```
*=============================================================
*PROC = COMPRS
*
*   PROCEDURE NAME: COMPRS
*
*   COMPRS EXAMINES THE CURRENT SENTENCE STACK AND DELETES
*   ALL DOUBLE TOUCHES FROM THE STACK. THIS PROCEDURE
*   PROCEEDS IN TWO PHASES.  FIRST, STORE THE TOP STACK
*   ELEMENT AND THEN SCAN THROUGH THE REST OF THE STACK
*   TO FIND AN IDENTICAL ELEMENT.
*   IF A MATCH IS FOUND, STASH >FF INTO THE
*   IMMEDIATE FLAG AREA OF BOTH ELEMENTS ON THE STACK.
*   NOW, POINT TO THE NEXT ELEMENT ON THE STACK WHICH HAS
*   NOT BEEN FLAGGED FOR DELETION, AND TRY TO FIND A MATCH
*   FOR IT.  THIS PROCEDURE CONTINUES UNTIL ALL ELEMENTS
*   HAVE BEEN EXAMINED.
*
*   THE SECOND PHASE OF COMPRS STARTS FROM THE BOTTOM OF
*   THE STACK AND PROCEEDS UPWARD.  ALL ELEMENTS WHICH
*   HAVE NOT BEEN FLAGGED FOR DELETION ARE COPIED DOWN
*   ONTO THE STACK, OVER WRITING ELEMENTS WHICH HAVE BEEN
*   FLAGGED FOR DELETION.
*
*   INPUT: CURRENT SENTENCE STACK
*   OUTPUT: COMPRESSED STACK
*   RESULT: DUPLICATE TOUCHES ARE ELIMINATED FROM THE STACK
*   TO CALL: BL @COMPRS
*
*=============================================================
```

```
*=============================================================
*PROC = SRTES
*
*   PROCEDURE NAME: SRTES
*
*   SRTES SORTS THE ELEMENT STACK BY THE SYNTACTIC CLASS OF
*   ITS ELEMENTS.  THE LOWEST CLASS IS SORTED TO THE TOP OF
*   THE STACK.  THE SORT IS DONE SIMPLY BY USING THE SYNTAX
*   CLASS OF EACH ELEMENT.
*
*   INPUT: SENTENCE STACK
*   OUTPUT: SORTED STACK
*   RESULT: STACK IS SORTED BY SYNTAX CLASS
*   TO CALL: BL @SRTES
*
*=============================================================
```

```
*=============================================================
*MOD = MHANDLER
*
*   MODULE MHANDLER CONTAINS THE MENU HANDLER ROUTINES.
*   ALL ROUTINES WHICH CREATE, MODIFY OR DELETE THE MENUS
*   IS CONTAINED HERE.
*
*=============================================================
```

```
*=============================================================
*DATA
*
*    MODULE MHANDLER DATA FIELDS
*
*
*=============================================================
```

```
*****************************************************
*PROC = DRMENU
*
*   PROCEDURE NAME: DRMENU
*
*   DRMENU IS THE ROUTINE WHICH MANAGES THE DRAWING OF
*   MENUS.  IF THE REQUESTED MENU IS THE CURRENT MENU, THEN
*   DRMENU CALLS RDMENU (RE-DRAW MENU).  IF THE REQUESTED
*   MENU IS NOT THE CURRENT MENU, DRMENU CHECKS TO SEE IF
*   THE REQUESTED MENU IS IN IMAGE MEMORY.  IF THIS IS THE
*   CASE, SWMENU (SWAP MENU) IS CALLED.  IF THE REQUESTED
*   MENU IS NOT THE CURRENT MENU AND IT IS NOT IN IMAGE
*   MEMORY, DRMENU WILL DRAW THE MENU.
*
*   DRMENU LOOPS THROUGH EACH MENU ELEMENT CONTAINED IN
*   THE MENU CONTROL TABLE (MCT) FOR THIS MENU.  IF THE
*   ELEMENT HAS NOT BEEN DISPLAYED BEFORE, THE DEFAULT
*   STATUS IS MOVED INTO THE NEXT STATUS FIELD.  IF THIS
*   MENU HAS BEEN DISPLAYED BEFORE, THEN SIMPLY CLEAR THE
*   CURRENT STATUS FIELD.
*
*   THE SUBSEQUENT MENU HANDLER ROUTINES KNOW WHAT FUNCTIONS
*   THEY MUST PERFORM BY THE CURRENT AND NEXT STATUS FIELDS.
*   EACH BIT IN THESE BYTES CORRESPONDS TO PART OF THE MENU
*   ELEMENT.  IF THE CORRESPONDING BITS IN THE CURRENT
*   STATUS EQUAL THE BITS IN THE NEXT STATUS, THEN THE
*   STATUS OF THIS PORTION OF THE ELEMENT IS UNCHANGED,
*   AND THE MENU ROUTINES RESPONSIBLE WILL DO NOTHING.
*   IF THE CORRESPONDING BITS ARE DIFFERENT,
*   THEN THE ASSOCIATED MENU ROUTINE WILL MAKE
*   THE ELEMENT CHANGES AND WILL UPDATE THE CURRENT STATUS.
*   IF A BIT IS ON (ONE) THEN IT SIGNALS THAT THE FEATURE IT
*   CORRESPONDS TO IS ON.  IF A BIT IS OFF (ZERO) THEN THIS
*   SIGNALS THAT THE FEATURE IS OFF.  WHEN THE BITS IN
*   THE TWO STATUS FIELDS DIFFER, THE NEXT STATUS BITS
*   DETERMINE WHAT WILL BE DONE.  THE CURRENT STATUS BITS
*   INDICATE HOW THE ELEMENT APPEARS CURRENTLY.
*
*   THE FOLLOWING CHART SHOWS THE FEATURES ASSOCIATED WITH
*   EACH BIT POSITION.  THESE BITS VALUES ARE IN THE
*   CURRENT, NEXT AND DEFAULT STATUS FIELDS OF THE MCT FILE.
*
*       BIT    HEX VALUE     ASSOCIATED FEATURE
*       ===    =========     ==================================
*         1        >80        ELEMENT BORDER ON/OFF
*         2        >40        ELEMENT BODY ON/OFF
*         3        >20        ELEMENT INSENSITIVITY ON/OFF
*         4        >10        ELEMENT INHIBIT ON/OFF
*         5        >08        ELEMENT TEXT ON/OFF
*         6        >04        RESERVED FOR FUTURE USE
*         7        >02        RESERVED FOR FUTURE USE
*         8        >01        NOT DRAWN ON/OFF (ONLY IN NEXT)
*
*   BIT 1--IF THIS BIT IS ON, THEN ELEMENT BORDER IS WHITE.
*      IF IT IS OFF, THE BORDER IS BLACK. IF BIT 1 IS OFF
```

```
*       IN THE CURRENT STATUS AND ON IN THE NEXT STATUS,
*       ROUTINE CBORD WILL TURN THE BORDER ON (TO WHITE).
*       IF IT IS ON IN THE CURRENT STATUS AND OFF IN THE
*       NEXT STATUS, THEN CBORD WILL TURN THE BORDER OFF.
*
*    BIT 2--IF THIS BIT IS ON, THEN THE ELEMENT BODY IS ON.
*       THE ON COLOR IS DICTATED EITHER BY MCT FIELD MCCON
*       OR BY THE ENTITY STATUS TABLE (EXPLAINED IN PROCEDURE
*       ESTADD). IF BIT 2 IS OFF, THEN THE ELEMENT BODY IS
*       OFF.  THE OFF COLOR IS ALWAYS INDICATED BY MCT FIELD
*       MCCOFF.  THE ROUTINE CBODY HANDLES THE BODY.
*
*    BIT 3--IF BIT 3 IS ON, THEN THE ELEMENT IS INSENSITIVE
*       TO THE TOUCH.  IF BIT 3 IS OFF, THEN IT IS SENSITIVE
*       AND IS THEREFORE A VALID SENTENCE ELEMENT.
*
*    BIT 4--IF BIT 4 IS ON, THEN THE DRAWING OF THE ELEMENT
*       IS INHIBITED, IT WILL NOT APPEAR ON THE MENU.  IF
*       IT IS OFF, THEN THE ELEMENT WILL BE DISPLAYED AS
*       PART OF THE MENU.  ROUTINE ZAPOUT HAS THE CAPABILITY
*       OF ERASING AN ALREADY DISPLAYED ELEMENT.  THE
*       DREL ROUTINE WILL DRAW AN ELEMENT THAT WAS PREVIOUSLY
*       INHIBITED.
*
*    BIT 5--IF THIS BIT IS ON, THEN THE ELEMENT'S TEXT FIELD
*       (IF ANY) WILL BE DRAWN INSIDE THE BODY OF THE
*       ELEMENT.  IF BIT 5 IS OFF, THEN THE TEXT WILL NOT
*       BE DISPLAYED.  ROUTINE CTEXT HANDLES THE ELEMENT
*       TEXT.
*
*    BIT 8--THIS BIT IS ONLY USED IN THE NEXT STATUS FIELD AS
*       AN INDICATOR THAT THIS ELEMENT HAS NEVER BEEN
*       DISPLAYED BEFORE.  THROUGH THIS BIT, ROUTINE DRMENU
*       KNOWS WHETHER TO USE THE DEFAULT STATUS BYTE OR
*       THE NEXT STATUS BYTE TO DRAW THE ELEMENT.
*
*
*    INPUT: RQMENU CONTAINS THE REQUESTED MENU ID.
*    OUTPUT: NONE
*    RESULT: MENU DISPLAYED ON SCREEN
*    TO CALL: BLWP @DRMENU
*
*================================================================
```

```
*=============================================================
*PROC = GMADDR
*
*    PROCEDURE NAME: GMADDR
*
*    GMADDR GETS THE MENU MDT (MENU DICTIONARY TABLE) ADDRESS
*    FROM A MENU ID.
*
*    INPUT: R0 CONTAINS MENU ID NUMBER
*    OUTPUT: MENUAD CONTAINS MDT ADDRESS, OR 0 IF NOT FOUND.
*    RESULT: NONE
*    TO CALL: BLWP @GMADDR    (EXTERNALLY)
*             BL   @GMA000    (WITHIN MHRTNS)
*
*=============================================================
```

```
*=============================================================
*PROC = MNEXT
*
*   PROCEDURE NAME: MNEXT
*
*   MNEXT DERIVES THE NEXT MENU NUMBER FROM FIELD MDNMM
*   (NEXT MENU MODIFIED) OR FIELD MDNM (NEXT MENU).  WHEN
*   THE MENU TABLES ARE ORIGINALLY CODED, EACH MENU IS
*   IN A CHAIN WITH A PRECEDING MENU AND A FOLLOWING MENU.
*   THE LOGICAL FOLLOWING (NEXT) MENU ID IS STORED IN MDT
*   FIELD MDNM.  THIS FIELD IS NOT CHANGED BY THE MENU
*   ROUTINES, AND AS LONG AS THE FIELD MDNMM IS ZERO, THEN
*   THE MENU MDNM WILL ALWAYS FOLLOW THIS MENU.  HOWEVER,
*   ON SOME OCCASIONS, THE MENU SOFTWARE WILL ALTER THE
*   NORMAL MENU FLOW.  WHEN THIS HAPPENS, A VALUE IS STORED
*   IN THE NEXT MENU MODIFIED (MDNMM) FIELD OF THE MDT. WHEN
*   MNEXT SEES A NON-ZERO MDNMM FIELD, THIS IS USED AS THE
*   NEXT MENU INSTEAD OF THE MDNM FIELD.  THUS ALTERING THE
*   MENU FLOW FROM ITS ORIGINAL STATE.
*
*   INPUT: NONE (CURRMA CONTAINS CURRENT MENU ADDRESS)
*   OUTPUT: PDMENU CONTAINS NEXT MENU PENDING VALUE
*   RESULT: NONE
*   TO CALL: BLWP @MNEXT
*
*=============================================================
```

```
*=============================================================
*PROC = ESTADD
*
*   PROCEDURE NAME: ESTADD
*
*   ESTADD DERIVES THE EST ENTRY ADDRESS FOR A GIVEN MENU
*   ELEMENT.
*
*   ALL MENU ELEMENTS WHICH CORRESPOND TO EQUIPMENT OR SOME
*   FUNCTION HAVE EST NUMBERS. THE ENTITY STATUS TABLE KEEPS
*   TRACK OF ELEMENTS WHOSE FUNCTION MAY BE GLOBAL OR WHOSE
*   STATUS MUST BE MAINTAINED. THE FOLLOWING TABLE DISCUSSES
*   SOME OF THE ASPECTS OF THE ENTITY STATUS TABLE.
*
*   FIELD-EXPLANATION
*   ====== ===================================================
*    ESID-THE ID NUMBER OF THE EST ENTRY
*
*  ESTYPE-EST ENTRIES CAN BE OF SEVERAL TYPES.
*         TYPE 0 = AN ELEMENT CONTROLLED BY OTHERS
*         TYPE 1 = A CONTROLLING ELEMENT
*         TYPE 2 = NEITHER CONTROLLED OR CONTROLLING
*         TYPE 3 = FRONT CAMERA PRESETS
*         TYPE 4 = LEFT CAMERA PRESETS
*         TYPE 5 = RIGHT CAMERA PRESETS
*
*    ESCB-IF ESTYPE NOT = 1 OR 2, THEN THE CONTROLLED BY
*         FIELD WILL EQUAL TO THE EST ID OF THE ELEMENT
*         WHICH IS CONTROLLING THIS ELEMENT.  WHEN THE
*         ELEMENT IS OFF OR NOT CONTROLLED, THE FIELD IS 0.
*
*   ESAME-ASSOCIATED MENU ELEMENT, NOT FUNCTIONING.
*
*  ESCOLR-FOR TYPE 1 ELEMENTS, ESCOLR IS THE UNIQUE COLOR
*         ASSOCIATED WITH A CONTROLLING ELEMENT.  WHEN IT
*         CONTROLS OTHERS, THEIR ESCOLR FIELD TAKES ON THE
*         VALUE OF THE CONTROLLING ESCOLR VALUE.
*
*   ESFON-ON FLAG.  WILL BE ONE (1) WHEN THE ELEMENT IS ON,
*         AND ZERO WHEN THE ELEMENT IS OFF.  THIS FIELD
*         OVERRIDES THE BODY BIT FLAG IN THE NEXT STATUS
*         FIELD.  IF NEXT STATUS SHOWS OFF, BUT ESFON IS ON,
*         THEN THE BODY BIT IN NEXT STATUS IS TURNED ON.
*
*  ESFPRE-PRESENT FLAG.  WHEN THIS FLAG IS ON (1) THEN THE
*         ELEMENT IS DRAWN ON THE MENU.  WHEN IT IS OFF,
*         DRAWING OF THE ELEMENT IS INHIBITED.  LIKE THE
*         ESFON FIELD, ESFPRE OVERRIDES THE INHIBIT BIT
*         FLAG IN THE NEXT STATUS FIELD.
*
*  ESDATA-DATA ASSOCIATED WITH THE ENTRY.  THIS FIELD IS
*         GENERALLY NOT USED BUT CAN BE VALUABLE ON AN
*         INDIVIDUAL BASIS.  ALL CAMERA PRESETS HAVE AN
*         ESDATA VALUE CORRESPONDING TO THEIR PRESET NUMBER.
*         AGAIN, THIS FIELD MAY HAVE VARIABLE INFORMATION
```

```
*          AND IS SPECIFIC TO AN ELEMENT.
*
*  ESCONF-CONFLICT FLAG.  ALL CONTROLLING ELEMENTS HAVE A
*          UNIQUE POWER OF 2 NUMBER ASSOCIATED WITH IT.
*          ELEMENTS WHICH MAY BE ASSOCIATED WITH A CERTAIN
*          CONTROLLING ELEMENT WILL HAVE THE BIT TURNED ON
*          IN ITS ESCONF FIELD ASSOCIATED WITH THE ELEMENTS
*          WHICH IT CAN BE CONTROLLED BY.
*
*    INPUT: R2 CONTAINS ELEMENT MCT ADDRESS
*    OUTPUT: ESADDR CONTAINS EST ADDRESS
*    RESULT: NONE
*    TO CALL: BLWP @ESTADD     (EXTERNALLY)
*          BL   @ESA000     (FROM WITHIN MHRTNS)
*
*================================================================
```

```
*=============================================================
*PROC = TOUCH
*
*    PROCEDURE NAME: TOUCH
*
*    TOUCH IS THE ROUTINE THAT FLIPS THE BORDER BIT OF THE
*    MCT ELEMENT TOUCHED.  IF THE ELEMENT'S BORDER WAS
*    ALREADY ON, THEN TOUCH TURNS IT OFF, AND VICE VERSA.
*
*    INPUT: MEADDR CONTAINS THE MCT ELEMENT ADDRESS
*    OUTPUT: NONE
*    RESULT: BORDER TURNS ON/OFF
*    TO CALL:  BLWP @TOUCH
*
*=============================================================
```

```
*=================================================
*PROC = LHERE
*
*   PROCEDURE NAME: LHERE
*
*   LHERE CREATES A LIGHT $H0,0; COMMAND AND THEN
*   PASSES IT TO LIGHT TO GIVE MENU ELEMENTS THEIR
*   OFFSETS.
*
*   INPUT: R0 CONTAINS X COORDINATE
*          R1 CONTAINS Y COORDINATE
*   RESULT: LIGHT HERE COMMAND IS ISSUED
*   TO CALL: BLWP @LHERE
*
*=================================================
*=================================================
```

A-106

```
*============================================================
*PROC = RDMENU
*
*    PROCEDURE NAME: RDMENU
*
*    RDMENU IS THE ROUTINE WHICH WILL RE-DRAW THE CURRENT
*    MENU.  REDRAWING IS SOMETIMES NECCESSARY TO MAKE MANY
*    CHANGES TO THE MENU WHERE INDIVIDUAL CALLS TO THE OTHER
*    MENU ROUTINES WOULD BE INEFFICIENT.  RDMENU LOOPS
*    THROUGH THE MCT ENTRIES CALLING EITHER DRE000 OR DREL2
*    FOR EACH ENTRY.  IF SOMETHING HAS CHANGED SINCE THE
*    LAST TIME, THE CHANGE WILL BE REFLECTED IN/ON THE MENU.
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: MENU IS REDRAWN
*    TO CALL: BLWP @RDMENU     (EXTERNALLY)
*             BL   @RDM000     (FROM WITHN MHRTNS)
*
*============================================================
```

```
*=============================================================
*PROC = DREL AND DREL2
*
*   PROCEDURE NAME: DREL
*
*   DREL IS THE ROUTINE WHICH DRAWS THE INDIVIDUAL MENU
*   ELEMENTS.  I IT FIRST RETRIEVES THE EST ENTRY (IF ANY)
*   AND SETS THE BODY AND INHIBIT NEXT STATUS FLAGS TO
*   CORRESPOND TO THE ESFUN AND ESFPRE FLAGS IN THE EST.
*   NEXT, IF THE INHIBIT FLAG IS ON, JUST RETURN AFTER
*   SETTING THE CURRENT STATUS TO THE NEXT STATUS.
*   NOW, GET THE DEL ADDRESS FOR THE ELEMENT AND PASS THE
*   LIGHT STRING TO LIGHT FOR DRAWING.  NOW, SEE IF THE
*   CURRENT STATUS EQUALS THE NEXT STATUS.  IF IT DOES,
*   THEN THIS ELEMENT HAS NOT CHANGED AND NO FURTHER
*   WORK IS REQUIRED, JUST RETURN.  IF THE CURRENT AND NEXT
*   FIELDS ARE NOT EQUAL, THEN CHECK IN TURN THE BORDER
*   BIT, BODY BIT, INHIBIT BIT AND THE TEXT BIT FOR
*   EQUALITY.  WHEN A DIFFERENCE IN FOUND, CALL THE
*   APPROPRIATE ROUTINE TO MAKE THE CHANGE.
*
*   INPUT: R2 CONTAINS THE MCT ADDRESS OF THE ELEMENT
*   OUTPUT: CURRENT STATUS FIELD IS SET
*   RESULT: THE ELEMENT IS DRAWN
*   TO CALL: BLWP @DREL     (EXTERNALLY)
*            BL   @DRE000   (FROM WITHIN MHRTNS)
*
*
*   PROCEDURE NAME: DREL2
*
*   DREL2 IS CALLED WHEN THE ELEMENT IS TO BE UPDATED ONLY
*   IF THE STATUS BYTES HAVE CHANGED.  WHEN DREL IS CALLED,
*   IT ALWAYS DRAWS THE ELEMENT, DREL2 ONLY MAKES CHANGES TO
*   THE ELEMENT.
*
*   INPUT: R2 CONTAINS MCT ADDRESS
*   OUTPUT: NONE
*   RESULT: CHANGED ELEMENTS ARE UPDATED
*   TO CALL: BL @DREL2
*
*=============================================================
```

```
*=============================================================
*PROC = DELEM
*
*    PROCEDURE NAME: DELEM
*
*=============================================================
```

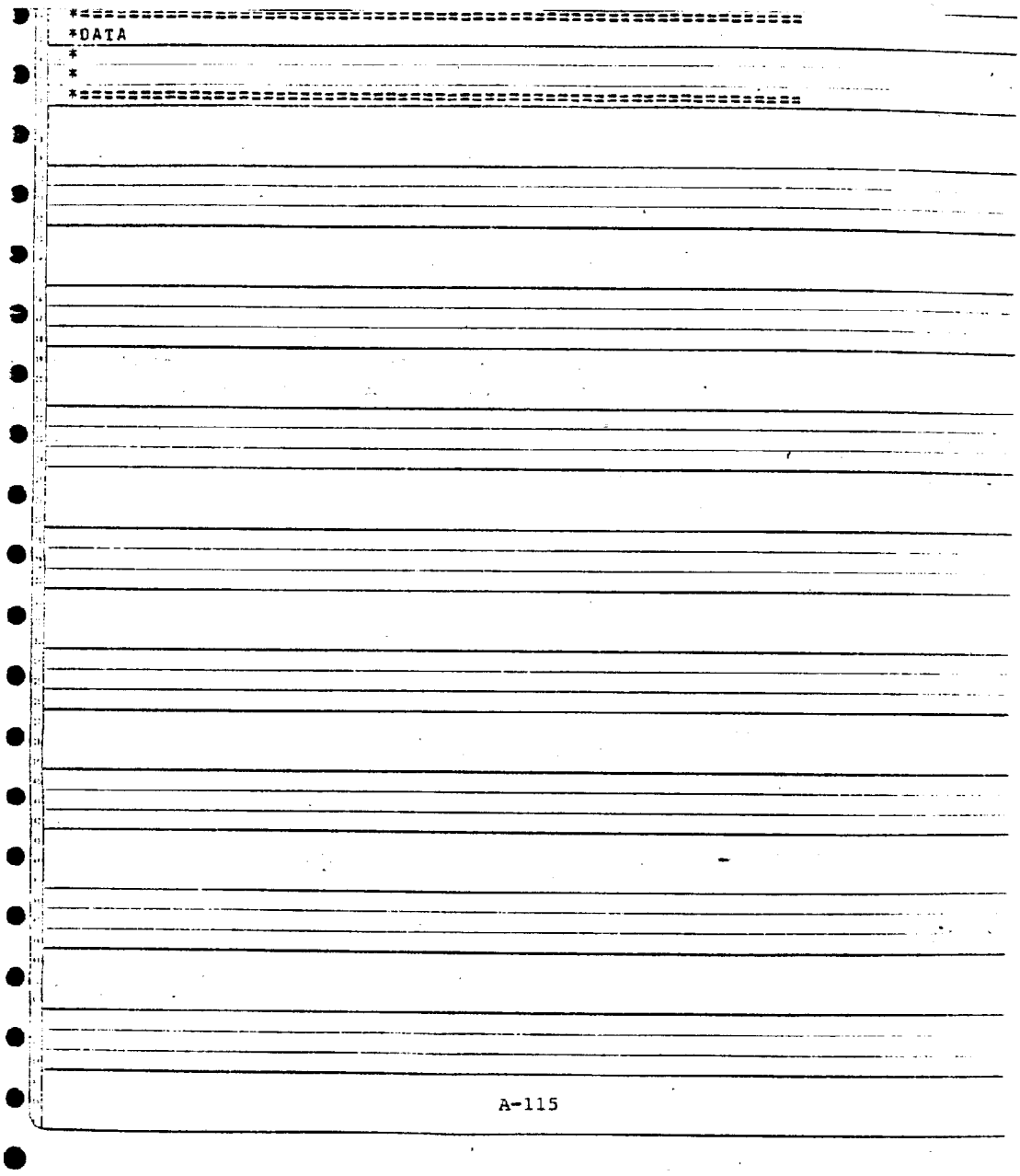
```
*=============================================================
*PROC = CBORD
*
*   PROCEDURE NAME: CBORD
*
*   CBORD IS THE ROUTINE WHICH EITHER TURNS THE BORDER ON
*   OR OFF.  IT FIRST CHECKS FOR A BORDER ON THE ELEMENT.
*   IF IT IS FOUND, UPDATE THE ICB FIELD WITH THE X AND Y
*   HERE (DEXFIL, DEYFIL) VALUES.  THEN, BASED ON THE BORDER
*   BIT IN THE NEXT STATUS, IT LOADS THE ADDRESS OF THE
*   LIGHT STRING TO TURN THE BORDER ON/OFF.  LIGHT IS NOW
*   CALLED TO IMPLEMENT THE COMMAND.
*
*   INPUT: R2 CONTAINS THE MCT ELEMENT ADDRESS
*   OUTPUT: NONE
*   RESULT: BORDER GOES WHITE OR BLACK
*   TO CALL: BL @CBORD
*
*=============================================================
```

```
*=============================================================
*PROC = CBODY
*
*   PROCEDURE NAME: CBODY
*
*   CBODY IS THE MENU ROUTINE WHICH CHANGES THE BODY
*   OF THE MENU ELEMENT.  WHEN THE BODY IS OFF, IT APPEARS
*   IN THE COLOR SPECIFIED IN THE MCCOFF FIELD OF THE MCT;
*   WHEN IT IS ON, THE COLOR CAN BE SPECIFIED EITHER IN THE
*   MCCON MCT FIELD, OR IF THAT FIELD IS BLACK (0), THEN
*   THE ENTITY ENTRY COLOR (ESCOLR) IS USED.  IF THIS FIELD
*   IS ZERO, THEN LOOK UP THE ESCOLR OF THE CONTROLLING
*   ELEMENT (IF ANY).  THIS WILL BE THE COLOR USED FOR
*   TURNING AN ELEMENT ON.  ONCE THE COLOR IS ASCERTAINED,
*   SIMPLY MOVE THE X AND Y HERE VALUES TO THE ICB.  THEN,
*   ADD THE DEL X AND Y FILL VALUES TO IT.  NOW LOAD THE
*   BODY CHANGE LIGHT COMMAND INTO R1 AND CALL LIGHT TO DO
*   THE WORK.
*
*   IF THERE IS TEXT INSIDE THE ELEMENT WHICH IS CURRENTLY
*   ON, THEN BEFORE THE ELEMENT BODY IS CHANGED, ERASE THE
*   TEXT.  THEN AFTER THE BODY IS CHANGED, ISSUE A LIGHT
*   DELAY COMMAND, AND THEN RE-DRAW THE TEXT.  THIS IS DONE
*   SO THAT THE INSIDE OF LETTERS SUCH AS THE "O" AND "A"
*   ARE FILLED IN WITH THE NEW COLOR.  THE DELAY IS TO GIVE
*   LIGHT TIME TO DO THE FILL BEFORE THE NEW TEXT IS DRAWN.
*
*   INPUT: R2 CONTAINS THE MCT ELEMENT ADDRESS
*   OUTPUT: NONE
*   RESULT: BODY GOES ON OR OFF
*   TO CALL: BLWP @CBODY     (EXTERNALLY)
*            BL   @CBD000    (FROM WITHIN MHRTNS)
*
*=============================================================
```

```
*===============================================================
*PROC = CTEXT
*
*   PROCEDURE NAME: CTEXT
*
*   CTEXT IS THE ROUTINE WHICH CHANGES THE TEXT IN THE MENU
*   ELEMENTS.  BASED UPON THE TEXT BIT IN THE NEXT STATUS
*   FIELD, CTEXT EITHER ERASES THE EXISTING TEXT OR DRAWS
*   THE TEXT.  TO ERASE TEXT, SIMPLY WRITE THE TEXT AGAIN
*   IN THE BACKGROUND COLOR (MCCCUR).  THIS CAUSES THE TEXT
*   TO DISAPPEAR.  TO WRITE TEXT, DISPLAY IT IN THE COLOR
*   DESIGNATED FOR TEXT (MCCTXT) FOR THIS ELEMENT.  TEXT
*   WHICH OCCURS INSIDE OF MENU ELEMENTS IS ALWAYS CENTERED
*   HORIZONTALLY WITHIN THE ELEMENT.  VERTICAL SPACING IS
*   LEFT UP TO THE USER.
*
*   INPUT: R2 CONTAINS MCT ELEMENT ADDRESS
*   OUTPUT: NONE
*   RESULT: TEXT IS EITHER ERASED OR DISPLAYED
*   TO CALL: BLWP @CTEXT    (EXTERNALLY)
*            BL   @CTE000   (FROM WITHIN MHRTNS)
*
*===============================================================
```

```
*===============================================================
*PROC = ZAPOUT
*
*   PROCEDURE NAME: ZAPOUT
*
*   ZAPOUT IS THE ROUTINE WHICH IS CAPABLE OF ERASING AN
*   ELEMENT WHICH IS CURRENTLY DISPLAYED ON THE MENU.  THIS
*   IS ACCOMPLISHED BY DOING A RECTANGULAR ZAP IN THE
*   BACKGROUND COLOR OVER THE ELEMENT TO BE ZAPPED.
*   THEREFORE, ELEMENTS WHICH CANNOT BE COMPLETLY COVERED
*   BY A RECTANGULAR AREA WHICH COVERS ONLY ONE BACKGROUND
*   COLOR, CANNOT BE ZAPPED.  THE DIMENSIONS OF THE
*   RECTANGLE USED FOR ZAPPING ARE THE X AND Y TOLERANCE
*   FIELDS IN THE DEL ENTRY.
*
*   INPUT: R2 CONTAINS THE MCT ELEMENT ADDRESS
*   OUTPUT: NONE
*   RESULT: THE MENU ELEMENT IS ERASED
*   TO CALL: BLWP @ZAPOUT     (EXTERNALLY)
*            BL   @ZAP000     (FROM WITHIN MHRINS)
*
*===============================================================
```

```
*================================================================
*MOD = MHSUPRTN
*
*    THIS MODULE CONTAINS SUPPORT ROUTINES WHICH ARE USED
*    BY OTHER PROCEDURES WITHIN THE MENU HANDLER SYSTEM.
*    ROUTINES WILL BE "PERMANENTLY" MAPPED IN WHENEVER THE
*    MENU HANDLER SOFTWARE IS IN.
*
*================================================================
```



A-115

```
*PROC = SLCTME
*
*   PROCEDURE NAME: SLCTME
*
*   SLCTME IS THE SUPPORT ROUTINE WHICH IMPLEMENTS
*   A GENERAL KEY ON ANY MENU.  SLCTME RECOGNIZES THE TOUCH
*   AND CLEARS THE IMMEDIATE AND DELAYED ACTION FLAGS.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: KEY BORDER REVERSES
*   TO CALL: BL @SLCTME
*
```

A-116

```
*============================================================
*PROC = PREVUE
*
*    PROCEDURE NAME: PREVUE
*
*    THIS PROCEDURE IS CALLED WHENEVER A NEW TOUCH HAS
*    BEEN ADDED TO THE STACK IF THE PREVIEW FLAG HAS
*    BEEN SET.  PREVUE WILL LOOK THROUGH THE STACK FOR
*    AN ELEMENT WHICH IS EITHER A SOURCE OR IS CURRENTLY
*    CONTROLLED BY A SOURCE.
*
*    IF SUCH AN ELEMENT IS FOUND, IT IS PREVIEWED.
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: KEY BORDER REVERSES
*    TO CALL: BL @PREVUE
*
*============================================================
```

```
*=============================================================
*PROC = STDFLT
*
*   PROCEDURE NAME: STDFLT
*
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: KEY BORDER REVERSES
*   TO CALL: BL @STDFLT
*
*=============================================================
```

A-118

```
*=============================================================
*PROC = FINDIT
*
*   PROCEDURE NAME: FINDIT
*
*   FINDIT IS A UTILITY PROCEDURE FOR STDFLT.  IT SEARCHES THROUGH
*   THE CURRENT MENU'S MCT ENTRIES TO FIND THE ELEMENT WHICH HAS
*   THE ENTITY ID CONTAINED IN R1.  IF R1 = 0 THEN SEARCH FOR
*   THE GO ELEMENT.  THE MCT ADDRESS IS STORED IN R2.
*
*   INPUT: R1 CONTAINS EST ID
*   OUTPUT: R2 CONTAINS MCT ELEMENT ADDRESS
*   RESULT: NONE
*   TO CALL: BL @FINDIT
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = EXECIT
*
*   PROCEDURE NAME: EXECIT
*
*   EXECIT IS A UTILITY PROCEDURE FOR STDFLT.  ITS PURPOSE IS
*   TO PERFORM EITHER THE IMMEDIATE OR DELAYED ACTION ROUTINE
*   ASSOCIATED WHICH EACH ELEMENT STACKED.  THIS SUB-PROCEDURE
*   IS FUNCTIONALLY EQUIVALENT TO SECTIONS WITHIN CCTSPI.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: ACTION ROUTINE EXECUTED
*   TO CALL: BL @EXECIT
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = BACK
*
*    PROCEDURE NAME: BACK
*
*    BACK IS THE SUPPORT ROUTINE WHICH IMPLEMENTS
*    THE BACK KEY ON THE MENUS.  BACK ERASES ALL CURRENT
*    TOUCHES, DELETES THE STACK, GETS THE PREVIOUS MENU
*    NUMBER FROM MDPM, AND FINALLY DRAWS THE NEW MENU.
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: ERASES ALL TOUCHES, DRAWS NEW MENU
*    TO CALL: BL @BACK
*
*=============================================================
```

```
*PROC = HELP
*
*   PROCEDURE NAME: HELP
*
*   HELP IS THE SUPPORT ROUTINE ASSOCIATED WITH
*   ALL HELP KEYS.  IF THE HELP KEY IS TOUCHED AND THERE IS
*   AN ELEMENT ON THE SENTENCE STACK, HELP WILL GIVE HELP
*   ON THE TOUCHED ELEMENT IF THIS ELEMENT HAS A NON-ZERO
*   MCHELP FIELD IN ITS MCT ENTRY.  IF THERE ARE NO ELEMENTS
*   ON THE SENTENCE, HELP WILL GIVE GENERAL HELP ON THE
*   MENU IF THE MDHELP FIELD IN THE MDT ENTRY IS NON-ZERO.
*   IF HELP CANNOT GIVE HELP, THEN A NO HELP AVAILABLE
*   MESSAGE IS DISPLAYED.
*
*   THE HELP KEY HAS AN ADDITIONAL FUNCTION OF DISPLAYING
*   PENDING MESSAGES (INDICATED BY A RED GO KEY).
*   GENERALLY, WHEN A MESSAGE IS GENERATED, IT IS NOT
*   DISPLAYED IMMEDIATELY, INSTEAD, A FLAG IS RAISED TO TELL
*   THE USER THAT AN ERROR IS PENDING.  TO RECEIVE PENDING
*   ERRORS, THE USER HAS ONLY TO TOUCH THE HELP KEY.
*
*   THE HELP KEY DOES NOT ERASE THE PENDING ERROR AND IF
*   HELP IS RE-TOUCHED THE SAME MESSAGE WILL BE RE-DISPLAYED
*   UNLESS OTHER HELP IS AVAILABLE IN THE FORM OF MDHELP
*   OR MCHELP.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: HELP MENU DISPLAYED, ERROR GENERATED OR ERROR
*           DISPLAYED.
*   TO CALL: BL @HELP
*
*****************************************************************
```

```
*=============================================================
*PROC = RESET
*
*   PROCEDURE NAME: RESET
*
*   RESET IS THE SUPPORT ROUTINE WHICH IMPLEMENTS
*   THE X KEY FUNCTIONS ON MANY OF THE MENUS.
*   IT PERFORMS THE FOLLOWING FUNCTIONS:
*
*       *  RE-TOUCHES ALL TOUCHED ELEMENTS
*       *  CLEARS AND RELEASES THE STACK
*       *  ERASES PENDING OR DISPLAYED ERRORS
*       *  ERASES CAMERA PRESETS, IF NECESSARY
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: MENU RETURNS TO PRE-SENTENCE STATE.
*   TO CALL: BL @RESET
*
*=============================================================
```

A-123

```
*=============================================================
*PROC = ZAPCAM
*
*   PROCEDURE NAME: ZAPCAM
*
*   ZAPCAM WILL ERASE ALL CAMERA PRESETS, TAKE THE CAMERA
*   OFF OF THE STACK, AND TURN THE ON CAMERA AND ON PRESET
*   OFF IS R0 IS SET TO 1.
*
*   INPUT: R0, WHEN =0, JUST ERASE TOUCHES AND PRESETS,
*          WHEN =1, TURN OFF AS WELL AS ERASE.
*          CAMTCH CONTAINS EITHER THE MCT ADDRESS OF THE
*          CAMERA OR THE ENTITY ID OF SAME.
*   OUTPUT: NONE
*   RESULT: CAMERA AND PRESET TOUCHES ARE ERASED.
*   TO CALL: ELWP @ZAPCAM
*
*=============================================================
```

```
*=============================================================
*PROC = LSENDS
*
*   PROCEDURE NAME: LSENDS
*
*   THIS ROUTINE IS CALLED WHEN A CHANGED HAS BEEN MADE
*   IN THE CONNECTIONS OF SINKS AND SOURCES.  IT CHECKS
*   THE ENTITY STATUS TABLE FOR ANY SOURCES WITH NO
*   ACTIVE SINKS AND DEACTIVATES THE SOURCE.  IT ALSO
*   CHECKS FOR DEACTIVATED SOURCES THAT HAVE SINKS
*   STILL ACTIVE AND GENERATES SYSTEM REQUESTS TO
*   DEACTIVATE THE SINKS.
*
*   INPUT:  ENTITY STATUS TABLE
*   OUTPUT: MODIFIED ENTITY STATUS TABLE
*           SYSTEM REQUESTS
*   RESULT: LOOSE ENDS OF DEVICE CONNECTIONS CLEANED UP
*   TO CALL:  BL @LSENDS
*
*=============================================================
*=============================================================
```

A-125

```
*=============================================================
*PROC = REFSTK
*
*   PROCEDURE NAME: REFSTK
*
*   REFSTK WILL REMOVE AN ELEMENT FROM THE SENTENCE
*   STACK.  THE ELEMENT TO BE REMOVED MCT ADDRESS IS
*   STORED IN R1 BEFORE CALLING THIS ROUTINE.
*
*   INPUT: R1 CONTAINS MCT ADDRESS OR ELEMENT
*   OUTPUT: NONE
*   RESULT: ELEMENT IS REMOVED FROM STACK
*   TO CALL: BL @REFSTK
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = CHELEM
*
*   PROCEDURE NAME: CHELEM
*
*   THIS PROCEDURE ALLOWS THE USER TO CHANGE (RE-DRAW,
*   RE-COLOR ETC.) A SINGLE MENU ELEMENT.  IF MORE THAN
*   ONE ELEMENT IS TO BE CHANGED, THE BEST PROCEDURE
*   TO USE IS RDMENU.
*
*   INPUT: CHELM CONTAINS ELEMENT MCT ADDRESS
*   OUTPUT: NONE
*   RESULT: ELEMENT IS RE-DRAWN
*   TO CALL: BLWP @CHELEM
*
*=============================================================
*=============================================================
```

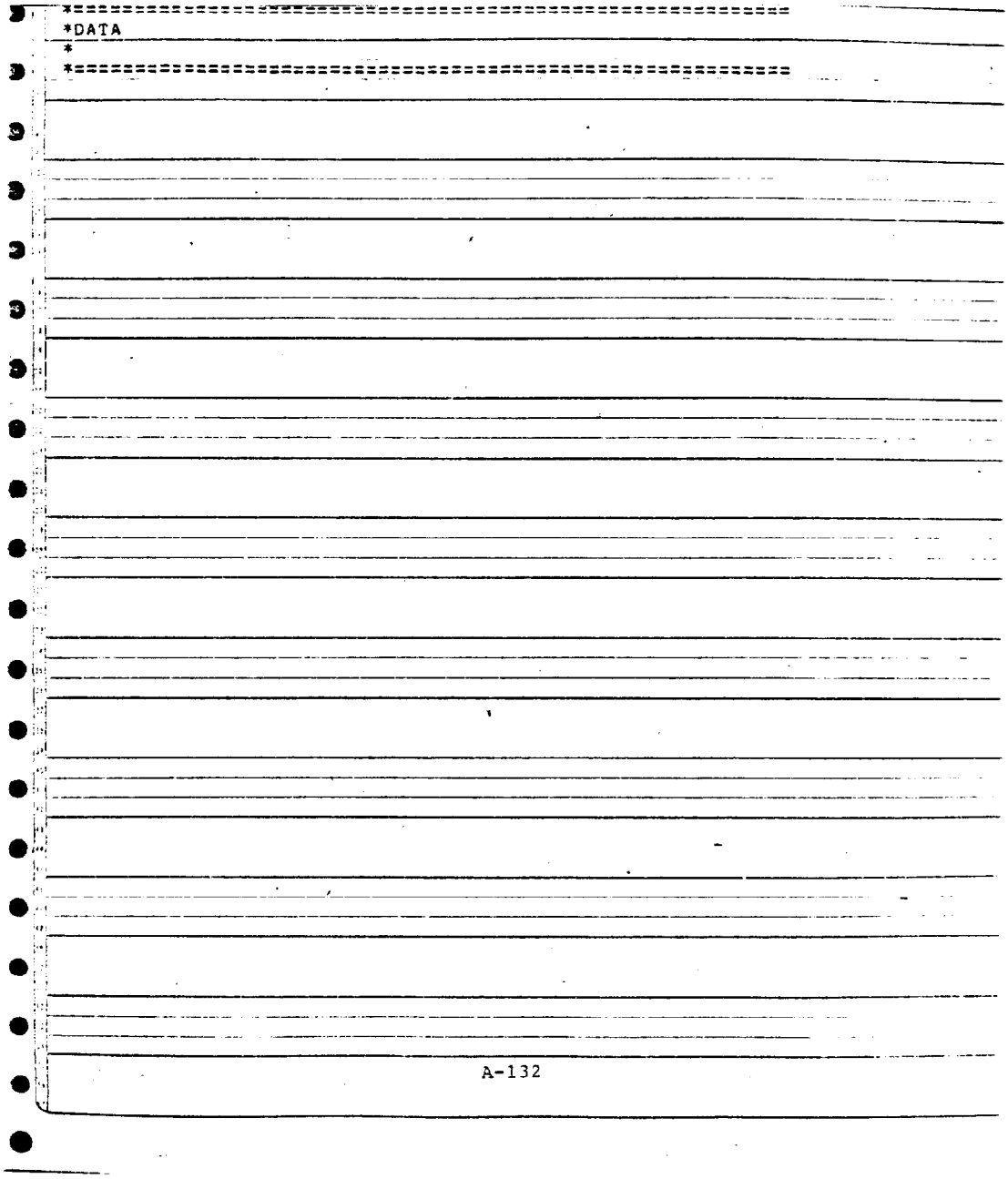
A-127

```
*=============================================================
*PROC = MHRHHD
*
*    PROCEDURE NAME: MHRHHD
*
*    THIS PROCEDURE WILL INTERPRET A MESSAGE FROM THE
*    HAND HELD DEVICE.  THE RECALL FROM STORAGE (NOT
*    AVAILABLE ON THE ALLSTATE TYPE SYSTEM) WILL EITHER
*    BE ADVANCED OR REVERSED BY THIS COMMAND.
*    THE MESSAGE TEXT IS AS FOLLOWS:
*
*         FC00 -- COMMAND CODE
*         0000 -- VIDEO/HI-RES CODE
*         0000 -- FORWARD/REVERSE CODE
*
*
*    INPUT: MESSAGE BUFFER IN R1
*    OUTPUT: NONE
*    RESULT: MENU IS UPDATED AND NEW IMAGE SHOWN
*    TO CALL: BLWP @MHRHHD
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = MHRSCP
*
*   PROCEDURE NAME: MHRSCP
*
*   THIS PROCEDURE WILL INTERPRET A MESSAGE FROM THE
*   SCP APT (NOT AVAILABLE ON ALLSTATE TYPE SYSTEMS).
*   THE MESSAGE FORMAT IS:
*
*     FB00 -- COMMAND CODE
*     0000 -- ENTITY ID OF OBJECT TO MODIFY
*     0000 -- ON/OFF FLAG
*     0000 -- ENTITY ID OF SOURCE (IF ANY)
*
*   INPUT: MESSAGE BUFFER IN R1
*   OUTPUT: NONE
*   RESULT: MENU IS UPDATED
*   TO CALL: BLWP @MHRSCP
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = SINIT
*
*   PROCEDURE NAME: SINIT
*
*   THIS PROCEDURE SENDS SYSTEM REQUEST MESSAGES TO
*   TURN OFF THE AUDIO AND ALL CONTROLLED BY ENTITIES.
*
*   INPUT: NONE
*   OUTPUT: SYSTEM MESSAGES
*   RESULT: ALL DEVICES ARE TURNED OFF
*   TO CALL: BL @SINIT
*
*=============================================================
*=============================================================
```

```
*=============================================================
*MOD = MHMAPRTN
*
*   THIS MODULE CONTAINS THE ROUTINES USED BY THE MENU
*   HANDLER SOFTWARE TO MAP MODULES/PROCEDURES IN TO
*   MEMORY.
*
*=============================================================
```


*DATA
*
A-132

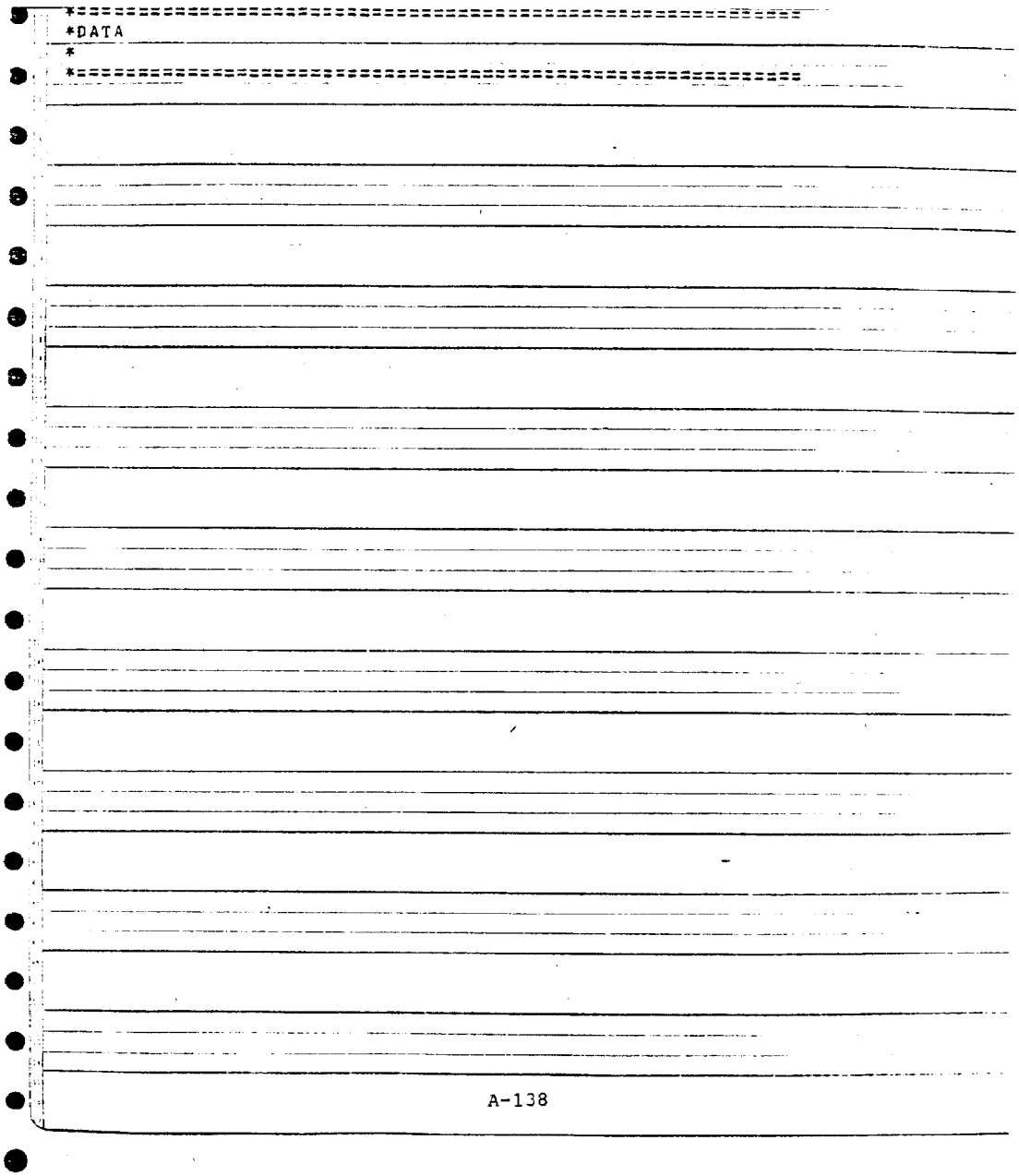
```
*=============================================================
*PROC = GDELAD
*
*   PROCEDURE NAME: GDELAD
*
*   GDELAD WILL DERIVE THE ADDRESS OF THE DISPLAY ELEMENT
*   LIBRARY (DEL) ENTRY FOR A GIVEN MENU ELEMENT.
*   THIS SECTION OF THE DEL WILL THEN BE MAPPED INTO
*   MEMORY.
*
*   INPUT: R2 CONTAINS THE ELEMENTS MCT ADDRESS
*   OUTPUT: DEADDR CONTAINS THE DEL ADDRESS
*   RESULT: NONE
*   TO CALL: BLWP @GDELAD   (EXTERNALLY)
*
*=============================================================
```

```
*============================================================
*PROC = GTELAD
*
*   PROCEDURE NAME: GTELAD
*
*   GTELAD WILL DERIVE THE ADDRESS OF THE TEXT ELEMENT
*   LIBRARY (TEL) ENTRY FOR A GIVEN MENU ELEMENT.
*   THIS SECTION OF THE TEL WILL THEN BE MAPPED INTO MEMORY.
*
*   INPUT: R3 CONTAINS TEXT ID NUMBER
*   OUTPUT: TLADDR CONTAINS THE TEL ADDRESS
*   RESULT: NONE
*   TO CALL: BLWP @GTELAD   (EXTERNALLY)
*
*============================================================
```

```
*=============================================================
*PROC = GMCTAD
*
*   PROCEDURE NAME: GMCTAD
*
*   GMCTAD WILL DERIVE THE ADDRESS OF THE MCT ENTRY
*   FOR A GIVEN MENU MDT ENTRY.
*
*   INPUT: R2 CONTAINS THE ELEMENTS MCT ADDRESS
*   OUTPUT: MCADDR CONTAINS THE MCT ADDRESS
*   RESULT: NONE
*   TO CALL: BLWP @GMCTAD   (EXTERNALLY)
*
*=============================================================
```

```
*=============================================================
*PROC = MHRMAP
*
*   MHRMAP IS THE PROCEDURE WHICH IMPLEMENTS MEMORY MAPPING
*   FOR THE MENU SOFTWARE.  IT WILL ALSO RESTORE THE LAST
*   MEMORY MAPPING IF REQUESTED.
*
*   INPUT: R0 CONTAINS THE MMS
*          R1 CONTAINS THE MMR
*
*          IF R0 IS LESS THAN 0, THEN THE LAST MEMORY
*          MAPPING WILL BE RESTORED.
*
*   OUTPUT: NONE
*   RESULT: MEMORY SEGMENTS MAPPED.
*   TO CALL: BLWP @MHRMAP     (EXTERNALLY)
*
*=============================================================
```

```
*=============================================================
*MOD = MHMSGS
*
*   MODULE MHMSGS MAINTAINS A LIBRARY OF ALL MESSAGES WHICH
*   ARE DISPLAYABLE BY THE TELE-CONFERENCING SYSTEM THROUGH
*   THE MENU DISPLAY SYSTEM.  IT ALSO MAINTAINS THE LIBRARY
*   OF MESSAGES WHICH ARE SENT TO OTHER ROUTINES TO
*   IMPLEMENT THE FUNCTIONS OF THE MENUS.  THIS MODULE
*   CONSISTS OF FIVE PROCEDURES. THEY ARE GRMST, ERRDSP,
*   ERRERS, GMSGAD AND MSGDSP.  THE FUNCTIONS OF EACH
*   PROCEDURE ARE DESCRIBED AT THE BEGINNING OF EACH PROC.
*
*=============================================================
```


*DATA
*
A-138

```
*==========================================================
*DATA
*
*==========================================================
```

```
*=============================================================
*DATA
*       REQUEST MESSAGE BLOCKS
*
*
*=============================================================
```

```
*=============================================================
*PROC = GRMST
*
*   PROCEDURE NAME: GRMST
*
*   GRMST SETS AND RESETS THE ERROR INDICATOR.  THE ERROR
*   INDICATOR IS INCORPORATED WITHIN THE GO KEY ELEMENT. IF
*   AN ERROR IS PENDING, THE "GO" CHARACTERS ARE DISPLAYED
*   IN RED.  IF THERE IS NOT AN ERROR PENDING, THEN THE
*   "GO" CHARACTERS ARE DISPLAYED IN GREEN.  WHEN AN ERROR
*   IS PENDING, TOUCHING THE "HELP" KEY WILL CAUSE THE ERROR
*   TO BE DISPLAYED ON THE LEFT SIDE OF THE CONTROL KEY
*   BOX.
*
*   INPUT: R0 = FLAG, FLAG = 0 FOR ERROR RESET,
*                     FLAG = 1 FOR ERROR SET.
*          R1 = MESSAGE NUMBER
*   OUTPUT: NONE
*   RESULT: GO ELEMENT TURNS RED (R0 = 1) OR
*           GREEN (R0 = 0).
*   TO CALL: BL @GRMST
*
*=============================================================
```

```
*=============================================================
*PROC = ERRDSP
*
*   PROCEDURE NAME: ERRDSP
*
*   ERRDSP CONVERTS A MESSAGE NUMBER (ERRMSG) INTO A MESSAGE
*   ADDRESS AND DISPLAYS THE MESSAGE ON THE MENU.
*   IT ALSO SETS ERRFLG TO INDICATE THAT AN ERROR IS
*   CURRENTLY BEING DISPLAYED.
*
*   INPUT: ERRMSG = MESSAGE NUMBER
*   OUTPUT: NONE
*   RESULT: MESSAGE IS DISPLAYED IN CONTROL KEY BOX
*   TO CALL: BL @ERRDSP
*
*=============================================================
```

```
*=============================================================
*PROC = ERRERS
*
*   PROCEDURE NAME: ERRERS
*
*   ERRERS WILL ERASE ANY MESSAGE DISPLAYED AND WILL RESET
*   THE MESSAGE INDICATORS AS WELL AS RESET THE GO KEY
*   ELEMENT TO ITS NORMAL STATUS.  ERRERS WILL ALSO RESTORE
*   THE CONTROL KEYS WHICH MAY HAVE BEEN ERASED BY THE
*   MESSAGE WHEN IT WAS DISPLAYED.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: MESSAGE IS ERASED, CHANGE GO KEY TO GREEN AND
*           RE-DRAW ANY CONTROL KEY WHICH WAS ERASED BY THE
*           DISPLAYED MESSAGE.
*   TO CALL: BL @ERRERS
*
*=============================================================
```

A-143

```
*=============================================================
*PROC = GMSGAD
*
*    PROCEDURE NAME: GMSGAD
*
*    GMSGAD RETURNS THE ADDRESS OF A REQUESTED RQB MESSAGE.
*
*    INPUT: R1 = RQB MESSAGE NUMBER
*    OUTPUT: R1 = ADDRESS OF RQB MESSAGE
*    RESULT: R1 IS REPLACED WITH ADDR OF RQB MSG
*    TO CALL: BL @GMSGAD
*
*=============================================================
```

A-144

```
*=============================================================
*PROC = MSGDSP
*
*   PROCEDURE NAME: MSGDSP
*
*   MSGDSP WILL EITHER CAUSE A MESSAGE TO BE DISPLAYED ON
*   THE MENU OR WILL INDICATE THAT AN ERROR IS PENDING.
*   THIS WILL BE THE GENERAL ROUTINE CALLED BY NON-MENU
*   HANDLING CODE.
*
*   INPUT: R0 = FLAG, WHEN 0 = DISPLAY MESSAGE NOW
*                     WHEN NOT 0 = INDICATE MESSAGE PENDING
*                     AND LET THE USER ASK FOR IT.
*          R1 = MESSAGE NUMBER
*   OUTPUT: NONE
*   RESULT: BASED ON R0, MESSAGE IS DISPLAYED OR INDICATED.
*   TO CALL: BLWP @MSGDSP
*
*=============================================================
```

A-145

```
*=============================================================
*MOD = MHIATBL
*
*    MODULE MHIATBL IS THE INDEX TABLE FOR THE IMMEDIATE
*    ACTION ROUTINES.
*
*=============================================================
```

```
*=============================================================
*DATA
*
*    IMMEDIATE ACTION ROUTINE INDEX TABLE
*
*=============================================================
```

```
*=============================================================
*MOD = MHCATBL
*
*   MODULE MHCATBL IS THE INDEX TABLE FOR THE DELAYED
*   ACTION ROUTINES.
*
*=============================================================
```

```
*=================================================
*DATA
*
*   DELAYED ACTION ROUTINE INDEX TABLE
*
*=================================================
```

```
*================================================================
*MOD = MHCITBL
*
*   THIS MODULE CONTAINS THE CONFERENCE INITIALIZATION
*   TABLE.  IT IS USED TO SPECIFY CAMERA PRESETS, DEFINES
*   THE USER SPECIFIC LOGO AND GOVERN THE MASTER MENU
*   DEFAULT STATES.
*
*================================================================
```

```
*=============================================================
*DATA
*
*   CIT DATA CONSISTS OF THE FOLLOWING INFORMATION FOR
*   EACH CAMERA (IN ORDER OF FRONT, LEFT AND RIGHT
*   CAMERAS).
*
*      *  NUMBER OF PRESETS
*         (FOR EACH PRESET)
*         *  X HERE COORDINATE
*         *  Y HERE COORDINATE
*         *  BACKGROUND COLOR
*
*
*   CIT DATA CONSISTS OF THE FOLLOWING INFORMATION FOR
*   EACH DEFAULT COMMAND FOR THE MASTER MENU.
*
*   *  ENTITY ID1, ENTITY ID2, ... ENTITY IDN, 0
*
*
*   CIT DATA ALSO CONSISTS OF A DEL-LIKE ENTRY FOR THE
*   USER SPECIFIC LOGO.
*
*=============================================================
```

```
*================================================================
*MOD = MHMDTBL
*
*   MODULE MHMDTBL CONTAINS THE MENU DICTIONARY TABLE.  THIS
*   TABLE CONTAINS MENU NUMBER, MCT ADDRESS, NUMBER OF
*   ELEMENTS ETC.
*
*================================================================
```

```
*=============================================================
*DATA
*
*   THE MENU DICTIONARY TABLE ENTRIES ARE INCLUDE BELOW.
*   THE ELEMENTS ARE:
*
*        OFFSET    DESCRIPTION
*        ======    ===========================================
*             0    MENU NUMBER (ID)
*             1    PREVIOUS MENU
*             2    NEXT MENU
*             3    NEXT MENU MODIFIED
*             4    MENU LOCATION (-1 = SCP, 0 = IN MEMORY,
*                                 +N = IN IMAGE MEMORY N)
*             5    NUMBER OF ELEMENTS
*             6    MENU FLAGS (NOT USED)
*             7    HELP MENU NUMBER (IF ANY)
*             8    MCT ADDRESS
*
*=============================================================
```

```
*=============================================================
*MOD = MHIMACT1
*
*   MODULE MIACT IS THE FIRST MODULE CONTAINING
*   IMMEDIATE ACTION ROUTINES.
*
*   MANY TOUCHABLE (ACTION) ELEMENTS ON THE MENUS HAVE AN
*   IMMEDIATE ACTION ROUTINE ASSOCIATED WITH IT.  THE
*   ROUTINE NUMBER IS IN POSITION MCIACT OF THE MENU CONTROL
*   TABLE.  THE CCTSPI PROCEDURE IS THE IMMEDIATE ACTION
*   ROUTINE DISPATCHER.
*
*   CCTSPI EXAMINES THE TOP ELEMENT ON THE STACK,
*   AND IF THE IMMEDIATE FLAG IS ON AND THIS ELEMENT HAS
*   AN IMMEDIATE ACTION ROUTINE ASSOCIATED WITH IT, CCTSPI
*   INVOKES THE ROUTINE.
*
*   IN THE COMMENTS ASSOCIATED WITH EACH IMMEDIATE ACTION
*   ROUTINE, INPUT AND OUTPUT IS LISTED AS "NONE" UNLESS
*   REGISTERS CONTAINS SPECIFIC VALUES (OR OTHER FIELDS).
*   ACTUALLY, THESES ACTION ROUTINES USE THE SENTENCE
*   STACK AS THEIR INPUT AND FREQUENTLY THEIR OUTPUT.
*   THE OTHER OUTPUT IS GENERALLY THE MENU SCREEN WHICH
*   IS EXPLAINED IN THE "RESULT:" SECTION.
*
*=============================================================
```

```
*=============================================================
*DATA
*
*        DATA FIELDS FOR 1AR001 - 1AR010
*
*=============================================================
```

```
*=============================================================
*PROC = IAR001
*
*   PROCEDURE NAME: IAR001
*
*   IAR001 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE YES KEY ON MENU C1.  IT CAUSES THE REVIEW MENUS
*   TO BE DISPLAYED.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: REVIEW MENU R1 IS DISPLAYED.
*   TO CALL: BL @IAR001
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = IAR002
*
*   PROCEDURE NAME: IAR002
*
*   IAR002 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE NO KEY ON MENU C1.  IT CAUSES THE MENU C2 TO BE
*   DISPLAYED, BYPASSING THE REVIEW MENUS.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: MENU C2 IS DISPLAYED.
*   TO CALL: BL @IAR002
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = IAR003
*
*   PROCEDURE NAME: IAR003
*
*   IAR003 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE BACK KEY FOR ALL MENUS EXCEPT C5.  IT CALLS THE
*   GENERAL BACK ROUTINE.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: RETURNS TO THE PREVIOUS MENU
*   TO CALL: BL @IAR003
*
*=============================================================
*=============================================================
```

A-158

```
*===============================================================
*PROC = IAR004
*
*    PROCEDURE NAME: IAR004
*
*    IAR004 IS THE IMMEDIATE ACTION ROUTINE FOR THE
*    WH  GO KEY WHICH CAUSES MOVEMENT TO NEXT MENU.  ITS
*    FUNCTIONDS  S ARE TO POP ITSELF FROM THE STACK
*    AND GET THE NEXT MENU NUMBER.
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: NEXT MENU IS DETERMINED
*    TO CALL: BL @IAR004
*
*===============================================================
*===============================================================
```

```
*=============================================================
*PROC = IAR005
*
*   PROCEDURE NAME: IAR005
*
*   IAR005 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE TOUCH OF A KEY WHICH IS MUTUALLY EXCLUSIVE OF
*   ANOTHER KEY.  IF THE OTHER KEY HAS BEEN SELECTED, IT
*   IS "TURNED OFF" AND THE NEW TOUCH IS RECOGNIZED.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: MUTUALLY EXCLUSIVE TOUCH
*   TO CALL: BL @IAR005
*
*=============================================================
```

```
*=============================================================
*PROC = IAR006
*
*   PROCEDURE NAME: IAR006
*
*   IAR006 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE CANCEL KEY ON MENU C2.  IF THERE'S AN ERROR, ONLY
*   THE ERROR WILL BE RESET; OTHERWISE, THE SESSION IS
*   TERMINATED BY DRAWING MENU C6.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: ERROR RESET OR MENU C6 DRAWN
*   TO CALL: BL @IAR006
*
*=============================================================
*=============================================================
```

```
*================================================================
*PROC = IAR007
*
*    PROCEDURE NAME: IAR007
*
*    IAR007 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*    THE HELP KEY FOR ALL MENUS EXCEPT C5.  A GENERAL HELP
*    ROUTINE IS CALLED BECAUSE THE C5 HELP ROUTINE IS IN
*    ANOTHER MODULE AND THIS AVOIDS DUPLICATE CODE.
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: CALL HELP ROUTINE
*    TO CALL: BL @IAR007
*
*================================================================
*================================================================
```

```
*============================================================
*PROC = IAR008
*
*   PROCEDURE NAME: IAR008
*
*   IAR008 IS THE IMMEDIATE ACTION ROUTINE FOR THE GO KEY
*   WHICH EXECUTES THE LEXICAL SCAN FOR THE STACK.  THE
*   DELAYED FLAG IS SET FOR THE TOP SENTENCE ELEMENT.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: LEXICAL SCAN EXECUTED, DELAYED FLAG SET
*   TO CALL: BL @IAR008
*
.ST (200,302)MHIMACT1.XAS
.PG;.INDEX PROC = IAR009
.LIT
*============================================================
*PROC = IAR009
*
*   PROCEDURE NAME: IAR009
*
*   IAR009 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE CANCEL KEY FOR THE REVIEW MENUS.  THE BRANCHING IS
*   DIFFERENT DEPENDING ON THE INITIAL REVIEW OR THE
*   REVIEW DURING A CONFERENCE.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: BRANCH OUT OF REVIEW
*   TO CALL: BL @IAR009
*
*============================================================
*============================================================
```

```
*=====================================================
*PROC = IAR010
*
*   PROCEDURE NAME: IAR010
*
*   IAR010 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE CANCEL KEY FOR THE MENUS C4, C5, C6, AND HELP
*   MENUS.  ANY ERROR IS RESET.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: ERROR RESET IF ANY
*   TO CALL: BL @IAR010
*
*=====================================================
*=====================================================
```

```
*=============================================================
#PROC = IAR028
*
*    PROCEDURE NAME: IAR028
*
*    THIS PROCEDURE DRAWS A YELLOW CIRCLE WHEREVER THE
*    SCREEN IS TOUCHED.
*
*    INPUT: XPOINT--X COORDINATE
*           YPOINT--Y COORDINATE
*    OUTPUT: NONE
*    RESULT: YELLOW CIRCLE IS DRAWN
*    TO CALL: BL @IAR028
*
*=============================================================
*=============================================================
```

A-165

```
*=============================================================
*MOD = MHIMACT2
*
*   MODULE MHIMACT2 IS A CONTINUATION OF THE IMMEDIATE ACTION
*   ROUTINES.
*
*=============================================================
```

```
*=============================================================
*DATA
*    DATA FOR PROCEDURES IAR011 - IAR018
*
*=============================================================
```

```
*=============================================================
*PROC = IAR011
*
*   PROCEDURE NAME: IAR011
*
*   IAR011 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE CAMERA KEYS ON THE MASTER MENU.
*
*   THE MENU CONTROL TABLE ENTRIES FOR THE CAMERAS AND ITS
*   PRESETS ARE CONSTRUCTED WITH THE CAMERA MCT FIRST,
*   FOLLOWED BY ITS PRESET MCT ENTRIES.
*
*   IF THE CAMERA TOUCHED IS A RE-TOUCH, THEN TURN THE
*   CAMERA OFF AND ERASE ALL OF ITS PRESETS.  IF A
*   CAMERA WAS TOUCHED AND ANOTHER CAMERA HAS ALREADY
*   BEEN TOUCHED, THEN AN ERROR IS INDICATED AND THIS
*   TOUCH REMOVED FROM THE STACK.
*
*   IF THIS IS A NEW CAMERA TOUCH, SAVE THE MCT ADDRESS IN
*   CAMTCH AND THE ENTITY ID INTO CONTRL.  NOW, GO THROUGH
*   THE NEXT "LOOP" (EITHER 2 OR 8) MCT ELEMENTS AND
*   DISPLAY THEM.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: CAMERA PRESETS ARE EITHER DISPLAYED (1 TOUCH)
*           OR ERASED IF THIS WAS A RE-TOUCH.
*   TO CALL: BL @IAR011
*
*=============================================================
```

```
*=============================================================
*PROC = IAR013
*
*   PROCEDURE NAME: IAR013
*
*   IAR013 IS CALLED WHEN THE CONFERENCE CONTROL MONITOR (PREVIEW)
*   KEY IS TOUCHED.
*
*   INPUT: STACK
*   OUTPUT: NONE
*   RESULT: A CONTROLLING ELEMENT MAY BE PREVIEWED
*   TO CALL: BL @IAR013
*
*=============================================================
```

```
*=============================================================
*PROC = IAR014
*
*   PROCEDURE NAME: IAR014
*
*   IAR014 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE CAMERA "GO TO PRESET" FUNCTION.  AFTER A CAMERA HAS
*   BEEN TOUCHED, ITS PRESETS ARE DRAWN.  IAR014 IS THE
*   IMMEDIATE ACTION ROUTINE ASSOCIATED WITH EACH PRESET
*   ELEMENT.  IT ALLOWS ONLY ONE PRESET TO BE TOUCHED AT A
*   TIME.  IF A SECOND PRESET IS TOUCHED WITHIN THE SENTENCE
*   IT TURNS OFF THE FIRST.  IAR014 THEN SENDS THE GO TO
*   PRESET MESSAGE SO THAT THE CAMERA WILL BE POINTING TO
*   THE DESIRED LOCATION BY THE TIME THE GO KEY IS TOUCHED.
*
*   INPUT: NONE
*   OUTPUT: MESSAGE TO VCMAPT "GO TO PRESET"
*   RESULT: CAMERA MOVE TO PRESET
*   TO CALL: BL @IAR014
*
*=============================================================
```

```
*=============================================================
*PROC = IAR015
*
*   PROCEDURE NAME: IAR015
*
*   IAR015 IS THE IMMEDIATE ACTION ROUTINE FOR THE
*   TV MONITORS.  IF ONE MONITOR (RIGHT OR LEFT) IS
*   TOUCHED, THE OTHER MONITOR ALSO HAS ITS BORDER
*   RECOGNIZE THE TOUCH.  THE TWO MONITORS ALWAYS ACT AS
*   ONE.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT:  BOTH MONITORS RECOGNIZE TOUCH TO ONE OR OTHER
*   TO CALL:  BL @IAR015
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = IAR016
*
*    PROCEDURE NAME: IAR016
*
*    IAR016 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*    THE MIKE KEY ON ALL MENUS.  TOUCHING THE MIKE KEY
*    REVERSES ITS STATUS LIKE AN ON/OFF SWITCH.
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: AUDIO IS TURNED ON AND OFF
*    TO CALL: BL @IAR016
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = IAR017
*
*   PROCEDURE NAME: IAR017
*
*   IAR017 IMPLEMENTS THE "OTHER" KEY ON THE MASTER MENU.
*   PENDING TOUCHES ARE CLEARED AND THE SELECT MENU DISPLAY
*   NUMBER IS STORED IN PDMENU SO THAT IT WILL BE
*   DRAWN NEXT.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: SELECT MENU DISPLAY IS DRAWN
*   TO CALL: BL @IAR017
*
*=============================================================
```

```
*PROC = IAR018
*
*   PROCEDURE NAME: IAR018
*
*   IAR018 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE CANCEL KEY FOR THE MASTER MENU--C3.  IT FIRST
*   CHECKS FOR A PENDING ERROR.  IF THERE IS ONE, ONLY THE
*   ERROR IS RESET. IF THERE'S NO ERROR, THE STACK IS
*   COMPRESSED AND THEN CHECKED FOR ENTRIES.  ALL ENTRIES
*   ARE CHECKED BEFORE PROCESSING TO MAKE SURE EACH
*   DEVICE IS ACTIVE AND CAN BE DISCONNECTED.  IF EVERY-
*   THING IS OK, THE STACK IS PROCESSED AND THE MESSAGES
*   ARE SENT TO DISCONNECT.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: ERROR RESET OR SELECTIONS DEACTIVATED
*   TO CALL: BL @IAR018
*
```

A-174

```
*================================================================
*MOD = MHIMACT3
*
*   MODULE MHIMACT3 IS A CONTINUATION OF THE IMMEDIATE ACTION
*   ROUTINES.  MHIMACT3 CONTAINS ROUTINES IAR019 - IAR027.
*
*================================================================
```

```
*-------------------------------------------------------
*DATA
*
*       DATA WORDS FOR IMMEDIATE ACTION ROUTINES 19 - 27.
*
*=======================================================
```

```
*=============================================================
*PROC = IAR019
*
*   PROCEDURE NAME: IAR019
*
*   IAR019 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE SELECT CAMERA KEYS ON THE CAMERA POSITIONING MENU
*   (MENU 12).  FIRST, TURN EVERYTHING ON THE MENU OFF AND
*   STOP ALL CAMERA MOTION (IF ANY).  THIS IS DONE BY
*   CALLING THE ROUTINE (IAR027) FOR THIS MENU.
*   NEXT, TURN THE TOUCHED CAMERA ON, GETS ITS VCM ID
*   (LEFT CAM = 1, RIGHT CAM = 2), INITIALIZE THE ZOOM/FOCUS
*   APT FUNCTION AND THEN CONNECT THE SELECTED CAMERA TO
*   THE CONFERENCE CONTROL MONITOR.  FINALLY RE-DRAW THE
*   MENU TO INDICATE WHICH CAMERA IS ON.
*
*   INPUT: NONE
*   OUTPUT: CAMERA MESSAGES
*   RESULT: CAMERA IS CONNECTED TO CCM
*   TO CALL: BL @IAR019
*
*   GLOBALS: CURRMA--USED HERE, SET IN DRMENU
*
*   PROCEDURES CALLED: IAR027, GMSGAD, SYSREQ, RDMENU
*
*   CALLED BY: CCTSP1
*
*=============================================================
```

```
*=============================================================
*PROC = I19A00
*
*   PROCEDURE NAME: I19A00
*
*   I19A00 IS AN AUXILIARY ROUTINE TO IAR019.  ITS FUNCTION IS
*   TO CONNECT OR DISCONNECT A CAMERA FROM THE CONFERENCE
*   MONITOR.
*
*   INPUT: R0, WHEN =0, DISCONNECT THE CAMERA
*              WHEN =2, CONNECT THE CAMERA
*   OUTPUT: MESSAGE TO CAMERA CONTROLLER
*   RESULT: CAMERA IS CONNECTED/DISCONNECTED
*   TO CALL: BL @I19A00
*
*=============================================================
*=============================================================
```

```
*=============================================================
*PROC = IAR020
*
*   PROCEDURE NAME: IAR020
*
*   IAR020 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE BACK KEY ON MENU 12 (CAMERA POSITIONING).  IT MUST
*   STOP ALL CAMERA MOTION AND DISCONNECT THE CAMERA FROM
*   THE CONFERENCE MONITOR.  THESE FUNCTIONS ARE HANDLED BY
*   IAR027 (MENU 12 X KEY) AND THEN IAR010 (GENERAL BACK
*   KEY) IS CALLED.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: STOP CAMERA MOTION, DISCONNECT CAMERA AND
*           RETURN TO THE MASTER MENU.
*   TO CALL: BL @IAR020
*
*=============================================================
```

```
*=============================================================
*PROC = IAR021
*
*    PROCEDURE NAME: IAR021
*
*    IAR021 IMPLEMENTS THE HELP KEY ON THE CAMERA POSITIONING
*    MENU (C05MNU).  BEFORE INVOKING THE HELP FUNCTION, THIS
*    ROUTINE FIRST TURNS OFF ANY CAMERA MOTION.
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: HELP MENU IS DRAWN
*    TO CALL: BL @IAR021
*
*=============================================================
```

```
*=============================================================
*PROC = IAR022
*
*   PROCEDURE NAME: IAR022
*
*   IAR022 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE CAMERA PRESET KEYS ON THE CAMERA POSITIONING MENU
*   (MENU 12).  IF A CAMERA HAS NOT BEEN TOUCHED BEFORE THIS
*   KEY, AN ERROR CONDITION ARISES AND THE PROCEDURE
*   RETURNS.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: MENU CHANGES TO REFLECT TOUCHES
*   TO CALL: BL @IAR022
*
*   GLOBALS: CURRMA--USED HERE, SET IN DRMENU
*
*   PROCEDURES CALLED: GRMST, RDMENU
*
*   CALLED BY: CCTSPI
*
*=============================================================
```

```
*=============================================================
*PROC = IAR023
*
*   PROCEDURE NAME: IAR023
*
*   IAR023 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE ZOOM KEYS ON THE CAMERA POSITIONING MENU (MENU 12).
*   IF A CAMERA IS NOT TOUCHED BEFORE THIS KEY, AN ERROR
*   CONDITION IS FLAGGED.  OTHERWISE, GET AND CONSTRUCT THE
*   ZOOM MESSAGE.  FROM THE ZOOM KEY'S MCT ID, INDEX INTO
*   TABLE ZOOM TO GET THE ZOOM INCREMENT AND DIRECTION.
*   MOVE THESE VALUES INTO THE ZOOM MESSAGE AND THEN SEND
*   IT.  RE-DRAW THE MENU SO THAT THE ZOOM KEY WILL BE ON.
*   FINALLY, TURN THE ZOOM KEY OFF AND REDRAW THE MENU
*   AGAIN.
*
*   INPUT: NONE
*   OUTPUT: ZOOM MESSAGE
*   RESULT: ZOOM KEY FLASHES AND CAMERA ZOOMS
*   TO CALL: BL @IAR023
*
*   GLOBALS: CURRMA--USED HERE, SET IN DRMENU
*
*   PROCEDURES CALLED: GRMST, GMSGAD, SYSREQ, RDMENU
*
*   CALLED BY: CCTSPI
*
*=============================================================
```

```
*=============================================================
*PROC = IAR024
*
*   PROCEDURE NAME: IAR024
*
*   IAR024 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE FOCUS KEYS ON THE CAMERA POSITIONING MENU (MENU 12).
*   IF A CAMERA IS NOT TOUCHED BEFORE THIS KEY, AN ERROR
*   CONDITION IS FLAGGED.  OTHERWISE, GET AND CONSTRUCT THE
*   FOCUS MESSAGE.  FROM THE FOCUS KEY'S MCT ID, INDEX INTO
*   TABLE FOCUS TO GET THE FOCUS INCREMENT AND DIRECTION.
*   MOVE THESE VALUES INTO THE FOCUS MESSAGE AND THEN SEND
*   IT.  RE-DRAW THE MENU SO THAT THE FOCUS KEY WILL BE ON.
*   FINALLY, TURN THE FOCUS KEY OFF AND REDRAW THE MENU
*   AGAIN.
*
*   INPUT: NONE
*   OUTPUT: FOCUS MESSAGE
*   RESULT: FOCUS KEY FLASHES AND CAMERA FOCUSES
*   TO CALL: BL @IAR024
*
*   GLOBALS: CURRMA--USED HERE, SET IN DRMENU
*
*   PROCEDURES CALLED: GRMST, GMSGAD, SYSREQ, RDMENU
*
*   CALLED BY: CCTSPI
*
*=============================================================
```

```
*=============================================================
*PROC = IAR025
*
*    PROCEDURE NAME: IAR025
*
*    IAR025 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*    THE EIGHT SLEW (PAN/TILT) KEYS ON THE CAMERA POSITIONING
*    MENU (MENU 12).  IF A CAMERA HAS NOT BEEN TOUCHED BEFORE
*    THE SLEW KEY, INDICATE AN ERROR AND RETURN.
*    OTHERWISE, GET THE CAMERA SLEW MESSAGE BUFFER FROM
*    GMSGAD.  NEXT, DETERMINE IF THIS IS A RE-TOUCH (I.E.
*    TURNING THE SLEW OFF).  IF IT IS, THEN MOVE THE OFF
*    VALUE INTO THE MESSAGE, SEND IT AND RETURN. NEXT,
*    CHECK TO SEE IF ANOTHER SLEW KEY IS CURRENTLY ON.  IF
*    ONE IS, TURN THE KEY OFF AND AGAIN SEND THE SLEW OFF
*    MESSAGE.  IN EITHER CASE, NOW TURN ON THE TOUCHED KEY,
*    CONSTRUCT THE SLEW MESSAGE, SEND IT AND RETURN.
*
*    INPUT: NONE
*    OUTPUT: SLEW MESSAGE
*    RESULT: SLEW KEYS ARE MAINTAINED AND CAMERA MOVES/STOPS.
*    TO CALL: BL @IAR025
*
*    GLOBALS: CURRMA--USED HERE, SET IN DRMENU
*
*    PROCEDURES CALLED: GRMST, GMSGAD, SYSREQ, RDMENU
*
*    CALLED BY: CCTSPI
*
*=============================================================
```

```
*=============================================================
*PROC = RSTPST
*
*    PROCEDURE NAME: RSTPST
*
*    RSTPST WILL ERASE ACTIVE PRESETS FROM THE MASTER MENU
*    (C03MNU) IF THEIR CAMERAS ARE RE-POSITIONED.
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: C03MNU PRESETS MAY BE ERASED
*    TO CALL: BL @RSTPST
*
*=============================================================
*=============================================================
```

A-185

```
*=============================================================
*PROC = IAR026
*
*   PROCEDURE NAME: IAR026
*
*   IAR026 TURNS OFF CAMERA MOTION, DISCONNECTS THE CAMERA
*   AND GOES ON TO THE NEXT MENU.
*
*  INPUT: NONE
*   OUTPUT: NONE
*   RESULT: NEXT MENU IS DRAWN
*   TO CALL: BL @IAR026
*
*=============================================================
```

```
*=============================================================
*PROC = IAR027
*
*   PROCEDURE NAME: IAR027
*
*   IAR027 IS THE IMMEDIATE ACTION ROUTINE WHICH IMPLEMENTS
*   THE SLEW OFF KEY ON THE CAMERA POSITIONING MENU (MENU 12).
*   IT TURNS OFF ALL CAMERA MOTION, AND CLEAR THE TOUCHED
*   PRESET KEYS.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT: ALL CAMERA ACTIVITY ENDS, ALL KEYS TURNED OFF
*   TO CALL: BL @IAR027
*
*   GLOBALS: CURRMA--USED HERE, SET IN DRMENU
*
*   PROCEDURES CALLED: GMSGAD, SYSREQ, RDMENU
*
*   CALLED BY: CCTSPI
*
*=============================================================
```

```
*MOD = MHDLACT
*
*   MODULE MHDLACT IS THE FIRST OF THE DELAYED ACTION ROUTINE
*   MODULES.  DELAYED ACTION ROUTINES ARE CALLED FROM THE
*   INTERPRETER AND SPECIFIED IN THE MENU CONTROL TABLE.
*   IF AN ELEMENT HAS AN ASSOCIATED DELAYED ACTION, FIELD
*   MCDACT OF THE MCT WILL CONTAIN A NON-ZERO VALUE.  THE
*   VALUE WILL CORRESPOND TO THE DELAYED ACTION ROUTINE
*   NUMBER.
*
*   THE DELAYED ACTION ROUTINES GENERALLY USE THE EXISTING
*   SENTENCE STACK AS ITS INPUT AND FOR OUTPUT THEY MODIFY
*   THE MENUS AND THE ENTITY STATUS TABLE.  THEY ALSO SEND
*   MESSAGES TO THE OTHER VARIOUS SYSTEM ROUTINES.
*
*=============================================================
```

```
*=================================================
*DATA
*
*
*=================================================
```

```
*=============================================================
*PROC = DAR001
*
*    DAR001 IS THE DELAYED ACTION ROUTINE TO HANDLE ALL
*    ELEMENTS ON THE MASTER MENU.  WHEN DAR001 IS CALLED,
*    THE ELEMENT ON THE TOP OF THE STACK IS THE
*    SIGNAL SOURCE.
*
*    SET EACH OF
*    THE FOLLOWING STACKED ELEMENTS TO BE ON AND CONTROLLED
*    BY THE SOURCE.  IF THE ELEMENT WAS FORMERLY CONTROLLED,
*    FIRST TURN IT OFF AND SEND THE DISCONNECT MESSAGE.
*    OTHERWISE, JUST TURN IT ON AND SEND THE CONNECT MESSAGE.
*
*    ONCE THE STACK HAS BEEN EXHAUSTED, PROCEDURE LSENDS
*    IS CALLED TO DISCONNECT ANY REMAINING LOOSE ENDS.
*    A LOOSE END IS DEFINED AS A SOURCE WITHOUT A SINK
*    OR A SINK WITHOUT A SOURCE.
*
*
*    INPUT: NONE
*    OUTPUT: NONE
*    RESULT: MENU AND EST REFLECT TOUCHES, AND DEVICES ARE
*            CONNECTED AND DISCONNECTED AS REQUIRED.
*    TO CALL: BL @DAR001
*
*=============================================================
```

```
*=====================================================
*PROC = DAR002
*
*   PROCEDURE NAME: DAR002
*
*   DAR002 IS THE DELAYED ACTION ROUTINE FOR THE
*   CIRCLES ON MENU C4 TO SELECT ANOTHER FUNCTION.
*   THE NEXT MENU IS DETERMINED BY ACCESSING THE
*   EST ENTRY FOR THE SELECTED CIRCLE.  PENDING
*   MENU IS SET AND THE STACK IS CLEARED.
*
*   INPUT: NONE
*   OUTPUT: NONE
*   RESULT:  NEXT MENU IS DETERMINED
*   TO CALL: BL @DAR002
*
*=====================================================
```

```
*=============================================================
*PROC = DAR003
*
*   PROCEDURE NAME: DAR003
*
*   DAR003 IS CALLED WHEN THE HI-RES SCREEN ELEMENT IS
*   FOUND IN A SENTENCE.  DAR003 WILL REPORT THAT THIS
*   ELEMENT MAY NOT BE USED IN THIS MANNER.
*
*   INPUT: NONE
*   OUTPUT: ERROR MESSAGE
*   RESULT: GO TURNS RED
*   TO CALL: BL @DAR003
*
*=============================================================
*=============================================================
```

```
*=====================================================
*PROC = DAR004
*
*   PROCEDURE NAME: DAR004
*
*   DAR004 IS CALLED WHEN THE CCM ELEMENT IS
*   FOUND IN A SENTENCE.  DAR004 WILL REPORT THAT THIS
*   ELEMENT MAY NOT BE USED IN THIS MANNER.
*
*   INPUT: NONE
*   OUTPUT: ERROR MESSAGE
*   RESULT: GO TURNS RED
*   TO CALL: BL @DAR004
*
*=====================================================
*=====================================================
```