```
* IDENTIFICATION DIVISION.
* PROGRAM-ID. DYNQSAM1
* AUTHOR. CRAIG SCHNEIDERWENT.
* INSTALLATION. WISCONSIN DEPARTMENT OF TRANSPORTATION
* DATE-WRITTEN. 06-JAN-2003
* REMARKS.
* THIS PROGRAM PROVIDES A SINGLE MODULE FOR BATCH
* IEF/COMPOSER/COOL:GEN/ADVANTAGE GEN/COBOL APPLICATIONS TO CALL
* IN ORDER TO PERFORM PHYSICAL SEQUENTIAL I/O.
*
* THIS MODULE CAN BE DYNAMICALLY CALLED.
*
* CALL DYNQSAM1-PGM USING
*     DDNAME
*     ACTION
*     IO-BUFFER
* END-CALL
*
* ACTION MUST BE A 1-BYTE FIELD WHOSE VALUE IS ONE OF THE FOLLOWING
*
* #ACTN_OPEN_READ EQU   C'1'
* #ACTN_OPEN_WRT  EQU   C'2'
* #ACTN_READ      EQU   C'3'
* #ACTN_WRT       EQU   C'4'
* #ACTN_CLOSE     EQU   C'5'
* #ACTN_TEST_MEM  EQU   C'6'
*
* DDNAME MUST BE AN 8-BYTE FIELD WHOSE VALUE IS THE DDNAME ON WHICH
* YOU WISH TO PERFORM THE INDICATED ACTION.  THIS IS THE DDNAME FROM
* THE JCL FOR THE FILE YOU WISH TO ACCESS.
*
* IO-BUFFER IS THE AREA YOU WISH WRITTEN TO DDNAME (IF ACTION WRITE IS
* SPECIFIED) OR THE AREA YOU WISH DATA PLACED INTO (IF ACTION READ
* IS SPECIFIED).  IT MUST BE SPECIFIED FOR ALL CALLS.
*
* FOR ACTION TEST IO-BUFFER MUST CONTAIN THE MEMBER NAME TO TEST FOR
* IN THE FIRST 8 BYTES, RIGHT PADDED WITH SPACES IF NECESSARY.  TO
* TEST FOR A MEMBER, FIRST OPEN THE FILE, THEN TEST FOR THE MEMBER.
*
* IF THE DDNAME SPECIFIES A VB (VARIABLE BLOCKED) FILE THE CALLER MUST
* ALLOW FOR THE RDW IN THE FIRST FOUR BYTES OF THE IO-BUFFER.  THE RDW
* CONSISTS OF A HALFWORD LENGTH FOLLOWED BY A HALFWORD OF X'0000'.
* THIS IS PARTICULARLY IMPORTANT WHEN THE FILE IS OPENED FOR WRITE, AS
* THE LENGTH MUST BE SET CORRECTLY BY THE CALLING PROGRAM.
*
* THIS MODULE CAN BE CALLED ANY NUMBER OF TIMES WITH DIFFERENT DDNAMES
```

```
* SPECIFIED, SUBJECT TO OPERATING SYSTEM AND STORAGE LIMITATIONS.  THIS
* MODULE IMPOSES NO LIMITS OF ITS OWN.
*
* RETURN CODES
*
* #ERR_DDNAME     EQU   1  DDNAME INVALID
* #ERR_ACTION     EQU   2  ACTION NOT UNDERSTOOD
* #ERR_IS_OPEN    EQU   3  FILE ALREADY OPEN (ACTION = OPEN)
* #ERR_NOT_OPEN   EQU   4  FILE NOT YET OPEN (ACTION ^= OPEN)
* #ERR_NO_DD      EQU   5  DD CARD NOT PRESENT IN JCL
* #ERR_CLOSE      EQU   6  R15 ^= 0 AFTER CLOSE (MSG IS DISPLAYED)
* #ERR_OPEN       EQU   7  R15 ^= 0 AFTER OPEN  (MSG IS DISPLAYED)
* #RC_EOF         EQU   9  END OF FILE
* #RC_NO_MEM      EQU  10  MEMBER NOT FOUND
*
* THIS MODULE INCLUDES A DEBUG TRACE FACILITY.  ADD A CARD TO YOUR
* EXECUTION JCL:
*
* //$#@DEBUG DD  DUMMY
*
* AND THIS MODULE WILL WRITE OUT DEBUG TRACE INFORMATION VIA CEEMOUT,
* WHICH USUALLY ENDS UP IN YOUR SYSOUT DD UNLESS YOU EXPLICTLY
* CHANGE IT.
*
* DEBUG TRACE RECORDS ARE OF THE FORM
*
* DYNQSAM1 DDDDDDDD A RRRRRRRR
*
* WHERE DDDDDDDD IS THE DDNAME THE CALLER PROVIDED
*       A        IS THE ACTION THE CALLER REQUESTED
*       RRRRRRRR IS THE RETURN CODE
*
* AND
*
* DYNQSAM1 DDDDDDDD A MMMMMMMM EEEEEEEE
*
* WHERE DDDDDDDD IS THE DDNAME THE CALLER PROVIDED
*       A        IS THE ACTION THE CALLER REQUESTED
*       MMMMMMMM IS THE ACTION BEING TAKEN BY THIS MODULE
*       EEEEEEEE IS THE DDNAME ON WHICH THE ACTION IS BEING TAKEN
*
*
* 04-MAY-2009 CRAIG SCHNEIDERWENT
* ADDED "TEST FOR A MEMBER IN A LIBRARY" FUNCTION.  ALSO ADDED A
* COUPLE OF EYECATCHERS IN ALLOCATED STORAGE TO MAKE THINGS EASIER
* TO FIND IN A CORE DUMP.
```

```
*
*
         DCBD  DSORG=PS       DSECT MAPPING FOR DCB
         IHADCBE              DSECT MAPPING FOR DCBE (31-BIT I/O)
         CEECAA               DSECT MAPPING FOR LE COMMON ANCHOR AREA
         CEEDSA               DSECT MAPPING FOR LE DYNAMIC SAVE AREA
         IHAPSA               DSECT MAPPING FOR PREFIX SAVE AREA
*
         MACRO
*
*  INTERNAL MACRO TO FACTOR OUT THE DISPLAY-MESSAGE-ABEND CODE
*
         DSPL_ABND &MODE,&ABND_CD,&ACTN,&RC,&DDNAME
         AIF   ('&MODE' EQ 'LE').LE
         AIF   ('&MODE' EQ 'NOT_LE').NOT_LE
         MNOTE 12,'MODE MUST BE LE OR NOT_LE'
         MEXIT
         AIF   ('&RC'(1,3) EQ 'R15').VCSREG
.NOT_LE  ANOP
         MVC   CEEMOUT_ACTN,&ACTN
         AIF   ('&DDNAME' EQ '').NO_DDNM
         MVC   CEEMOUT_DDNAME,&DDNAME
.NO_DDNM ANOP
         MVC   CEEMOUT_MSG02,OUT_MSG02A
         AIF   ('&RC'(1,3) EQ 'R15').R15
         L     R15,&RC
         MVC   CEEMOUT_MSG02,OUT_MSG02B
.R15     ANOP
         CVD   R15,R15_PACKED               MAKE
         OI    R15_PACKED+7,X'0F'               REGISTER 15
         UNPK  CEEMOUT_R15,R15_PACKED                        READABLE
         CALL  CEEMOUT,(CEEMOUT_MSG,CEEMOUT_DEST_CD,LE_FC),            X
               MF=(E,CALL_LIST)
         AIF   ('&ABND_CD' NE '0').ABND
         MEXIT
.LE      ANOP
         CALL  CEEMSG,(LE_FC,CEEMSG_DEST_CD,CEEMSG_LE_FC),             X
               MF=(E,CALL_LIST)
         AIF   ('&ABND_CD' NE '0').ABND
         MEXIT
.ABND    ANOP
         MVC   CEE3ABD_CD,&ABND_CD          SAVE ABEND CODE
         CALL  CEE3ABD,(CEE3ABD_CD,CEE3ABD_CLEANUP),MF=(E,CALL_LIST)
         MEXIT
         MEND
*
```

```
$LSTELMT DSECT
*
* THIS DSECT MAPS THE LINKED LIST OF DDNAME, DCB AND DCBE
*
EYECATCH DS    CL16
DDNAME   DS    CL8
NEXT     DS    F
PREV     DS    F
MYDCB    DS    XL(DCBLNGQS)   LENGTH IS DEFINED IN DCBD DSECT
MYDCBE   DS    XL(DCBEEND-DCBE)
#LSTELMT EQU   *-$LSTELMT
*
         COPY  ASMMSP         IBM STRUCTURED PROGRAMMING MACROS
*
         IEANTASM             IBM NAME-TOKEN SERVICES EQUATES
*
         YREGS                IBM REGISTER EQUATES
*
* LOCAL-STORAGE (PER INVOCATION)
*
LOCAL_STORAGE   DSECT
                DS    XL(CEEDSASZ)   LEAVE SPACE FOR DSA FIXED PART
MY_EYE_CATCHER  DS    CL16
RC              DS    F    DUH
#ERR_DDNAME     EQU   1    DDNAME INVALID
#ERR_ACTION     EQU   2    ACTION NOT UNDERSTOOD
#ERR_IS_OPEN    EQU   3    FILE ALREADY OPEN (ACTION = OPEN)
#ERR_NOT_OPEN   EQU   4    FILE NOT YET OPEN (ACTION ^= OPEN)
#ERR_NO_DD      EQU   5    DD CARD NOT PRESENT IN JCL
#ERR_CLOSE      EQU   6    R15 ^= 0 AFTER CLOSE (MSG IS DISPLAYED)
#ERR_OPEN       EQU   7    R15 ^= 0 AFTER OPEN  (MSG IS DISPLAYED)
#RC_EOF         EQU   9    END OF FILE
#RC_NO_MEM      EQU  10  MEMBER NOT FOUND
LINKADDR        DS    A
@PARM_DDNAME    DS    A
PARM_DDNAME     DS    CL8
@PARM_ACTION    DS    A
PARM_ACTION     DS    CL1
#ACTN_OPEN_READ EQU   C'1'
#ACTN_OPEN_WRT  EQU   C'2'
#ACTN_READ      EQU   C'3'
#ACTN_WRT       EQU   C'4'
#ACTN_CLOSE     EQU   C'5'
#ACTN_TEST_MEM  EQU   C'6'
@PARM_BUFFER    DS    A           ADDRESS OF CALLER'S I/O AREA
CK_FOR_DD_SA    DS    A           TINY SAVE AREA
```

```
ALCT_SA          DS    A           TINY SAVE AREA
DEALCT_SA        DS    A           TINY SAVE AREA
SEARCH_SA        DS    A           TINY SAVE AREA
CALL_LIST        CALL  ,(,,,,,,,,,),MF=L
OPEN_LIST        OPEN  (,),MODE=31,MF=L
CLOSE_LIST       CLOSE (,),MF=L,MODE=31
NT_USER_NAME     DS    0XL16
                 DS    CL4
                 DS    A
NT_MY_ASCB       DS    A
                 DS    A
NT_USER_TOKEN    DS    0XL16
@TASK_STORAGE    DS    A
                 DS    3F
NT_RC            DS    F
DEVTY_WORK_AREA  DS    0F
                 DS    XL24
LE_FC            DS    3XL4        LE FEEDBACK CODE
CEEMSG_LE_FC     DS    3XL4        LE FEEDBACK CODE FOR CEEMSG
EYE_CATCH_01     DS    CL8
@MEM             DS    A           ADDRESS OF ALLOCATED MEMORY
BLDL_PRFX        DS    CL8         REQUIRED FOR NOCONNECT
BLDL_LIST        DS    0F
BLDL_FF          DS    H
BLDL_LL          DS    H
BLDL_NAME        DS    CL8
BLDL_TT          DS    CL2
BLDL_R           DS    CL1
BLDL_K           DS    CL1
BLDL_Z           DS    CL1
BLDL_C           DS    CL1
#BLDL_LIST       EQU   *-BLDL_LL
@BLDL_LIST       DS    A
CEEMSG_DEST_CD   DS    F
CEEMOUT_DEST_CD  DS    F
CEE3ABD_CD       DS    F
CEE3ABD_CLEANUP  DS    F
ZERO             DS    F           DUH
R0_SAVED         DS    F
R15_PACKED       DS    D
CEEMOUT_MSG      DS    0D
CEEMOUT_MSG_LN   DS    H
CEEMOUT_MSG_TXT  DS    0CL80
CEEMOUT_MSG01    DS    CL18
CEEMOUT_ACTN     DS    CL8
CEEMOUT_MSG02    DS    CL5
```

```
CEEMOUT_R15     DS    CL8
CEEMOUT_MSG03   DS    CL8
CEEMOUT_DDNAME  DS    CL8
CEEMOUT_MSG04   DS    CL30
CEEDBUG_MSG     DS    0D
CEEDBUG_MSG_LN  DS    H
CEEDBUG_MSG_TXT DS    0CL80
CEEDBUG_ID      DS    CL8
CEEDBUG_FILL01  DS    CL1
CEEDBUG_DDNAME  DS    CL8
CEEDBUG_FILL02  DS    CL1
CEEDBUG_ACTN    DS    CL1
CEEDBUG_FILL03  DS    CL1
CEEDBUG_R15     DS    CL8
CEEDBUG_FILL04  DS    CL1
CEEDBUG_DDNAME1 DS    CL8
CEEDBUG_FILL05  DS    CL43
#LOCAL_STORAGE  EQU   *-LOCAL_STORAGE
*
* TASK STORAGE  (PERSISTS UNTIL END OF JOB STEP)
*
TASK_STORAGE    DSECT
LIST_HEAD       DS    A           START OF LINKED LIST
LIST_TAIL       DS    A           END OF LINKED LIST
LISTMEMSZ       DS    F           SIZE OF A LIST ELEMENT
HEAPID          DS    F           SET BY CEECRHP
CEECRHP_OPT     DS    F
DEBUG_SW        DS    H           DEBUG MODE?
#TASK_STORAGE   EQU   *-TASK_STORAGE
*
DYNQSAM1 CEEENTRY MAIN=NO,PPA=PGM_PROLOG,AUTO=#LOCAL_STORAGE
DCBELMT  USING $LSTELMT,R9
         USING LOCAL_STORAGE,R13
*
         MVC   MY_EYE_CATCHER,=C'DYNQSAM1WORKDATA'
         MVC   EYE_CATCH_01,=C'@MEM--->'
         LR    R8,R1                 MAKE A TEMPORARY COPY
         ST    R8,LINKADDR           KEEP PARAMETER LIST ADDRESS
         L     R2,LINKADDR           WORKING OFF R2
         MVC   @PARM_DDNAME,0(R2)    ADDRESS OF PARM 1
         MVC   @PARM_ACTION,4(R2)    ADDRESS OF PARM 2
         MVC   @PARM_BUFFER,8(R2)    ADDRESS OF PARM 3
         L     R2,@PARM_DDNAME       ADDRESS OF DDNAME PARM
         MVC   PARM_DDNAME,0(R2)     SAVE DDNAME
         L     R2,@PARM_ACTION       ADDRESS OF ACTION PARM
         MVC   PARM_ACTION,0(R2)     SAVE ACTION
```

```
*             INITIALIZE SOME LOCAL_STORAGE FIELDS
         MVC   NT_USER_NAME,USER_NAME
         XR    R2,R2
         MVC   NT_MY_ASCB,PSAAOLD-PSA(R2)
         XC    RC,RC                   CLEAR RETURN CODE
         XC    R15_PACKED,R15_PACKED
         XC    NT_USER_TOKEN,NT_USER_TOKEN
         XC    OPEN_LIST(4),OPEN_LIST
         XC    CLOSE_LIST(4),CLOSE_LIST
         XC    BLDL_PRFX,BLDL_PRFX
         XC    BLDL_FF,BLDL_FF
         XC    BLDL_LL,BLDL_LL
         XC    BLDL_NAME,BLDL_NAME
         XC    BLDL_TT,BLDL_TT
         XC    BLDL_R,BLDL_R
         XC    BLDL_K,BLDL_K
         XC    BLDL_Z,BLDL_Z
         XC    BLDL_C,BLDL_C
         MVC   ZERO,FWORD_ZERO
         MVC   CEEMSG_DEST_CD,FWORD_TWO
         MVC   CEEMOUT_DEST_CD,FWORD_TWO
         MVC   CEE3ABD_CD,FWORD_ZERO
         MVC   CEE3ABD_CLEANUP,FWORD_ONE
         MVC   CEEMOUT_MSG_LN,ERR_MSG_LN
         MVC   CEEMOUT_MSG01,OUT_MSG01
         MVC   CEEMOUT_MSG03,OUT_MSG03
         INITCHAR FIELD=CEEMOUT_DDNAME,CHAR=C' '
         MVC   CEEMOUT_MSG04,OUT_MSG04
         INITCHAR FIELD=CEEDBUG_MSG_TXT,CHAR=C' '
*             BRAIN-DEAD EDIT OF DDNAME
         IF    (CLI,PARM_DDNAME,EQ,C' '),OR,                           X
               (CLI,PARM_DDNAME,EQ,X'00'),OR,                          X
               (CLI,PARM_DDNAME,EQ,X'FF') THEN
               MVI   RC+3,#ERR_DDNAME
               B     GOBACK
         ENDIF
*        RETRIEVE POINTER TO TASK_STORAGE USING NAME-TOKEN SERVICES
         CALL  IEANTRT,(NT_LEVEL,NT_USER_NAME,                         X
               NT_USER_TOKEN,NT_RC),MF=(E,CALL_LIST)
         SELECT CLI,NT_RC+3,EQ
           WHEN (IEANT_OK)
             L     R10,@TASK_STORAGE    MAKE USING STATEMENT TRUE
           WHEN (IEANT_NOT_FOUND)
*            POINTER NOT FOUND - FIRST TIME IN - ALLOCATE TASK_STORAGE
             CALL  CEEGTST,(ZERO,TASKMEMSZ,                            X
               @MEM,LE_FC),MF=(E,CALL_LIST)
```

```
            IF    (CLC,LE_FC,NE,=3XL4'0') THEN
                  DSPL_ABND LE,=F'6'
            ENDIF
*           STORE POINTER TO TASK_STORAGE FOR USE IN LATER EXECUTIONS
            MVC   @TASK_STORAGE,@MEM
            L     R10,@TASK_STORAGE    MAKE USING STATEMENT TRUE
            USING TASK_STORAGE,R10
            CALL  IEANTCR,(NT_LEVEL,NT_USER_NAME,                      X
              NT_USER_TOKEN,ZERO,NT_RC),MF=(E,CALL_LIST)
            IF    (CLI,NT_RC+3,EQ,IEANT_OK) THEN
                  CNOP  2,4
            ELSE
                  L     R15,NT_RC
                  DSPL_ABND NOT_LE,=F'7',IEANTCR_LIT,NT_RC
            ENDIF
*           INITIALIZE SOME TASK_STORAGE FIELDS
            MVC   CEECRHP_OPT,HEAP_BELOW
            XC    LIST_HEAD,LIST_HEAD
            XC    LIST_TAIL,LIST_TAIL
            XC    HEAPID,HEAPID
            XC    LISTMEMSZ,LISTMEMSZ
            MVI   LISTMEMSZ+3,#LSTELMT
            MVC   DEBUG_SW,=H'0'
            DEVTYPE DEBUG_DD,DEVTY_WORK_AREA
            IF    (CHI,R15,EQ,0) THEN
                  MVC   DEBUG_SW,=H'1'
            ENDIF
          OTHRWISE
            DSPL_ABND NOT_LE,=F'5',IEANTRT_LIT,NT_RC
        ENDSEL
        IF    (CLC,DEBUG_SW,EQ,=H'1') THEN
              MVC   CEEDBUG_ID,MYNAME
              MVC   CEEDBUG_MSG_LN,DBUG_MSG_LN
              MVC   CEEDBUG_DDNAME,PARM_DDNAME
              MVC   CEEDBUG_ACTN,PARM_ACTION
        ENDIF
        BAS   R14,CK_FOR_DD       TEST TO SEE IF DD IS PRESENT IN JCL
        IF    (CLC,RC,NE,=F'0') THEN
              B     GOBACK        DD NOT PRESENT - EXIT
        ENDIF
        XR    R9,R9               CLEAR REGISTER 9
        BAS   R14,SEARCH_DCBELMT
*             R9 WILL NOW CONTAIN THE ADDRESS OF THE DCBELMT LIST
*             CORRESPONDING TO THE PARM_DDNAME, IF THE DD HAS
*             ALREADY BEEN OPENED, OR 0 IF IT HAS NOT BEEN OPENED
        SELECT CLI,PARM_ACTION,EQ
```

```
            WHEN (#ACTN_OPEN_READ)
              LTR   R9,R9                 DO WE HAVE A POINTER?
              BZ    OPEN_FOR_READ         NO - BRANCH
              MVI   RC+3,#ERR_IS_OPEN
            WHEN (#ACTN_OPEN_WRT)
              LTR   R9,R9                 DO WE HAVE A POINTER?
              BZ    OPEN_FOR_WRITE        NO - BRANCH
              MVI   RC+3,#ERR_IS_OPEN
            WHEN (#ACTN_READ)
              LTR   R9,R9                 DO WE HAVE A POINTER?
              BNZ   READ_INPUT            YES - BRANCH
              MVI   RC+3,#ERR_NOT_OPEN
            WHEN (#ACTN_WRT)
              LTR   R9,R9                 DO WE HAVE A POINTER?
              BNZ   WRITE_OUTPUT          YES - BRANCH
              MVI   RC+3,#ERR_NOT_OPEN
            WHEN (#ACTN_CLOSE)
              LTR   R9,R9                 DO WE HAVE A POINTER?
              BNZ   CLOSE_DATASET         YES - BRANCH
              MVI   RC+3,#ERR_NOT_OPEN
            WHEN (#ACTN_TEST_MEM)
              LTR   R9,R9                 DO WE HAVE A POINTER?
              BNZ   TEST_MEMBER           YES - BRANCH
              MVI   RC+3,#ERR_NOT_OPEN
            OTHRWISE
              MVI   RC+3,#ERR_ACTION      UNRECOGNIZED ACTION
          ENDSEL
          B     GOBACK
OPEN_FOR_READ  EQU *
          BAS   R14,ALCT_DCBELMT      MAKE R9 POINT TO NEW DCBELMT ITEM
          IF    (CLC,DEBUG_SW,EQ,=H'1') THEN
              MVC   CEEDBUG_R15,=CL8'OPENREAD'
              MVC   CEEDBUG_DDNAME1,DCBELMT.DDNAME
              CALL  CEEMOUT,(CEEDBUG_MSG,CEEMOUT_DEST_CD,LE_FC),        X
                MF=(E,CALL_LIST)
              INITCHAR FIELD=CEEDBUG_DDNAME1,CHAR=C' '
          ENDIF
          LA    R2,DCBELMT.MYDCB      POINT TO THE DCB
          MVI   OPEN_LIST,B'10000000' END OF PARMLIST INDICATOR
          OPEN  ((R2),INPUT),MODE=31,MF=(E,OPEN_LIST)
          IF    (CHI,R15,NE,0) THEN
                DSPL_ABND NOT_LE,0,OPEN_LIT,R15,PARM_DDNAME
                MVI   RC+3,#ERR_OPEN
          ENDIF
          B     GOBACK
OPEN_FOR_WRITE EQU *
```

```
         BAS   R14,ALCT_DCBELMT      MAKE R9 POINT TO NEW DCBELMT ITEM
         IF    (CLC,DEBUG_SW,EQ,=H'1') THEN
             MVC   CEEDBUG_R15,=CL8'OPEN WRT'
             MVC   CEEDBUG_DDNAME1,DCBELMT.DDNAME
             CALL  CEEMOUT,(CEEDBUG_MSG,CEEMOUT_DEST_CD,LE_FC),        X
               MF=(E,CALL_LIST)
             INITCHAR FIELD=CEEDBUG_DDNAME1,CHAR=C' '
         ENDIF
         LA    R2,DCBELMT.MYDCB      POINT TO THE DCB
         MVI   OPEN_LIST,B'10000000' END OF PARMLIST INDICATOR
         OPEN  ((R2),OUTPUT),MODE=31,MF=(E,OPEN_LIST)
         IF    (CHI,R15,NE,0) THEN
               DSPL_ABND NOT_LE,0,OPEN_LIT,R15,PARM_DDNAME
               MVI   RC+3,#ERR_OPEN
         ENDIF
         B     GOBACK
CLOSE_DATASET  EQU *
         IF    (CLC,DEBUG_SW,EQ,=H'1') THEN
             MVC   CEEDBUG_R15,=CL8'CLOSE   '
             MVC   CEEDBUG_DDNAME1,DCBELMT.DDNAME
             CALL  CEEMOUT,(CEEDBUG_MSG,CEEMOUT_DEST_CD,LE_FC),        X
               MF=(E,CALL_LIST)
             INITCHAR FIELD=CEEDBUG_DDNAME1,CHAR=C' '
         ENDIF
         LA    R2,DCBELMT.MYDCB       POINT TO THE DCB
         MVI   CLOSE_LIST,B'10000000' END OF PARMLIST INDICATOR
         CLOSE ((R2)),MODE=31,MF=(E,CLOSE_LIST)
         IF    (CHI,R15,NE,0) THEN
               DSPL_ABND NOT_LE,0,CLOSE_LIT,R15,PARM_DDNAME
               MVI   RC+3,#ERR_CLOSE
         ENDIF
         BAS   R14,DEALCT_DCBELMT    FILE CLOSED, DEALLOCATE LIST ENTRY
         B     GOBACK
TEST_MEMBER    EQU *
         IF    (CLC,DEBUG_SW,EQ,=H'1') THEN
             MVC   CEEDBUG_R15,=CL8'TEST    '
             MVC   CEEDBUG_DDNAME1,DCBELMT.DDNAME
             CALL  CEEMOUT,(CEEDBUG_MSG,CEEMOUT_DEST_CD,LE_FC),        X
               MF=(E,CALL_LIST)
             INITCHAR FIELD=CEEDBUG_DDNAME1,CHAR=C' '
         ENDIF
         MVC   BLDL_FF,HWORD_ONE      ONE ENTRY IN BLDL LIST
         L     R4,@PARM_BUFFER        ADDRESS OF MEMBER NAME
         MVC   BLDL_NAME,0(R4)        MEMBER NAME TO TEST FOR
         L     R4,BLDL_SZ             LENGTH OF BLDL_LIST
         STH   R4,BLDL_LL             SIZE OF BLDL LIST
```

```
         LA    R5,BLDL_LIST
         LA    R2,DCBELMT.MYDCB       POINT TO THE DCB
         BLDL  (R2),(R5),NOCONNECT    TEST FOR EXISTENCE
         IF    (CHI,R15,NE,0) THEN
               ST    R0,R0_SAVED
               MVI   RC+3,#RC_NO_MEM
         ENDIF
         B     GOBACK
READ_INPUT     EQU *
         L     R3,@PARM_BUFFER     POINT TO THE I/O BUFFER
         LA    R2,DCBELMT.MYDCB    POINT TO THE DCB
         GET   (R2),(R3)
         XC    RC,RC               CAN'T FIND DOC FOR RETURN CODES
         B     GOBACK
WRITE_OUTPUT   EQU *
         L     R3,@PARM_BUFFER     POINT TO THE I/O BUFFER
         LA    R2,DCBELMT.MYDCB    POINT TO THE DCB
         PUT   (R2),(R3)
         XC    RC,RC               CAN'T FIND DOC FOR RETURN CODES
         B     GOBACK
END_OF_FILE    EQU *
         MVI   RC+3,#RC_EOF
         B     GOBACK
CK_FOR_DD      EQU *                  LOOK FOR DD STATEMENT IN JCL
         ST    R14,CK_FOR_DD_SA
         DEVTYPE PARM_DDNAME,DEVTY_WORK_AREA
         SELECT CHI,R15,EQ
           WHEN (0)
                CNOP 0,4              DD IS PRESENT
           WHEN (4)
                MVI  RC+3,#ERR_NO_DD  DD IS NOT PRESENT
           OTHRWISE
                DSPL_ABND NOT_LE,=F'4',DEVTYPE_LIT,R15,PARM_DDNAME
         ENDSEL
         L     R14,CK_FOR_DD_SA
         BR    R14
SEARCH_DCBELMT EQU *                  SEARCH LIST FOR PARM DDNAME
         ST    R14,SEARCH_SA
         L     R9,LIST_HEAD
         DO    WHILE=(CHI,R9,NE,0)
               DOEXIT (CLC,DCBELMT.DDNAME,EQ,PARM_DDNAME)
               L  R9,DCBELMT.NEXT
         ENDDO
         L     R14,SEARCH_SA
         BR    R14
ALCT_DCBELMT   EQU *
```

```
*                         ON RETURN R9 POINTS TO NEWLY ALLOCATED
*                         STORAGE MAPPED BY $LSTELMT DSECT TO BE
*                         USED WITH DCBELMT USING STATEMENT
         ST    R14,ALCT_SA
TAILELMT USING $LSTELMT,R6
NEWELMT  USING $LSTELMT,R8
NEWDCB   USING IHADCB,R7
         IF    (CLC,HEAPID,EQ,=F'0') THEN
               CALL  CEECRHP,(HEAPID,ZERO,ZERO,                        X
               CEECRHP_OPT,LE_FC),MF=(E,CALL_LIST)
*              CREATE HEAP FOR CEEGTST
               IF    (CLC,LE_FC,NE,=3XL4'0') THEN
                     DSPL_ABND LE,=F'1'
               ENDIF
         ENDIF
         CALL  CEEGTST,(HEAPID,LISTMEMSZ,@MEM,LE_FC),MF=(E,CALL_LIST)
*              ALLOCATE STORAGE
         IF    (CLC,LE_FC,NE,=3XL4'0') THEN
               DSPL_ABND LE,=F'2'
         ENDIF
         L     R8,@MEM           MAKE USING STATEMENT TRUE
         XR    R0,R0
         L     R6,LIST_TAIL      MAKE USING STATEMENT TRUE
         IF    (CHI,R6,EQ,0)  THEN
               ST    R8,LIST_HEAD
         ELSE
               ST    R8,TAILELMT.NEXT
         ENDIF
         ST    R8,LIST_TAIL
         ST    R6,NEWELMT.PREV   POINT TO PREVIOUS ELEMENT IN CHAIN
         ST    R0,NEWELMT.NEXT   THERE IS NO NEXT ELEMENT - CLEAR PTR
         LA    R7,NEWELMT.MYDCB  MAKE USING STATEMENT TRUE
         MVC   NEWELMT.EYECATCH,=C'DYNQSAM1$LSTELMT'
         MVC   NEWELMT.DDNAME,PARM_DDNAME
         MVC   NEWELMT.MYDCB,MODL_DCB      COPY MODEL DCB INTO LIST
         MVC   NEWDCB.DCBDDNAM,PARM_DDNAME PLUNK DDNAME IN DCB
         LA    R2,NEWELMT.MYDCBE           ADDRESS OF DCBE
         ST    R2,NEWDCB.DCBDCBE           MAKE DCB POINT TO DCBE
         MVC   NEWELMT.MYDCBE,MODL_DCBE    COPY MODEL DCBE INTO LIST
         LR    R9,R8                       MAKE USING STATEMENT TRUE
         DROP  NEWDCB
         DROP  NEWELMT
         DROP  TAILELMT
         IF    (CLC,DEBUG_SW,EQ,=H'1') THEN
             MVC   CEEDBUG_R15,=CL8'ALLOCATE'
             MVC   CEEDBUG_DDNAME1,PARM_DDNAME
```

```
             CALL  CEEMOUT,(CEEDBUG_MSG,CEEMOUT_DEST_CD,LE_FC),        X
               MF=(E,CALL_LIST)
             INITCHAR FIELD=CEEDBUG_DDNAME1,CHAR=C' '
          ENDIF
          L     R14,ALCT_SA
          BR    R14
DEALCT_DCBELMT   EQU *
          ST    R14,DEALCT_SA
NEXTELMT USING $LSTELMT,R8
OLDELMT  USING $LSTELMT,R7  THIS IS THE ONE WE WILL DECHAIN AND FREE
PREVELMT USING $LSTELMT,R6
          LR    R7,R9            MAKE USING STATEMENT TRUE
          IF    (CLC,DEBUG_SW,EQ,=H'1') THEN
              MVC   CEEDBUG_R15,=CL8'FREEING '
              MVC   CEEDBUG_DDNAME1,OLDELMT.DDNAME
              CALL  CEEMOUT,(CEEDBUG_MSG,CEEMOUT_DEST_CD,LE_FC),        X
               MF=(E,CALL_LIST)
             INITCHAR FIELD=CEEDBUG_DDNAME1,CHAR=C' '
          ENDIF
          L     R8,OLDELMT.NEXT  MAKE USING STATEMENT TRUE
          L     R6,OLDELMT.PREV  MAKE USING STATEMENT TRUE
          IF    (CLC,OLDELMT.NEXT,EQ,=F'0') THEN
                ST   R6,LIST_TAIL
          ELSE
                MVC  NEXTELMT.PREV,OLDELMT.PREV BREAK CHAIN
          ENDIF
          IF    (CLC,OLDELMT.PREV,EQ,=F'0') THEN
                ST   R8,LIST_HEAD
          ELSE
                MVC  PREVELMT.NEXT,OLDELMT.NEXT BREAK CHAIN
          ENDIF
          ST    R7,@MEM
          CALL  CEEFRST,(@MEM,LE_FC),MF=(E,CALL_LIST)
*               FREE STORAGE
          IF    (CLC,LE_FC,NE,=3XL4'0') THEN
                DSPL_ABND LE,=F'3'
          ENDIF
          DROP  OLDELMT
          DROP  NEXTELMT
          DROP  PREVELMT
          L     R14,DEALCT_SA
          BR    R14
GOBACK    EQU   *
          IF    (CLC,DEBUG_SW,EQ,=H'1') THEN
              L     R15,RC
              CVD   R15,R15_PACKED               MAKE
```

```
             OI    R15_PACKED+7,X'0F'               REGISTER 15
             UNPK  CEEDBUG_R15,R15_PACKED                       READABLE
             CALL  CEEMOUT,(CEEDBUG_MSG,CEEMOUT_DEST_CD,LE_FC),        X
               MF=(E,CALL_LIST)
          ENDIF
          CEETERM RC=RC
*
* CONSTANTS
*
PGM_PROLOG  CEEPPA VER=01,REL=00,MOD=00
FWORD_ZERO  DC     F'0'
FWORD_ONE   DC     F'1'
FWORD_TWO   DC     F'2'
HEAP_BELOW  DC     F'76'  HEAP(,,BELOW,)
TASKMEMSZ   DC     A(#TASK_STORAGE)
BLDL_SZ     DC     A(#BLDL_LIST)
ERR_MSG_LN  DC     AL2(L'CEEMOUT_MSG_TXT)
DBUG_MSG_LN DC     AL2(L'CEEDBUG_MSG_TXT)
LE_FC_OK    DC     3XL4'0'
MYNAME      DC     CL8'DYNQSAM1'
OPEN_LIT    DC     CL8'OPEN    '
CLOSE_LIT   DC     CL8'CLOSE   '
TEST_LIT    DC     CL8'TEST    '
DEVTYPE_LIT DC     CL8'DEVTYPE '
IEANTCR_LIT DC     CL8'IEANTCR '
IEANTRT_LIT DC     CL8'IEANTRT '
OUT_MSG01   DC     CL18'DYNQSAM1 ERROR IN '
OUT_MSG02A  DC     CL5' R15='
OUT_MSG02B  DC     CL5' RC ='
OUT_MSG03   DC     CL8' DDNAME='
OUT_MSG04   DC     CL30' '
DEBUG_DD    DC     CL8'$#@DEBUG'
            DC     0D
USER_NAME   DC     0XL16
            DC     CL4'UDOT'
            DC     A(DYNQSAM1)
            DC     D'0'
NT_LEVEL    DC     A(IEANT_PRIMARY_LEVEL)
HWORD_ONE   DC     H'1'
            LTORG
*
* FILE SECTION
*
MODL_DCB    DCB DSORG=PS,MACRF=(GM,PM),DCBE=MODL_DCBE,DDNAME=NONE
MODL_DCBE   DCBE RMODE31=BUFF,EODAD=END_OF_FILE
            END
```