 3270 datastreams under TSO. All screens in PANELBMS are 3270 datastreams. Since I use reverse video, your terminal (or more likely your emulator) must be capable of handling it. As far as I have seen, all PC emulators do, so you should have no problem with it. Only very old 'green' real terminals may have trouble dealing with reverse video.

SOME PRACTICAL NOTES

PANELBMS is not meant to cover all the possibilities and options that BMS has. However, it deals with the most commonly used, and should be able to handle most needs. In particular, it does not handle multiple DFHMDI macros, nor things like PICIN or PICOUT. Also, it only supports 24 by 80 screen sizes, and field lengths up to 79 bytes. Apart from that, it creates a BMS ready to be assembled, with all the fields, positions, lengths, initial texts, and attributes, which you can always modify by hand if you need to. Do not forget that I automatically create 'stop bytes', fields with only one byte that are used to mark the end of another field. So if you used to create them by hand, just forget about them – they will be there in the BMS source.

The colour 'white', in BMS terms, is called 'neutral'. You may notice that for defining the colours of a field in my drawing screen, I use the initial letter of each colour. Following that logic, white is specified with a 'W'. If you want the colour to be 'default', just leave this field blank.

The copybook I create is more elegant and easy to read than IBM's. You may notice that instead of COBOL levels 01 02 03 I chose levels 01 05 07. This is because on many occasions a copy is part of a greater structure (like a commarea, for example), so you only need to modify the 01 to something else, instead of modifying all levels.

Editor's note: the code for this article will be published next month.

Luis Paulo Figueiredo Sousa Ribeiro
Edinfor (Portugal)

CICS news

Landmark Systems is shipping its TMON for Unix System Services monitor, which is designed to make it possible to change parameters and abort processes without leaving the monitoring console.

It identifies problems, bottlenecks, and availability issues in OS/390 USS resources and enables immediate action to be taken to correct them.

From one workstation, users can access CICS, DB2, IMS, MVS, TCP/IP, or VTAM data to monitor performance.

It monitors key activity components such as global I/O buffers, HFS data sets, processes, threads, and TCP/IP stacks. An exception monitor automates the problem detection process by creating alerts whenever pre-defined problems occur.

For further information contact:
Landmark Systems, 12700 Sunrise Valley Drive, Reston, VA 20191-5804, USA.
Tel: (703) 464 1300.
URL: http://www.landmark.com/products/tmonuss.shtml.

* * *

Candle has announced that the company will provide day-one solution support for CICS Transaction Server for z/OS Version 2 (CICS TS V2) and DB2 Universal Database Server for OS/390 and z/OS, Version 7.

The company also reaffirmed its commitment to IBM's workload software pricing structure for its CICS, DB2, and IMS solutions.

Candle's day-one support for z/OS Version 1.1 includes OMEGAMON II availability and performance monitors for MVS, CICS, DB2, IMS, DBCTL, SMS, and VTAM, OMEGAVIEW for 3270, OMEGAVIEW II for the Enterprise, OMEGACENTER Gateway, AF/OPERATOR, and AF/REMOTE.

Day-one support for CICS TS V2 DB2 UDB Server for OS/390 and z/OS is covered by OMEGAMON II for CICS and DB2 and CCC for CICS and DB2plex, respectively.

For further information contact:
Candle, 201 N Douglas St, El Segundo, CA 90245, USA.
Tel: (310) 535 3600.
URL: http://www.candle.com/news_events/press_releases/mainframe/dayone_zOS_033001.html.

* * *

IBM has announced Version 1.1 of its WebSphere Studio Asset Analyzer for z/OS, described as a development tool for application understanding, impact analysis, and connector builder assistance.

It can be used to Web-enable S/390 CICS and batch assets, and takes advantage of existing Web and COBOL skills.

It helps understand the effects when a software change is made to enterprise systems. It can find the source code that's affected and assesses the scope of a change from JCL to transaction to tables, and provides impact analysis information to plan changes, make changes, and trace through the effects of changes.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/ad/enterprise.