```
iRMX 86 PL/M-86 V1.0 COMPILATION OF MODULE MASTER
OBJECT MODULE PLACED IN TAPSDEV/MASTER.OBJ
COMPILER INVOKED BY:  :SYSTEM:PLM86 TAPSDEV/MASTER.SRC COMPACT
```

```
  1          MASTER: DO;
  2    1     RQSCREATESMAILBOX: PROCEDURE(TASKTK,RTN_CODE) WORD EXTERNAL;
  3    2                        DECLARE TASKTK WORD,RTN_CODE POINTER;
  4    2                        END RQSCREATESMAILBOX;
  5    1     RQSASATTACHSFILE:  PROCEDURE(USER,CONNECTION,PATH_NAME,
                                      RESMBX,RTN_CODE) EXTERNAL;
  6    2                        DECLARE PATH_NAME POINTER,RTN_CODE POINTER;
  7    2                        DECLARE USER WORD,CONNECTION WORD,RESMBX WORD;
  8    2                        END RQSASATTACHSFILE;
  9    1     RQSASOPEN:         PROCEDURE(CONNECT,MODE,SHARE,RESPONSE,RTN_CODE)
                                EXTERNAL;
 10    2                        DECLARE CONNECT WORD,MODE WORD,SHARE BYTE,
                                      RESPONSE WORD,RTN_CODE POINTER;
 11    2                        END RQSASOPEN;
 12    1     RQSASREAD:         PROCEDURE(CONNECT,BUFF_PTR,COUNT,RESPONSE,RTN_CODE)
                                EXTERNAL;
 13    2                        DECLARE CONNECT WORD,BUFF_PTR POINTER,COUNT WORD,
                                      RESPONSE WORD,RTN_CODE POINTER;
 14    2                        END RQSASREAD;
 15    1     RQSASSEEK:         PROCEDURE(CONNECT,MODE,HI_ORDER,LOW_ORDER,RESPONSE,
                                      RTN_CODE) EXTERNAL;
 16    2                        DECLARE CONNECT WORD,MODE WORD,HI_ORDER WORD,
                                      LOW_ORDER WORD,RESPONSE WORD,RTN_CODE POINTER;
 17    2                        END RQSASSEEK;
 18    1     RQSASCLOSE:        PROCEDURE(CONNECT,RESPONSE,RTN_CODE) EXTERNAL;
 19    2                        DECLARE CONNECT WORD,RESPONSE WORD,RTN_CODE POINTER;
 20    2                        END RQSASCLOSE;
 21    1     RQSGETSTASKSTOKENS: PROCEDURE(TASKN,STATUS) WORD EXTERNAL;
 22    2                           DECLARE TASKN WORD,STATUS POINTER;
 23    2                           END RQSGETSTASKSTOKENS;
 24    1     RQSDELETESSEGMENT: PROCEDURE(OBJECT,RTN_CODE) EXTERNAL;
 25    2                        DECLARE OBJECT WORD,RTN_CODE POINTER;
 26    2                        END RQSDELETESSEGMENT;
 27    1     RQSCREATESSEGMENT: PROCEDURE(SIZE,RTN_CODE) WORD EXTERNAL;
 28    2                        DECLARE SIZE WORD,RTN_CODE POINTER;
 29    2                        END RQSCREATESSEGMENT;
 30    1     RQSLOOKSUPSOBJECT: PROCEDURE(TASKTK,OBJ_NAME,TIME,RTN_CODE) WORD EXTERNAL;
 31    2                        DECLARE TASKTK WORD,OBJ_NAME POINTER,TIME WORD,
                                      RTN_CODE POINTER;
 32    2                        END RQSLOOKSUPSOBJECT;

 33    1     RQSCREATESTASK:    PROCEDURE(PRIORITY,CODE_SEG,DATA_SEG,STACK_SEG,
                                      STACK_SIZE,FLAGS,RTN_CODE) WORD EXTERNAL;
 34    2                        DECLARE PRIORITY BYTE,CODE_SEG POINTER,
                                      DATA_SEG WORD,STACK_SEG POINTER,
                                      STACK_SIZE WORD,FLAGS WORD,RTN_CODE
                                      POINTER;
 35    2                        END RQSCREATESTASK;
 36    1     RQSGETSTYPE:       PROCEDURE(OBJECT,RTN_CODE) WORD EXTERNAL;
 37    2                        DECLARE OBJECT WORD,RTN_CODE POINTER;
 38    2                        END RQSGETSTYPE;
```

~~2400~~ 2500 - TMJ
:1E2
OE = Return from Receive Msg
DAD = START TASK

```
 39   1      RQ$A$LOAD:          PROCEDURE(CONNECT,RESPONSE,RTN_CODE)EXTERNAL;
 40   2                          DECLARE CONNECT WORD,RESPONSE WORD,
                                        RTN_CODE POINTER;
 41   2                          END RQ$A$LOAD;
 42   1      RQ$RECEIVE$MESSAGE: PROCEDURE(MAIL_BOX,TIME,RESPONSE,STATUS) WORD EXTERNAL;
 43   2                           DECLARE MAIL_BOX WORD,TIME WORD,RESPONSE POINTER,
                                          STATUS POINTER;
 44   2                           END RQ$RECEIVE$MESSAGE;
 45   1      RQ$SEND$MESSAGE:    PROCEDURE(MAIL_BOX,OBJECT,RESPONSE,RTN_CODE) EXTERNAL;
 46   2                          DECLARE MAIL_BOX WORD,OBJECT WORD,RESPONSE WORD,
                                        RTN_CODE POINTER;
 47   2                          END RQ$SEND$MESSAGE;
 48   1      RQ$CATALOG$OBJECT:  PROCEDURE(JOB,OBJECT,NAME,RTN_CODE) EXTERNAL;
 49   2                          DECLARE JOB WORD,OBJECT WORD,NAME POINTER,
                                        RTN_CODE POINTER;
 50   2                          END RQ$CATALOG$OBJECT;
 51   1      RQ$EXIT$IO$JOB:     PROCEDURE(TYPE,MESSAGE,RTN_CODE) EXTERNAL;
 52   2                          DECLARE TYPE WORD,MESSAGE POINTER,RTN_CODE POINTER;
 53   2                          END RQ$EXIT$IO$JOB;
 54   1      RQ$SET$EXCEPTION$HANDLER: PROCEDURE(EXCPSTRUC,RTN_CODE) EXTERNAL;
 55   2                                DECLARE (EXCPSTRUC,RTN_CODE) POINTER;
 56   2                                END RQ$SET$EXCEPTION$HANDLER;
 57   1      RQ$GET$SIZE:        PROCEDURE(SEGMENT,RTN_CODE) WORD EXTERNAL;
 58   2                          DECLARE SEGMENT WORD,RTN_CODE POINTER;
 59   2                          END RQ$GET$SIZE;
 60   1      RQ$GET$DEFAULT$USER: PROCEDURE(JOB,RTN_CODE) WORD EXTERNAL;
 61   2                           DECLARE JOB WORD,RTN_CODE POINTER;
 62   2                           END RQ$GET$DEFAULT$USER;
 63   1      RQ$GET$DEFAULT$PREFIX: PROCEDURE(JOB,RTN_CODE) WORD EXTERNAL;
 64   2                             DECLARE JOB WORD,RTN_CODE POINTER;
 65   2                             END RQ$GET$DEFAULT$PREFIX;
 66   1      RQ$DELETE$MAILBOX: PROCEDURE(MBX_TOKEN,RTN_CODE) EXTERNAL;
 67   2                         DECLARE MBX_TOKEN WORD,RTN_CODE POINTER;
 68   2                         END RQ$DELETE$MAILBOX;
 69   1      RQ$UNCATALOG$OBJECT: PROCEDURE(JOBTKN,NAME,RTN_CODE) EXTERNAL;
 70   2                           DECLARE JOBTKN WORD,RTN_CODE POINTER;
 71   2                           DECLARE NAME POINTER;
 72   2                           END RQ$UNCATALOG$OBJECT;
 73   1      RQ$DELETE$TASK:     PROCEDURE(TASKID,RTN_CODE) EXTERNAL;
 74   2                          DECLARE TASKID WORD,RTN_CODE POINTER;
 75   2                          END RQ$DELETE$TASK;
 76   1      BINAS:              PROCEDURE(SOURCE,RESULT,LENGTH) EXTERNAL;
 77   2                          DECLARE SOURCE POINTER,RESULT POINTER,LENGTH WORD;
 78   2                          END BINAS;
 79   1      TAPSER: PROCEDURE(TYPE,MSGTKN,MESSAGE) EXTERNAL;
 80   2              DECLARE TYPE WORD,MSGTKN WORD,MESSAGE POINTER;
 81   2              END TAPSER;

 82   1      DECLARE LOCAL_MBX WORD PUBLIC; /*FOR BIOS RESULTS RETURN*/


              $INCLUDE (:DO:TAPSDEV/COMMON.SRC)
        =     $SAVE NOLIST

              $INCLUDE (:DO:TAPSDEV/GMF.EXT)
        =     $SAVE NOLIST
              $INCLUDE (:DO:TAPSDEV/MMXP86.SRC)
```

```
=    /*****************************************************************
=    *                                                               *
=    *                    INCLUDE FILE MMXPRT.LIT                    *
=    *                    =======================                    *
=    *                                                               *
=    *   THIS INCLUDE FILE DEFINES THE MMX, MIP AND GMF SYSTEM PORT NAMES    *
=    *                                                               *
=    *   USED IN THE MSP OPERATING SYSTEM. INCLUDE THIS FILE IN YOUR CODE    *
=    *                                                               *
=    *   TO MAKE IT INDEPENDENT OF CHANGES IN PORT NAMES OR DEVICES.         *
=    *                                                               *
=    *****************************************************************/
=            /* SPU-0 IS DEVICE 0 IT CURRENTLY HAS 13 SYSTEM      */
=            /* PORTS USED AS SHOWN BELOW                         */

135  1  =    DECLARE

=       XTH$RETURN$PORT      LITERALLY  '000H', /* XTH BIOS RECEIVE PORT        */
=       FLIPPY$RETURN$PORT   LITERALLY  '001H', /* CPIO FLIPPY RECEIVE PORT    */
=       PRINTER$RETURN$PORT  LITERALLY  '002H', /* CPIO PRINTER RECEIVE PORT    */
=       TMC                  LITERALLY  '003H', /* TAPS MASTER CONTROLLER       */
=       CM                   LITERALLY  '004H', /* TAPS CM                      */
=       TAC                  LITERALLY  '005H', /* TAPS APPLICATION CONTROLLER  */
=       AM1                  LITERALLY  '006H', /* APPLICATION MANAGER #1       */
=       AM2                  LITERALLY  '007H', /* APPLICATION MANAGER #2       */
=       AM3                  LITERALLY  '008H', /* APPLICATION MANAGER #3       */
=       IO1                  LITERALLY  '009H', /* I/O MANAGER #1               */
=       IO2                  LITERALLY  '00AH', /* I/O MANAGER #2               */
=       TS1                  LITERALLY  '00BH', /* TAPS SPARE PORT #1           */
=       TS2                  LITERALLY  '00CH', /* TAPS SPARE PORT #2           */

=            /* CPIO IS DEVICE 1 IT CURRENTLY HAS 2 SYSTEM       */
=            /* PORTS USED AS SHOWN                              */

=       FLIPPY$XMIT$PORT     LITERALLY  '100H', /* CPIO FLIPPY XMIT PORT        */
=       PRINTER$XMIT$PORT    LITERALLY  '101H', /* CPIO PRINTER XMIT PORT       */

=            /* 544 IS DEVICE 2 IT HAS 1 SYSTEM PORT             */

=       XTH$I544$PORT1       LITERALLY  '200H'; /* XTH BIOS XMIT PORT */
=     /* XTH$I544$PORT1 SHOULD BE SET TO 200H IF 544 BOARD*/

      $INCLUDE (:DO:TAPSDEV/TCBDAT.SRC)
=     $SAVE

136  1  =      DECLARE TAPS$MASTER LITERALLY 'TMC';
137  1  =      DECLARE TAPS$CM     LITERALLY 'CM';
138  1  =      DECLARE TAPS$AM1 LITERALLY 'AM1';
=     /*
=     ******DEFINES THE TCB FOR XTH INTERFACE MESSAGE*****
=     */
139  1  =      DECLARE  TCB$PART$ONE  LITERALLY '
=         FUNCTION$CODE        BYTE,
=         FUNCTION$FLAGS       BYTE,
=         IND$DATA$PTR         POINTER,
=         DATA$LENGTH      WORD,
```

```
          =        COUNT$TRANSFERRED   WORD,
          =        NAME$PTR        POINTER',

          =        TCB$PART$TWO  LITERALLY  '
          =        EXCEPTION           WORD,
          =        IORS$T          WORD,
          =        TIT$ENTRY$TOKEN     WORD,
          =        RESERVED(4) WORD,
          =        TERM$STATE  BYTE,
          =       TT$INDEX   BYTE,
          =       MSG$DEST$MBX  WORD,
          =       TERM$NAME$544(16)    BYTE,
          =        MESS$START$544(80)  BYTE',
          =    /*

          =    */
          =        TCB$AREA$DATA LITERALLY 'STRUCTURE(
          =            TCB$PART$ONE,
          =            TCB$PART$TWO)';

          =    /*
          =    ****************DEFINES THE STRUCTURE OF AN AM MESSAGE********
          =    ****************SENT BY CM TO ACCOMPLISH A TRANSACTION********
          =    */
140    1  =     DECLARE AM$MESSAGE$AREA LITERALLY 'STRUCTURE(
          =            TCB$PART$ONE,EXCEPTION WORD,
          =        RESERVED(6) WORD,TERM$STATE BYTE,
          =            TT$INDEX BYTE,MSG$DEST$MBX WORD,
          =              TERM$NAME$544(16) BYTE,
          =            TT_INDEX BYTE,TWA_READ_FLAG BYTE,
          =            IO_SUFF_PTR POINTER,TWA_PTR POINTER)';
          =    /*
          =    ***********DEFINES THE MESSAGE STRUCTURE OF A MESSAGE*********
          =    ***********TO AN AM PROCESS BY CM AT AM INITIALIZATION********
          =    **************TIME*********
          =    */
141    1  =     DECLARE AM$STARTUP$MSG LITERALLY 'STRUCTURE(
          =                        PART1(6) BYTE,DATA_LENGTH WORD,
          =                        PART2(6) BYTE,EXCEPTION WORD,
          =                        PART3(32) BYTE,NAME(16) BYTE,
          =                        PORT_ID WORD,AM_TABLES WORD)';
          =    /*
          =    *********REDEFINES THE AM STARTUP MESSAGE FOR THE AM*****************
          =    *********THIS IS NECESSARY SINCE MASTER REBUILDS THE***************
          =    ******** THE MESSAGE WHEN IT STARTS THE TASK        ***************
          =    */
142    1  =    DECLARE AM$START$MSG LITERALLY 'STRUCTURE(
          =                                       TCB (49) BYTE,
          =                                       AM_PORT_ID WORD,
          =                                       AM_TABLES WORD)';

          =    /*
          =    ********************SPECIAL FUNCTION CODES******************
          =    */
143    1  =    DECLARE RETURN_FROM_CM LITERALLY '077H';
144    1  =    DECLARE MUST_READ LITERALLY '0F0H';
145    1  =    DECLARE INIT_TWA LITERALLY '0F1H';
```

```
146   1  =    DECLARE TAPS_DOWN LITERALLY 'OFEH';
147   1  =    DECLARE CM_LOGON  LITERALLY 'OFFH';
148   1  =    DECLARE TASK_START LITERALLY 'OFDH';
         =    /*
         =    ********************TERMINAL STATUS CODES**********************
         =    */
149   1  =    DECLARE TAPS_TERM LITERALLY '-4';
         =    /*
         =    ************************ DEFINE TAPSER CODES*****************
         =    */
150   1  =    DECLARE TERMOP_ERR LIT '136',
         =            SCREENS_ERR LIT'137',
         =            CMDOWN      LIT '134',
         =            SYSTEMS_ERR LIT '138',
         =            TRANS_ERR  LIT '140',
         =            TWAREAD    LIT '139',
         =            TWA_ERR    LIT '139',
         =            OPENER     LIT '106',
         =            BADFC      LIT '107',
         =            NOTACT     LIT '110';
         =    $RESTORE
151   1       DECLARE FSTIME WORD INITIAL(0);

152   1       DECLARE TERM_NAME(16) BYTE DATA('                ');
153   1       DECLARE CMFLAG BYTE;
154   1       DECLARE DOWN_FLAG BYTE;


155   1       DECLARE NAME(*) BYTE DATA(10,'TAPSMASTER');
156   1       DCL RDFCN WORD;
157   1       DCL TTC  WORD;
158   1       DCL MSGTKN WORD;
159   1       DCL IPMBOX WORD;
160   1       DCL MSGSEG WORD;
161   1       DCL STATS WORD;
162   1       DCL OUTKN WORD;
163   1       DCL MODE  WORD;
164   1       DCL J     WORD;
165   1       DCL I     WORD;
166   1       DCL K     WORD;
167   1       DCL SHARE BYTE;
168   1       DCL ICOUNT WORD;
169   1       DCL RJTKN WORD;
170   1       DCL DO_CONNECTION WORD;
171   1       DCL TAPS_USER WORD;
172   1       DCL EX_OVERIDE STRUCTURE(OFFSET WORD,
                                         BASE WORD,
                                         MODE BYTE);
173   1       DCL INTKN WORD;
174   1       DCL IORPTR POINTER;
175   1       DCL IORPTS STRUCTURE(OFFSET WORD,BASE WORD) AT (@IORPTR);
176   1       DCL IOREST BASED IORPTR STRUCTURE(STATUS WORD,USTATS WORD,
                                                 ACTUAL WORD);
177   1       DCL IPMPTR POINTER;
178   1       DCL IPMPTS STRUCTURE(OFFSET WORD,BASE WORD) AT (@IPMPTR);
179   1       DCL IPMSG BASED IPMPTR TCB$AREA$DATA;
180   1       DCL TAPSCM$MBX WORD; /*TAPS$CM GLOBAL MAILBOX*/
```

```
181   1             DCL SEGTKN WORD;
182   1             DCL MSGPTR POINTER;
183   1             DCL MSGPTS STRUCTURE(OFFSET WORD,BASE WORD) AT (@MSGPTR);
184   1             DCL THEMSG BASED MSGPTR TCB$AREA$DATA;
185   1             DCL PATH_NAME(20)BYTE INITIAL('                    ');

186   1       DCL LIST_LENGTH LITERALLY '8';
187   1       DCL TASK_MBX WORD;
188   1       DCL TASK_LIST (LIST_LENGTH) STRUCTURE(NAME (8) BYTE,
                                                  CODE      WORD,
                                                  DATASEG   WORD,
                                                  STACK     WORD,
                                                  MSGSEG    WORD,
                                                  TASKID    WORD);
189   1       DCL TASK_ACTIVE BYTE;


190   1           DCL STATUS WORD;
191   1          DCL    MAILBOX$FLAGS        WORD;
192   1          DCL    INITIAL$MSG(*)    BYTE DATA(1BH,',',1BH,29H,
                                                 'TAPS MASTER  TERMINAL',
                                          1BH,'(',1BH,'G<                ',1BH,'=!5',
                                          1BH,'(',1BH,'G<                ',
                                          1BH,'G','0',1BH,'=7 ',1BH,'G<                  ',
                                                                                  ',
                                    1BH,'= 6',1BH,'&',1BH,'B',1BH,'"',0DH);/* 1BH = ESC CHAR*/
193   1          DCL    CLEAR$SCREEN(*) BYTE DATA(1BH,'*',1BH,'"');
194   1          DCL    HI_MESSAGE(*) BYTE DATA(1BH,'F','TPS System running',CR,
                                          1BH,'C',' iTPS HI V1.0',CR,LF);
195   1          DCL    REFRESH$MSG(*) BYTE DATA(1BH,'= 6','                  ',
                                                 1BH,'=!6','                ',1BH,'= 6',1BH,'"');
196   1       DECLARE UNLOCK (2) BYTE DATA(1BH,'"');
197   1          DCL    INIT_FLAG BYTE;
198   1          DCL    TCB$SIZE    WORD;
199   1          DCL    MASTER$MBX WORD PUBLIC;
200   1          DCL    MESSAGE$PTR    POINTER;
201   1          DCL    MESSAGE$RECV$PTR    POINTER;
202   1          DCL    OPEN$CONNECTION    WORD;
203   1          DCL    TCB$AREA BASED MESSAGE$PTR    TCB$AREA$DATA;
204   1          DCL    MESS$LENGTH        WORD;
205   1          DCL    MESS$RECV$LENGTH    WORD;
206   1          DCL    DUMMY    WORD;
207   1          DCL    FUNCODE WORD;
208   1          DCL    RETURN$MBOX WORD;

          /*************************************************************************/
          /*  OPEN    TERMINAL USING TCB SENT FROM TAPS LOGON CUSP                 */
          /*************************************************************************/
209   1       OP$LOGSTERM: PROCEDURE(MESG$PTR,RTN_CODE) ;
210   2             DECLARE (MESG$PTR,RTN_CODE) POINTER;
211   2             DECLARE MESG BASED MESG$PTR TCB$APEA$DATA;
212   2             DECLARE USTATS BASED RTN_CODE WORD;

213   2          MESS$LENGTH = 128 ;
214   2          MESG.IND$DATA$PTR = OFFFFH;
```

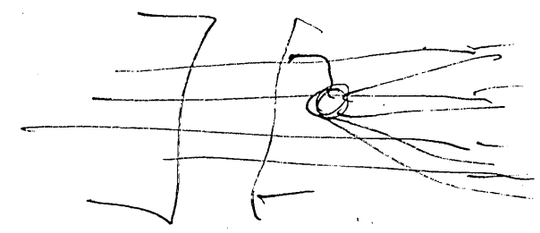
DCL Myself lite
DCL who$Knows1 "
DCL who$Knows2 "
DCL Root$job "

```
215   2 7       MESG.FUNCTION$CODE = 4;  /*  OPEN      */

216   2       SEND$MESSAGE:
                CALL MQ$SEND$MESSAGE(
                            OPEN$CONNECTION,
                            MESG$PTR,
                            MESS$LENGTH,
                            TAPS$MASTER,
                            @STATUS);
217   2           CALL MQ$DELETE$RCB(MESG$PTR,@STATUS);
218   2       USTATS = STATUS;

219   2       RETURN;
220   2       END OP$LOG$TERM;
              /*************************************************************************/
              /*  CLOSE   TERMINAL                                                     */
              /*************************************************************************/
221   1       CL$LOG$TERM: PROCEDURE(MESG$PTR,RTN_CODE) ;
222   2             DECLARE (MESG$PTR,RTN_CODE) POINTER;
223   2             DECLARE MESG BASED MESG$PTR TCB$AREA$DATA;
224   2             DECLARE USTATS BASED RTN_CODE WORD;

225   2           MESS$LENGTH = 32 + 16 ;
226   2           MESG.IND$DATA$PTR = 0FFFFH;

227   2           MESG.FUNCTION$CODE = 5;  /*  CLOSE     */

228   2       SEND$MESSAGE:
                CALL MQ$SEND$MESSAGE(
                            OPEN$CONNECTION,
                            MESG$PTR,
                            MESS$LENGTH,
                            TAPS$MASTER,
                            @STATUS);
229   2           CALL MQ$DELETE$RCB(MSG$PTR,@STATUS);
230   2       USTATS = STATUS;

231   2           RETURN;
232   2       END CL$LOG$TERM;
              /*************************************************************************/
              /*   WRITE TERMINAL BLOCK MODE                                           */
              /*************************************************************************/

233   1       WRTERM: PROCEDURE(BUFFER,COUNT);
234   2             DECLARE BUFFER POINTER;
235   2             DECLARE COUNT  WORD;

236   2           MESS$LENGTH = COUNT + 48;
237   2           MSGPTR=BUFFER;
238   2           MESS$LENGTH=MESS$LENGTH;
239   2           THEMSG.FUNCTION$CODE = 1;  /*  WRITE     */
240   2           THEMSG.DATA$LENGTH = COUNT;

241   2           CALL MQ$SEND$MESSAGE(
                            OPEN$CONNECTION,
                            BUFFER,
```

```
                        MESS$LENGTH,
                        TAPS$MASTER,
                        @STATUS);
242   2           CALL MQ$DELETE$RCB(BUFFER,@STATUS);
243   2         RETURN;
244   2         END WRTERM;

          /*************************************************************************/
          /*   READ TERMINAL BLOCK MODE                                            */
          /*************************************************************************/


245   1       RDTERM: PROCEDURE(BUFFER,COUNT,ACTUAL);
246   2               DECLARE (BUFFER,ACTUAL) POINTER;
247   2               DECLARE COUNT WORD;

248   2               DECLARE RESULT BASED ACTUAL WORD;

249   2           MESS$LENGTH = 32 + 16 + COUNT;
250   2           MSGPTR=BUFFER;
251   2           THEMSG.FUNCTION$CODE = 00; /*  READ      */
252   2           THEMSG.DATA$LENGTH = COUNT;
253   2           THEMSG.FUNCTION$FLAGS = 0;   /*  BLOCK MODE*/
254   2           CALL MQ$SEND$MESSAGE(
                        OPEN$CONNECTION,
                        BUFFER,
                        MESS$LENGTH,
                        TAPS$MASTER,
                        @STATUS);

255   2           CALL MQ$DELETE$RCB(BUFFER,@STATUS);

256   2           RETURN;
257   2       END RDTERM;


          /*************************************************************************/
          /* ROUTINE TO CHECK RCB SIZE AND GET A NEW ONE IF ITS TO SMALL  */
          /*************************************************************************/

258   1       CHECK_RCB_SIZE: PROCEDURE(REQ_SIZE);

259   2       DECLARE REQ_SIZE WORD;
260   2       DECLARE NEW_RCB POINTER;
261   2       DECLARE CUR_SIZE WORD;

262   2       CUR_SIZE=RQ$GET$SIZE(MSGPTS.BASE,@STATUS);
263   2       IF CUR_SIZE-16<REQ_SIZE THEN DO;
265   3          NEW_RCB=MQ$CREATE$RCB(REQ_SIZE,@STATUS);
266   3          CALL MOVB(MSGPTR,NEW_RCB,48+THEMSG.COUNT$TRANSFERRED);
267   3          CALL MQ$DELETE$RCB(MSGPTR,@STATUS);
268   3          MSGPTR=NEW_RCB;
269   3       END;
270   2       RETURN;
271   2       END CHECK_RCB_SIZE;
```

```
                /*****************************************************************/
                /*  START A TASK                                                 */
                /*****************************************************************/

 272    1       START_TASK: PROCEDURE(FILE_NAME,RTN_MSG);
 273    2            DECLARE FILE_NAME POINTER,RTN_MSG WORD;
 274    2          DCL NAME (*)BYTE DATA(10,'START_TASK');
 275    2          DCL LSGPTR POINTER;
 276    2          DCL LSGPTS STRUCTURE(OFFSET WORD,BASE WORD) AT (@LSGPTR);
 277    2          DCL TLS BASED LSGPTR STRUCTURE(EXCODE WORD,
                                                  RECCNT WORD,
                                                  ERRREC BYTE,
                                                  UNDEFS WORD,
                                                  INIP WORD,
                                                  CSB WORD,
                                                  STKOFF WORD,
                                                  SSB WORD,
                                                  SSZ WORD,
                                                  DSB WORD);

 278    2          DCL TTC    WORD;
 279    2          DCL TYPE_CODE WORD;
 280    2          DCL LDSEG WORD;
 281    2          DCL RTTKN WORD;

 282    2          DCL PRIORITY BYTE;
 283    2          DCL DATA_SEG POINTER;
 284    2          DCL DATA_SEG_PTS STRUCTURE(OFFSET WORD,BASE WORD) AT (@DATA_SEG);
 285    2          DCL CODE_SEG POINTER;
 286    2          DCL CODE_SEG_PTS STRUCTURE(OFFSET WORD,BASE WORD) AT (@CODE_SEG);
 287    2          DCL STACK_SEG POINTER;
 288    2          DCL STACK_SEG_PTS STRUCTURE(OFFSET WORD,BASE WORD) AT (@STACK_SEG);
 289    2          DCL STACK_SIZE WORD;
 290    2        SEGTKN=0;
 291    2          CALL RQ$A$ATTACH$FILE(TAPS_USER,DO_CONNECTION,FILE_NAME,
                                          LOCAL_MBX,@STATUS);
 292    2               IF STATUS<>0 THEN GOTO ERROR_EXIT;
 294    2          RDFCN=RQ$RECEIVE$MESSAGE(LOCAL_MBX,0FFFFH,0,@STATUS);
 295    2          TYPE_CODE=RQ$GET$TYPE(RDFCN,@STATUS);
 296    2               IF TYPE_CODE=6 THEN DO;
 298    3                                   STATUS=107;
 299    3                                   SEGTKN=RDFCN;
 300    3                                   GOTO ERROR_EXIT;
 301    3                                   END;
 302    2          CALL RQ$A$LOAD(RDFCN,LOCAL_MBX,@STATUS);
 303    2               IF STATUS<>0 THEN GOTO ERROR_EXIT;
 305    2           LDSEG = RQ$RECEIVE$MESSAGE(LOCAL_MBX,0FFFFH,@TTC,@STATUS);
 306    2           SEGTKN=LDSEG;
 307    2                  IF STATUS<>0 THEN GOTO ERROR_EXIT;
 309    2           LSGPTS.BASE=LDSEG;
 310    2           LSGPTS.OFFSET=0;
 311    2           IF TLS.EXCODE <> 0 THEN DO;
 313    3               STATUS=TLS.EXCODE;
 314    3               GOTO ERROR_EXIT;
 315    3           END;
```

```
316   2          PRIORITY=150;
317   2          DATA_SEG_PTS.BASE=TLS.DSB;
318   2          DATA_SEG_PTS.OFFSET=0;
319   2          STACK_SEG_PTS.BASE=TLS.SSB;
320   2          STACK_SEG_PTS.OFFSET=0;
321   2          STACK_SIZE=TLS.SSZ;
322   2          CODE_SEG_PTS.BASE=TLS.CSB;
323   2          CODE_SEG_PTS.OFFSET=TLS.INIP;
324   2          RTTKN = RQSCREATESTASK(PRIORITY,
                                          CODE_SEG,
                                          DATA_SEG_PTS.BASE,
                                          STACK_SEG,
                                          STACK_SIZE,
                                           0,
                                          @STATUS);
325   2            IF STATUS<>0 THEN GOTO ERROR_EXIT;
                  /*FIND AN EMPTY LIST ENTRY AND SAVE THE SEGMENT VALUES*/
                  /*SO WE CAN DELETE THEM LATER                         */
327   2            DO I=0 TO LIST_LENGTH-1;
328   3              IF TASK_LIST(I).CODE=0 THEN DO;
330   4                 CALL MOVB(@PATH_NAME(6),@TASK_LIST(I).NAME,8);
331   4                 TASK_LIST(I).CODE=CODE_SEG_PTS.BASE;
332   4                 TASK_LIST(I).DATASEG=DATA_SEG_PTS.BASE;
333   4                 TASK_LIST(I).STACK=STACK_SEG_PTS.BASE;
334   4                 TASK_LIST(I).MSGSEG=IPMPTS.BASE;
335   4                 TASK_LIST(I).TASKID=RTTKN;
336   4                 I=LIST_LENGTH;
337   4               END;
338   3             END;
339   2          CALL RQSDELETESSEGMENT(LDSEG,@STATUS);
340   2        RETURN;
341   2        ERROR_EXIT:
                        CALL TAPSER(STATUS,RTN_MSG,@NAME);
342   2                  IF SEGTKN<>0 THEN CALL RQSDELETESSEGMENT(SEGTKN,@STATUS);
344   2                  RETURN;
345   2        END START_TASK;

                  /*************************************************************/
                  /*   MAIN PROCEDURE CODE                                     */
                  /*************************************************************/

346   1              RJTKN=RQSGETSTASKSTOKENS(0,@STATS);
347   1              EX_OVERIDE.OFFSET=0;
348   1              EX_OVERIDE.BASE=0;
349   1              EX_OVERIDE.MODE=0;
350   1              CALL RQSSETSEXCEPTIONSHANDLER(@EX_OVERIDE,@STATUS);

351   1              LOCAL_MBX=RQSCREATESMAILBOX(0,@STATUS);
352   1              MASTERSMBX=MQSACTIVATESMAILBOX(TAPSSMASTER,@STATUS);

353   1              OPENSCONNECTION=MQSOPENSDESTINATION(XTHSI544SPORT1,@STATUS);
354   1              CALL RQSCATALOGSOBJECT(0,MASTERSMBX,@(10,'TAPSMASTER'),@STATS);

355   1              TAPS_USER=RQSGETSDEFAULTSUSER(0,@STATUS);
356   1              DO_CONNECTION=RQSGETSDEFAULTSPREFIX(0,@STATUS);
357   1              CALL SETB(0,@TASK_LIST,144);
```



root

rcv/resp mbx

output port

?

?

```
358   1            CMFLAG=FALSE; /*SET FLAG TO INDICATE CM NOT YET ACTIVE*/

359   1            DO FOREVER;
360   2               MESSAGE$RECV$PTR=MQ$RECEIVE$MESSAGE(MASTER$MBX,
                                                          0FFFFH,
                                                          .MESS$RECV$LENGTH,
                                                          .DUMMY,
                                                          .STATUS);
361   2               MSGPTR=MESSAGE$RECV$PTR;
362   2               I=THEMSG.TERM$STATE;
363   2     DOIT: DO CASE I;
364   3     /*CASE 0*/ DO;  /*START A MASTER SUBTASK AND SEND IT A MESSAGE*/
                           /*CHECK FOR SPECIAL COMMANDS TO TAPS MASTER*/

365   4                CALL CHECK_RCB_SIZE(128);

366   4                CALL SETB(' ',.PATH_NAME,16);
367   4                TASK_ACTIVE=FALSE;

368   4                J=1;
369   4                DO WHILE THEMSG.MESS$START$544(J)<>' '
                                  AND J < 16;
370   5                   PATH_NAME(J)=THEMSG.MESS$START$544(J);
371   5                   J=J+1;
372   5                END;
373   4                PATH_NAME(0)=J-1;

                          /*CHECK FOR VALID TAPS COMMAND*/

374   4                 IF CMP$B(.THEMSG.MESS$START$544(1),.('TAPS'),4)<>0FFFFH THEN DO;
376   5                      CALL TAPSER(109,MSGPTS.BASE,.NAME);
377   5                      I=6;
378   5                      GOTO DOIT;
379   5                      END;

                         /* CHECK IF ITS A REQUEST TO STOP TAPS */

380   4                IF CMP$B(.THEMSG.MESS$START$544(6),.('DOWN'),4)=0FFFFH THEN DO;
382   5                       I=10;
383   5                       IF CMFLAG<>TRUE THEN DO;
385   6                          CALL TAPSER(NOTACT,MSGPTS.BASE,.NAME);
386   6                          GOTO DOIT;
387   6                       END;
388   5                       I=12;
389   5                      DOWN_FLAG=TRUE;
390   5                      CMFLAG=FALSE;
391   5                      THEMSG.TTSINDEX=0FEH;
392   5                      GOTO DOIT;
393   5                      END;

                         /* CHECK FOR A REQUEST TO SWITCH FROM MASTER TO CM */

394   4                IF CMP$B(.THEMSG.MESS$START$544(6),.('LOGON'),5)=0FFFFH THEN DO;
396   5                      I=9;
397   5                      THEMSG.FUNCTION$CODE=3;
398   5                      GOTO DOIT;
399   5                      END;
```

```
                        /* CHECK FOR A REQUEST TO SWITCH FROM MASTER TO HI */

400   4                 IF CMPB(@THEMSG.MESS$START$544(6),@('LOGOFF'),6)=0FFFFH THEN DO;
402   5                       I=10;
403   5                       GOTO DOIT;
404   5                       END;

                             /* ITS NONE OF THE ABOVE SO IT MUST BE A MASTER */
                             /* TERMINAL FUNCTION REQUEST SO LETS DO IT      */

                             /*FIRST WE GET A SEGMENT AND CONSTRUCT A MESSAGE*/

405   4                 J=16;
406   4                 DO WHILE THEMSG.MESS$START$544(J) = 1FH OR
                                 THEMSG.MESS$START$544(J) = 1CH;
407   5                    J=J+1;
408   5                 END;
409   4                  K=0;
410   4                  IPMPTR=MQ$CREATE$RCB(256,@STATS);
411   4                  CALL SETB(0,IPMPTR,32);
412   4                  CALL MOVB(@THEMSG.TERM$NAME$544,@IPMSG.TERM$NAME$544,16);
413   4                  IPMSG.IND$DATA$PTR=0FFFFFH;
414   4                  IPMSG.TERM$STATE=0;
415   4                  IPMSG.EXCEPTION=0;
416   4                  IPMSG.MSG$DEST$MBX=THEMSG.MSG$DEST$MBX;
417   4                DO WHILE J<THEMSG.COUNT$TRANSFERRED+1 AND THEMSG.MESS$START$544(J)<>' ';
418   5                     IPMSG.MESS$START$544(K+1)=THEMSG.MESS$START$544(J);
419   5                     J=J+1;
420   5                     K=K+1;
421   5                 END;
422   4                 IPMSG.MESS$START$544(0)=K;
423   4                 IPMSG.EXCEPTION=0;
424   4                 IPMSG.DATA$LENGTH=208;

                        /*NEXT WE CHECK TO SEE IF THIS TASK IS ALREADY*/
                        /*ACTIVE AND IF SO FIND HIS MAILBOX AND SEND  */

425   4                 DO I=0 TO LIST_LENGTH-1;
426   5                   IF CMPB(@PATH_NAME(6),@TASK_LIST(I).NAME,
                                 PATH_NAME(0)-5)=0FFFFH
                            THEN DO;
428   6                          IPMBOX=RQ$LOOK$UP$OBJECT(0,@PATH_NAME,0,@STATUS);
429   6                          TASK_ACTIVE=TRUE;
430   6                          I=LIST_LENGTH;
431   6                          END;
432   5                       ELSE TASK_ACTIVE=FALSE;
433   5               END;
434   4                  IF TASK_ACTIVE=TRUE THEN GOTO SENDIT;
436   4                 IPMBOX=RQ$CREATE$MAILBOX(0,@STATS);
437   4                  CALL RQ$CATALOG$OBJECT(0,IPMBOX,@PATH_NAME,@STATUS);
438   4                 CALL MQ$DELETE$RCB(MSGPTR,@STATUS);

439   4                 CALL START_TASK(@PATH_NAME,IPMPTS.BASE);
440   4                      IF IPMSG.EXCEPTION<>0 THEN DO;
442   5                               I=6;
443   5                               CALL RQ$UNCATALOG$OBJECT(0,@PATH_NAME,@STATUS);
```

```
444   5                              CALL RQ$DELETE$MAILBOX(IPMBOX,@STATUS);
445   5                              MSGPTR=IPMPTR;
446   5                              GOTO DOIT;
447   5                              END;
448   4                  SENDIT:
                          CALL RQ$SEND$MESSAGE(IPMBOX,IPMPTS.BASE,MASTER$MBX,@STATS);

449   4                  END;
450   3       /*CASE 1*/ DO;  /* SEND MESSAGE TO TERMINAL AND SET TERMINAL    */
                           /* STATE TO SEND TERMINAL RESPONSE TO ORIGINATOR*/
451   4                  THEMSG.TERM$STATE=2;
452   4                  CALL WRTERM(MSGPTR,THEMSG.DATA$LENGTH);
453   4                  END;
454   3       /*CASE 2*/ DO; /* READ THE TERMINAL INPUT IN RESPONSE TO THE*/
                           /* MESSAGE JUST SENT                        */
455   4                  THEMSG.TERM$STATE=3;
456   4                  CALL RDTERM(MSGPTR,80,@ICOUNT);
457   4                  END;
458   3       /*CASE 3*/ DO; /*FORWARD THE MESSAGE TO THE MSG$DEST$MBX */
                           /*SPECIFIED IN THE TCB JUST RECEIVED*/

459   4                  IPMBOX=THEMSG.MSG$DEST$MBX;
460   4                  CALL RQ$SEND$MESSAGE(IPMBOX,MSGPTS.BASE,MASTER$MBX,@STATUS);
461   4                  END;
462   3       /*CASE 4*/ DO;  /*RESERVED FOR TERMINAL SWITCH TO CM CONTROL*/
463   4                  THEMSG.TERM$STATE=13;
464   4                  THEMSG.DATA$LENGTH=80;
465   4                  MESS$LENGTH=THEMSG.DATA$LENGTH+48;
466   4                  CALL MQ$SEND$MESSAGE(TAPSCM$MBX,MSGPTR,MESS$LENGTH,0,@STATUS);
467   4                  END;
468   3       /*CASE 5*/ DO;  /*DISPLAY MESSAGE ON TERMINAL AND DO NOTHING ELSE*/
469   4                  THEMSG.TERM$STATE=19;
470   4                  CALL WRTERM(MSGPTR,THEMSG.DATA$LENGTH);
471   4                  END;
472   3       /*CASE 6*/ DO;  /*WRITE AN ERROR MESSAGE TO THE TERMINAL*/
473   4                  THEMSG.TERM$STATE=7;
474   4                  CALL WRTERM(MSGPTR,THEMSG.DATA$LENGTH);
475   4                  END;
476   3       /*CASE 7*/ DO;   /*PROMPT FOR ANOTHER TAPS COMMAND*/
477   4                  IF THEMSG.FUNCTION$CODE=RETURN_FROM_CM THEN
478   4                    INIT_FLAG=TRUE;
479   4                  THEMSG.TERM$STATE=8;
480   4                  IF INIT_FLAG=TRUE THEN DO; /*DO INITIAL*/
482   5                  CALL CHECK_RCB_SIZE(208);
483   5                  CALL MOVB(@INITIAL$MSG,@THEMSG.MESS$START$544,148);
484   5                  INIT_FLAG=FALSE;
485   5                  CALL WRTERM(MSGPTR,148);
486   5                 END;
                         ELSE /*DO REFRESH*/
487   4                    DO;
488   5                    CALL CHECK_RCB_SIZE(96);
489   5                    CALL MOVB(@REFRESH$MSG,@THEMSG.MESS$START$544,45);
490   5                    CALL WRTERM(MSGPTR,45);
491   5                    END;
492   4                  END;
493   3       /*CASE 8*/ DO;  /*ISSUE A READ TO A TERMINAL*/
494   4                  THEMSG.TERM$STATE=0;
```

```
                    CALL RDTERM(MSGPTR,36,@ICOUNT);
                    END;
/*CASE 9*/  DO; /*RESERVED FOR TAPS LOGON*/

                    IF THEMSG.FUNCTION$CODE=1 THEN THEMSG.TERM$STATE=17;
                    ELSE THEMSG.TERM$STATE=18;
                    CALL OP$LOG$TERM(MSGPTR,@STATUS);
                    END;
/*CASE 10*/DO; /*PREPARE FOR TERMINAL LOGOFF BY SENDING A CLEAR */
                      /*SCREEN AND SET TO CONVERSATIONAL MODE MESSAGE   */
                    CALL CHECK_RCB_SIZE(96);
                       /*CHECK IF THE TERMINAL BELONGS TO TAPS*/
                       /* IF SO SEND TAPS DOWN MESSAGE AND THATS*/
                       /* THATS ALL FOR THIS GUY                */

                    IF THEMSG.MSG$DEST$MBX=0FFFFH THEN DO;
                       THEMSG.TERM$STATE=19;
                       CALL TAPSER(134,MSGPTS.BASE,@NAME);
                       CALL MOVB(@CLEAR$SCREEN,@THEMSG.MESS$START$544(0),4);
                       CALL WRTERM(MSGPTR,THEMSG.DATA$LENGTH);
                    END;
                  ELSE DO;
                       THEMSG.TERM$STATE=13;
                       CALL MOVB(@CLEAR$SCREEN,@THEMSG.MESS$START$544,42);
                       CALL WRTERM(MSGPTR,42);
                      END;

                    END;
/*CASE 11*/ DO;
                       CALL MOVB(@PATH_NAME(1),
                                 @THEMSG.MESS$START$544(4),16);
                       CALL MOVB(@REFRESH$MSG,@THEMSG.MESS$START$544(0),4);
                       THEMSG.TERM$STATE=19;
                       CALL WRTERM(MSGPTR,16+4);

                    END;
/*CASE 12*/DO;/*TAPS DOWN PROCESSING STARTS WITH MSG TO CM*/

                    MESS$LENGTH=MESS$RECV$LENGTH;

                    CALL MQ$SEND$MESSAGE(TAPSCM$MBX,MSGPTR,MESS$LENGTH,
                                                           0,@STATUS);
                    CALL MQ$DELETE$RCB(MSGPTR,@STATUS);
                    END;
/*CASE 13*/DO; /*TERMINAL DISCONNECTED FROM TAPS */
                    THEMSG.TERM$STATE=16;
                    CALL CL$LOG$TERM(MSGPTR,@STATUS);
                    END;
/*CASE 14*/DO; /*CM STARTUP COMPLETE */
                    TAPSCM$MBX=MQ$OPEN$DESTINATION(TAPS$CM,@STATUS);
                    CMFLAG=TRUE;
                    THEMSG.TERM$STATE=6;
                    I=6;
                    GOTO DOIT;
                   END;
/*CASE 15*/DO;/*DELETE SUBORDINATE TASK ON REQUEST*/
```

```
541   4                 TASK_MBX=RQ$LOOKUP$OBJECT(0,@THEMSG.MESS$START$544,
                                                  0FFFFH,@STATUS);
542   4                 CALL RQ$UNCATALOG$OBJECT(0,@THEMSG.MESS$START$544,@STATUS);
543   4                 CALL RQ$DELETE$MAILBOX(TASK_MBX,@STATUS);
544   4                  DO I = 0 TO LIST_LENGTH-1;
545   5                     IF CMPB(@THEMSG.MESS$START$544(6),
                                  @TASK_LIST(I).NAME,8)=0FFFFH THEN DO;
547   6                       CALL RQ$DELETE$SEGMENT(TASK_LIST(I).CODE,@STATUS);
548   6                       CALL RQ$DELETE$SEGMENT(TASK_LIST(I).DATASEG,@STATUS);
549   6                       CALL RQ$DELETE$SEGMENT(TASK_LIST(I).STACK,@STATUS);
550   6                       IPMPTS.BASE=TASK_LIST(I).MSGSEG;
551   6                       CALL MQ$DELETE$RCB(IPMPTR,@STATUS);
552   6                       CALL RQ$DELETE$TASK(TASK_LIST(I).TASKID,@STATUS);
553   6                       TASK_LIST(I).CODE=0;
554   6                       CALL SETB(0,@TASK_LIST(I),8);
555   6                       I=LIST_LENGTH;
556   6                     END;
557   5                   END;
558   4                   IF THEMSG.FUNCTION$CODE = TASK_START THEN DO;
560   5                     CALL MOVB(@THEMSG.MESS$START$544(16),
                                    @THEMSG.MESS$START$544(1),16);
561   5                     THEMSG.TERM$STATE=0;
562   5                     CALL WRTERM(MSGPTR,16);
563   5                   END;
564   4                   IF CMPB(@THEMSG.MESS$START$544(1),@('TAPS/CM'),7)=0FFFFH
                           AND DOWN_FLAG=TRUE THEN DO; /*PROMPT THE MASTER TERMINAL*/
566   5                     I=6;
567   5                     CALL TAPSER(CMDOWN,MSGPTS.BASE,@NAME);
568   5                     GOTO DOIT;
569   5                     END;
570   4                   IF THEMSG.TT$INDEX=7 THEN DO;
572   5                      I=7;
573   5                      INIT_FLAG=TRUE;
574   5                      GOTO DOIT;
575   5                      END;
576   4                    CALL MQ$DELETE$RCB(MSGPTR,@STATUS);
577   4                   END;
578   3      /*CASE 16*/  DO;
579   4                   DUMMY = RQ$CREATE$SEGMENT(16,@STATUS);
580   4                   CALL RQ$SEND$MESSAGE(THEMSG.MSG$DEST$MBX,
                                              DUMMY,0,@STATUS);
581   4                   CALL MQ$DELETE$RCB(MSGPTR,@STATUS);
582   4                   END;
583   3      /*CASE 17*/  DO;
584   4                   IF THEMSG.EXCEPTION<>0 THEN DO;
586   5                             CALL TAPSER(OPENER,MSGPTS.BASE,@NAME);
587   5                             THEMSG.TERM$STATE=10;
588   5                             GOTO END$CASE$17;
589   5                             END;

590   4                   INIT_FLAG=TRUE;
591   4                   THEMSG.TERM$STATE=7;

592   4                   END$CASE$17:
                           I =THEMSG.TERM$STATE;
593   4                   GOTO DOIT;
```

```
 594   4                    END;
 595   3       /*CASE 18*/ DO;

 596   4                    IF CMFLAG<>TRUE THEN DO;
 598   5                      CALL TAPSER(NOTACT,MSGPTS.BASE,@NAME);
 599   5                      THEMSG.TERM$STATE=10;
 600   5                      GOTO END$CASE$18;
 601   5                      END;
 602   4                    THEMSG.TERM$STATE=4;
 603   4                    THEMSG.TT$INDEX=0FFH;

 604   4                     END$CASE$18:
                             I = THEMSG.TERM$STATE;
 605   4                     GOTO DOIT;

 606   4                     END;
 607   3       /*CASE 19*/  DO;/*RELEASE THE RCB AND WAIT FOR NEXT MESSAGE*/

 608   4                     CALL MQ$DELETE$RCB(MSGPTR,@STATUS);
 609   4                     END;

 610   3         END;
 611   2        END;
 612   1       END MASTER;
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 0E90H   3728D
     CONSTANT AREA SIZE = 0139H    313D
     VARIABLE AREA SIZE = 0121H    289D
     MAXIMUM STACK SIZE = 0020H     32D
     991 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```