Structured Programming in Assembler Language

By Charles S. Davis
csdavis@csdbiz.com
612-247-1313

Table of Contents

Introduction

A set of structured macros have been implemented that provide the following benefits:

- Productivity
  - Easier to write and test new code
  - Easier to analyze and maintain existing code

  - Clarity
    - Structured Programming means “organized” programming. Use of these macros allow you to organize your code into “building blocks” rendering it easier to write, test and maintain.
    - Structured code “looks” better. The nesting structure supported by these macros makes it clear to the reader which code is associated with various conditions.
    - Eliminates “spaghetti” code. These macros will eliminate most branches and therefore, most procedure labels.
    - Significant reduction in:
      - Procedure Labels
      - Switches

  - Powerful programming metaphors available only in high-level programming languages such as COBOL, C and Visual Basic.
    - IF, ELSE, ENDIF logic
    - Do loops
    - Select statement similar to COBOL’s Evaluate or VB’s Select Case

  - Compatibility
    - Works in batch or CICS
    - Should not conflict with existing programs

Most of these macros are part of the IBM High Level Assembler Toolkit Feature. They have been augmented with the “PERFORM” macros that I added to the collection.

Macro Summary

The macros are organized in the following categories;

- IF macro set:
  - IF
  - ELSE
  - ENDIF

- DO macro set:
  - DO Declare the start of a Loop
  - DOEXIT Leave the loop
  - ENDDO Declare the end of a loop

- SELECT macro set:
  - SELECT Start Select sequence
  - WHEN Declare Select condition
  - OTHRWISE Declare default Select condition
  - ENDSEL Terminate Select sequence

- PERFORM macro set:
  - PERFORM Perform a paragraph
  - PARA Declare the start of a paragraph
  - ENDPARA Declare the end of a paragraph
  - EXITPARA Go to the end of the current paragraph
  - SETCC Set condition code for a Paragraph or a File
  - PFMLIST Declare paragraph work areas

- Advanced Feature macro set:
  - IFERROR Tests VSAM Return code (with VSAMIO macro)

Structured Programming Introduction

The topic of Structured Programming is huge and controversial. Most programmers would agree that structured programming is desirable, but no two will readily agree on which parts of the structured programming philosophy to embrace.

Generally, the goal for all programming methodologies is to:

- Improve programmer productivity
- Simplify maintenance
- Facilitate code reuse

The techniques employed to achieve the above goals include:

- Organization of a program into modules or paragraphs
- Use of DO/ENDO to control loops
- Use of IF, ELSE, ENDIF and SELECT to manage program logic
- Elimination of GO Tos (Branch)
- Significantly reduce the number of labels and switches

Using these macros and structured programming techniques allow you to write a “readable”, organized program. “Readable” means someone else or even you can read your program 3 months later. No doubt, all programs can be read, but, most programmers would agree that it would be desirable to write programs requiring less study and research in order to be maintained in the future.

Implementation of Structured Programming.

Existing Programs

I do not recommend that you restructure an existing program just because you are making a few changes. A rewrite of a program should only be considered if:

- Many changes are required
- The program has been patched to the point of being unmanageable
- You need to free up a register or two.

Yet, you can still use these macros in those parts of the program that you are changing.

If you are adding a new function or process, new code would be a candidate to being placed inside a paragraph.

If you need to make a simple logic change, the IF, ELSE and ENDIF macros free you from the burden of creating more labels in a program already overloaded with labels. The fewer labels in a program, the easier it is to maintain.

New Programs

I recommend that all new programs be structured. You have the opportunity to create an organized program from the ground up and you don’t have to do extensive research as would be required when rewriting an existing program.

Batch / CICS

The structured programming macros can be used in both batch as well as CICS

Structured Macro Illustration

Main Line Processing controlled by a DO Loop

```
MAINLINE EQU   *
         PERFORM P6000_GET_FILE1       Priming read
         DO UNTIL=(CLI,FILE1_CC,EQ,CCEOF)
            IF (TM,ZCJ,X'01',O)        If managed account
               PERFORM P3000_PROCESS
               PERFORM P7000_PRINT_DETAIL
               COUNT 'Managed Accounts Processed'
            ENDIF
            PERFORM P6000_GET_FILE1    Get Next FILE1 record
         ENDDO
         B     EOJ
```

A one-byte condition code is automatically generated for each File and Paragraph. This code is tested with a CLI and set with the SETCC macro

IF statement using TM

Start of a Paragraph

```
P3000_PROCESS    PARA
         MVC   DetAcct+0(3),ZKEY       Move Acct No to Print Line
         MVI   DetAcct+3,C'-'
         MVC   DetAcct+4(6),ZKEY

         MVC   DetName1,ZNA1           Move in Name & Addr info
         MVC   DetName2,ZNA2

         Select CLI,ZCE,EQ
            When (C'A')
               MVC   DetAcTyp(11),=C'Association'
            When (C'B')
               MVC   DetAcTyp(04),=C'401k'
            When (C'C')
               MVC   DetAcTyp(15),=C'Tennants/Common'
            When (C'D')
               MVC   DetAcTyp(09),=C'Community'
            When (C'E')
               MVC   DetAcTyp(10),=C'Entireties'
            . . . . . .
            When (C'Z')
               MVC   DetAcTyp(09),=C'Custodial'
            Othrwise
               MVC   DetAcTyp(1),ZCE
         ENDSEL
```

SELECT Statement

You may code as many WHEN clauses as you need. You may code as many assembler statements and macros (even move structured macros) between the WHEN clauses.

IF, ELSE, ENDIF Structure

```
         IF (CLI,ZFNDCDE,GT,C' ')
            MVC   DetFund(1),ZFNDCDE
         ELSE
            MVC   DetFund,=C'None'
         ENDIF
```

Simple IF Statement

```
         IF (CLI,ZZIP,EQ,X'55')        Is this a Metro Zip Code?
            MVI   DetMetro,C'Y'
         ENDIF
         ENDPARA
```

End of a Paragraph

Standard VSAM IO command followed by IFERROR

```
P6000_GET_FILE1   PARA
         VSAMIO READNEXT,FILE1,WORKA=ZREC
         IFERROR (EQ,EOF),ERR=VERR
            SETCC FILE1,EOF
         ELSE
            COUNT 'FILE1 Records Read'
         ENDIF
         ENDPARA


P7000_PRINT_DETAIL    PARA
         BAL   R6,PRINTLN
         ENDPARA
```

IFERROR replaces 'VSAMIO ERROR' converting it to an IF statement that must be followed with an ENDIF and optionally an ELSE.

Code standard 'VSAMIO ERROR' keywords along with EQ OR NE inside the parenthesis.

Without Using Structured Macros

This following code demonstrates how the program on the previous page would be written without structured macros.

```
MAINLOOP EQU   *
         CLI   EOFSW,EOF               Are we at EOF
         BE    EOJ                        Yes, done
         TM    ZCJ,X'01'               Skip record if not managed
         BNO   NEXTREC                    Not managed, skip it
         BAL   R6,PROCESS              Process output line
         BAL   R6,PRINTLN               Print the line
         COUNT 'Managed Accounts Processed'
NEXTREC  EQU   *
         BAL   R6,GETFILE1             Read next FILE1 record
         B     MAINLOOP

*********************************************************************
*        P r o c e s s                                              *
*********************************************************************
PROCESS  EQU   *
         MVC   DetAcct+0(3),ZKEY       Move Acct No to Print Line
         MVI   DetAcct+3,C'-'
         MVC   DetAcct+4(6),ZKEY
         MVC   DetName1,ZNA1           Move in Name & Addr info
         MVC   DetName2,ZNA2

CHKA     CLI   ZCE,C'A'
         BNE   CHKB
         MVC   DetAcTyp(11),=C'Association'
         B     CHKDONE
CHKB     CLI   ZCE,C'B'
         BNE   CHKC
         MVC   DetAcTyp(04),=C'401k'
         B     CHKDONE
CHKC     CLI   ZCE,C'C'
         BNE   CHKD
         MVC   DetAcTyp(15),=C'Tennants/Common'
         B     CHKDONE
. . . . . .
CHKZ     CLI   ZCE,C'Z'
         BNE   CHKOTHER
         MVC   DetAcTyp(09),=C'Custodial'
         B     CHKDONE
CHKOTHER EQU   *
         MVC   DetAcTyp(1),ZCE
CHKDONE  EQU   *

         CLI   ZFNDCDE,C' '            Is there a fund code
         BNH   NOFUND                     No, Indicate None
         MVC   DetFund(1),ZFNDCDE         Yes, Move in Fund Code
         B     FUNDX
NOFUND   EQU   *
         MVC   DetFund,=C'None'        No Fund code
FUNDX    EQU   *

         CLI   ZZIP,X'55'              Is this a Metro Zip Code?
         BNE   NONMETRO                   No, Not Metro
         MVI   DetMetro,C'Y'              Yes, Indicate Metro
NONMETRO EQU   *
         BR    R6

*********************************************************************
*        Get Next FILE1 Record                                      *
*********************************************************************
GETFILE1 EQU   *
         VSAMIO READNEXT,FILE1,WORKA=ZREC
         VSAMIO ERROR,(R1),EOF=GETFILE7,ERR=VERR
         COUNT 'FILE1 Records Read'
         B     GETFILE9
GETFILE7 EQU   *
         MVI   EOFSW,EOF               Indicate EOF
GETFILE9 EQU   *
         BR    R6
```

IF, ELSE, ENDIF

These 3 macros allow for classic "If, Then, Else" processing used in all high-level languages. Each IF statement starts with an IF macro followed by one or more assembler statements, macros or even more IF, ELSE, ENDIF macros (nested IF statements). The IF statement must be terminated with and ENDIF macro. You may optionally place an ELSE macro between the IF and ENDIF macros.

Example 1. Simple IF Statement using CLI and CLC

```
IF (CLI,ZCE,EQ,C'A')            |   IF (CLC,ZCE,EQ,=C'A')
   MVC   ACCTYP(5),=C'Assoc"    |      MVC   ACCTYP(5),=C'Assoc'
ENDIF                           |   ENDIF
```

Example 2. IF, ELSE, ENDIF Statement using TM

```
 IF (TM,ZCA,X'01',O)
    MVC   DVPTYP(19),=C'Delivery vs Payment"
 ELSE
    MVC   DVPTYP(7),=C'Not DVP"
 ENDIF
```

Note that with TM and other instructions that set the condition code, the condition indicator is placed AFTER the 2nd operand, not between the operands as with CLI and CLC

Example 3. Any instruction that sets the condition code can be used:

```
IF (SP,COUNTER,=P'1',Z)         |   IF (AR,R2,R3,Z)
   MVI   DONE,X'FF'             |      MVI   DONE,X'FF
ENDIF                           |   ENDIF
```

Example 4. Compound IF statements can be used with OR /AND:

```
 IF (CLI,ZCA,EQ,C'A'),OR,(CLI,ZCA,EQ,C'B')
   MVI   DONE,X'FF'
ENDIF
```

However, if all conditions don't fit in one line, they must be continued using the same rules as all other macros - X in column 72 and continue in column 16. This is not a problem, but it can get in the way of a carefully crafted indentation structure.

```
IF (CLI,ZCE,EQ,C'A'),                                    X
      OR,(CLI,ZCE,EQ,C'B'),                              X
      OR,(CLI,ZCE,EQ,C'C'),                              X
      OR,(CLI,ZCE,EQ,C'D')
   MVI   . . . .
ENDIF
```

Example 5. Nested IF Statements

```
   IF (CLI,ZCA,EQ,C'A')
      MVC   ACCTYP(5),=C'Assoc"
      IF (TM,ZCB,X'02',O)
         MVI   DONE,X'FF'
      ENDIF
   ENDIF
```

Example 6. Previously Set Condition Codes

```
   CPQ   QTY1,QTY1F,QTY2,QTY2F
   IF (EQ)
      MVC   QTYSTAT(5),=C'Equal'
   ENDIF
```

IF - Example 1

Simple IF Statement using CLI and CLC

Source Code

```
IF (CLI,ZCE,EQ,C'A')
   MVC   ACCTYP(5),=C'Assoc"
ENDIF
```

Expanded Code

```
                                    420          IF (CLI,ZCE,EQ,C'A')
00054C 95C1 7CC7      01061         421+              CLI        ZCE,C'A'
000550 4770 71C0            0055A   422+               BC 15-8,#@LB1
000554 D204 7C13 77B8 00FAD 00B52   423               MVC   DetActyp(5),=C'Assoc'
                                    424          ENDIF
                            0055A   425+#@LB1        EQU *
```

Source Code

```
IF (CLC,ZCE,EQ,=C'A')
   MVC   ACCTYP(5),=C'Assoc'
ENDIF
```

Expanded Code

```
                                    427          IF (CLC,ZCE,EQ,=C'A')
00055A D500 7CDF 77CD 01079 00B67   428+              CLC        ZCE,=C'A'
000560 4770 71D0            0056A   429+               BC 15-8,#@LB3
000564 D204 7C2B 77C8 00FC5 00B62   430               MVC   DetActyp(5),=C'Assoc'
                                    431          ENDIF
                            0056A   432+#@LB3        EQU *
```

IF - Example 2

IF, ELSE, ENDIF Statement using TM

Source Code

```
          IF (TM,ZCA,X'01',O)
             MVC   DetActyp(3),=C'DVP'
          ELSE
             MVC   DetActyp(7),=C'Not DVP'
          ENDIF
```

Expanded Code

```
                                        434          IF (TM,ZCA,X'01',O)
00056A 9101 7D03      0109D             435+               TM         ZCA,X'01'
00056E 47E0 71E2            0057C       436+              BC 15-1,#@LB5
000572 D202 7C53 77E6 00FED 00B80       437              MVC   DetActyp(3),=C'DVP'
                                        438          ELSE
000578 47F0 71E8            00582       439+              BC   15,#@LB7
0057C                                   440+#@LB5         EQU  *
00057C D206 7C53 77F5 00FED 00B8F       441              MVC   DetActyp(7),=C'Not DVP'
                                        442          ENDIF
                            00582       443+#@LB7         EQU *
```

IF - Example 3

IF in conjunction with any instruction that sets the condition code

Source Code

```
         IF  (SP,COUNTER,=P'0',NZ)
             MVI   DONESW,X'FF'
         ENDIF
```

Expanded Code

```
                                      445          IF  (SP,COUNTER,=P'0',NZ)
000582 FB40 7A3E 7810 00DD8 00BAA     446+               SP         COUNTER,=P'0'
000588 4780 71F6            00590     447+              BC 15-7,#@LB8
00058C 92FF 7CF8      01092           448              MVI   DONESW,X'FF'
                                      449          ENDIF
                            00590     450+#@LB8        EQU *
```

Source Code

```
         IF  (ICM,R1,B'0001',ZCA,NZ)
             MVI   DVPSW,C'X'
         ENDIF
```

Expanded Code

```
                                      452          IF  (ICM,R1,B'0001',ZCA,NZ)
000590 BF11 7D1B            010B5     453+               ICM        R1,B'0001',ZCA
000594 4780 7202            0059C     454+              BC 15-7,#@LB10
000598 92E7 7CF9      01093           455              MVI   DVPSW,C'X'
                                      456          ENDIF
                            0059C     457+#@LB10       EQU *
```

Source Code

```
         IF  (AR,R2,R3,Z)
             MVI   ERRSW,C'E'

         ENDIF
```

Expanded Code

```
                                      459          IF  (AR,R2,R3,Z)
00059C 1A23                           460+               AR         R2,R3
00059E 4770 720C            005A6     461+              BC 15-8,#@LB12
0005A2 92C5 7CFA      01094           462              MVI   ERRSW,C'E'
                                      463          ENDIF
                            005A6     464+#@LB12       EQU *
```

IF - Example 4

Compound IF Statements

Source Code

```
         IF (CLI,ZCA,EQ,C'A'),OR,(CLI,ZCA,EQ,C'B')
            MVI   DONESW,X'FF'

         ENDIF
```

Expanded Code

```
                                 467          IF (CLI,ZCA,EQ,C'A'),OR,(CLI,ZCA,EQ,C'B')
0005A6 95C1 7D53      010ED      468+                CLI        ZCA,C'A'
0005AA 4780 721C            005B6 469+                BC         8,#@LB15
0005AE 95C2 7D53      010ED      470+                CLI        ZCA,C'B'
0005B2 4770 7220            005BA 471+              BC 15-8,#@LB14
                            005B6 472+#@LB15          EQU *
0005B6 92FF 7D30      010CA      473              MVI   DONESW,X'FF'
                                 474           ENDIF
                            005BA 475+#@LB14       EQU *
```

Source Code

```
         IF (CLI,ZCE,EQ,C'A'),                          X
            OR,(CLI,ZCE,EQ,C'B'),                       X
            OR,(CLI,ZCE,EQ,C'C'),                       X
            OR,(CLI,ZCE,EQ,C'D')
            MVI   DONESW,C'D'

         ENDIF
```

Expanded Code

```
                                 477          IF (CLI,ZCE,EQ,C'A'),                X
                                                    OR,(CLI,ZCE,EQ,C'B'),          X
                                                    OR,(CLI,ZCE,EQ,C'C'),          X
                                                    OR,(CLI,ZCE,EQ,C'D')
0005BA 95C1 7D57      010F1      478+                CLI        ZCE,C'A'
0005BE 4780 7240            005DA 479+                BC         8,#@LB17
0005C2 95C2 7D57      010F1      480+                CLI        ZCE,C'B'
0005C6 4780 7240            005DA 481+                BC         8,#@LB17
0005CA 95C3 7D57      010F1      482+                CLI        ZCE,C'C'
0005CE 4780 7240            005DA 483+                BC         8,#@LB17
0005D2 95C4 7D57      010F1      484+                CLI        ZCE,C'D'
0005D6 4770 7244            005DE 485+              BC 15-8,#@LB16
                            005DA 486+#@LB17          EQU *
0005DA 92C4 7D30      010CA      487              MVI   DONESW,C'D'
                                 488           ENDIF
                            005DE 489+#@LB16       EQU *
```

IF - Example 5

Nested If Statements

Source Code

```
          IF (CP,COUNTER,EQ,=P'0')
             MVI   DONESW,C'X'
             IF (CLI,ZCE,EQ,C'A')
                MVC   ERRSW,C'E'
             ELSE
                MVC   ERRSW,C' '
                IF (CP,ZZIP,EQ,=P'0')
                   MVC   PRTZIP,=C'None '
                ELSE
                   UNPK  PRTZIP,ZZIP
                   OI    PRTZIP+4,X'F0'
                ENDIF
             ENDIF
          ENDIF
```

Expanded Code

```
                                          492          IF (CP,COUNTER,EQ,=P'0')
0005DE F940 7ABE 7890 00E58 00C2A   493+                CP         COUNTER,=P'0'
0005E4 4770 7288            00622   494+               BC 15-8,#@LB18
0005E8 92E7 7D78      01112         495               MVI   DONESW,C'X'
                                    496               IF (CLI,ZCE,EQ,C'A')
0005EC 95C1 7DA7      01141         497+                 CLI        ZCE,C'A'
0005F0 4770 7264            005FE   498+               BC 15-8,#@LB20
0005F4 D200 7D7A 00C5 01114 000C5   499                  MVC   ERRSW,C'E'
                                    500               ELSE
0005FA 47F0 7288            00622   501+                BC   15,#@LB22
                            005FE   502+#@LB20       EQU  *
0005FE D200 7D7A 0040 01114 00040   503                  MVC   ERRSW,C' '
                                    504                  IF (CP,ZZIP,EQ,=P'0')
000604 F920 7D97 7890 01131 00C2A   505+                 CP         ZZIP,=P'0'
00060A 4770 727E            00618   506+               BC 15-8,#@LB23
00060E D204 7D7B 7891 01115 00C2B   507                      MVC   PRTZIP,=C'None '
                                    508                  ELSE
000614 47F0 7288            00622   509+                BC   15,#@LB25
                            00618   510+#@LB23       EQU  *
000618 F342 7D7B 7D97 01115 01131   511                      UNPK  PRTZIP,ZZIP
00061E 96F0 7D7F      01119         512                      OI    PRTZIP+4,X'F0'
                                    513                  ENDIF
                            00622   514+#@LB25       EQU *
                                    515               ENDIF
                            00622   516+#@LB22       EQU *
                                    517           ENDIF
                            00622   518+#@LB18       EQU *
```

IF - Example 6

Using Previously Set Condition Code

Source Code

```
         CPQ   QTY1,QTY1F,QTY2,QTY2F
         IF (EQ)
            MVC   QTYSTAT(5),=C'Equal'
         ELSE
            MVC   QTYSTAT(9),=C'Not Equal'
         ENDIF
```

Expanded Code

```
                                    521          CPQ   QTY1,QTY1F,QTY2,QTY2F
000622 F922 7DB0 7DB6 0114A 01150   522+         CP    QTY1,QTY2    COMPARE INT QTY
000628 4770 7298            00632   523+         BNE   *+10         BR IF NE
00062C F922 7DB3 7DB9 0114D 01153   524+         CP    QTY1F,QTY2F  COMPARE
                                    525          IF (EQ)
000632 4770 72A6            00640   526+             BC 15-8,#@LB26
000636 D204 7DBC 78B6 01156 00C50   527             MVC   QTYSTAT(5),=C'Equal'
                                    528          ELSE
00063C 47F0 72AC            00646   529+             BC   15,#@LB28
                            00640   530+#@LB26       EQU  *
000640 D208 7DBC 78BB 01156 00C55   531             MVC   QTYSTAT(9),=C'Not Equal'
                                    532          ENDIF
                            00646   533+#@LB28      EQU *
```

DO Loops

DO and ENDDO make up a "Do Loop". There are 3 variations of the Do Loop. DO UNTIL, DO WHILE and DO INF. All of them must be terminated with ENDDO. The DOEXIT macro may be used inside a Do Loop to exit from the loop.

DO UNTIL means to do everything between the DO and ENDDO until the condition becomes true. The loop is terminated when the condition becomes true.

DO WHILE means to do everything between the DO and ENDDO while the condition is true. The loop is terminated when the condition becomes false.

Do Loops are controlled (terminated) with a single condition statement that is built into the DO macro. If you use the UNTIL option, that IF statement is executed at the end of the loop - where the ENDDO is placed guaranteeing the the loop will be executed at least once. If you use the WHILE option, the IF statement is tested immediately before the loop starts - meaning that the loop may never be executed.

Only one condition may be placed in an UNTIL or WHILE clause.

Example 1. DO UNTIL & WHILE Statements

```
PERFORM P6000_READ     Priming Read     |   PERFORM P6000_READ    Priming Read
DO UNTIL=(CLI,EOFSW,EQ,EOF)             |   DO WHILE=(CLI,EOFSW,NE,EOF)
   PERFORM P3000_PROCESS                |      PERFORM P3000_PROCESS
   PERFORM P7000_WRITE_PRINTER          |      PERFORM P7000_WRITE_PRINTER
   PERFORM P6000_READ                   |      PERFORM P6000_READ
ENDDO                                   |   ENDDO
```

Note the logic difference between UNTIL and WHILE.

'Until EOF' requires EQ where 'WHILE EOF' requires NE.

Example 2. Nested Do Loops

```
PERFORM P6000_READ_HEADER          Priming Read on Driver File
DO WHILE=(CLI,EOF1SW,NE,C'E')      Loop until EOF on driver
   PERFORM P3000_PRINT_HEADER      Print Header line
   PERFORM P6100_POINT_DETAIL         Start Browse on Detail File
   PERFORM P6200_READNEXT_DETAIL      Priming Read on Detail File
   DO WHILE=(CLI,EOF2SW,NE,C'E')        Stop on EOF or key change
      PERFORM P3100_PRINT_DETAIL        Print Detail line
      PERFORM P6200_READNEXT_DETAIL     Read next detail record
    ENDDO
   PERFORM P6000_READ_HEADER       Read next driver record
ENDDO
```

Nested DO Loops are handy for dealing with a sequential "driver" file and skip-sequential processing on a secondary file - such as customer records (driver) and customer position records (secondary). The above example is a classic Outer and Inner Loop structure.

Example 3. DO INF Statement with DOEXIT

```
PERFORM P6000_READ                 Priming Read
DO INF
   DOEXIT (CLI,EOFW,EQ,C'Y')    Exit at EOF
   PERFORM P3000_PROCESS
   PERFORM P7000_WRITE_PRINTER
   PERFORM P6000_READ
ENDDO
```

DO INF sets up an infinite loop. You need to code something to manually break out of the loop. The DOEXIT macro will branch to the instruction AFTER the ENDDO macro. If you do not want to use DOEXIT, then you must code your own branch statement to a label.

DO - Example 1

DO using UNTIL and WHILE

Note - Only one condition allowed in an UNTIL or WHILE clause.

Source Code

```
     PERFORM P6000_GET_FILE1        Priming read
     DO UNTIL=(CLI,FILE1_CC,EQ,CCEOF)
        PERFORM P6000_GET_FILE1     Get Next FILE1 record
        PERFORM P3000_PROCESS
        PERFORM P7000_PRINT_DETAIL
     ENDDO
```

Expanded Code

```
                                    541          DO UNTIL=(CLI,FILE1_CC,EQ,CCEOF)
                              00652 542+#@LB30       EQU   *
000652 41E0 72C4              0065E 544              PERFORM P6000_GET_FILE1  Get Next Rec
00065E 41E0 72D0              0066A 548              PERFORM P3000_PROCESS
00066A 41E0 72DC              00676 552              PERFORM P7000_PRINT_DETAIL
                                    557          ENDDO
000676 95C5 7EDF      01279         558+                CLI        FILE1_CC,CCEOF
00067A 4770 72B8              00652 559+                BC         15-8,#@LB30
```

In this example, the 3 PERFORM statements inside the DO Loop are executed at least once and will continue to be executed as long as FILE1_CC is NOT EOF

This is the same as 'BNE'

Source Code

```
     PERFORM P6000_GET_FILE1        Priming read
     DO WHILE=(CLI,FILE1_CC,NE,CCEOF)
        PERFORM P6000_GET_FILE1     Get Next FILE1 record
        PERFORM P3000_PROCESS
        PERFORM P7000_PRINT_DETAIL

     ENDDO
```

This branch instruction is used to test the condition BEFORE the code inside the DO Loop is executed

Expanded Code

```
                                    567          DO WHILE=(CLI,FILE1_CC,NE,CCEOF)
00068A 47F0 7318              006B2 568+                 BC    15,#@LB32
                              0068E 569+#@LB33           EQU  * 570
00068E 41E0 7300              0069A 571              PERFORM P6000_GET_FILE1  Get Next Rec
00069A 41E0 730C              006A6 575              PERFORM P3000_PROCESS
0006A6 41E0 7318              006B2 579              PERFORM P7000_PRINT_DETAIL
                                    584          ENDDO
0006B2 95C5 7EDF      01279         585+#@LB32           CLI        FILE1_CC,CCEOF
0006B6 4770 72F4              0068E 586+                 BC         7,#@LB33
```

In this example, the 3 PERFORM statements inside the DO Loop are executed as long as FILE1_CC is NOT set to EOF. It is possible that the instructions inside the DO Loop may not get executed at all

This is the same as 'BNE'

DO - Example 2

Nested DO Loops

Source Code

```
PERFORM P6000_READ_HEADER          Priming Read on Driver
DO WHILE=(CLI,EOF1SW,NE,C'E')      Loop until EOF on Driver
   PERFORM P3000_PRINT_HEADER
   PERFORM P6100_POINT_DETAIL      Start Browse on Detail
   PERFORM P6200_READNEXT_DETAIL   Priming Read on Detail
   DO WHILE=(CLI,EOF2SW,NE,C'E')   Stop on EOF or key change
      PERFORM P3100_PRINT_DETAIL      Print Detail line
      PERFORM P6200_READNEXT_DETAIL   Read next detail rec
   ENDDO
   PERFORM P6000_READ_HEADER       Read next driver record
ENDDO
```

Expanded Code

```
0006BA 41E0 732C            006C6   591          PERFORM P6000_READ_HEADER  Priming Read
                                    596          DO WHILE=(CLI,EOF1SW,NE,C'E') Outer Loop
0006C6 47F0 7384            0071E   597+                BC    15,#@LB35
                            006CA   598+#@LB36          EQU  *
0006CA 41E0 733C            006D6   600             PERFORM P3000_PRINT_HEADER
0006D6 41E0 7348            006E2   604             PERFORM P6100_POINT_DETAIL
0006E2 41E0 7354            006EE   608             PERFORM P6200_READNEXT_DETAIL
                                    613             DO WHILE=(CLI,EOF2SW,NE,C'E')
0006EE 47F0 7370            0070A   614+                BC    15,#@LB38
                            006F2   615+#@LB39          EQU  *
0006F2 41E0 7364            006FE   617                PERFORM P3100_PRINT_DETAIL
0006FE 41E0 7370            0070A   621                PERFORM P6200_READNEXT_DETAIL
                                    626             ENDDO
00070A 95C5 7EFF      01299         627+#@LB38          CLI        EOF2SW,C'E'
00070E 4770 7358            006F2   628+                BC         7,#@LB39
000712 41E0 7384            0071E   630             PERFORM P6000_READ_HEADER
                                    635          ENDDO
00071E 95C5 7EFE      01298         636+#@LB35          CLI        EOF1SW,C'E'
000722 4770 7330            006CA   637+                BC         7,#@LB36
```

DO - Example 3

DO INF with DOEXIT

Source Code

```
       PERFORM P6000_GET_FILE1        Priming read
          DO INF                      Infinite Loop
             DOEXIT (CLI,FILE1_CC,EQ,CCEOF)  Exit at EOF
             DOEXIT (CLC,ZKEY,GE,=C'900')    Stop on 900
             PERFORM P3000_PROCESS
             PERFORM P7000_PRINT_DETAIL
             PERFORM P6000_GET_FILE1
           ENDDO
```

SELECT

The SELECT set of macros provide a classic “Select Case” environment, similar to EVALUATE in COBOL. The SELECT macro sets up a prototype compare instruction without the 2nd operand. The WHEN macro supplies the 2nd operand of the compare instruction for each case you want to test. The OTHRWISE macro captures any condition not tested with WHEN. ENDSEL terminates the sequence.

Example 1. SELECT using CLI

Source Code

```
SELECT CLI,ZCE,EQ
   WHEN (C'A')
      MVC   DetAcTyp(11),=C'Association'
   WHEN (C'B')
      MVC   DetAcTyp(04),=C'401k'
   WHEN (C'C')
      MVC   DetAcTyp(15),=C'Tennants/Common'
   WHEN (C'D')
      MVC   DetAcTyp(09),=C'Community'
   OTHRWISE
      MVC   DetAcTyp(1),ZCE
ENDSEL
```

Expanded Code

```
                                        683          Select CLI,ZCE,EQ
                                        684             When (C'A')
000774 95C1 7FC7      01361             685+         CLI           ZCE,C'A'
000778 4770 73EC            00786       686+         BC    15-8,#@LB47
00077C D20A 7EDB 7A7F 01275 00E19       687                MVC   DetAcTyp(11),=C'Association'
                                        688             When (C'B')
000782 47F0 7428            007C2       689+         B     #@LB46             SKIP TO END
                            00786       690+#@LB47         EQU *
01-WHEN
000786 95C2 7FC7      01361             691+         CLI           ZCE,C'B'
00078A 4770 73FE            00798       692+         BC    15-8,#@LB49
00078E D203 7EDB 79C6 01275 00D60       693                MVC   DetAcTyp(04),=C'401k'
                                        694             When (C'C')
000794 47F0 7428            007C2       695+         B     #@LB46             SKIP TO END
                            00798       696+#@LB49         EQU *
000798 95C3 7FC7      01361             697+         CLI           ZCE,C'C'
00079C 4770 7410            007AA       698+         BC    15-8,#@LB51
0007A0 D20E 7EDB 7A8A 01275 00E24       699                MVC   DetAcTyp(15),=C'Tennants/Common'
                                        700             When (C'D')
0007A6 47F0 7428            007C2       701+         B     #@LB46             SKIP TO END
                            007AA       702+#@LB51         EQU *
0007AA 95C4 7FC7      01361             703+         CLI           ZCE,C'D'
0007AE 4770 7422            007BC       704+         BC    15-8,#@LB53
0007B2 D208 7EDB 7A99 01275 00E33       705                MVC   DetAcTyp(09),=C'Community'
                                        706             Othrwise
0007B8 47F0 7428            007C2       707+         B     #@LB46             SKIP TO END
                            007BC       708+#@LB53         EQU *
0007BC D200 7EDB 7FC7 01275 01361       709                MVC   DetAcTyp(1),ZCE
                                        710          ENDSEL
                            007C2       711+#@LB46          EQU *
```

PERFORM Macros

The “Perform” set of macros allow you to declare COBOL-type paragraphs and Perform them as subroutines. They also support the setting and testing of condition codes associated with paragraphs. The macros in this set are:

PERFORM	Performs a paragraph (Similar to ‘BAL Rx,Name’)
PARA	Declares the beginning of a paragraph
ENDPARA	Declares the end of a paragraph
EXITPARA	Branches to the end of the current paragraph
SETCC	Sets a condition code associated with the current paragraph or a file
PFMLIST	Defines work areas needed by the PERFORM and SETCC macros

These macros provide the following benefits:

- Simulates COBOL-like Perform logic
- Allows unlimited nesting of performs providing that there are NO recursive performs (same rule as in COBOL and many other high-level languages).
- Does not tie up any registers.
- Enhances the ability of the programmer to structure a program.
- Supports the generation and setting of condition codes
- Works well with IBM's Structured macros

Rules for using these macros:

- A paragraph MUST be Performed. If you branch to a paragraph or let your program flow into a paragraph, the program will ABEND with an addressing exception when the ENDPARA macro is encountered. This is done to enforce the integrity of the paragraph structure.
- Unless you are exiting your program, you must always take the perform exit ('ENDPARA'). Good structured practices strongly recommend doing this.

Example.

```
          PERFORM P6000_GET_FILE1       Priming read
          DO WHILE=(CLI,FILE1_CC,NE,CCEOF)
             PERFORM P6000_GET_FILE1    Get Next FILE1 record
             PERFORM P3000_PROCESS
             PERFORM P7000_PRINT_DETAIL
          ENDDO


P6000_GET_FILE1   PARA
          VSAMIO READNEXT,FILE1,WORKA=ZREC
          IFERROR (EQ,EOF),ERR=VERR
             SETCC FILE1,EOF
          ELSE
             COUNT 'FILE1 Records Read'
          ENDIF
          ENDPARA
```

PERFORM - Example

Source Code

(See previous page)

Expanded Code

```
                                        564          PERFORM P6000_GET_FILE1
00067E 41E0 72F0            0068A       565+         LA    R14,*+12
000682 50E0 7F6E            01308       566+         ST    R14,#Save_P6000_GET_FILE1
000686 47F0 765E            009F8       567+         B     P6000_GET_FILE1
00068A 47F0 7318            006B2       569          DO WHILE=(CLI,FILE1_CC,NE,CCEOF)
                                        573             PERFORM P6000_GET_FILE1
00068E 41E0 7300            0069A       574+         LA    R14,*+12
000692 50E0 7F6E            01308       575+         ST    R14,#Save_P6000_GET_FILE1
000696 47F0 765E            009F8       576+         B     P6000_GET_FILE1
00069A 41E0 730C            006A6       578             PERFORM P3000_PROCESS
0006A6 41E0 7318            006B2       582             PERFORM P7000_PRINT_DETAIL
0006B2 95C5 7F7F      01319             587          ENDDO



                                        974 P6000_GET_FILE1   PARA
                            009F8       975+P6000_GET_FILE1 EQU *
                                        976          VSAMIO READNEXT,FILE1,WORKA=ZREC
                                        990          IFERROR (EQ,EOF),ERR=VERR
000A18 91DF 101E      0001E             991+               TM         30(R1),255-32
000A1C 4780 768A            00A24       992+              BC 15-7,#@LB103
000A20 47F0 77FE            00B98       993+             B     VERR
                            00A24       994+#@LB103      EQU *
000A24 9120 101E      0001E             995+               TM         30(R1),32
000A28 4780 769A            00A34       996+              BC 15-7,#@LB105
                                        997              SETCC FILE1,EOF
000A2C 92C5 7F7F      01319             998+         MVI   FILE1_CC,CCEOF
                                        999          ELSE
000A30 47F0 76A6            00A40  1000+              BC   15,#@LB107
                            00A34  1001+#@LB105       EQU  *
                                   1002              COUNT 'FILE1 Records Read'
                                   1006         ENDIF
                                   1008         ENDPARA
000A40 58E0 7F6E            01308  1009+         L     R14,#SAVE_P6000_GET_FILE1
000A44 D203 7F6E 79CA 01308 00D64  1010+         MVC   #SAVE_P6000_GET_FILE1,=F'0'
000A4A 07FE                        1014+         BR    R14
```

EXITPARA

The EXITPARA macro is used to leave a paragraph without coding a branch instruction and an associated label. It is accomplished by branching to an internal generated label in the ENDPARA macro. Although the generated code is exactly the same as if it was coded manually, it does serve to reduce the number of labels and subsequent label references in a program.

Source Code

```
P3100_PROCESS   PARA
         MVC   DetAcct+0(3),ZKEY       Move Acct No to Print Line
         MVI   DetAcct+3,C'-'
         MVC   DetAcct+4(6),ZKEY

         MVC   DetName1,ZNA1           Move in Name & Addr info
         MVC   DetName2,ZNA2

         IF (TM,ZCJ,X'01',NO)          If not managed account
             EXITPARA                     Exit
         ENDIF
         Select CLI,ZCE,EQ
            When (C'A')
               MVC   DetAcTyp(11),=C'Association'
            When (C'B')
               MVC   DetAcTyp(04),=C'401k'
            Othrwise
               MVC   DetAcTyp(1),ZCE
         ENDSEL

         ENDPARA
```

Expanded Code

```
                                      959 P3100_PROCESS   PARA
                              009F8   960+P3100_PROCESS EQU *
01-PARA
0009F8 D202 7F2F 8013 012C9 013AD   961          MVC   DetAcct+0(3),ZKEY
0009FE 9260 7F32      012CC         962          MVI   DetAcct+3,C'-'
000A02 D205 7F33 8013 012CD 013AD   963          MVC   DetAcct+4(6),ZKEY
000A08 D21D 7F4B 8068 012E5 01402   965          MVC   DetName1,ZNA1
000A0E D21D 7F6A 8086 01304 01420   966          MVC   DetName2,ZNA2
                                    968          IF (TM,ZCJ,X'01',NO)  If not managed
000A14 9101 8034      013CE         969+               TM         ZCJ,X'01'
000A18 4710 7686            00A20   970+              BC 15-14,#@LB103
                                    971               EXITPARA             Exit
000A1C 47F0 76B0            00A4A   972+         B     P3100_PROCESS_Exit
                                    973          ENDIF
                              00A20 974+#@LB103     EQU *
                                    975          PRINT NOGEN
000A20 95C1 802F      013C9         977             When (C'A')
000A28 D20A 7F3B 7ADF 012D5 00E79   980                MVC   DetAcTyp(11),=C'Association'
000A2E 47F0 76B0            00A4A   981             When (C'B')
000A3A D203 7F3B 7A26 012D5 00DC0   986                MVC   DetAcTyp(04),=C'401k'
000A40 47F0 76B0            00A4A   987             Othrwise
000A44 D200 7F3B 802F 012D5 013C9   990                MVC   DetAcTyp(1),ZCE
                                    991          ENDSEL
                                    994          ENDPARA
                              00A4A 995+P3100_PROCESS_Exit EQU *
000A4A 58E0 7FCE            01368   996+         L     R14,#SAVE_P3100_PROCESS
000A4E D203 7FCE 7A2A 01368 00DC4   997+         MVC   #SAVE_P3100_PROCESS,=F'0'
000A54 07FE                        1001+         BR    R14
```

SETCC Macro

This macro sets a condition code associated with any file or paragraph. SETCC is used to add structured programming functionality by standardizing the way condition codes are set in a subroutine and tested by the caller. It supports the premise that a subroutine should have a return code that can be tested making it easier for the caller to determine the outcome of the subroutine (or function).

SETCC generates a MVI instruction that 'sets' the condition code. The condition code specified must be one of these names:

EOF	NotEOF	Found	NotFnd	Good	Bad	Pass	Fail
True	False	Ok	NotOk	EQ	NE	Equal	NotEQ
Z	NZ	Zero	Nzero	Low	High	NotLow	NotHi

The names generated are generated for you in the PFMLIST macro. All of them are prefaced with CC. In the SETCC macro, specification of CC is optional. If you don't supply the CC prefix, the macro will do it for you. Therefore specifying ‘Fail’ as a condition code will generate 'CCFail’.

There are two formats of this macro:

- Format 1 - Set File Condition Code - **'SETCC FILE1,EOF'**

  This format sets a condition code associated with any file declared using VSAMIO file macros. The PFMLIST macro generates a list of one-byte condition codes for each delcared file. The names of these condition codes look like this:

```
                    FILE1_CC   DC    C' '
```

  Macro Format:

```
                    SETCC FILE1,EOF
      generates >>> MVI   FILE1_CC,CCEOF
```

- Format 2 - Set Paragraph Condition Code - **'SETCC EOF’**

  This format sets a condition code associated with the current Paragraph. The PFMLIST macro generates a list of one-byte condition codes for each paragraph declared with the PARA macro. The names of these condition codes look like this:

```
                    P4200_Search_CC  DC    C' '
```

  Macro Format:

```
                    SETCC FOUND
      generates >>> MVI   P4200_Search_CC,CCFOUND
```

PFMLIST Macro

The PFMLIST macro is coded at the end of your program to generate the following fields:

- Save Areas for each paragraph (PARA macro) - 4-byte fullword for each paragraph
- Condition Codes for each paragraph - 1byte per paragraph
- Condition Codes for each file - 1 byte per file declared with VSAMIO macros. This includes RDNA, RDSEC and RDSYM.
- Constants for commonly used conditions (True, False, EOF, NotEof, etc)

```
                          1516          PFMLIST
                          1517+*************************************************
                          1518+*        PERFORM Save Cells                      *
                          1519+*************************************************
0012ED 000000
0012F0                    1520+         DC    0F'0'
                    012F0 1521+PFMLIST  EQU   *
0012F0 00000000           1522+#SAVE_P6000_READ_HEADER              DC    F'0'
0012F4 00000000           1523+#SAVE_P3000_PRINT_HEADER             DC    F'0'
0012F8 00000000           1524+#SAVE_P3100_PRINT_DETAIL             DC    F'0'
0012FC 00000000           1525+#SAVE_P6100_POINT_DETAIL             DC    F'0'
001300 00000000           1526+#SAVE_P6200_READNEXT_DETAIL          DC    F'0'
001304 00000000           1527+#SAVE_P3000_PROCESS                  DC    F'0'
001308 00000000           1528+#SAVE_P6000_GET_FILE1                DC    F'0'
00130C 00000000           1529+#SAVE_P7000_PRINT_DETAIL             DC    F'0'
                          1530+PFMLISTL EQU   *-PFMLIST

                          1532+*************************************************
                          1533+*        Condition Codes                         *
                          1534+*************************************************
001310                    1535+         DC    0F'0'
                          1536+CCCodes  EQU   *
001310 00                 1537+P6000_READ_HEADER_CC                 DC   X'00'
001311 00                 1538+P3000_PRINT_HEADER_CC                DC   X'00'
001312 00                 1539+P3100_PRINT_DETAIL_CC                DC   X'00'
001313 00                 1540+P6100_POINT_DETAIL_CC                DC   X'00'
001314 00                 1541+P6200_READNEXT_DETAIL_CC             DC   X'00'
001315 00                 1542+P3000_PROCESS_CC                     DC   X'00'
001316 00                 1543+P6000_GET_FILE1_CC                   DC   X'00'
001317 00                 1544+P7000_PRINT_DETAIL_CC                DC   X'00'

                          1546+*************************************************
                          1547+*        File IO Switches                        *
                          1548+*************************************************
001318                    1549+         DC    0F'0'
                    01318 1550+IOCodes  EQU   *
001318 00                 1551+PRINTER_CC       DC    X'00'
001319 00                 1552+FILE1_CC         DC    X'00'

                          1554+*************************************************
                          1555+*        Condition Code Values                   *
                          1556+*************************************************
                    000C5 1557+CCEOF    EQU   C'E'
                    000D5 1558+CCNotEOF EQU   C'N'
                    000E8 1559+CCFound  EQU   C'Y'
                    000D5 1560+CCNotfnd EQU   C'N'
                    000E8 1561+CCGood   EQU   C'Y'
                    000D5 1562+CCBad    EQU   C'N'
                    000E8 1563+CCPass   EQU   C'Y'
                    000D5 1564+CCFail   EQU   C'N'
                    000E8 1565+CCTrue   EQU   C'Y'
                    000D5 1566+CCFalse  EQU   C'N'
                    000E8 1567+CCOk     EQU   C'Y'
                    000D5 1568+CCNotOk  EQU   C'N'
                    000E8 1569+CCEQ     EQU   C'Y'
                    000D5 1570+CCNE     EQU   C'N'
                    000E8 1571+CCEqual  EQU   C'Y'
                    000D5 1572+CCNotEq  EQU   C'N'
                    000E9 1573+CCZ      EQU   C'Z'
                    000D5 1574+CCNZ     EQU   C'N'
                    000E9 1575+CCZero   EQU   C'Z'
                    000D5 1576+CCNZero  EQU   C'N'
                    000D3 1577+CCLow    EQU   C'L'
                    000C8 1578+CCHigh   EQU   C'H'
                    0006E 1579+CCNotLow EQU   C'>'
                    0004C 1580+CCNotHi  EQU   C'<'
```

IFERROR Macro

The IFERROR macro is used to replace VSAMIO ERROR when you want to capture a recoverable error condition with an IF statement instead of a Branch and Label as required by VSAMIO ERROR. The format of IFERROR is:

```
IFERROR (operator,condition),ERR=label
ELSE   (optional)
ENDIF
```

Operator may be:

- EQ Equal
- NE Not Equal

Condition may be one of the following (Same names as used in VSAMIO ERROR)

- NOREC No record found
- NOTFND No record found
- EOF End of File
- SEQERR Sequence Error (during LOAD)
- DUPREC Duplicate Record - when attempting to insert a new record

ERR=Label serves the same purpose as in VSAMIO ERROR:

- Declare the name of a label to handle unrecoverable errors.

Example Using VSAMIO ERROR

```
P6000_GET_FILE1   PARA
         VSAMIO READNEXT,FILE1,WORKA=ZREC
         VSAMIO ERROR,(R1),EOF=P6000_EOF,ERR=VERR
         COUNT 'FILE1 Records Read'
         B     P6000_EXIT

P6000_EOF   EQU   *
         SETCC FILE1,EOF

P6000_EXIT  EQU   *
         ENDPARA
```

Note that the VSAMIO ERROR technique require the use of 2 extra labels and a branch instruction

Example Using IFERROR

```
P6000_GET_FILE1   PARA
         VSAMIO READNEXT,FILE1,WORKA=ZREC
         IFERROR (EQ,EOF),ERR=VERR
            SETCC FILE1,EOF
         ELSE
            COUNT 'FILE1 Records Read'
         ENDIF
         ENDPARA
```

Note that no extra labels are required.

IFERROR Example

Source Code

```
P6000_GET_FILE1   PARA
         VSAMIO READNEXT,FILE1,WORKA=ZREC
         IFERROR (EQ,EOF),ERR=VERR
            SETCC FILE1,EOF
         ELSE
            COUNT 'FILE1 Records Read'
         ENDIF
         ENDPARA
```

Expanded Code

```
                                    976          VSAMIO READNEXT,FILE1,WORKA=ZREC
                                    990          IFERROR (EQ,EOF),ERR=VERR
000A18 91DF 101E      0001E         991+               TM          30(R1),255-32
000A1C 4780 768A            00A24   992+              BC 15-7,#@LB103
000A20 47F0 77FE            00B98   993+             B     VERR
                            00A24   994+#@LB103      EQU *
000A24 9120 101E      0001E         995+               TM          30(R1),32
000A28 4780 769A            00A34   996+              BC 15-7,#@LB105
                                    997             SETCC FILE1,EOF
000A2C 92C5 7F7F      01319         998+          MVI   FILE1_CC,CCEOF
                                    999           ELSE
000A30 47F0 76A6            00A40  1000+              BC   15,#@LB107
                            00A34  1001+#@LB105       EQU  *
                                   1002             COUNT 'FILE1 Records Read'
                                   1006          ENDIF
```

Fatal Error test. This checks for any bit other than the one requested in parenthesis for non-zero,

Error Condition Test. The desired condition is tested here.

Note that SETCC automatically adds the _CC extension to the file name and prefixes the condition (EOF) with CC

IFERROR actually needs to check for two conditions. First, it needs to see if there is a fatal error; in which case it branches to the label specified in the ERR parameter. Then it checks to see if the desired condition (EOF, NOREC, etc.) has occurred.

The fatal-error test is accomplished by testing all bits except for the one specified in parenthesis. All of these bits should be zero. If one or more are on, it means a fatal or unexpected error has occurred and a branch to the ERR label is taken.

The actual condition test is handled by a single-bit TM instruction for an ON condition. The IFERROR macro uses an in-line IF macro to do this, requiring you code ELSE (optional) and ENDIF macros right after the IFERROR macro.

Sample Program Using Structured Macros (1)

```
*************************************************************************
*                                                                       *
*  Program CSDSTR1      Structured Macro Demonstration Program          *
*                                                                       *
*  This program utilizes Structured Macros to demonstrate their use.   *
*  It can be contrasted with program CSDSTR2 which is essentially       *
*  same program - only it does NOT use structured macros.               *
*                                                                       *
*  This program reads each FILE1 record, selecting those that are       *
*  managed accounts. For each selected FILE1 record, it prints one      *
*  detail line. To speed up processing, the program stops after         *
*  printing 100 records.                                                *
*                                                                       *
*************************************************************************
         PRINT NOGEN
         LEVEL 1
         START X'108'
FILE1    VSAMIO DEFINE,TYPE=KSDS,IO=INPUT,ACCESS=SEQ
         STARTUP PRINT=132
         HDR   INITIALIZE,1            Initialize Page Headings



*************************************************************************
*        Mainline Processing                                            *
*************************************************************************
MAINLINE EQU   *
         ZAP   COUNTER,=P'100'
         PERFORM P6000_GET_FILE1       Priming read
         DO UNTIL=(CLI,FILE1_CC,EQ,CCEOF)
            IF (TM,ZCJ,X'01',O)        If managed account
               PERFORM P3000_PROCESS
               PERFORM P7000_PRINT_DETAIL
               COUNT 'Managed Accounts Processed'
               IF (SP,COUNTER,=P'1',Z)
                  SETCC FILE1,CCEOF
               ENDIF
            ENDIF
            PERFORM P6000_GET_FILE1    Get Next FILE1 record
         ENDDO
         B     EOJ
```

Sample Program Using Structured Macros (2)

```
*********************************************************************
*        P r o c e s s                                              *
*********************************************************************
P3000_PROCESS    PARA
         MVC   DetAcct+0(3),ZKEY       Move Acct No to Print Line
         MVI   DetAcct+3,C'-'
         MVC   DetAcct+4(6),ZKEY

         MVC   DetName1,ZNA1           Move in Name & Addr info
         MVC   DetName2,ZNA2
         Select CLI,ZCE,EQ
            When (C'A')
               MVC   DetAcTyp(11),=C'Association'
            When (C'B')
               MVC   DetAcTyp(04),=C'401k'
            When (C'C')
               MVC   DetAcTyp(15),=C'Tennants/Common'
            When (C'D')
               MVC   DetAcTyp(09),=C'Community'
            When (C'E')
               MVC   DetAcTyp(10),=C'Entireties'
            When (C'F')
               MVC   DetAcTyp(09),=C'Fiduciary'
            When (C'G')
               MVC   DetAcTyp(04),=C'Bank'
            When (C'H')
               MVC   DetAcTyp(14),=C'Investment Co.'
            When (C'I')
               MVC   DetAcTyp(15),=C'Invest. Counsel'
            When (C'J')
               MVC   DetAcTyp(09),=C'Joint Tenant'
            When (C'K')
               MVC   DetAcTyp(15),=C'Investment Club'
            When (C'L')
               MVC   DetAcTyp(07),=C'Pension'
            When (C'M')
               MVC   DetAcTyp(15),=C'Commercial Bank'
            When (C'N')
               MVC   DetAcTyp(12),=C'Savings Bank'
            When (C'O')
               MVC   DetAcTyp(10),=C'Sole Prop.'
            When (C'P')
               MVC   DetAcTyp(11),=C'Partnership'
            When (C'S')
               MVC   DetAcTyp(14),=C'Single Account'
            When (C'Z')
               MVC   DetAcTyp(09),=C'Custodial'
            Othrwise
               MVC   DetAcTyp(1),ZCE
         ENDSEL

         IF (CLI,ZFNDCDE,GT,C' ')
            MVC   DetFund(1),ZFNDCDE
         ELSE
            MVC   DetFund,=C'None'
         ENDIF

         IF (CLI,ZZIP,EQ,X'55')        Is this a Metro Zip Code?
            MVI   DetMetro,C'Y'
         ENDIF
         ENDPARA
```

Sample Program Using Structured Macros (3)

```
*************************************************************************
* 6000_Get_FILE1                                        Subroutine      *
*                                                                       *
* Description:                                                          *
*     This routine reads the next sequential FILE1 record.              *
*                                                                       *
* Requirements:                                                         *
*     DTR 'FILE1,Z' must contain the last FILE1 record read.            *
*                                                                       *
* Return Code:                                                          *
*     The built-in switch 'FILE1_CC' will be set to 'EOF' after the     *
*     last record has been read.                                        *
*                                                                       *
*************************************************************************
         PRINT GEN
P6000_GET_FILE1   PARA
         VSAMIO READNEXT,FILE1,WORKA=ZREC
         IFERROR (EQ,EOF),ERR=VERR
            SETCC FILE1,EOF
         ELSE
            COUNT 'FILE1 Records Read'
         ENDIF
         ENDPARA

*************************************************************************
* 7000_Print_Detail                                     Subroutine      *
*                                                                       *
* Description:                                                          *
*     This routine prints a detail line using PRINTLN.                  *
*                                                                       *
* Requirements:                                                         *
*     The detail line (Detail1) must be previously formatted.           *
*                                                                       *
* Return Code:                                                          *
*     None                                                              *
*                                                                       *
*************************************************************************
P7000_PRINT_DETAIL    PARA
         BAL   R6,PRINTLN
         ENDPARA

         PRINT NOGEN
*************************************************************************
*        P R I N T L N     R O U T I N E                                *
*************************************************************************
         PRINTLN                                                       *
               PRINTER=PRINTER,    PRINTER NAME                        *
               OUT=Detail1,        PRINTER WORK AREA                   *
               LNCNT=LNCNT1,       LINE COUNTER                        *
               LNMAX=57,           MAX LINES PER PAGE                  *
               PGCNT=PGCNT1,       PAGE COUNTER                        *
               PGFMT=HD1PG,        FORMATTED PAGE NUMBER IN HEADER     *
               HD=(HD1A,HD1B,1,HD2,HD3),                               *
               SK=NO,              SKIP ROUTINES                       *
               SP=1,               SPACE ROUTINES                      *
               L=R6,               LINK REGISTER                       *
               CLEAR=YES,          Clear Print Line after Printing     *
               SUFFIX=             LABEL SUFFIX
         EJECT
```

Sample Program Using Structured Macros (4)

```
*********************************************************************
*        PROCESS VSAM ERROR                                         *
*********************************************************************
VERR     EQU   *
         MSG   'VSAM ERROR ',(34(R1),50),L=R6
         B     EOJ

EOJ      EQU   *
         PRINT NOGEN
         ENDPROG
         EJECT
         PRINT GEN
*********************************************************************
*        WORKING STORAGE                                            *
*********************************************************************
COUNTER  DC    PL5'0'

*********************************************************************
*        Print Lines and Related Work Areas                         *
*********************************************************************
         HDR   DEFINE,1,TITLE1='Name and Address Report',          *
               TITLE2='for managed accounts'

HD2      DS    0CL132
         DC    C'Acct No     Account Type    Name Field 1          *
                    Name Field 2                  Fund Metro?'
         DC    (132-(*-HD2))CL1' '

HD3      DS    0CL132
         DC    C'----------- --------------- -------------------------X
               ---- ------------------------------ ---- ------'
         DC    (132-(*-HD3))CL1' '

LNCNT1   DC    PL2'0'
PGCNT1   DC    PL2'0'

         DC    c' '
Detail1  DS    0CL132
DetAcct  DS    CL11
         DS    C
DetAcTyp DS    CL15
         DS    c
DetName1 DS    CL30
         DS    C
DetName2 DS    CL30
         DS    C
DetFund  DS    CL4
         DS    CL3
DetMetro DS    C
         DC    (132-(*-Detail1))CL1' '
         eject
         PFMLIST

*********************************************************************
*        F I L E 1           NAME AND ADDRESS FILE                  *
*********************************************************************
         DC    0D'0'
         DTR   FILE1,Z        Generate Copy Book
         END   BEGIN
```

Sample Program Without Structured Macros (1)

```
*********************************************************************
*                                                                   *
*  Program CSDSTR2      Structured Macro Demonstaration Program     *
*                                                                   *
*  This program is used to contrast it to CSDSTR1, a program written  *
*  using Structured Macros. This program does NOT use Structured    *
*  Macros.                                                          *
*                                                                   *
*  This program reads each FILE1 record, selecting those that are   *
*  managed accounts. For each selected FILE1 record, it prints one  *
*  detail line. To speed up processing, the program stops after     *
*  printing 100 records.                                            *
*                                                                   *
*********************************************************************
         PRINT NOGEN
         LEVEL 1
         START X'108'
FILE1    VSAMIO DEFINE,TYPE=KSDS,IO=INPUT,ACCESS=SEQ
         STARTUP PRINT=132
         HDR   INITIALIZE,1            Initialize Page Headings

MAINLINE EQU   *
         ZAP   COUNTER,=P'100'         Setup record limit counter
         BAL   R6,GETFILE1             Priming read
MAINLOOP EQU   *
         CLI   EOFSW,EOF               Are we at EOF
         BE    EOJ                        Yes, done
         TM    ZCJ,X'01'               Skip record if not managed
         BNO   NEXTREC                    Not managed, skip it
         BAL   R6,PROCESS              Process output line
         BAL   R6,PRINTLN              Print the line
         COUNT 'Managed Accounts Processed'
         SP    COUNTER,=P'1'           Reduce limit counter
         BNZ   NEXTREC
         MVI   EOFSW,EOF               Force EOF
NEXTREC  EQU   *
         BAL   R6,GETFILE1             Read next FILE1 record
         B     MAINLOOP
```

Sample Program Without Structured Macros (2)

```
*********************************************************************
*        P r o c e s s                                              *
*********************************************************************
PROCESS  EQU   *
         MVC   DetAcct+0(3),ZKEY       Move Acct No to Print Line
         MVI   DetAcct+3,C'-'
         MVC   DetAcct+4(6),ZKEY
         MVC   DetName1,ZNA1           Move in Name & Addr info
         MVC   DetName2,ZNA2

CHKA     CLI   ZCE,C'A'
         BNE   CHKB
         MVC   DetAcTyp(11),=C'Association'
         B     CHKDONE
CHKB     CLI   ZCE,C'B'
         BNE   CHKC
         MVC   DetAcTyp(04),=C'401k'
         B     CHKDONE
CHKC     CLI   ZCE,C'C'
         BNE   CHKD
         MVC   DetAcTyp(15),=C'Tennants/Common'
         B     CHKDONE
CHKD     CLI   ZCE,C'D'
         BNE   CHKE
         MVC   DetAcTyp(09),=C'Community'
         B     CHKDONE
CHKE     CLI   ZCE,C'E'
         BNE   CHKF
         MVC   DetAcTyp(10),=C'Entireties'
         B     CHKDONE
CHKF     CLI   ZCE,C'F'
         BNE   CHKG
         MVC   DetAcTyp(09),=C'Fiduciary'
         B     CHKDONE
CHKG     CLI   ZCE,C'G'
         BNE   CHKH
         MVC   DetAcTyp(04),=C'Bank'
         B     CHKDONE
CHKH     CLI   ZCE,C'H'
         BNE   CHKI
         MVC   DetAcTyp(14),=C'Investment Co.'
         B     CHKDONE
CHKI     CLI   ZCE,C'I'
         BNE   CHKJ
         MVC   DetAcTyp(15),=C'Invest. Counsel'
         B     CHKDONE
CHKJ     CLI   ZCE,C'J'
         BNE   CHKK
         MVC   DetAcTyp(09),=C'Joint Tenant'
         B     CHKDONE
CHKK     CLI   ZCE,C'K'
         BNE   CHKL
         MVC   DetAcTyp(15),=C'Investment Club'
         B     CHKDONE
CHKL     CLI   ZCE,C'L'
         BNE   CHKM
         MVC   DetAcTyp(07),=C'Pension'
         B     CHKDONE
CHKM     CLI   ZCE,C'M'
         BNE   CHKN
         MVC   DetAcTyp(15),=C'Commercial Bank'
         B     CHKDONE
CHKN     CLI   ZCE,C'N'
         BNE   CHKO
         MVC   DetAcTyp(12),=C'Savings Bank'
         B     CHKDONE
```

Sample Program Without Structured Macros (3)

```
CHKO     CLI   ZCE,C'O'
         BNE   CHKP
         MVC   DetAcTyp(10),=C'Sole Prop.'
         B     CHKDONE
CHKP     CLI   ZCE,C'P'
         BNE   CHKS
         MVC   DetAcTyp(11),=C'Partnership'
         B     CHKDONE
CHKS     CLI   ZCE,C'S'
         BNE   CHKZ
         MVC   DetAcTyp(14),=C'Single Account'
         B     CHKDONE
CHKZ     CLI   ZCE,C'Z'
         BNE   CHKOTHER
         MVC   DetAcTyp(09),=C'Custodial'
         B     CHKDONE
CHKOTHER EQU   *
         MVC   DetAcTyp(1),ZCE
CHKDONE  EQU   *

         CLI   ZFNDCDE,C' '            Is there a fund code
         BNH   NOFUND                     No, Indicate None
         MVC   DetFund(1),ZFNDCDE         Yes, Move in Fund Code
         B     FUNDX
NOFUND   EQU   *
         MVC   DetFund,=C'None'        No Fund code
FUNDX    EQU   *

         CLI   ZZIP,X'55'              Is this a Metro Zip Code?
         BNE   NONMETRO                   No, Not Metro
         MVI   DetMetro,C'Y'              Yes, Indicate Metro
NONMETRO EQU   *
         BR    R6


*********************************************************************
*        Get Next FILE1 Record                                      *
*********************************************************************
GETFILE1 EQU   *
         VSAMIO READNEXT,FILE1,WORKA=ZREC
         VSAMIO ERROR,(R1),EOF=GETNADA7,ERR=VERR
         COUNT 'FILE1 Records Read'
         B     GETNADA9
GETNADA7 EQU   *
         MVI   EOFSW,EOF               Indicate EOF
GETNADA9 EQU   *
         BR    R6

*********************************************************************
*        P U T P R     R O U T I N E                                *
*********************************************************************
         PRINTLN                                                    *
               PRINTER=PRINTER,    PRINTER NAME                     *
               OUT=Detail1,        PRINTER WORK AREA                *
               LNCNT=LNCNT1,       LINE COUNTER                     *
               LNMAX=57,           MAX LINES PER PAGE               *
               PGCNT=PGCNT1,       PAGE COUNTER                     *
               PGFMT=HD1PG,        FORMATTED PAGE NUMBER IN HEADER  *
               HD=(HD1A,HD1B,1,HD2,HD3),                            *
               SK=NO,              SKIP ROUTINES                    *
               SP=1,               SPACE ROUTINES                   *
               L=R6,               LINK REGISTER                    *
               CLEAR=YES,          Clear Print Line after Printing  *
               SUFFIX=             LABEL SUFFIX
```

Sample Program Without Structured Macros (4)

```
*********************************************************************
*        PROCESS VSAM ERROR                                         *
*********************************************************************
VERR     EQU   *
         MSG   'VSAM ERROR ',(34(R1),50),L=R6
         B     EOJ

EOJ      EQU   *
         PRINT NOGEN
         ENDPROG
         EJECT
         PRINT GEN
*********************************************************************
*        WORKING STORAGE                                            *
*********************************************************************
EOFSW    DS    C
EOF      EQU   C'E'
COUNTER  DC    PL5'0'

*********************************************************************
*        Print Lines and Related Work Areas                         *
*********************************************************************
         HDR   DEFINE,1,TITLE1='Name and Address Report',           *
               TITLE2='for managed accounts'

HD2      DS    0CL132
         DC    C'Acct No     Account Type    Name Field 1           *
                    Name Field 2                  Fund Metro?'
         DC    (132-(*-HD2))CL1' '

HD3      DS    0CL132
         DC    C'----------- --------------- -------------------------X
               ---- ------------------------------ ---- ------'
         DC    (132-(*-HD3))CL1' '

LNCNT1   DC    PL2'0'
PGCNT1   DC    PL2'0'

         DC    c' '
Detail1  DS    0CL132
DetAcct  DS    CL11
         DS    C
DetAcTyp DS    CL15
         DS    c
DetName1 DS    CL30
         DS    C
DetName2 DS    CL30
         DS    C
DetFund  DS    CL4
         DS    CL3
DetMetro DS    C
         DC    (132-(*-Detail1))CL1' '
         EJECT
*********************************************************************
*        F I L E 1           NAME AND ADDRESS FILE                  *
*********************************************************************
         DC    0D'0'
         DTR   FILE1,Z      Generate File Copy Book

         END   BEGIN
```

CICS Considerations

Structured Macros work well in CICS as well as batch. Only slight modifications to a CICS program are required.

The structured macros require the **PFMLIST** macro to declare necessary work areas used by the other macros. However, **PFMLIST**:

- Must come after all other structured macros and
- Must be in Dynamic Storage and
- Not be part of the COMMAREA

Most assembler CICS programs have the Dynamic Storage (**DFHEISTG**) declared in the front of the program, rendering a **PFMLIST** macro useless if it comes before other structured macros. To solve this problem:

- Extend the DFHEISTG DSECT. Insert the following statements just before the END statement:

```
DFHEISTG DSECT
         PFMLIST
```

  Be sure that PFMLIST is NOT part of the COMMAREA

CICS Example using Structured Macros

```
*********************************************************************
*                                                                   *
*   Receive Map   Using HANDLE CONDITION           Subroutine       *
*                                                                   *
*********************************************************************
P7050_Map_In_Handle   Para
         EXEC CICS                                                 X
                   HANDLE CONDITION                                X
                   MAPFAIL(P7057_MAP_IN_MAPFAIL)

         EXEC CICS                                                 X
                   RECEIVE                                         X
                   MAPSET   ('TSTMAPA')                            X
                   MAP      ('TSTMAPA')                            X
                   INTO     (TSTMAPAI)
         EXITPARA                      Exit this Paragraph

P7057_Map_In_Mapfail   EQU   *
         MLONG TSTMAPAS,FILL=00,L1=TSTMAPAL   Clear Map Area to 00

         ENDPARA
```