70.2.1 PL/I Coding Guideline

ADM Services, ADM Technical Support

Date issued: November 1, 2000 (Wed.)

Contents

Chapter 1

Preface

1.1 Title

Standard No. 70.2.1

Japanese Title PL/I Coding Guideline

English Title PL/I Coding Guideline

1.2 Outline

To set a coding standard for the programs developed and maintained by ADM Services.

1.3 Purpose

To make the PL/I program easier to understand and to improve its quality.

1.4 Scope

Applies to the PL/I-coded programs (batch and J-AAS Screen Handler type) created and maintained by ADM Services.

1.5 Owner

ADM Services

1.6 Issuer

ADM Services ADM Technical Support

1.7 Deviation from this Standard

The issuer must be contacted and notified.

1.8 Related Standards

ADM Services Standard 50.5.1 "Naming Procedures"
ADM Services Standard 70.2.2 "Standard Micro Subroutines Usage Procedure"

1.9 Classification

Guideline

1.10 Revision history

Table 1.1: Revision history

Revision date	Revision content
May 1, 1987	
November 1, 2000	Total revision

1.11 History

March 1987: Subcontracted from the Technology Promotion Department and a draft was made by the Productivity Committee's Standards Subdivision in 1987. This draft was issued after review by the related departments (all managers in the Information Systems Department and Technical Support Planning).

Chapter 2

Content

2.1 Program structure

The basic elements of programs such as DCLs and EXECS should be in the following sequence:

Batch

1		Program Headings (Compile List Headings)
	XXXXXX:	PROC (–) OPTIONS(MAIN);
2		Program Identification (Overall comments)
3		DCLs of OS files
4		Parameter DCLs
5		DCL (INCB) for the input region
6		DCL for Report Headings
7		Other DCLs
8		%INCLUDE SYSLIB2(SMONERR1) ;
9		ON Condition
10		MAIN PROCESS
11		SUBROUTINES
12		Standard Error Processing
	END	XXXXXX;

1. Program Headings (Compile List Headings)

   - The first row of the source program should be written in the following format:

     /*Program No. : Program Name Security class */

     This statement is displayed on all pages of the Compile List.
   - One of the following Security Classes should be used as instructed:

- IBM CONFIDENTIAL
- IBM CONFIDENTIAL RESTRICTED
- REGISTERED IBM CONFIDENTIAL

For a SENSITIVE program, the following should be stated after the Security Class as follows:

Example: IBM CONFIDENTIAL(SENSITIVE)

2. Comments should be specified as follows:

list 1: Sample batch program

```
1/* ST10SAMP : SAMPLE PROGRAM 1                          security class  */
0ST10SAMP : PROC(PARM) OPTIONS(MAIN);
0/*-------------------------------------------------------------------*/
 /*   PROJECT NO.                 ST10     <--project number          */
 /*   PROGRAM NO.                 ST10SAMP <--program number          */
 /*   PROGRAM NAME                SAMPLE PROGRAM 1                    */
 /*   SENSITIVE                   YES or NO                           */
 /*   PROGRAMMED BY                programmer name                     */
 /*   WITTEN DATE                 yy/mm/dd                            */
 /*-------------------- FUNCTION NARRATIVE -----------------------------*/
 /*                                                                    */
 /*   MAINTENANCE OF DOCUMENT FILE                                     */
 /*      1) GET DROPPED JOBGROUP INFORMATION FROM SCHEDULE FILE        */
 /*      2) DELETE DOCUMENTS OF DROPPED JOBGROUP FROM DOCUMENT FILE    */
 /*      3) MOVE DOCUMENTS OF DROPPED JOBGROUP FROM DOCUMENT FILE      */
 /*         UNLOAD FILE TO HISTORY FILE                                */
 /*      4) NAME LOG REPORT IF PARAMETER REQUESTS THE REPORT          */
 /*                                                                    */
 /*-------------------- INPUT FILE DESCRIPTION -------------------------*/
 /*   (DDNAME)           (CRDNAME)         (FILE DESCRIPTION)          */
 /*    SSTMS              JRSTMS0           MONTHLY SCHEDULE FILE      */
 /*    SSTDE              JRSTDE0           DOCUMENT FILE              */
 /*    SSTUN              JRSTUN0           DOCUMENT UNLOAD FILE       */
 /*-------------------- UPDATE FILE DESCRIPTION ------------------------*/
 /*   (DDNAME)           (CRDNAME)         (FILE DESCRIPTION)          */
 /*    SSTHI              JRSTHI0           DOCUMENT HISTORY FILE      */
 /*-------------------- OUTPUT FILE DESCRIPTION ------------------------*/
 /*   (DDNAME)           (CRDNAME)         (FILE DESCRIPTION)          */
 /*    SSTLG                                LOG REPORT                 */
 /*-------------------- CALLING MODULE INFORMATION ---------------------*/
 /*   (MODULE)                             (MODULE DESCRIPTION)        */
 /*    SC02U22            EXTERNAL          ENQ/DEQ SUBROUTINE         */
 /*    SC02U06            EXTERNAL        PDS DIRECTORY READ SUBROUTINE*/
 /*-------------------- PARAMETER/COMMON AREA DCL ----------------------*/
 /*    (PTR/LABEL)        (CRDNAME)         (DESCRIPTION)              */
 /*     PARM                                 EXEC PARAM(LISTING OPTION)*/
 /*-------------------- EXIT -------------------------------------------*/
```

```
/*     EXIT NORMAL: Normal end condition   */
/*     EXIT ERROR: Abend condition and the error code, EXIT name (see below)*/
/*-------------------- CHANGE ACTIVITY --------------------------------*/
/*     Revision history (see below)                                     */
/*---------------------------------------------------------------------*/
```

- A sample EXIT ERROR comment

list 2: A sample EXIT ERROR comment

```
ERROR CONDITION                  ERROR CODE            EXIT(Label)
1. INPUT RECORD (A FILE) = 0      RETURN CODE = 16      SMONERR(ABEND)
2. JOB CODE TABLE OVERFLOW        RETURN CODE = 20      SMONERR(ABEND)
```

- How to write a revision history
  Search Key Programmer Revision Date Reason Content of change
  xxxxxxx Exxxxx yy/mm/dd xxxxxxx.....
  - Search key
    A key that identifies the changed part of a program. Usually, a planning request number or a CPMA number is used. A search key is written in the changed portion part TSO/ISPF.
  - Programmer
    The employee number and name of the programmer who made the change.
  - Change date
    The day the program was changed in yy/mm/dd.
  - Reason for the change, and its content
    Should be described in detail.

3. DCLs for the OS file

   Refer to the section on "DCLs for the file" in this guide.

4. DCL (receiving parameters in the EXEC)

   - Except for the PCBs, the parameters are received as follows and made into DCLs.

Table 2.1: PARM

1	2	3	4	5	6	7	8	9	10	11	12
		A		B			C				

list 3: Sample DCL parameters

```
SAMPLE : PROC(PARAM)OPTIONS(MAIN);
         -
DCL   PARAM    CHAR(12)
      A        CHAR(2)        DEF PARAM POS(3),
      B        CHAR(3)        DEF PARAM POS(5),
      C        CHAR(5)        DEF PARAM POS(8);
      -
```

5. Input region DCL

   The layout stored in the CRD(Common Record Description) is used and coded as follows:

   list 4: Sample: DCL for the input region

```
DCL     1     Structure Name      STATIC,
          INCB UNDEF,N,CRD Name, , ,Field Name1, Field Name 2,....;
                                    ****************************
                                  Specify only the field name you are going to use.
                                  You cannot write a comment in this row.
```

   Notes:

   (a) The last three digits of the CRD Name are used for the Structure Name.

   (b) Refer to ADM Standard 70.2.2 "Procedure for Using Standard Macros" on how to use the INCB macro.

6. DCL for report headings

   - The report heading can be made by reading the DCL or an external file. We will explain how to make it by using DCL.

     To make a heading, first make one row of a constant (132 bytes for a printer and 80 bytes for the usual GRIs) into a DCL and enter it one row at a time, using the PUT command.

     list 5: Sample: DCL for Report Heading - DCL 1

```
DCL  1 HEAD1               STATIC,         /* TITLE LINE 1 */
       2 H1                CHAR(20) INIT('                    '),
       2 H2                CHAR(20) INIT('SALES LIST          '),
       2 H3                CHAR(20) INIT(' '),
       2 H4                CHAR(20) INIT(' '),
       HEAD1X              CHAR(80) DEF HEAD1 POS(1);
```

```
DCL  1 HEAD2              STATIC,        /* TITLE LINE 2 */
      2 H1                CHAR(20) INIT(' '),
      2 H2                CHAR(20) INIT(' '),
      2 H3                CHAR(20) INIT(' '),
      2 H4                CHAR(10) INIT('AS OF DATE'),
      2 ASOF              PIC'99/99/99',
      2 H5                CHAR(2),
     HEAD2X               CHAR(80) DEF HEAD2 POS(1);

DCL  1 HEAD3              STATIC,         /* ITEM HEADING */
      2 H1                CHAR(20) INIT('  BO SALES# POINT   '),
      2 H2                CHAR(20) INIT('QTY SALES  PANK TOT '),
      2 H3                CHAR(20) INIT('AL CUST# CUSTOMER N '),
      2 H4                CHAR(20) INIT('AME                 '),
      HEAD1X              CHAR(80) DEF HEAD3 POS(1);
```

- The Heading should be PUT within the ON condition.

list 6: Sample of Report Heading DCL 2

```
ON ENDPAGE(LIST1) BEGIN;
   page counter=page counter+1;
   IF page counter=1 THEN PUT FILE(LIST1) EDIT(HEAD1X) (A); (Note)
                     ELSE PUT FILE(LIST1) EDIT(HEAD1X) (A)PAGE;
   PUT FILE(LIST1) EDIT(HEAD2X) (A);
   PUT FILE(LIST1) EDIT(HEADX3) (A);
   -
   -
   END;
```

Note: If this rule is not obeyed, a blank page is generated on the first page and the report put on the GRI would be hard to use.

- The first heading should be printed by the SIGNAL ENDPAGE(Filename).
- When the GRI is being used, the report should have 58 rows, since the blank (unnecessary) pages will not be printed when output to a remote printer. Specify the number of rows in a report as follows:
  OPEN FILE(Filename) PAGESIZE(58);

7. Other DCLs

- The DCLs should be written in the following sequence:
  (a) ENTRY Name DCL
      - The program-unique subroutine name

- ADM Service Standard subroutine name
- BUILTIN FUNCTION

(b) The DCL of a constant

(c) The DCL of a work area

8. %INCLUDE SYSLIB2(SMONERR1);

   - A standard error-processing program

     For details on how to use the macro, refer to ADM Services Standard 70.2.2, "Standard Macro Usage Guide."

9. The subroutines should come after the MAIN PROCESS.

10. A coding for standard error processing

    - The following coding should come at the end of the program, and the final program processing should pass here.

      list 7: Standard error processing sample

```
%INCLUDE SYSLIB2(SMHEAD3);
%INCLUDE SYSLIB2(SMCHECK5);
%XTWOWEY=3;
 SMEND7(END);
```

2.2 Coding format

1. Program page/line control

   - Use column 1 for control.
   - Specify SOURCE MARGIN=(2,72,1) as the default. Do not change this default value.
     - ' ' SKIP(1)
     - 0 SKIP(2)
     - - SKIP(3)
     - + no skip
     - 1 Page break
   - Place a blank (0) before a cluster of DCLs or a processing break, to make the program easier to read.
   - Place a one (1) before a subroutine or before a large block within a procedure to continue to the next page.
   - Use column 1 rather than U%PAGE or %SKIP since it does not interrupt the source code.

2. Coding start column

   - In principle, the start column should be fixed.
   - DCL

Table 2.2: DCL

Column	2	7	25	50	72
	DCL	Identifier Name	Attribute	/*	*/

   - Execution statement

Table 2.3: Execution statement

Column	2	5	50	72
	Label	A=B	/*	*/

3. Comments

   - Comments should be written in logical units.
   - In principle, every DCL should contain a comment.
   - The IF statement should contain a comment.
   - In principle, the comment should be in English.
   - If a comment cannot be written on one line, it can overflow to the next line.
   - A full line comment should be placed before a subroutine or a large block.
   - The END; should show to which portion it corresponds to.
   - The comment should be logically described.

4. Each row should contain only one statement.

5. Indenting and nesting

   - The IF THEN ELSE nests and the DO nests should be indented.
   - How to write the DO and END statements

list 8: DO and END statement sample 1

```
DO I=1 TO 10;
    DO J=1 TO 10;
        X(I,J) = Y(I,J);
    END;
END;
```

list 9: DO and END statement sample 2

```
IF A=B
    THEN DO;
         X=0;
         Y=1;
    END;
    ELSE DO;
         X=1;
         Y=0;
    END;
```

- The use of the IF statement is discouraged. Use SELECT instead.

list 10: SELECT sample (corresponding to sample 2 above)

```
SELECT (A);
    WHEN (B) DO;
             X=0;
             Y=1;
             END;
    OTHER    DO;
             X=1;
             Y=0;
             END;
END;
```

- The DO, IF, and the procedure nests should each be less than level 5.
- Nests over level 3 or DO and SELECT statements over 20 rows should be labeled, and each should contain a corresponding END statement.

list 11: DO nest sample (with label)

```
DO1 : DO I=1 TO 10;
      -
      -
      DO2 DO J=1 TO 20;
          -
          -
          END DO2;
      -
      -
      END DO1;
```

list 12: SELECT sample (with label)

```
SEL1 : SELECT(-);
         WHEN(-)
         -
         -
         -
         -
         -
     END SEL1;
```

2.3 Variables and attributes

1. Naming
   - Use a full name in English, or its abbreviation..
   - Use an easy name.
   - Less than 10 characters.
   - The underscore (_) can be used, but not excessively.
2. Attribute
   - In principle, do not use FLOAT.
   - In principle, do not use a label variable.
   - In principle, the calculated data should be the following:
     - FIXED BIN(15,0)
     - FIXED BIN(31,0)
     - FIXED DEC(n,m)
   - The single-character variable I-N should be used as a DO control variable in the form FIXED BIN(15,0).
   - The attribute should be explicitly declared (however, the AUTO, REAL, and INT defaults can be omitted).
   - The variable DCLs, in general
     - All variables should be explicitly declared by use of DCL.
     - Variables unique to the subroutine should be declared (by use of DCL) within that subroutine (using INTRNAL).
     - In principle, only the "Full Qualify" variables should be used.

2.4 Structured programming

1. Do not use GO TO.

   - Except in the following situations:
     - When exiting from a DO loop (or LEAVE).
     - When going to the end of a module.
     - When going from the ON condition area to a end of a module.
     - When the IF THEN ELSE nesting becomes too complex.
     - When the programming becomes complex.
   - The input, processing, and output should be done in the following pattern:

     list 13: Input, processing, output pattern sample

```
-
-
ON ENDFILE(IN) EOFIN='1'B;
-
-
READ FILE(IN) INTO(INAREA);      -- Read the first record.
DO WHILE(EOFIN);
   input count=input count + 1
   -
   Processing
   -
   -
   -
   READ FILE(IN) INTO(INAREA);   -- Read the succeeding records
END;
```

   - When setting an EOF flag using ON ENDFILE (filename)
     When the program is not at EOF, use 0.
     When the program is at EOF, use 1.

2. SP statement

   - SELECT statement
     Use the SELECT statement if the IF statement opens the same item.

     list 14: SP statement sample (IF statement)

```
IF  OPE='*' THEN C=A+B;
ELSE
IF  OPE='-' THEN C=A-B;
ELSE
```

```
IF  OPE='*' THEN C=A*B;
ELSE
IF  OPE='/' THEN C=A/B;
ELSE DO;
    MESSAGE='OPERATION CODE ERROR';
    PUT EDIT(MESSAGE) (A);
END;
```

IF statements above should be re-written with SELECT statement as follows.

list 15: SP statement sample (SELECT statement)

```
SELECT(OPE);
    WHEN('+') C=A+B;
    WHEN('-') C=A-B;
    WHEN('*') C=A*B;
    WHEN('/') C=A/B;
    OTHER   DO;
                MESSAGE='OPERATION CODE ERROR'
                PUT EDIT(MESSAGE) (A);
            END;
END;
```

The SELECT statement executes only one function of multiple functions; which one, depends on the condition.

list 16: Format

```
SELECT(condition);
    WHEN(value)     action-1;
    WHEN(value)     action-2;
    -
    -
    OTHER           action-n;
    END;
```

Explanation: The item you can write after executing one action for which condition=value, can be as follows:

- A single statement
- DO group
- IF statement
- BEGIN block

– ON statement
– SELECT statement

The condition for each case should be clearly shown.

list 17: Sample when the OPE is either +, -, *, or /.

```
SELECT(OPE);
    WHEN('+') C=A+B;
    WHEN('-') C=A-B;
    WHEN('*') C=A*B;
    OTHER    C=A/B;
Attention: If the variable OPE is a slash / or takes any value other than +, -, or *, ther
/ will be executed. This will lead to confusion.
The samples on the previous pages show all the different cases.
```

- The use of the DO UNTIL statement is discouraged.
- LEAVE statement

list 18: Use this statement to exit from the DO loop

```
DO I=1 TO 20;
-
  IF A=B THEN LEAVE;  --Exit from this DO group and go to the step after END.
  -
 END;
```

list 19: Exit from the DO loop nest

```
A : DO I=1 TO 20;
    B : DO J=1 TO 40;
    -
        LEAVE B;     -- Exit from the label B DO group.
    -
    END;
    -
    LEAVE A;     -- Exit from the label A Do group..
    -
END;
```

3. Whenever possible, include only one segment on each page (100 rows).
   segment — Procedure
   MAIN LOGIC

2.5 Input of file and output to file

1. DCL
   - Do not specify ENV except as necessary for file editing (for VSAM, etc.)
   - Attributes other than ENV should be explicitly declared (DCL).
   - The SYSPRINT should be explicitly declared by use of DCL.
     Error tracing information is output to SYSPRINT. If you declare the SYSPRINT, then the corresponding DD statement becomes necessary, and you will not forget to add it to the JCL.
   - You can use SYSPRINT only to print out AUDIT Reports and Error Information. You cannot use it to print regular reports.
2. Input/output layout DCLs
   - Use CRD.
   - In principle, include only data used by the program.
3. Input statement
   - In principle, use READ.
4. Output statement
   - In principle, use PUT for the print output.
     You should put the headings and the DETAIL row of set forms into an area of one row (80 bytes, 132 bytes)
     Example: PUT FILE (LIST) EDIT (HEAD1) (A); /* HEAD1:one row */
   - For details on printing the headings, see "Program Structure-6. DCL for report headings."
   - Use WRITE unless the output is to be printed.
5. Do not use BY NAME
   - If you use BY NAME, you will have no way to know which data items you have copied; you will have to look at the DCL each time.
   - When there is a structural change in the DCL, the subject of the copy for BY NAME may change. Such a change may cause an error.
6. Do not use the file variable.
7. In principle, do not use PLISORT (external SORT) in a program. The processing is easier to understand if you separate a step into multiple steps in the JCL, for instance the sorting part from the program.

2.6 Program audit report

- At the time of a normal program end or an abend, print out at least the following items:

  1. Input record count/file
  2. Output record count/file
  3. Updated records count/file
  4. Error record count, etc.

  Also, print out the equation showing how the input, output, and error records in the comment are related.

  Example: Output record = Input record + Error record

  Output record = Updated record + Error record

- The above is done by using the ADM Services Standard Macro. Items that cannot be output by use of the ADM Services Standard Macro, such as Comments, should be output by use of PUT EDIT.

list 20: ADM Services Standard Macro Usage Sample

```
SAMPLE:PROC OPTIONS(MAIN);
    -
    -
DCL     AFILE_INPUT_COUNT   FIXED BIN(31,0) INIT(0),
        BFILE_UPDATE_COUNT  FIXED BIN(31,0) INIT(0),
        BFILE_ADD_COUNT     FIXED BIN(31,0) INIT(0),
        BFILE_CHANGE_COUNT  FIXED BIN(31,0) INIT(0),
        BFILE_DELETE_COUNT  FIXED BIN(31,0) INIT(0),
        BFILE_OUTPUT_COUNT  FIXED BIN(31.0) INIT(0);
           -
%INCLUDE SYSLIB2(SMONERR1);
    READ FILE (AFILE)INTO(AIN);
    AFILE_INPUT_COUNT=AFILE_INPUT_COUNT+1;
        -
EOJ:                                    /* END OF JOB PROCESS */
%INCLUDE SYSLIB2(SMHEAD3);
SMCOUNT4(AFILE_INPUT_COUNT,     The variable specified here
         BFILE_UPDATE_COUNT,    is output to the //SYSPRINT
         BFILE_ADD_COUNT,       when the program comes to a normal end or an abend.
         BFILR_CHANGE_COUNT,
         BFILE_DELETE_COUNT,
         BFILE_OUTPUT_COUNT,END);
PUT EDIT('AFILE_INPUT_COUNT ===> INPUT COUNTER OF AAAA FILE',
         'BFILE_UPDATE_COUNT===> UPDATE(ADD,CHANGE,DELETE)',
         'COUNT OF BBBB FILE',
```

```
            'BFILE_UPDATE_COUNT===> ADD_COUNT+CHANGE_COUNT+',
            'DELETE_COUNT',
            -
            -)
            (SKIP,A,SKIP,A,A,SKIP,A,A,-);
    %INCLUDE SYSLIB(SHCHECK5);
    %XTWOWAY=3
    SMEND7(END);
    END program;
```

- If the program ends abnormally, you should ABEND it by use of the ADM Services Standard Macro.
- In the following cases, print out the message and then abend.
  - TABLE oversized
  - Data Base Status abnormality.
  - Other cases where the operation cannot be continued.

  In this case, you should include such items as cause of error and information on the location of the error (such as table name, statement label, DLI return code) in the SMCONST2 USER TEXT so that you can trace the origin of the error.

list 21: Coding sample

```
    SAMPLE:PROC OPTIONS(MAIN);

    DCL     TBL_A(10)       CHAR(3);
            TBL_A_CTR       FIXED BIN(31,0) INIT(0);
    -
    %INCLUDE SYSLIB2(SMONERR1);
    -
    -
    DO I=1 TO X CTR;
T1 : IF TBL_A_CTR > 10 THEN SMCONST2(TABLE_A SIZE OVERFLOW AT T1);
    TBL_A_CTR=TBL_A_CTR+1;
    TBL_A(TBL_A_CTR)=TBL_X(I);
    -
END;
    -
EOJ:
    %INCLUDE SYSLIB2(SMHEAD3);
    SMCOUNT4(......,TBL_A_CTR,END);
    %INCLUDE SYSLIB2(SMCHECK5);
```

```
%XTWOWEY=3;
SMEND7(END);
END SAMPLE;
```

2.7 Subject to %INCLUDE

Items to %INCLUDE

- The data definition used by regular and multiple programs should should be %INCLUDE.
  - SSA
  - PCB itself
  - PCB/SH (screen handler) interface
  - User SPA
  - Regular constants (DL/I CALL Function, etc.)
- The EXEC statement sets used by regular and multiple programs should not use %INCLUDE but should be subroutines.

2.8 Reusability of the online programs

Should be reusable

- The programs, whether they are JAAS-TP type online programs or not, must be reusable.
- During online processing, programs that already exist in the MPP need not be reloaded from the load module library if they are coded for reuse.

  Method: The variables that change their values within a program should be specified as AUTOMATIC attributes. If the variables are STATIC attributes, they should be initialized at the start of a program as such, not by use of INIT. (If the variables are AUTOMATIC, a new area is allocated to them every time there is a block activity. However, if they are STATIC, their area can be allocated only once, and the value used in the preceding execution is used.)

list 22: Sample: Reusable codings

```
DCL FLAG FIXES BIN(15,0) INIT(O);

    IF FLAG=0 THEN DO;
                    FLAG=1;
                    CALL A;
                END;
                ELSE DO;
                    FLAG=0;
                    CALL B;
                END;
```

list 23: Sample: Unreusable codings

```
DCL FLAG FIXES BIN(15,0) INIT(OB) STATIC;
```

If you write the program as shown above, the preceding result still remains. Therefore the program will not be FLAG=0 initially.

2.9 Caution when calling the IMS by use of CMD CALL or GCMD CALL

1. Scope: Programs that are calling the IMS by use of CMD/GCMD CALL or programs that are using MOOLTM..
2. Status: The IMS reschedules the message if an input message from a program is in flight when the IMS has to be restarted because of a problem such as a system down. The subject program then sends a GU/CHKP CALL to the message. Then, if the message is read correctly, the IMS sends a Cx status to the program.
3. CODING SAMPLE

list 24: Coding Sample

```
 CALL PLITDLI (COUNT, 'GU', IOPB, IOAREA); ........'GU' MSG CALL

 IF IOPCB.STATUS='  '  |                  ---
    IOPCB.STATUS='CE' |                     |
    IOPCB.STATUS='CF' |                     |
    IOPCB.STATUS='CI'  |           __        |
    IOPCB.STATUS='CG' |             |       | Additional statements
    IOPCB.STATUS='CJ' |            | Note|
```

```
   IOPCB.STATUS='CK' |            |       |
   IOPCB.STATUS='CL' |          __|    ___|
   THEN CALL NOMAL_RTN; ........NORMAL processing
   ELSE
IF IOPCB.STATUS='QC'
   THEN CALL MSGEND_RTN; .........NO MORE MSG processing
   ELSE CALL BAD_STATUS; .........ERROR STATUS processing
   -
   -
   -
```

Note: If the message is not from an AOI USER EXIT, the CG, CJ, CK, or CL check is unnecessary.

- A Cx related status code.

  When the application program processes a transaction that is defined to the security maintenance utility as an AOI transaction, the following additional PCB status codes can be returned on a GU call:

  - CE means that a message has been rescheduled after a command (CMD) call has been issued before a synchronization point has been reached.
  - CF means that a message had been scheduled for a transaction.
  - CG means that a message originated from an AOI user exit.
  - CI means that both CE and CF have been returned.
  - CJ means that both CE and CG have been returned.
  - CK means that both CF and CG have been returned.
  - CL means that CE, CF and CG have been returned.

2.10 Logic pattern

1. Control break

   figure2.1Refer to "Control break.."

2. Matching

   list 25: Sample: Matching

```
ON ENDFILE(MAIN BEGIN);
    IF SUB WAS EOF THEN GOTO EOJ;
    MAIN key=highest
    END

ON ENDFILE(SUB) BEGIN
```

```
        IF MAIN WAS EOF THEN GOTO EOJ;
        SUB key=highest
        END
```

list 26: Sample: Reading the main file or the subfile

```
DO WHILE (EOF)
    DO WHILE(MAIN key < SUB key)
        MAIN < SUB processing
        MAIN FILE reading
    END

    DO WHILE(MAIN key < SUB key)
        MAIN FILE reading
        SUB FILE reading
    END

    DO WHILE(MAIN key < SUB key)
        MAIN < SUB processing
        SUB FILE reading
    END
END
```

2.11 Frame

Frames for Matching, Selection, and Total are available in "SC02.DS.FRAME."

- SC02$10
  1. Function: AUDIT
  2. Replacement portion
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : AS OF DATE FILE INPUT AREA AUTO CRD MEMBER NAME
     $INPFL01 : INPUT FILE
     $OUTFL01 - 06 : OUTPUT FILE
     $LSTFL01 : LIST FILE
- SC02$21
  1. Function: SELECTION

2. Replacement portion
   $A : PROCEDURE NAME
   $B : EXTERNAL PARAMETER
   $C : INPUT FILE AUTO CRD MEMBER NAME
   $D : OUTPUT FILE AUTO CRD MEMBER NAME
   $INPFL01 : INPUT FILE
   $OUTFL01 : OUTPUT FILE

- SC02$21A

  1. Function: SELECTION(No Internal Read)
  2. Replacement portion
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : INPUT FILE AUTO CRD MEMBER NAME
     $D : OUTPUT FILE AUTO CRD MEMBER NAME
     $INPFL01 : INPUT FILE
     $OUTFL01 : OUTPUT FILE

- SC02$22

  1. Function: SELECTION (for two input files)
  2. Replacement portion
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $G : ROUTINE NAME
     $H : ROUTINE NAME
     $INPFL01 : INPUT FILE
     $OUTFL01 : OUTPUT FILE
     $INPFL02 : INPUT FILE
     $OUTFL02 : OUTPUT FILE

- SC02$23

  1. Function: SELECTION (for three input files)
  2. Replacement portion
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $I : ROUTINE NAME
     $J : ROUTINE NAME
     $K : ROUTINE NAME
     $INPFL01 : INPUT FILE
     $OUTFL01 : OUTPUT FILE

$INPFL02 : INPUT FILE
$OUTFL02 : OUTPUT FILE
$INPFL03 : INPUT FILE
$OUTFL03 : OUTPUT FILE

- SC02$24

  1. Function: SELECTION (for four input files)
  2. Replacement portion
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $K : ROUTINE NAME
     $L : ROUTINE NAME
     $M : ROUTINE NAME
     $N : ROUTINE NAME
     $INPFL01 : INPUT FILE
     $OUTFL01 : OUTPUT FILE
     $INPFL02 : INPUT FILE
     $OUTFL02 : OUTPUT FILE
     $INPFL03 : INPUT FILE
     $OUTFL03 : OUTPUT FILE
     $INPFL04 : INPUT FILE
     $OUTFL04 : OUTPUT FILE

- SC02$311

  1. Function: MATCHING (1:1 one output file)
  2. Replacement portion
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : MAIN FILE CRD MEMBER NAME
     $D : SUB FILE AUTO CRD MEMBER NAME
     $MANFL01 : MAIN FILE
     $SUBFL01 : SUB FILE
     $OUTFL01 : OUTPUT FILE * Also included in the MODEL function

- SC02$312

  1. Function: MATCHING (1:1 two output files)

2. Replacement portion
   $A : PROCEDURE NAME
   $B : EXTERNAL PARAMETER
   $C : MAIN FILE CRD MEMBER NAME
   $D : SUB FILE AUTO CRD MEMBER NAME
   $MANFL01 : MAIN FILE
   $SUBFL01 : SUB FILE
   $OUTFL01 : OUTPUT FILE
   $OUTFL02 : OUTPUT FILE

- SC02$32

  1. Function: MATCHING (1:2 one output file)
  2. Replacement portion
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : MAIN FILE CRD MEMBER NAME
     $D : SUB FILE-1 AUTO CRD MEMBER NAME
     $E : SUB FILE-2 AUTO CRD MEMBER NAME
     $MANFL01 : MAIN FILE
     $SUBFL01 : SUB FILE
     $SUBFL02 : SUB FILE
     $OUTFL01 : OUTPUT FILE

- SC02$33

  1. Function: MATCHING (1:3 one output file)
  2. Replacement portion
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : MAIN FILE CRD MEMBER NAME
     $D : SUB FILE-1 AUTO CRD MEMBER NAME
     $E : SUB FILE-2 AUTO CRD MEMBER NAME
     $F : SUB FILE-3 AUTO CRD MEMBER NAME
     $MANFL01 : MAIN FILE
     $SUBFL01 : SUB FILE
     $SUBFL02 : SUB FILE
     $SUBFL03 : SUB FILE
     $OUTFL01 : OUTPUT FILE

- SC02$34

  1. Function: MATCHING (1:4 one output file)

2. Replacement part
   $A : PROCEDURE NAME
   $B : EXTERNAL PARAMETER
   $C : MAIN FILE CRD MEMBER NAME
   $D : SUB FILE-1 AUTO CRD MEMBER NAME
   $E : SUB FILE-2 AUTO CRD MEMBER NAME
   $F : SUB FILE-3 AUTO CRD MEMBER NAME
   $G : SUB FILE-4 AUTO CRD MEMBER NAME
   $MANFL01 : MAIN FILE
   $SUBFL01 : SUB FILE
   $SUBFL02 : SUB FILE
   $SUBFL03 : SUB FILE
   $SUBFL04 : SUB FILE
   $OUTFL01 : OUTPUT FILE

- SC02$35

  1. Function: MATCHING (1:5 one output file)
  2. Replacement part
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : MAIN FILE CRD MEMBER NAME
     $D : SUB FILE-1 AUTO CRD MEMBER NAME
     $E : SUB FILE-2 AUTO CRD MEMBER NAME
     $F : SUB FILE-3 AUTO CRD MEMBER NAME
     $G : SUB FILE-4 AUTO CRD MEMBER NAME
     $H : SUB FILE-5 AUTO CRD MEMBER NAME
     $MANFL01 : MAIN FILE
     $SUBFL01 : SUB FILE
     $SUBFL02 : SUB FILE
     $SUBFL03 : SUB FILE
     $SUBFL04 : SUB FILE
     $SUBFL05 : SUB FILE
     $OUTFL01 : OUTPUT FILE

- SC02$36

  1. Function: MATCHING (1:6 one output file)
  2. Replacement part
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : MAIN FILE CRD MEMBER NAME

$D : SUB FILE-1 AUTO CRD MEMBER NAME
$E : SUB FILE-2 AUTO CRD MEMBER NAME
$F : SUB FILE-3 AUTO CRD MEMBER NAME
$G : SUB FILE-4 AUTO CRD MEMBER NAME
$H : SUB FILE-5 AUTO CRD MEMBER NAME
$I : SUB FILE-6 AUTO CRD MEMBER NAME
$MANFL01 : MAIN FILE
$SUBFL01 : SUB FILE
$SUBFL02 : SUB FILE
$SUBFL03 : SUB FILE
$SUBFL04 : SUB FILE
$SUBFL05 : SUB FILE
$SUBFL06 : SUB FILE
$OUTFL01 : OUTPUT FILE

- SC02$39

  1. Function: MATCHING (1:9 one output file)
  2. Replacement part
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : MAIN FILE CRD MEMBER NAME
     $D : SUB FILE-1 AUTO CRD MEMBER NAME
     $E : SUB FILE-2 AUTO CRD MEMBER NAME
     $F : SUB FILE-3 AUTO CRD MEMBER NAME
     $G : SUB FILE-4 AUTO CRD MEMBER NAME
     $H : SUB FILE-5 AUTO CRD MEMBER NAME
     $H : SUB FILE-6 AUTO CRD MEMBER NAME
     $H : SUB FILE-7 AUTO CRD MEMBER NAME
     $H : SUB FILE-8 AUTO CRD MEMBER NAME
     $H : SUB FILE-9 AUTO CRD MEMBER NAME
     $MANFL01 : MAIN FILE
     $SUBFL01 : SUB FILE
     $SUBFL02 : SUB FILE
     $SUBFL03 : SUB FILE
     $SUBFL04 : SUB FILE
     $SUBFL05 : SUB FILE
     $SUBFL06 : SUB FILE
     $SUBFL07 : SUB FILE
     $SUBFL08 : SUB FILE
     $SUBFL09 : SUB FILE
     $OUTFL01 : OUTPUT FILE

- SC02$42

  1. Function: MATCHING (2:1 one output file)
  2. Replacement part
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : MAIN FILE-1 CRD MEMBER NAME
     $D : MAIN FILE-2 CRD MEMBER NAME
     $E : SUB FILE AUTO CRD MEMBER NAME
     $MANFL01 : MAIN FILE
     $MANFL02 : MAIN FILE
     $SUBFL01 : SUB FILE
     $OUTFL01 : OUTPUT FILE
     $OUTFL02 : OUTPUT FILE

- SC02$43

  1. Function: MATCHING (3 : 1 three output files)
  2. Replacement part
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $C : MAIN FILE-1 CRD MEMBER NAME
     $D : MAIN FILE-2 CRD MEMBER NAME
     $E : MAIN FILE-3 CRD MEMBER NAME
     $F : SUB FILE AUTO CRD MEMBER NAME
     $MANFL01 : MAIN FILE
     $MANFL02 : MAIN FILE
     $MANFL03 : MAIN FILE
     $SUBFL01 : SUB FILE
     $OUTFL01 : OUTPUT FILE
     $OUTFL02 : OUTPUT FILE
     $OUTFL03 : OUTPUT FILE

- SC02$60

  1. Function: TOTAL
  2. Replacement part
     $A : PROCEDURE NAME
     $B : EXTERNAL PARAMETER
     $E : MACRO CRD MEMBER NAME OF INPUT FILE
     $F : MACRO CRD MEMBER NAME OF OUTPUT FILE
     $G : INPUT RECORD FIELD - KEY1 (LOWEST KEY)

$H : INPUT RECORD FIELD - KEY2 v
$J : INPUT RECORD FIELD - KEY3 v
$K : INPUT RECORD FIELD - KEY4 v
$L : INPUT RECORD FIELD - KEY5 v
$M : INPUT RECORD FIELD - KEY6 v
$N : INPUT RECORD FIELD - KEY7 v
$P : INPUT RECORD FIELD - KEY8 v
$Q : INPUT RECORD FIELD - KEY9 (HIGHEST KEY)
$U : INPUT RECORD FIELD - TOTAL COUNT AREA 1
$V : INPUT RECORD FIELD - TOTAL COUNT AREA 2
$W : INPUT RECORD FIELD - TOTAL COUNT AREA 3
$SUBFL01 : INPUT FILE
$OUTFL01 : OUTPUT FILE * Also in the MODEL function

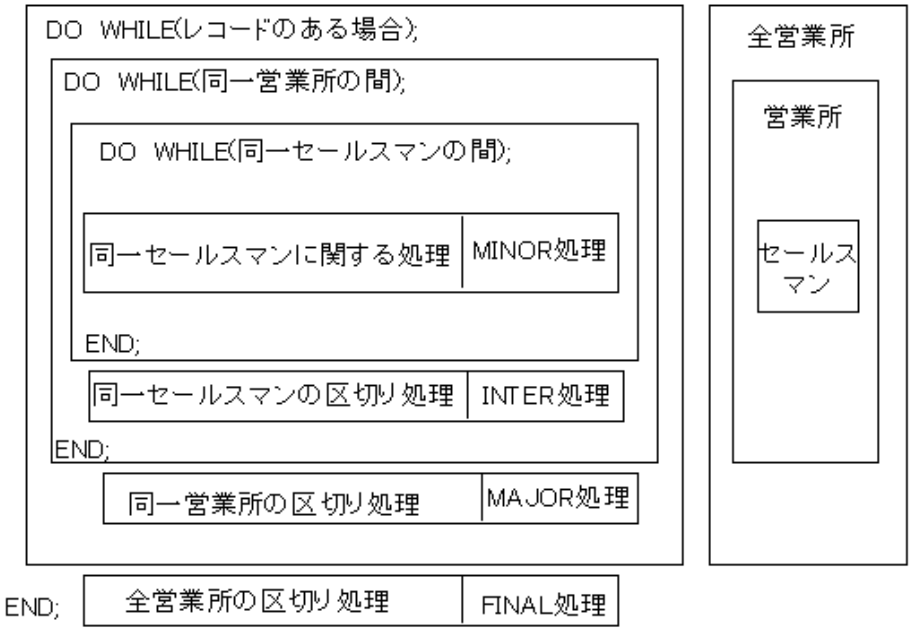

Figure 2.1: Control break

Chapter 3

Appendix

3.1 Sample coding

list 27: SAMPLE

```
 /* ST80F40 : GDE - DOCUMENT DB MAINT.     IBM INTERNAL USE ONLY      */00010000
0ST80F40: PROC (DEPCB) OPTIONS (MAIN);                                  00020000
0/*--------------------------------------------------------------------*/00030000
 /*     PROJECT NUMBER        :   ST80                                 */00040000
 /*     PROJECT NAME          :   GDE                                  */00050000
 /*     MODULE  NUMBER        :   ST80F40                              */00060000
 /*     MODULE  NAME          :   DOCUMENT DB MAINTENANCE              */00070000
 /*     PROGRAMMER            :   RYOKA SYSTEMS INC.                   */00080000
 /*     WRITTEN DATE          :   82/06/30                             */00090000
 /*     UPDATE                :                                        */00100000
 /*     * DATA BASE DATABASE * (CRD NAME)  :   JSSTDEDB (JRSTDE0)      */00110000
 /*--------------------------------------------------------------------*/00120000
0DCL  SSTMS FILE KEYED RECORD INPUT ENV(INDEXED),/* MSF               */00130000
      SSTCT FILE RECORD OUTPUT;                  /* IEBCOPY CTL FILE  */00140000
 %INCLUDE ST80PDE;                                                      00150000
 %INCLUDE ST80SDE0;                                                     00160000
0DCL  1 JRSTDE0 STATIC,                                                 00170000
      INCB UNDEF,N,JRSTDE0;                                             00180000
0DCL  1 JRSTMS0 STATIC,                                                 00190000
      INCB UNDEF,N,JRSTMS0;                                             00200000
0DCL  SCO2U12           ENTRY,                 /* PDS DIRECTORY READ*/00210000
      INTSRT            ENTRY,                 /* INTERNAL SORT     */00220000
      PLITDLI           ENTRY;                 /* DL/I INTERFACE    */00230000
0DCL  ADDR              BUILTIN,               /* BUILTIN FUNCTION  */00240000
      HIGH              BUILTIN,               /* BUILTIN FUNCTION  */00250000
      INDEX             BUILTIN,               /* BUILTIN FUNCTION  */00260000
      SUBSTR            BUILTIN,               /* BUILTIN FUNCTION  */00270000
      PLIDUMP           BUILTIN;               /* BUILTIN FUNCTION  */00280000
```

```
0%INCLUDE ST80CDLI;                                                    00290000
0DCL  CTREC             CHAR(80) STATIC;         /* CONTROL STATMENT  */00300000
0DCL  1 PARM,                                    /* PARAMETER SET     */00310000
        2 DDNAME        CHAR(08),                /* DDDNAME           */00320000
        2 SELKEY        CHAR(08),                /* SELECTION KEY     */00330000
        2 TBLADDR       PTR,                     /* TABLE ADDRESS     */00340000
        2 TBLCNT        FIXED BIN(15),           /* TABLE COUNT       */00350000
        2 RETCD         CHAR(01),                /* RETURN CODE       */00360000
      PARMPT            PTR;                     /* PARM AREA PTR     */00370000
0DCL  MEMTBL(2000)      CHAR(08) BASED(TBLADDR); /* MEMBER TBL        */00380003
0DCL  MBRNM             CHAR(07) STATIC;         /* MEMBER NAME WORK  */00390000
0DCL  EOFMS             BIT(1)   STATIC,         /* SW_EOF MSF FILE   */00400000
      EOFMEM            BIT(1)   STATIC,         /* SW_EOF TBL.POS.   */00410000
      RDMEM             CHAR(04) STATIC,         /* SW_READ MSF FILE  */00420000
      RDMSF             CHAR(04) STATIC;         /* SW_READ TBL.POS.  */00430000
0DCL  MEMPOS            FIXED BIN(15) STATIC,    /* MEMPOS COUNT      */00440000
      DROP_MBR_CNT      FIXED BIN(31) STATIC,    /* DROP MBR COUNT    */00450000
      PDS_MBR_CNT       FIXED BIN(31) STATIC,    /* MEMBER COUNT      */00460000
      I                 FIXED BIN(15) STATIC;    /* INDEX WORK        */00470000
1/*-------------------------------------------------------------------*/00480000
 /*              MAIN      PROCESS                                    */00490000
 /*-------------------------------------------------------------------*/00500000
0%INCLUDE SYSLIB2(SMONERR1);                     /* ERROR DCL         */00510000
0   ON ENDFILE (SSTMS) BEGIN;                    /* ON END FILE MSF   */00520000
      EOFMS = '1'B;                              /* ON EOF SW         */00530000
      JRSTMS0.RJGDA = HIGH(7);                   /* SET HIGH KEY      */00540000
      END;                                       /*                   */00550000
0   OPEN FILE(SSTMS),                            /* OPEN FILE MSF     */00560000
         FILE(SSTCT);                            /* IEBCOPY CTL FILE  */00570000
0   EOFMS  = '0'B;                               /* INIT MSF EOF      */00580000
    EOFMEM = '0'B;                               /* INIT MEMTBL EOF   */00590000
    RDMSF  = 'READ';                             /* INIT MSF READ     */00600000
    RDMEM  = 'READ';                             /* INIT MEMTBL READ  */00610000
1   CALL MEMSET;                                 /* SET MEMBER TABLE  */00620000
0   MEMPOS = 0;                                  /* CLEAR POSITION    */00630000
0   CTREC = '  COPY OUTDD=SSTHI,INDD=SSTUN';     /* FIRST DATA        */00640000
    WRITE FILE(SSTCT) FROM(CTREC);               /* WRITE CTL FILE    */00650000
0LBL_READ:                                       /*                   */00660000
0   IF (RDMSF = 'READ')                          /* WHEN MSF BE READ  */00670004
       THEN DO;                                                          00680004
       IF (EOFMS) THEN;                                                  00690004
          ELSE READ FILE(SSTMS) INTO(JRSTMS0);   /* READ MSF          */00700004
       END;                                                              00710004
0   IF (RDMEM = 'READ')                          /* WHEN MEMBER TABLE */00720004
      THEN DO;                                                           00730004
      IF (EOFMEN) THEN;                                                  00740004
        ELSE DO;                                 /*        BE READ    */00750004
        MEMPOS = MEMPOS + 1;                     /* COUNT TBL POS     */00760004
        IF MEMPOS > PDS_MBR_CNT                  /* WHEN EOF OF TABLE */00770004
        THEN DO;                                 /*                   */00780004
```

```
        MBRNM = HIGH(7);                          /* SET HIGH TO MBRNM */00790004
        EOFMEM = '1'B;                            /* SET ON TO EOF SW  */00800004
        END;                                      /*                   */00810004
        ELSE                                      /*                   */00820004
        MBRNM = TRANSLATE(MEMTBL(MEMPOS),'-','@');/*SET MEMBER NAME   */00830004
                                                  /* TRANSLATE '@'=>'-'*/00840004
        END;                                                            00850004
      END;                                        /*                   */00860004
0   IF JRSTMS0.RJGDA < MBRNM                      /* WHEN MSF ONLY     */00870004
      THEN DO;                                    /*                   */00880004
      RDMSF = 'READ';                             /* ON READ MSF SW    */00890004
      RDMEM = 'PASS';                             /* OFF READ TBL SW   */00900004
      END;                                        /*                   */00910004
0   IF JRSTMS0.RJGDA = HIGH(7) THEN;             /* WHEN NOT EOF  &   */ 00920004
       ELSE DO;                                                         00930004
       IF JRSTMS0.RJGDA = MBRNM                   /*         MATCH     */00940005
          THEN DO;                                /*                   */00950005
          RDMSF = 'READ';                         /* ON READ MSF SW    */00960004
          RDMEM = 'READ';                         /* ON READ TBL SW    */00970004
          END;                                                          00980004
       END;                                       /*                   */00990004
0   IF JRSTMS0.RJGDA > MBRNM                      /* WHEN MEMBER TABLE */01000004
      THEN DO;                                    /*            ONLY   */01010004
      CALL DOCDEL;                                /* DOCUMENT DB DELETE*/01020004
      CALL COPYREQ;                               /* IEBCOPY CTL STMT  */01030004
      RDMSF = 'PASS';                             /* OFF READ MSF SW   */01040004
      RDMEM = 'READ';                             /* ON READ TBL SW    */01050004
      END;                                        /*                   */01060004
0   IF JRSTMS0.RJGDA = HIGH(7) &                  /* WHEN MSF NOT EOF  */01070004
       MBRNM         = HIGH(7) THEN;              /* OR  MEMTBL NOT EOF*/01080005
      ELSE GOTO LBL_READ;                         /* PROCESS NEXT DATA */01090004
0   PUT PAGE    EDIT('ST80F40 SSTUN(UNLOAD DOCUMENT)  MEMBER',          01100004
                  TBLCNT) (X(5),A,X(2),P'ZZZ9'); /* PUT TBL MBR COUNT */01110004
    PUT SKIP(2) EDIT('        SSTCT(IEBCOPY CTL.STMT.)MEMBER',          01120004
          DROP_MBR_CNT) (X(5),A,X(2),P'ZZZ9');   /* PUT DROP MBR COUNT*/01130004
0   CLOSE FILE(SSTMS),                            /* CLOSE FILE MSF    */01140004
          FILE(SSTCT);                            /* IEBCOPY CTL FILE  */01150004
1/*-------------------------------------------------------------------*/01160004
 /*           SET MEMBER TABLE (SUB)                                  */01170004
 /*-------------------------------------------------------------------*/01180004
0MEMSET: PROC;                                    /* MEMSET SUB PROC   */01190004
0   DDNAME = 'SSTUN';                             /* INIT DDNAME       */01200004
    SELKEY = ' ';                                 /* CLEAR KEY         */01210004
    TBLCNT = 0;                                   /* CLEAR TBL COUNT   */01220004
    PARMPT = ADDR(PARM);                          /* SET PARM PTR      */01230004
0   CALL SC02U12(PARMPT);                         /* PDS READ SUB      */01240004
0   IF RETCD = ' ' THEN;                          /* WHEN ERROR        */01250004
      ELSE                                        /*                   */01260004
      SMCONST2(ST80F40 UNLOAD DOC. READ ERROR);   /* PDS READ ERR ABEND*/01270004
0   IF TBLCNT > 2000                              /* WHEN TABLE OVER   */01280004
```

```
      THEN                                        /*                   */01290004
      SMCONST2(ST80F40 STUN MBR TBL OVER 2000);  /* MEMBER TBL OVER   */01300004
0   CALL INTSRT(MEMTBL,MEMTBL,TBLCNT);           /* MEMBER TBL SORT   */01310004
0   PDS_MBR_CNT = TBLCNT;                        /* SET TBBLE COUNT   */01320004
0END MEMSET;                                     /* END MEMSET        */01330004
1/*-------------------------------------------------------------------*/01340004
 /*           DOCUMENT DB DELETE (SUB)                               */01350004
 /*-------------------------------------------------------------------*/01360004
0DOCDEL: PROC;                                   /* DOCDEL SUB PROC   */01370004
0   DEOSSA.DE0BEG = '(';                         /* SET DOCUMENT SEG  */01380004
    DEOSSA.DE00PR = '>=';                        /*  OF DOCUMENT DB   */01390004
    DEOSSA.DE0KEY = (2)' ' || MBRNM || (27)' ';  /*                   */01400004
0   CALL PLITDLI (CNT4,GHU,DEPCB,JRSTDE0,DEOSSA);/* GHU DOCUMENT SEG  */01410004
                                                 /*  OF DOCUMENT DB   */01420004
    SELECT(DE.STATUS);                           /* STATUS CODE ?     */01430004
      WHEN(' ');                                 /* WHEN NORMAL       */01440004
      OTHER DO;                                  /* WHEN DB ERROR     */01450004
        CALL DBERR(DEPCB,GHU);                   /* SET ERROR MESSAGE */01460004
        RETURN;                                  /*                   */01470004
        END;                                     /*                   */01480004
    END;                                         /* END SELECT(STATUS)*/01490004
0   DO WHILE (JRSTDE0.RJGDA = MBRNM &            /* DO LOOP           */01500004
              DE.STATUS = ' ');                  /*                   */01510004
0     CALL PLITDLI(CNT3,DLET,DEPCB,JRSTDE0);     /* DLET DOCUMENT SEG */01520004
                                                 /*  OF DOCUMENT DB   */01530004
      SELECT(DE.STATUS);                         /* STATUS CODE ?     */01540004
        WHEN(' ');                               /* WHEN NORMAL       */01550004
        OTHER DO;                                /* WHEN DB ERROR     */01560004
          CALL DBERR(DEPCB,DLET);                /* SET ERROR MESSAGE */01570004
          RETURN;                                /*                   */01580004
          END;                                   /*                   */01590004
      END;                                       /* END SELECT(STATUS)*/01600004
0     CALL PLITDLI(CNT3,GHN,DEPCB,JRSTDE0);      /* GHN DOCUMENT SEG  */01610004
                                                 /*  OF DOCUMENT DB   */01620004
      SELECT(DE.STATUS);                         /* STATUS CODE ?     */01630004
        WHEN('  ');                              /* WHEN NORMAL       */01640004
        WHEN('GB');                              /* WHEN EOF          */01650004
        OTHER DO;                                /* WHEN DB ERROR     */01660004
          CALL DBERR(DEPCB,GHN);                 /* SET ERROR MESSAGE */01670004
          RETURN;                                /*                   */01680004
          END;                                   /*                   */01690004
      END;                                       /* END SELECT(STATUS)*/01700004
    END;                                         /* END DO LOOP       */01710004
0END DOCDEL;                                     /* END DOCDEL        */01720004
1/*-------------------------------------------------------------------*/01730004
 /*           IEBCOPY CONTROL STATEMENT SET (SUB)                    */01740004
 /*-------------------------------------------------------------------*/01750004
0COPYREQ: PROC;                                  /* COPYREQ SUB PROC  */01760004
0   I = INDEX(MEMTBL(MEMPOS),' ');               /* SEARCH REAL LENGTH*/01770004
    IF I = 0                                     /* 7 BYTE FULL ?     */01780004
```

```
0     THEN CTREC = '  SELECT MEMBER=(('          /* SET CONTROL       */01790004
                || MEMTBL(MEMPOS) || ',,R))';    /*       STETMENT    */01800004
0     ELSE CTREC = '  SELECT MEMBER=(('          /* SET CONTROL       */01810004
               || SUBSTR(MEMTBL(MEMPOS),1,(I-1)) /*       STETMENT    */01820004
               || ',,R))';                       /*                   */01830004
0   WRITE FILE(SSTCT) FROM(CTREC);              /* WRITE CTL FILE    */01840004
    DROP_MBR_CNT = DROP_MBR_CNT + 1;            /* COUNT DROP MBR    */01850004
0END COPYREQ;                                   /* END COPYREQ       */01860004
1/*-------------------------------------------------------------------*/01870004
 /*     DATA BASE CALL ERROR           (SUB)                         */01880004
 /*-------------------------------------------------------------------*/01890004
0DBERR : PROC(ERRPCB,ERRFUNC);                  /* DBERR SUB PROC    */01900004
0DCL  1 ER              BASED(ERRPCB),          /* ERROR DB PCB      */01910004
        2 DBNAME        CHAR(08),               /* DB NAME           */01920004
        2 SEGLVL        CHAR(02),               /* SEGMENT LEVEL NO  */01930004
        2 STATUS        CHAR(02),               /* STATUS CODE       */01940004
        2 PROCOP        CHAR(04),               /* PROCESSING OPTION */01950004
        2 JCBPTR        CHAR(04),               /* RESERVED          */01960004
        2 SEGNAME       CHAR(08),               /* SEGMENT NAME      */01970004
        2 FBKLEN        FIXED BIN(31),          /* KFBA LENGTH       */01980004
        2 SENSEG#       FIXED BIN(31),          /* SENSITIVE SEGMENT#*/01990004
        2 KFBA          CHAR(47);               /* KEY FEEDBACK AREA */02000004
 DCL  ERRPCB            PTR;                    /* ERR PCB PTR       */02010004
0DCL  ERRFUNC           CHAR(04);               /* ERROR DB FUNCTION */02020004
    PUT SKIP(2) EDIT(SUBSTR(ER.DBNAME,5,2),' DL/I ',ERRFUNC,          02030004
                  ' CALL ERROR STATUS = ',ER.STATUS,                  02040004
                  ',  KFBA = ',ER.KFBA)(X(10),(7)A);                  02050004
                                                /* SET ERROR MESSAGE */02060004
0END DBERR;                                     /* DBERR END         */02070004
1                                                                     02080004
 %INCLUDE SYSLIB2(SMHEAD3);                     /* ERROR DCL         */02090004
 SMCOUNT4(PDS_MBR_CNT,DROP_MBR_CNT,END);        /* ERROR DCL         */02100004
 %INCLUDE SYSLIB2(SMCHECK5);                    /* ERROR DCL         */02110004
 %XTWOWAY = 3;                                  /*                   */02120004
 SMEND7(END);                                   /*                   */02130004
0END ST80F40;                                   /* PROGRAM END       */02140004
```