```
DISPLAY ELEMENT LIBRARY

        ARRAY   DEL (1:NUMBER OF ELEMENT SHAPES) OF DEL-ENTRY
        RECORD  DEL-ENTRY       [DISPLAY ELEMENT LIBRARY]
                .XTOL,          [X TOLERANCE]
                .YTOL,          [Y TOLERANCE]
                .NSE,           [NUMBER OF SUB-ELEMENTS]
                .LENGTH: INTEGER        [LENGTH OF LIGHT STRING]
                ARRAY   DEL.SUB (1:NSE) OF DEL.SUB-ENTRY
                RECORD DEL.SUB-ENTRY    [DISPLAY ELEMENT LIBRARY]
                        .CLASS,         [SUB-ELEMENT TYPE]
                        .XFILL,         [X FILL OFFSET]
                        .YFILL,         [Y FILL OFFSET]
                        .XTEXT,         [X TEXT OFFSET]
                        .YTEXT: INTEGER [Y TEXT OFFSET]
                        .TEXT: STRING(LENGTH)   [LIGHT STRING]
```

```
TEXT ELEMENT LIBRARY

        ARRAY   TEL     (1:NUMBER OF TEXT STRINGS) OF TEL-ENTRY
        RECORD  TEL-ENTRY       [TEXT ELEMENT LIBRARY]
                .LENGTH: INTEGER        (LENGTH OF TEXT STRING)
                .TEXT: STRING (LENGTH)  (TEXT STRING)
                .EOT: INTEGER           (END OF TEXT STRING)
```

A-6

```
MESSAGE TABLES

        ARRAY   MESSAGE (1:LSTERR) OF MESSAGE-RECORD
        RECORD  MESSAGE-RECORD
                .TEXT: STRING (VARIABLE)
                .EOM:  INTEGER              [END OF MESSAGE]

        ARRAY   RQB (1:17) OF RQB-RECORD
        RECORD  RQB-RECORD
                .ACTION,                    [ACTION CODE]
                .DESTINATION,               [DESTINATION APT/DHT]
                .LENGTH,                    [LENGTH OF REMAINING RQB]
                .MODIFIER,                  [CAMERA NUMBER, IF APPLICABLE]
                .CODE,                      [RQB CODE]
                .PARAMETERS: INTEGER        [RQB PARAMETERS]

        ARRAY   STACK (1:16) OF STACK RECORD
        RECORD  STACK-RECORD                [SENTENCE/ELEMENT STACK]
                .IFLAG,                     [IMMEDIATE ACTION FLAG]
                .DFLAG,                     [DELAYED ACTION FLAG]
                .ELEMENT: INTEGER           [ELEMENT MCT ADDRESS]

ATAD
PROC CTPAPT (ALT INPUT-MESSAGE)

        DATA
                RECORD INPUT-MESSAGE
                        .TEXT(24) :STRING

                SCALAR X,Y,MSG-ID,MSG-FLAG :INTEGER

                CONSTANT MINUS1 := -1 :INTEGER

        ATAD

        <REENABLE INTERRUPTS AFTER DISPATCH>
        RUN MHRMAP (USE MINUS1)

        CASE
                <TYPE OF INPUT-MESSAGE>

        SELECT (FULL-INPUT-MESSAGE.HEADER.ORIG-TASK = <DHTCTP>)
                RUN DCDCTP (ALT FULL-INPUT-MESSAGE, CTPCB)
                RUN CNVCTP (ALT CTPCB)
                X := CTPCB.XL
                Y := CTPCB.YL
                RUN CCTSPI (ALT X,Y)

        SELECT (INPUT-MESSAGE.TEXT(1:1) = 254)
                MSG-FLAG := INPUT-MESSAGE.TEXT(2:1)
                IF
                        INPUT-MESSAGE.TEXT(3:1) < '0'
                THEN

                                A-7
```

```
                    MSG-ID := INPUT-MESSAGE.TEXT(1:2)
               ELSE
                    MSG-ID := <ADDRESS OF INPUT-MESSAGE.TEXT(4:1)>
               FI
               RUN MSGDSP (USE MSG-FLAG, MSG-ID)
               RUN RELBUF (ALT FULL-INPUT-MESSAGE)

          ELSE
               RUN RELBUF (ALT FULL-INPUT-MESSAGE)
          ESAC

CORP  (ACTUALLY, BRANCH TO DISPAT)
```

A-8

```
PROC SYSREQ (USE RQB; ALT REPLY-ADDR)
        DATA
                RECORD INPUT-MESSAGE
                        .TEXT(24) :STRING
                SCALAR RC,INTVL,REPLY-ADDR :INTEGER
                SCALAR GOT-BUF  :LOGICAL

        ATAD

        GOT-BUF := FALSE
        WHILE
                NOT GOT-BUF
        DO
                RUN GETBUF (ALT FULL-INPUT-MESSAGE, GOT-BUF)
                IF
                        NOT GOT-BUF
                THEN
                        RC := RC-TIMEOUT
                        WHILE
                                RC = RC-TIMEOUT
                        DO
                                RUN WAITAC (USE <ACTBUF>; ALT RC)
                                IF
                                        RC <> RC-TIMEOUT AND RC <> RC-SUCCESS
                                THEN
                                        RUN ABORT
                                FI
                                RUN MHRMAP (USE MINUS1)
                        OD
                FI
        OD

        FULL-INPUT-MESSAGE.HEADER.DEST-TASK := RQB.DST
        INPUT-MESSAGE.TEXT(1:2) = RQB.CMD
        N := RQB.LEN / 2 - 1
        <COPY N WORDS FROM RQB.PRM TO INPUT-MESSAGE.TEXT(3:2N)>
```

```
        CASE
                RQB.AC

        SELECT (RQB.AC = 0) [ORDINARY SEND]
                RUN SEND (ALT FULL-INPUT-MESSAGE)

        SELECT (RQB.AC > 0) [SENDAC]
                RUN SENDAC (USE RQB.AC; ALT FULL-INPUT-MESSAGE, REPLY-ADDR, RC
                IF
                        RC <> RC-SUCCESS AND RC <> RC-TIMEOUT
                THEN
                        RUN ABORT
                FI

                WHILE
                        RC = RC-TIMEOUT
                DO
                        RUN WAITAC (USE RQB.AC; ALT REPLY-ADDR, RC)

                        IF
                                RC <> RC-SUCCESS AND RC <> RC-TIMEOUT
                        THEN
                                RUN ABORT
                        FI

                        RUN MBRMAP (USE MINUS1)
                OD

        SELECT (RQB.AC < 0) [SENDTM]
                INTVL := - RQB.AC * 6    [GIVEN IN .1 SEC INTERVALS]
                RUN SENDTM (USE INTVL; ALT FULL-INPUT-MESSAGE, RC)
                IF
                        RC <> RC-SUCCESS
                THEN
                        RUN ABORT
                FI

                RUN MBRMAP (USE MINUS1)
        ESAC

CORP
```

```
PROC DCDCTP (ALT FULL-INPUT-MESSAGE, CTPCB)
        DATA
                RECORD INPUT-MESSAGE
                        .TEXT(24) :STRING
                SCALAR NUM-STRING(5) :STRING
                SCALAR NUM,LEN :INTEGER
        ATAD

        LEN := 5
        NUM-STRING := INPUT-MESSAGE.TEXT(1:5)
        RUN ASCBIN (USE NUM-STRING,LEN; ALT NUM,RC)
        CTPCB.XT := NUM

        NUM-STRING := INPUT-MESSAGE.TEXT(6:5)
        RUN ASCBIN (USE NUM-STRING,LEN; ALT NUM,RC)
        CTPCB.YT := NUM

        RUN RELBUF (ALT FULL-INPUT-MESSAGE)
CORP

PROC CNVCTP (ALT CTPCB)
        CTPCB.XL := CTPCB.XLMN + ABS(CTPCB.XT - CTPCB.XTMN) * CTPCB.DXL / CTP
        CTPCB.YL := CTPCB.YLMN + ABS(CTPCB.YT - CTPCB.YTMN) * CTPCB.DYL / CTP
CORP
```

A-11

```
PROC INTMHR              [INITIATES THE MENU HANDLER MODULE]
     DATA
          SCALAR  CLRCCM: STRING(V)
          SCALAR  COLORS: STRING(V)
          CONSTANT FSTMEN: INTEGER        [FIRST MENU ID]
     ATAD
     ICB := <LIGHT ICB ADDRESS>
     RUN LIGHT (USE CLRCCM)              [CLEAR THE MONITOR]
     RUN LIGHT (USE COLORS)              [DEFINES THE COLORS]
     ROMEMU := FSTMEN
     RUN DRMENU
CORP
```

```
PROC MINIT                    [RESETS MENU HANDLER VALUES]
        DATA
                SCALAR  LOOP,
                        LOOP1,
                        ELEMENT,
                        MENU: INTEGER
                CONSTANT NUMMEN,          [NUMBER OF MENUS]
                         ESTENT: INTEGER          [NUMBER OF EST ENTRIES]
                ARRAY   DEFEST (1:ESTENT) OF DEFEST-ENTRY
                RECORD  DEFEST-ENTRY
                        .FON,
                        .FPRE: INTEGER
        ATAD
        MENU := MDT
                LOOP := NUMMEN
        WHILE                         [RESET MENU DICTIONARY TABLE]
                        LOOP > 0
        DO
                <RESET MDT VALUES>
                ELEMENT := MDT(MENU).MCT
                LOOP1 := MDT(MENU).NEL
                WHILE                 [RESET MENU CONTROL TABLE ELEMENTS]
                        LOOP1 > 0
                DO
                        <RESET MCT(ELEMENT) VALUES>
                        LOOP := LOOP - 1
                        ELEMENT := ELEMENT + MCTLEN
                OD
                LOOP := LOOP -1
                MENU := MENU + MDTLEN
        OD
        ESADDR := EST
        LOOP := ESTENT
        LOOP1 := 0
        WHILE                         [RESET ENTITY STATUS TABLE]
                LOOP > 0
        DO
                LOOP1 := LOOP1 + 1
                EST(ESADDR).CB := 0
                EST(ESADDR).FON := DEFEST(LOOP1).FON
                EST(ESADDR).FPRE := DEFEST(LOOP1).FPRE
                ESADDR := ESADDR + ESTLEN
                LOOP := LOOP -1
        OD
        <RESET SYSTEM PARAMETERS>
        <LOAD PRESET COORDINATES FROM CONFERENCE INITIALIZATION
                TABLE TO MENU C3 ELEMENTS>
        <PFLAG(1-2,1-2) := 0>
CORP
```

```
PROC DRAW-MENU (USE MENU)
    DATA
        SCALAR CURR-MENU: INTEGER (CURRENT MENU DISPLAYED)
        SCALAR N-ICBS: INTEGER (NUMBER OF ICBS AVAILABLE)
        SCALAR @MCT: INTEGER
        SCALAR LOOP: INTEGER
        SCALAR POS: INTEGER
        SCALAR MIN-USAGE: INTEGER
        SCALAR LOOP-MIN: INTEGER
        SCALAR DROP-MENU: INTEGER
    ATAD
    IF
        MENU = FSTMEN
    THEN
        RUN MINIT
    FI
    IF
        CURR-MENU = MENU
    THEN
        RUN REDRAW-MENU (USE MENU)
    FI
    IF
        MENU-DICT (MENU).FLAGS.ICB <> 0
    THEN
        RUN SWAP-MENU (USE MENU)
    FI
    POS := 0
    LOOP := 0
    LOOP-MIN := 1
    MIN-USAGE := 999
    DO
        LOOP := LOOP + 1
        IF
            IMAGE-ARRAY (LOOP).MENU-ID = 0 (EMPTY ICB)
        THEN
            POS := LOOP
        ELSE
            IF
                IMAGE-ARRAY (LOOP).USAGE < MIN-USAGE
            THEN
                MIN-USAGE := IMAGE-ARRAY (LOOP).USAGE
                LOOP-MIN := LOOP
            FI
        FI
    UNTIL
        POS <> 0 OR LOOP = N-ICBS
    OD
    IF
        POS = 0 (NO EMPTY SLOTS)
    THEN
        POS := LOOP-MIN
        DROP-MENU := IMAGE-ARRAY(POS).MENU-ID
        MENU-DICT(DROP-MENU).FLAGS.ICB := 0

                                A-14
```

```
                @MCT := MENU-DICT(CURR-MENU).MCT
                LOOP := 0
                WHILE
                        LOOP =< MENU-DICT (CURR-MENU).N-MCT-ELEMENTS
                DO
                        LOOP := LOOP + 1
                        @MCT-->MCT(LOOP).CURR-STATUS := 0
                OD
        FI
        MENU-DICT(MENU).FLAGS.ICB := POS
        MENU-DICT(MENU).PREV-MENU := CURR-MENU
        IMAGE-ARRAY(POS).MENU-ID := MENU
        IMAGE-ARRAY(POS).USAGE := 1
        RUN LIGHT [< SET ICB NUMBER, IMAGE-ARRAY(POS).ICB-NO>]
        LOOP := 0
        WHILE
                LOOP =< MENU-DICT(MENU).N-MCT-ELEMENTS
        DO
                LOOP := LOOP + 1
                RUN DRAW-ELEMENT (USE MENU, LOOP)
        OD
        CURR-MENU := MENU
        FI
                MENU = C03MNU AND C03DRW = 0
        THEN
                RUN STDFLT
                C03DRW := C03DRW + 1
        FI
        RUN LIGHT [< SET ICB   TO DISPLAY>]
CCRP
```

```
PROC GMADDR (USE MENUID)          [GET MDI ADDRESS]
        DATA
        ATAL
        MENUAL := (MENUID - 1) * MDILEN + MDI
CORP
```

```
PROC MNEXT (ALT PDMENU)          [GET NEXT MENU ID]
        DATA
        ATAD
        IF
                MDT(CURRMA).NMM <> 0
        THEN
                PDMENU := MDT(CURRMA).NMM
                MDT(CURRMA).NMM := 0
        ELSE
                PDMENU := MDT(CURRMA).NM
        FI
CORP
```

A-17

```
PROC ESTADD (USE MEADDR, ALT ESADDR)
        DATA
                SCALAR  ID: INTEGER
        ATAD
        ID := MCT(MEADDR).EID
        ESADDR := (ID -1) * ESTLEN + ES1
CORP
```

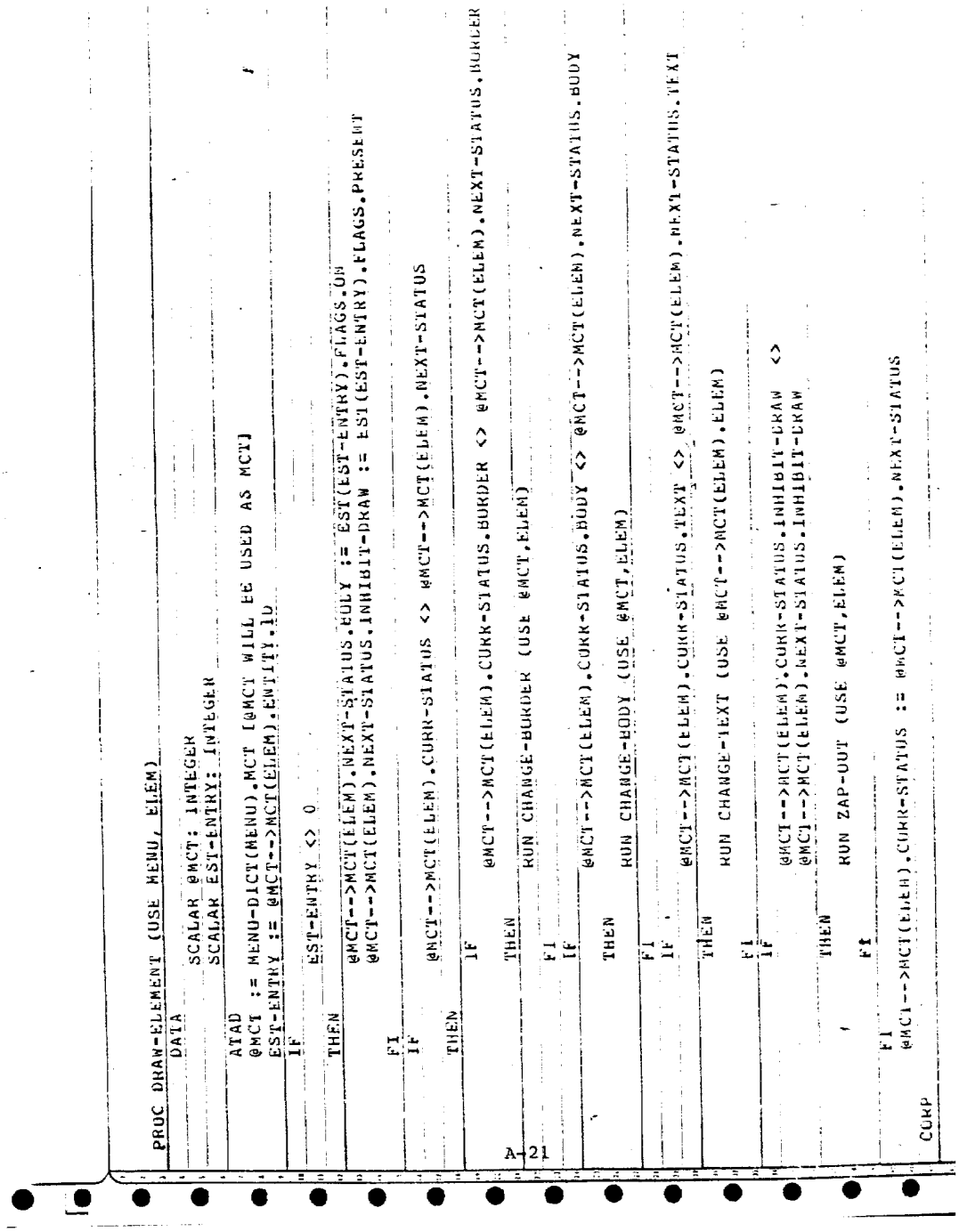
A-18

```
PROC REDRAw-MENU (USE MENU)
        DATA
                SCALAR CURR-MENU: INTEGER
                SCALAR LOOP: INTEGER
        ATAD
        LOOP := 0
        WHILE
                LOOP =< MENU-DICT(MENU).N-MCT-ELEMENTS
        DC
                LOOP := LOOP + 1
                RUN DRAW-ELEMENT (USE MENU, LOOP)
        OD
CORP
```

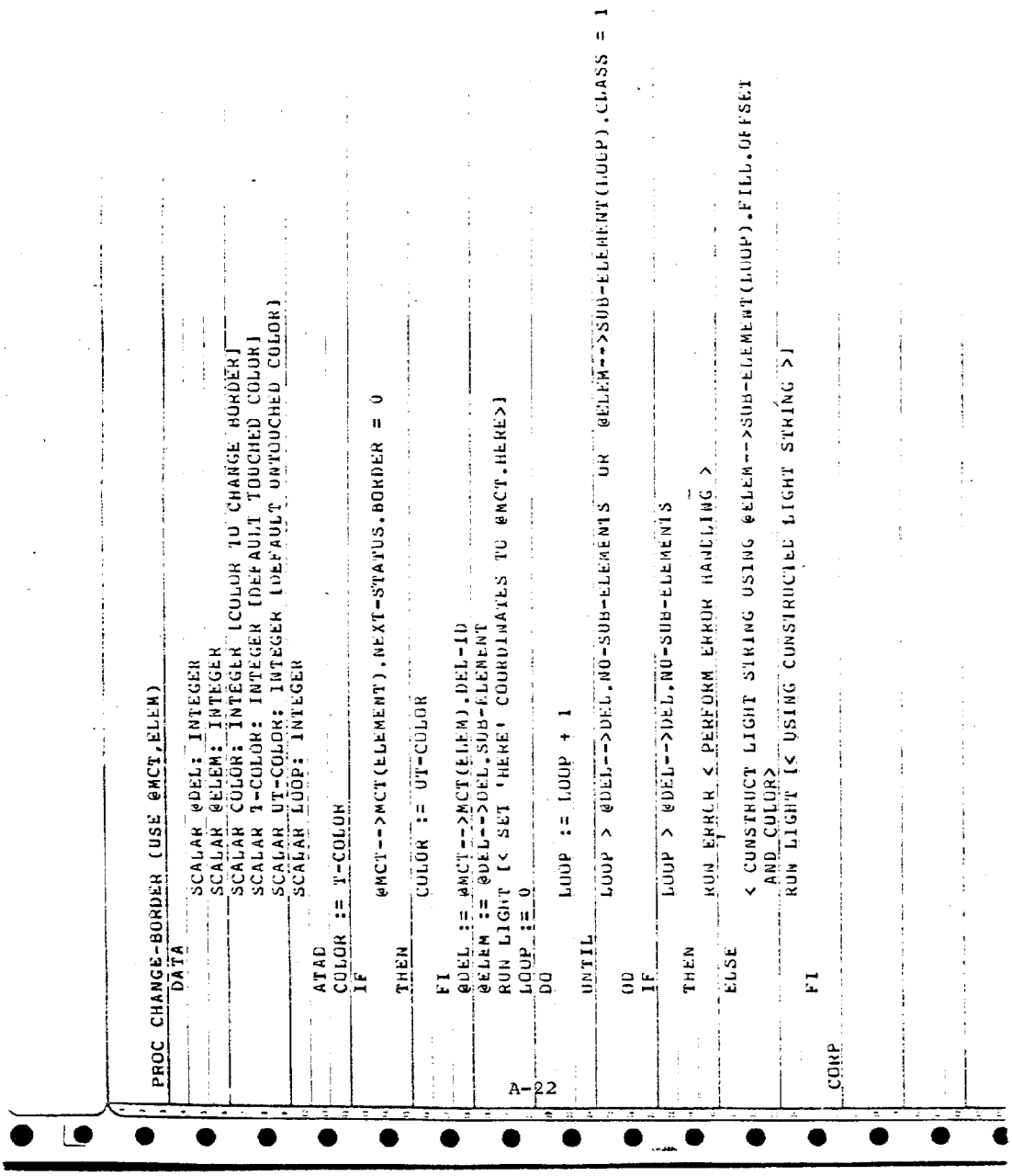
A-19

```
PROC SWAP-MENU (USE MENU)
        DATA
                SCALAR CURR-MENU: INTEGER
                SCALAR IMAGE-LOC: INTEGER
                SCALAR CURR-ICB: INTEGER (CURRENT ICB IN DISPLAY)
                SCALAR LOOP: INTEGER
        ATAD
        MENU-DICT(MENU).PREV-MENU := CURR-MENU
        IMAGE-LOC := MENU-DICT(MENU).FLAGS.ICB
        IMAGE-ARRAY(IMAGE-LOC).USAGE := IMAGE-ARRAY(IMAGE-LOC).USAGE + 1
        RUN LIGHT [<SET DRAW ICB TO IMAGE-LOC>]
        LOOP := 0
        WHILE
                LOOP =< MENU-DICT(MENU).N-MCT-ELEMENTS
        DO
                LOOP := LOOP + 1
                RUN DRAW-ELEMENT (USE MENU, LOOP)
        OD
        CURR-ICB := IMAGE-LOC
        CURR-MENU := MENU
        RUN LIGHT [<SET CURR-ICB TO DISPLAY>]
CORP
```

A-20



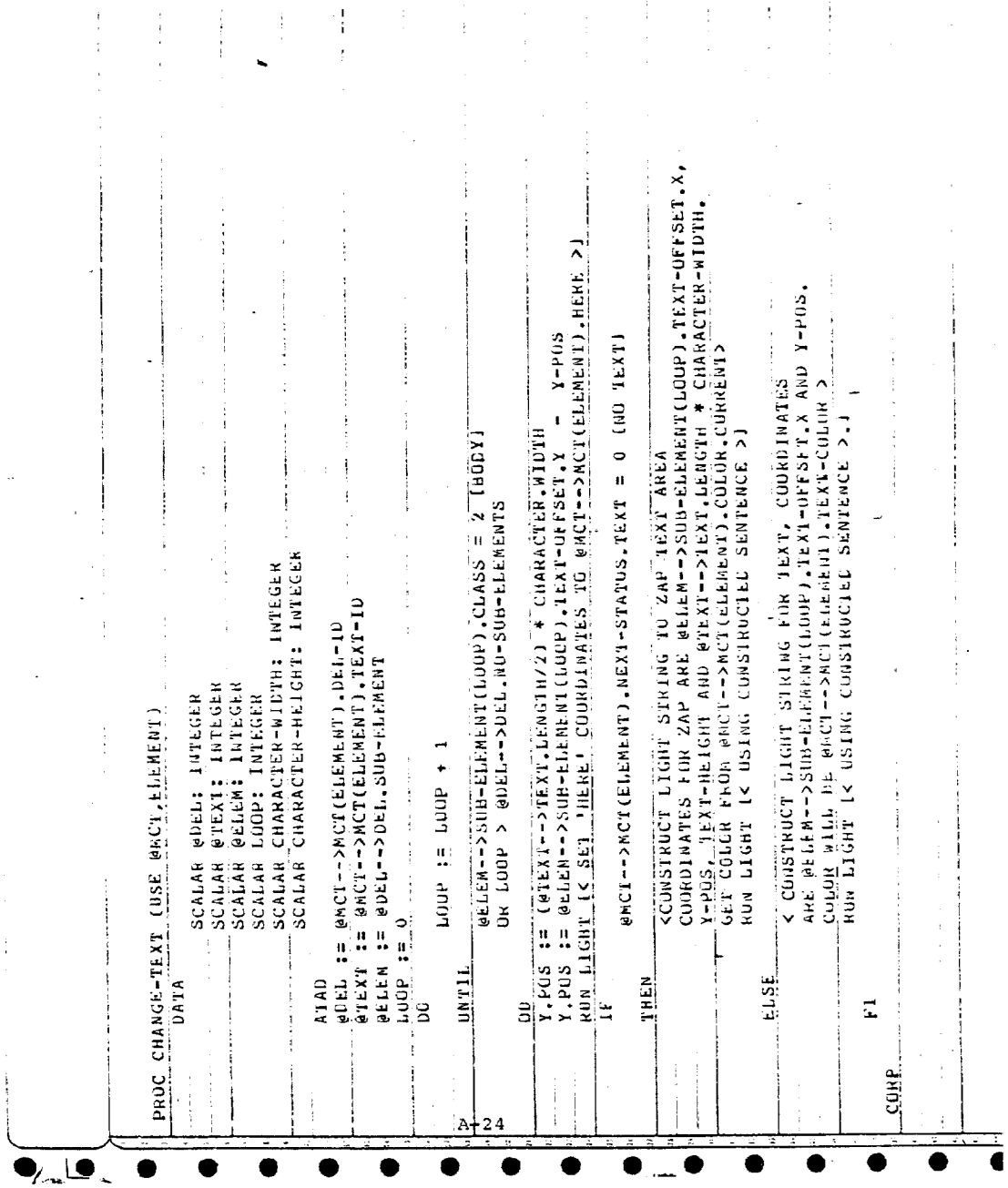
```
PROC DRAW-ELEMENT (USE MENU, ELEM)
  DATA
    SCALAR @MCT: INTEGER
    SCALAR EST-ENTRY: INTEGER
  ATAD
  @MCT := MENU-DICT(MENU),MCT [@MCT WILL BE USED AS MCT]
  EST-ENTRY := @MCT-->MCT(ELEM),ENTITY.ID
  IF
    EST-ENTRY <> 0
  THEN
    @MCT-->MCT(ELEM),NEXT-STATUS.BODY := EST(EST-ENTRY),FLAGS.ON
    @MCT-->MCT(ELEM),NEXT-STATUS.INHIBIT-DRAW := EST(EST-ENTRY),FLAGS.PRESENT
  FI
  IF
    @MCT-->MCT(ELEM),CURR-STATUS <> @MCT-->MCT(ELEM),NEXT-STATUS
  THEN
    IF
      @MCT-->MCT(ELEM),CURR-STATUS.BORDER <> @MCT-->MCT(ELEM),NEXT-STATUS.BORDER
    THEN
      RUN CHANGE-BORDER (USE @MCT,ELEM)
    FI
    IF
      @MCT-->MCT(ELEM),CURR-STATUS.BODY <> @MCT-->MCT(ELEM),NEXT-STATUS.BODY
    THEN
      RUN CHANGE-BODY (USE @MCT,ELEM)
    FI
    IF
      @MCT-->MCT(ELEM),CURR-STATUS.TEXT <> @MCT-->MCT(ELEM),NEXT-STATUS.TEXT
    THEN
      RUN CHANGE-TEXT (USE @MCT-->MCT(ELEM),ELEM)
    FI
    IF
      @MCT-->MCT(ELEM),CURR-STATUS.INHIBIT-DRAW <>
      @MCT-->MCT(ELEM),NEXT-STATUS.INHIBIT-DRAW
    THEN
      RUN ZAP-OUT (USE @MCT,ELEM)
    FI
    @MCT-->MCT(ELEM),CURR-STATUS := @MCT-->MCT(ELEM),NEXT-STATUS
  FI
CORP
```



```
PROC CHANGE-BORDER (USE @MCT,ELEM)
    DATA
        SCALAR @DEL: INTEGER
        SCALAR @ELEM: INTEGER
        SCALAR COLOR: INTEGER (COLOR TO CHANGE BORDER)
        SCALAR T-COLOR: INTEGER (DEFAULT TOUCHED COLOR)
        SCALAR UT-COLOR: INTEGER (DEFAULT UNTOUCHED COLOR)
        SCALAR LOOP: INTEGER
    ATAD
    COLOR := T-COLOR
    IF
        @MCT-->MCT(ELEMENT).NEXT-STATUS.BORDER = 0
    THEN
        COLOR := UT-COLOR
    FI
    @DEL := @MCT-->MCT(ELEM).DEL-ID
    @ELEM := @DEL-->DEL.SUB-ELEMENT
    RUN LIGHT (< SET 'HERE' COORDINATES TO @MCT.HERE>)
    LOOP := 0
    DO
        LOOP := LOOP + 1
    UNTIL
        LOOP > @DEL-->DEL.NO-SUB-ELEMENTS OR @ELEM-->SUB-ELEMENT(LOOP).CLASS = 1
    OD
    IF
        LOOP > @DEL-->DEL.NO-SUB-ELEMENTS
    THEN
        RUN ERROR < PERFORM ERROR HANDLING >
    ELSE
        < CONSTRUCT LIGHT STRING USING @ELEM-->SUB-ELEMENT(LOOP).FILL.OFFSET
          AND COLOR>
        RUN LIGHT (< USING CONSTRUCTED LIGHT STRING >)
    FI
CORP
```

A-22

```
PROC CHANGE-BODY (USE @MCT,ELEM)
        DATA
                SCALAR @DEL: INTEGER
                SCALAR @ELEM: INTEGER
                SCALAR COLOR: INTEGER
                SCALAR EST-ENTRY: INTEGER
                SCALAR CONTROLLER: INTEGER
        ATAD
        @DEL := @MCT-->MCT(ELEM).DEL-ID
        @ELEM := @DEL-->DEL.SUB-ELEMENT
        EST-ENTRY := @MCT-->MCT(ELEM).ENTITY.ID
        COLOR := @MCT-->MCT(ELEM).COLOR.BASE
        IF
                @MCT-->MCT(ELEM).NEXT-STATUS.BODY = 1 [ACTIVE]
        THEN
                IF
                        EST(EST-ENTRY).TYPE = 0 [CONTROLLED]
                THEN
                        CONTROLLER := EST(EST-ENTRY).CONTROLLED-BY
                        COLOR := EST(CONTROLLER).COLOR
                        EST(EST-ENTRY).COLOR := COLOR
                ELSE
                        COLOR := EST(EST-ENTRY).COLOR [NOT CONTROLLED]
                FI
        FI
        RUN LIGHT [< SET 'HERE' COORDINATES TO @MCT.HERE >]
        LOOP := 0
        WHILE
                LOOP =< @DEL-->DEL.NO-SUB-ELEMENTS
        DO
                LOOP := LOOP + 1
                IF
                        @ELEM-->SUB-ELEMENT(LOOP).CLASS = 2
                THEN
                        < CONSTRUCT LIGHT STRING WITH COLOR AND
                        @DEL.SUB-ELEMENT(LOOP).FILL-OFFSETS >
                        RUN LIGHT [< USE CONSTRUCTED STRING >]
                FI
        OD
        @MCT-->MCT(ELEMENT).COLOR.CURRENT := COLOR
```


PROC CHANGE-TEXT (USE @MCT,ELEMENT)
DATA
SCALAR @DEL: INTEGER
SCALAR @TEXT: INTEGER
SCALAR @ELEM: INTEGER
SCALAR LOOP: INTEGER
SCALAR CHARACTER-WIDTH: INTEGER
SCALAR CHARACTER-HEIGHT: INTEGER
ATAD
@DEL := @MCT-->MCT(ELEMENT).DEL-ID
@TEXT := @MCT-->MCT(ELEMENT).TEXT-ID
@ELEM := @DEL-->DEL.SUB-ELEMENT
LOOP := 0
DO
LOOP := LOOP + 1
UNTIL
@ELEM-->SUB-ELEMENT(LOOP).CLASS = 2 (BODY)
OR LOOP > @DEL-->DEL.NO-SUB-ELEMENTS
OD
Y.POS := (@TEXT-->TEXT.LENGTH/2) * CHARACTER.WIDTH
Y.POS := @ELEM-->SUB-ELEMENT(LOOP).TEXT-OFFSET.Y - Y-POS
RUN LIGHT (< SET 'HERE' COORDINATES TO @MCT-->MCT(ELEMENT).HERE >)
IF
@MCT-->MCT(ELEMENT).NEXT-STATUS.TEXT = 0 (NO TEXT)
THEN
<CONSTRUCT LIGHT STRING TO ZAP TEXT AREA
COORDINATES FOR ZAP ARE @ELEM-->SUB-ELEMENT(LOOP).TEXT-OFFSET.X,
Y-POS, TEXT-HEIGHT AND @TEXT-->TEXT.LENGTH * CHARACTER-WIDTH.
GET COLOR FROM @MCT-->MCT(ELEMENT).COLOR.CURRENT>
RUN LIGHT (< USING CONSTRUCTED SENTENCE >)
ELSE
< CONSTRUCT LIGHT STRING FOR TEXT, COORDINATES
ARE @ELEM-->SUB-ELEMENT(LOOP).TEXT-OFFSET.X AND Y-POS.
COLOR WILL BE @MCT-->MCT(ELEMENT).TEXT-COLOR >
RUN LIGHT (< USING CONSTRUCTED SENTENCE >.)
FI
CORP
A-24

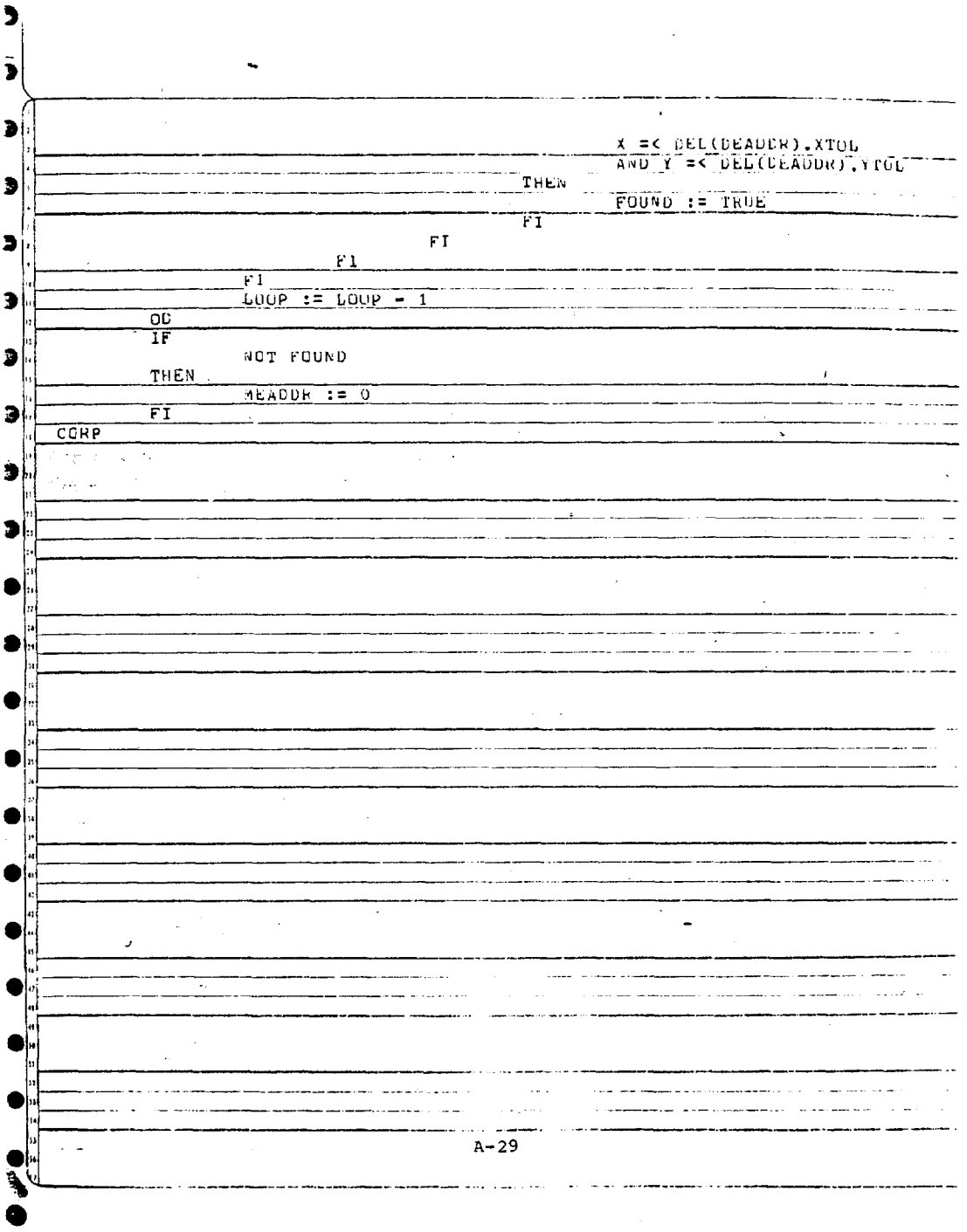
```
PROC ZAP-OUT (USE @MCT,ELEM)
        DATA
                SCALAR @DEL: INTEGER
        ATAD
        RUN LIGHT [< SET 'HERE' TO @MCT-->MCT(ELEM).HERE >]
        @DEL := @MCT-->MCT(ELEM).DEL-ID
        <CONSTRUCT LIGHT STRING TO ZAP AREA,
        COORDINATES ONE 0,0,@DEL.TOLERANCE.X AND
        @DEL-->DEL.TOLERANCE,Y, COLOR WILL BE @MCT-->MCT(ELEM).COLOR.BASE >
        RUN LIGHT [< USE CONSTRUCTED SENTENCE >]
CCRP
```

```
PROC TOUCH (USE  ELEMENT; OUT:RC)
        DATA
                SCALAR RC, ELEMENT: INTEGER
                SCALAR @MCT: INTEGER
        ATAD
        @MCT := MENU-DICT(CURR-MENU),MCT
        RC := @MCT-->MCT(ELEMENT).CURR-STATUS.BORDER
        IF
                RC = 0
        THEN
                @MCT-->MCT(ELEMENT).NEXT-STATUS.BORDER := 1
        ELSE
                @MCT-->MCT(ELEMENT).NEXT-STATUS.BORDER := 0
        FI
        RUN DRAW-ELEMENT (USE  @MCT,ELEMENT)
CORP
```

A-26

```
PROC CCISPI                [TOUCH INTERPRETER]
        DATA
                SCALAR  ON: INTEGER
                        MORE: LOGICAL
        ATAD
        ON := 1
        RUN FINDEL (ALT MEADDR)  [LOCATE TOUCHED ELEMENT]
        IF
                <ELEMENT FOUND>
        THEN
                RUN COMPRS
        FI
        <STACK ELEMENT>
        STACK(TOP).IFLAG := ON
        MORE := TRUE
        WHILE
                MORE
        DO
                IF
                        <STACK EMPTY>
                THEN
                        MORE := FALSE
                ELSE
                        IF
                                STACK(TOP).IFLAG = ON
                        THEN
                                RUN <IMMEDIATE ACTION ROUTINE>
                        ELSE
                                IF
                                        STACK(TOP).DFLAG = ON
                                THEN
                                        RUN <DELAYED ACTION ROUTINE>
                                ELSE
                                        MORE := FALSE
                                FI
                        FI
                FI
                IF
                        NOT MORE
                THEN
                        RUN MNEXT (ALT PDMENU)
                        RQMENU := PDMENU
                        RUN DRMENU (USE RQMENU)
                FI
        OD
CORP
```

```
PROC GACTAD (USE ACTION.RTN, ALT ACTION.ADDR)   [MAP ACTION ROUTINES]
        DATA
                SCALAR  ORIGIN,
                        MMS.NUMBER,
                        MMR.NUMBER: INTEGER
        ATAD
        IF
                ACTION.RTN <> 0
        THEN
                ORIGIN := MMSTAB(ACTION.RTN)
                MMS.NUMBER := MMSTAB(ACTION.RTN + 1)
                BLOCKS := MMSTAB(ACTION.RTN + 2)
                MMR.NUMBER := ORIGIN/512 - 64
                WHILE
                        BLOCKS > 0
                DO
                        RUN MRHMAP (USE MMS.NUMBER, MMR.NUMBER)
                        MMS.NUMBER := MMS.NUMBER + 1
                        MMR.NUMBER := MMR.NUMBER + 1
                        BLOCKS := BLOCKS - 1
                OD
        FI
CORP
PROC FINDEL (USE X.TOUCH, Y.TOUCH)      [FIND THE TOUCHED ELEMENT]
        DATA
                SCALAR  LOOP,
                        X,
                        Y,
                        X.TOUCH,
                        Y.TOUCH: INTEGER
                SCALAR  FOUND: LOGICAL
        ATAD
        LOOP := MDT(CURRMA).NEL
        FOUND := FALSE
        MEADDR := MDT(CURRMA).MCT - MCTLEN
        WHILE
                LOOP > 0  AND NOT FOUND
        DO
                MEADDR := MEADDR + MCTLEN
                IF
                        MCT(MEADDR).DEL <> 0    [HAS A DEL ENTRY]
                THEN
                        IF
                                MCT(MEADDR).CURRENT.INSENSITIVE = <OFF>  AND
                                MCT(MEADDR).CURRENT.INHIBIT = <OFF>
                        THEN
                                X := X.TOUCH - MCT(MEADDR).XHERE
                                Y := Y.TOUCH - MCT(MEADDR).YHERE
                                IF
                                        X >= 0 AND Y >= 0
                                THEN
                                        RUN GUELAD (USE MEADDR, ALT DEADDR)
                                IF
                                        A-28
```



```
                                                X =< DEL(DEADDR).XTOL
                                                AND Y =< DEL(DEADDR).YTOL
                                        THEN
                                                FOUND := TRUE
                                    FI
                                FI
                            FI
                        FI
                        LOOP := LOOP - 1
                OD
                IF
                        NOT FOUND
                THEN
                        MEADDR := 0
                FI
CORP
```

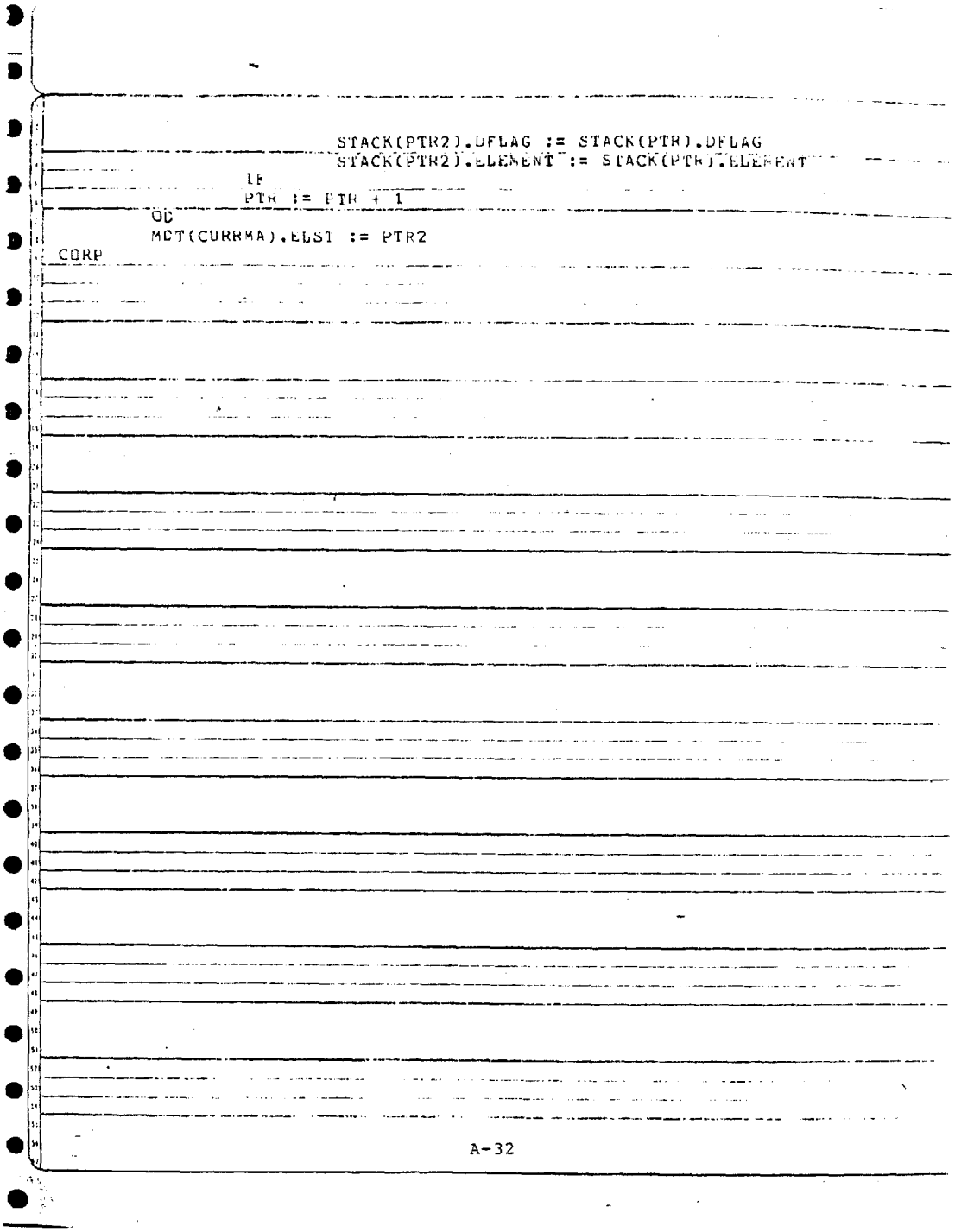
A-29

```
PROC LEXSCN      (LEXICAL SCANNER ROUTINE)
        DATA
                 SCALAR LEXRC,
                        ELEMENT: INTEGER
                 SCALAR FOUND: LOGICAL
        ATAD
        LEXRC := 0
        RUN COMPRS
        IF
                 <STACK EMPTY>
        THEN
                 LEXRC := 1
        ELSE
                 RUN SRTES
                 ELEMENT := STACK(TOP).ELEMENT
                 IF
                          MCT(ELEMENT).SC = 2 AND <NUMBER STACKED = 1>
                 THEN
                          LEXRC := 2
                          ERRMSG := 7
                          RUN GRMST
    ELSE
            IF
                     MCT(ELEMENT).SC <> 0
            THEN
                     RUN ESTADD (USE ELEMENT, ALT ESADDR)
                     IF
                              EST(ESADDR).CB <> 0
                     THEN
                              FOUND := FALSE
                              MEADDR := MDT(CURRMA).MCT
                              WHILE
                                       (NOT FOUND) AND (<MORE ELEMENTS>)
                              DO
                                       IF
                                                MCT(MEADDR).ESID = EST(ESADDR).ID
                                       THEN
                                                <STACK ELEMENT>
                                                FOUND := TRUE
                                       ELSE
                                                MEADDR := MEADDR + MCTLEN
                                       FI
                              OD
                                   ELSE
                                            LEXRC := 2
                                            ERRMSG := 8
                                            RUN GRMST
                                   FI
                          FI
                 FI
        FI
CORP
```

A-30

```
PROC COMPRS                 (COMPRESS SENTENCE STACK)
        DATA
                SCALAR  ELEMENT,
                        TOP,
                        PTR,
                        PTR2: INTEGER
                SCALAR  MORE,
                        MATCH: LOGICAL
        ATAC
        TOP := MDT(CURRMA).ELST
        MORE := TRUE
        WHILE
                MORE
        DO
                ELEMENT := STACK(TOP).ELEMENT
                PTR := TOP
                MATCH := FALSE
                WHILE
                        PTR > MDT(CURRMA).ESTK  OR  NOT MATCH
                DO
                        PTR := PTR - 1
                        IF
                                ELEMENT = STACK(PTR).ELEMENT
                        THEN
                                STACK(PTR).IFLAG = <DELETE>
                                STACK(TOP).IFLAG = <DELETE>
                                MATCH := TRUE
                        FI
                OD
                TOP := TOP - 1
                WHILE
                        STACK(TOP).IFLAG = <DELETE>  AND
                        MORE
                DO
                        TOP := TOP - 1
                        IF
                                TOP = MDT(CURRMA).ESTK
                        THEN
                                MORE := FALSE
                        FI
                OD
        OD
        PTR := MDT(CURRMA).ESTK
        PTR2 := 0
        TOP := MDT(CURRMA).ELST
        WHILE
                PTR =< TOP
        DO
                IF
                        STACK(PTR).IFLAG <> <DELETE>
                THEN
                        PTR2 := PTR2 + 1
                        STACK(PTR2).IFLAG := STACK(PTR).IFLAG
```

A-31



```
                    STACK(PTR2).DFLAG := STACK(PTR).DFLAG
                    STACK(PTR2).ELEMENT := STACK(PTR).ELEMENT
                IF
                PTR := PTR + 1
            OD
            MST(CURRMA).ELS1 := PTR2
        CORP
```

A-32

```
PROC SRTES                    [SORT SENTENCE STACK]
        DATA
                ARRAY TEMP.STACK(1:10)
                SCALAR I,SC,PTR: INTEGER
        ATAD
        I := 0
        SC := 0
        WHILE
                SC < 5
        DO
                PTR := STACK.BOTTOM
                WHILE
                        PTR < STACK.TOP
                DO
                        MEADDR := STACK(TOP).ELEMENT
                        IF
                                MCT(MEADDR).SC = SC
                        THEN
                                I := I + 1
                                TEMP.STACK(I) := STACK(PTR)
                        FI
                        PTR := PTR + 1
                OD
                SC := SC + 1
        OD
        PTR := 0
        WHILE
                PTR <= I
        DO
                STACK(PTR) := TEMP.STACK(PTR)
                PTR := PTR + 1
        OD
CORP
```

A-33

```
PROC GDELAD (USE MEADDR)          [GET DISPLAY ELEMENT ADDRESS AND MAP DEL]
     DATA
          SCALAR   DEL.ID,
                   DEL.INDEX,
                   MMS.NUMBER,
                   DEL.ORIGIN,
                   MAP.ORIGIN,
                   SDEL,              [LOCATION OF DEX AND DEL]
                   MMR.NUMBER: INTEGER
          ARRAY    DEX (1:NUMBER OF DEL ENTRIES) OF DEX-RECORD
          RECORD   DEX-RECORD
                   .ADDRESS: INTEGER
     ATAD
     DEL.ID = MCT(MEADDR).DEL
     IF
          DEL.ID = 1
     THEN
          DEADDR = LOGO.ADDR        [FROM CONF INIT TABLE]
     ELSE
          DEADDR := 0
          MMS.NUMBER := SDEL(2)
          MMR.NUMBER := MAPDEL/512 + 64
          RUN MMRMAP (USE MMS.NUMBER, MMR.NUMBER)
          MAP.ORIGIN := MOD(DEX,512) + MAPDEL
          DEL.INDEX := DEL.ID * 2 + MAP.ORIGIN
          DEADDR := DEX(DEL.INDEX).ADDRESS
          DEL.ORIGIN := MMSTAB(SDEL + 1)
          MMS.NUMBER := (DEADDR - DEL.ORIGIN)/512 + MMS.NUMBER
          MMR.NUMBER := MAPDEL/512 + 64
          RUN MMRMAP (USE MMS.NUMBER, MMR.NUMBER)
          MMS.NUMBER := MMS.NUMBER + 1
          MMR.NUMBER := MMR.NUMBER + 1
          RUN MMRMAP (USE MMS.NUMBER, MMR.NUMBER)
          DEADDR := MOD(DEADDR,512) + MAPDEL
     FI
CORP
```

```
PROC GTELAD (USE MEADDR)          [GET TEXT ELEMENT ADDRESS AND MAP TEL]
        DATA
                SCALAR  TEL.ID,
                        TEL.INDEX,
                        MMS.NUMBER,
                        TEL.ORIGIN,
                        MAP.ORIGIN,
                        STEL,           [LOCATION OF TEX AND TEL]
                        MMR.NUMBER: INTEGER
                ARRAY   TEX (1:NUMBER OF TEL ENTRIES) OF TEX-RECORD
                RECORD  TEX-RECORD
                        .ADDRESS: INTEGER
        ATAD

        TLADDR := 0
        MMS.NUMBER := STEL(2)
        MMR.NUMBER := MAPTEL/512 + 64
        RUN MMRMAP (USE MMS.NUMBER, MMR.NUMBER)
        MAP.ORIGIN := MOD(TEX,512) + MAPTEL
        TEL.ID := MCT(MEADDR).TEL
        TEL.INDEX := TEL.ID * 2 + MAP.ORIGIN
        TLADDR := TEX(TEL.INDEX).ADDRESS
        TEL.ORIGIN := MMSTAB(STEL + 1)
        MMS.NUMBER := (TLADDR - TEL.ORIGIN)/512 + MMS.NUMBER
        MMR.NUMBER := MAPTEL/512 + 64
        RUN MMRMAP (USE MMS.NUMBER, MMR.NUMBER)
        MMS.NUMBER := MMS.NUMBER + 1
        MMR.NUMBER := MMR.NUMBER + 1
        RUN MMRMAP (USE MMS.NUMBER, MMR.NUMBER)
        TLADDR := MOD(TLADDR,512) + MAPTEL
CORP
```

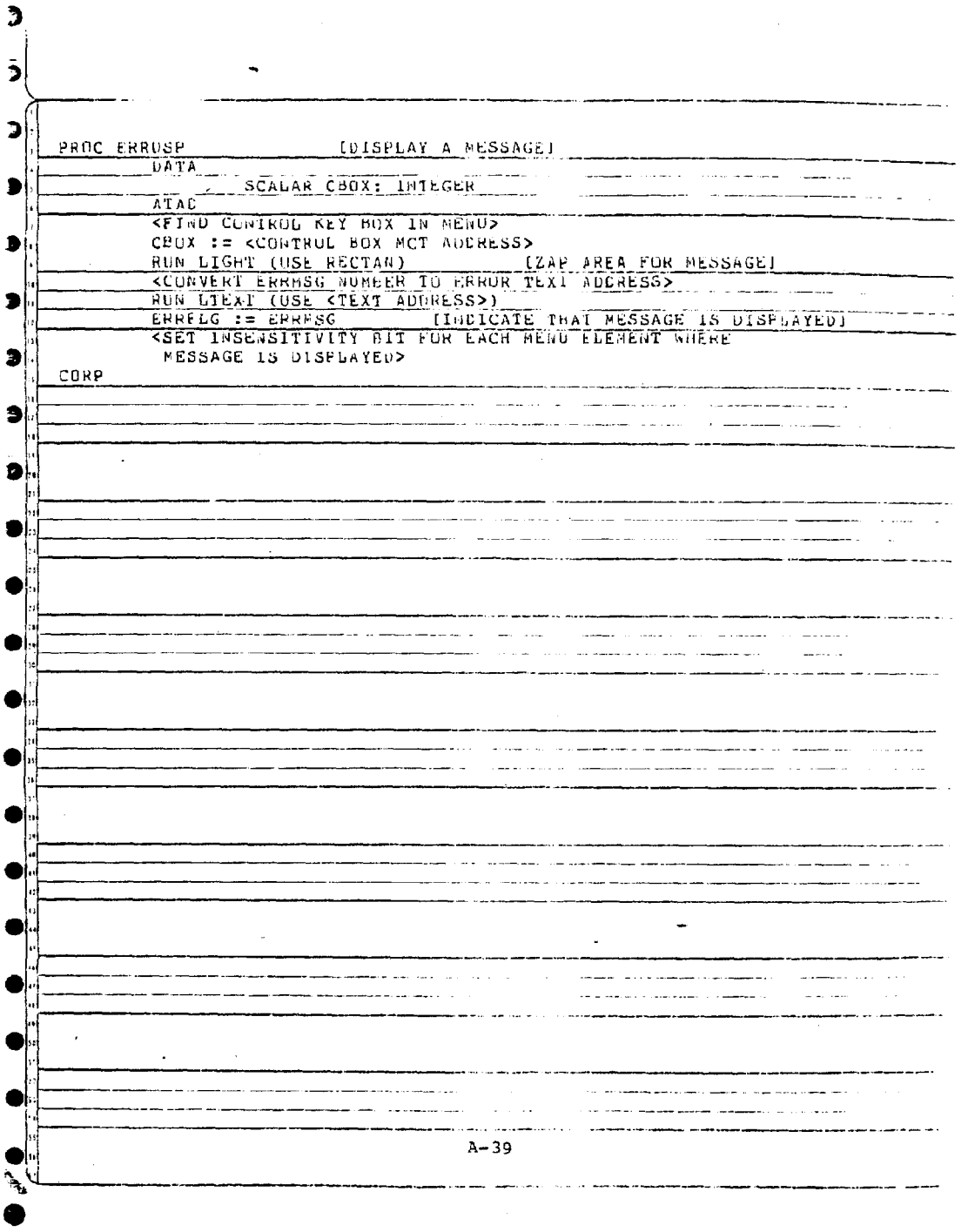
```
PROC GMCTAD (USE MCADDR)          [DO MCT MAPPING AND COMPUTE ADDRESSES]
        DATA
                SCALAR  MCT.ORIGIN,
                        MMS.NUMBER,
                        MMR.NUMBER,
                        SMCTF,          [MCT MAPPED LOCATION FROM LINKER]
                        LOOP: INTEGER
        ATAD
        MCT.ORIGIN := MMSTAB(SMCTF)
        MMS.NUMBER := MMSTAB(SMCTF+1)
        MMS.NUMBER := (MCADDR - MCT.ORIGIN)/512 + MMS.NUMBER
        MMR.NUMBER := MAPMCT/512 - 64
        LOOP := 4
        WHILE
                LOOP >0
        DO
                RUN MMRMAP (USE MMS.NUMBER, MMR.NUMBER)
                MMS.NUMBER := MMS.NUMBER + 1
                MMR.NUMBER := MMR.NUMBER + 1
                LOOP := LOOP -1
        OD
        MCADDR := MOD(MCADDR,512) + MAPMCT
CORP
```

A-36

```
PROC MMRMAP (USE MMS, MMR)        [PERFORM MEMORY MAPPING]
     DATA
          SCALAR   LOOP: INTEGER
          ARRAY    MAPTAB (1:32) OF MAPTAB-ENTRY
          RECORD   MAPTAB-ENTRY
                   .MMS
     ATAD
     IF
          MMS <> 0
     THEN
          MMS := (MMS/8)*16 + MOD(MMS,8)
          MMS := INVERT(MMS)              [1'S COMPLIMENT]
          >F002 := MMR
          >F004 := MMS
          MAPTAB(MMR).MMS := MMS
     ELSE
          LOOP := 0
          WHILE
               LOOP < 32
          DO
               IF
                    MAPTAB(LOOP).MMS <> 0
               THEN
                    >F002 := LOOP
                    >F004 := MAPTAB(LOOP).MMS
               FI
               LOOP := LOOP + 1
          OD
     FI
CORP
```

```
PROC GRRST (USE FLAG, ERR.NO)
     DATA
          SCALAR  ERR.NO,
          SAVEGO: INTEGER
     ATAD
     IF
          ERRMSG <> 0  OR  ERR.NO <> 0
     THEN
          <LOCATE GO KEY ELEMENT IN MENU>
          SAVEGO := <MCT ADDRESS OF GO KEY>
          IF
               FLAG = <RESET>
          THEN
               IF
                    MCT(SAVEGO).NEXT.BODY <> <OFF>
               THEN
                    MCT(SAVEGO).NEXT.BODY = <OFF>
                    RUN CBODY (USE SAVEGO)
                    MCT(SAVEGO).CURRENT := MCT(SAVEGO).NEXT
               ELSE
                    RUN ERRERS      (ERASE ERROR MSG)
                    ERRMSG := 0
               FI
          ELSE
               ERRMSG := MSG.NO
               IF
                    <GO INDICATOR IS OFF>
               THEN
                    <SET GO INDICATOR>
               FI
               IF
                    <A MSG ALREADY DISPLAYED>
               THEN
                    RUN ERRERS
               FI
          FI
     FI
CORP
```

A-38



```
PROC ERRDSP                [DISPLAY A MESSAGE]
     DATA
          SCALAR CBOX: INTEGER
     ATAC
     <FIND CONTROL KEY BOX IN MENU>
     CBOX := <CONTROL BOX MCT ADDRESS>
     RUN LIGHT (USE RECTAN)          [ZAP AREA FOR MESSAGE]
     <CONVERT ERRMSG NUMBER TO ERROR TEXT ADDRESS>
     RUN LTEXT (USE <TEXT ADDRESS>)
     ERRFLG := ERRMSG          [INDICATE THAT MESSAGE IS DISPLAYED]
     <SET INSENSITIVITY BIT FOR EACH MENU ELEMENT WHERE
      MESSAGE IS DISPLAYED>
CORP
```

```
PROC ERREKS                    [ERASE A DISPLAYED MESSAGE]
        DATA
                SCALAR  LOOP: INTEGER
                        FOUND: LOGICAL
        ATAD
        IF
                ERRFLG <> 0
        THEN
                RUN LIGHT (USE RECTAN)          [ZAP MESSAGE AREA]
                LOOP := 3
                ELEMENT := CBOX
                FOUND := FALSE
                WHILE
                        LOOP >0  OR  NOT FOUND
                DO
                        ELEMENT := ELEMENT + MCILEN
                IF
                        IF
                                MCT(ELEMENT).DEL = GOKEY
                        THEN
                                FOUND := TRUE
                        ELSE
                                MCT(ELEMENT).CURRENT := 0
                                MCT(ELEMENT).NEXT := MCT(ELEMENT).DEFAULT
                                RUN DREL (USE ELEMENT)  [REDRAW ELEMENT]
                        FI
                        LOOP := LOOP - 1
                OD
                ERRFLG := 0
        FI
CORP
```

```
PROC GMSGAD (USE RQBID, ALT RQB)          GET RQB MESSAGE ADDRESS
        DATA
                SCALAR  INDEX: INTEGER
        ATAC
        INDEX := RQBID * 2 + RQBLKS
        RQB := RQBREC(INDEX)
CORP
```

A-41

```
PROC MSGDSP (USE_FLAG, MSG.NO)
        DATA
                SCALAR  MSG.NO,
                        MSG.ADDR,
                        POS: INTEGER
                SCALAR  MSG024: STRING(66)
                SCALAR  FINISHED: LOGICAL
        ATAD
        IF
                MSG.NO > 4000
        THEN                            (MSG.NO IS AN ADDRESS)
                MSG.ADDR := MSG.NO
                MSG.NO := 024
                POS := 0
                NUM.CHAR := 0
                FINISHED := FALSE
                WHILE
                        NOT FINISHED
                DO
                        MSG024(POS) := MSG.ADDR(POS)
                        IF
                                MSG024(POS) = 0  OR
                                POS > 66
                        THEN
                                FINISHED := TRUE
                        ELSE
                                POS := POS + 1
                        FI
                OD
        FI
        IF
                FLAG <> 0
        THEN
                RUN GRHST (USE 1)
        ELSE
                RUN ERRDSP
        FI
CORP
```

```
PROC RESET                    [GENERAL CANCEL FUNCTION]
        DATA
                SCALAR  PTR: INTEGER
        ATAD
        PTR := MDT(CURRMA).ELST          [TOP ELEMENT ON STACK]
        WHILE
                PTR <> MDT(CURRMA).ESTK
        DO
                MEADDR := STACK(PTR).ELEMENT     [GET STACKED ELEMENT]
                IF
                        MCT(MEADDR).CURRENT.BORDER = <ON>
                THEN
                        RUN TOUCH (USE MEADDR)
                FI
                PTR := PTR - 1
        OD
        IF
                CAMTCH = 0                [CAMERA IN LAST SENTENCE?]
        THEN
                MEADDR = CAMTCH + MCTLEN          [POINT TO FIRST PRESET]
                WHILE
                        CAMLOP > 0
                DO
                        IF
                                MCT(MEADDR).CURRENT.BODY = <OFF>
                        THEN
                                RUN ESTADD (USE MEADDR, ALT ESADDR)
                                EST(ESADDR).ON := <OFF>
                                EST(ESADDR).FPRE := <OFF>
                                EST(ESADDR).CE := 0
                                RUN GDELAD (USE MEADDR, ALT DEADDR)
                                RUN ZAPOUT (USE DEADDR)
                                MCT(MEADDR).NEXT := MCT(MEADDR).DEFAULT
                                MCT(MEADDR).CURRENT := MCT(MEADDR).DEFAULT
                        FI
                        CAMLOP := CAMLOP - 1
                OD
        FI
        CAMTCH := 0
        CURPRE := 0
        STACK.TOP := STACK.BOTTOM
        RUN GRMST (USE 0)          [ERASE MESSAGES]
CORP
```

```
PROC PREVUE
        DATA
                SCALAR PTR, PREVIEW.ID: INTEGER
        ATAD
        PTR := STACK.TOP
        PREVIEW.ID := 0
        WHILE
                (PTR > STACK.BOTTOM) AND (PREVIEW.ID = 0)
        DO
                MEADDR := STACK(PTR).ELEMENT
                RUN ESTADD (USE MEADDR, ALT ESADDR)
                IF
                        ESADDR <> 0
                THEN
                        IF
                                EST(ESADDR).TYPE = 1
                        THEN
                                PREVIEW.ID := EST(ESADDR).ID
                        ELSE
                                IF
                                        EST(ESADDR).CB <> 0
                                THEN
                                        PREVIEW.ID := EST(ESADDR).CB
                                FI
                        FI
                FI
        OD
        IF
                PREVIEW.ID <> 0
        THEN
                <SEND PREVIEW MESSAGE>
                PREVIEW.FLAG := OFF
        FI
CORP
```

```
PROC SLCTME (USE MEADDR)      (SELECT MUTUALLY EXCLUSIVE)
        DATA
                SCALAR CONFLICT, PTR: INTEGER
        ATAD
        RUN ESTADD (USE MEADDR, ALT ESADDR)
        CONFLICT := EST(ESADDR).CONF
        IF
                CONFLICT <> 0
        THEN
                PTR := STACK.BOTTOM
                WHILE
                        PTR <= STACK.TOP
                DO
                        ELEMENT := STACK(PTR).ELEMENT
                        RUN ESTADD (USE ELEMENT, ALT ESADDR)
                        IF
                                CONFLICT <> EST(ESADDR).CONF
                        THEN
                                RUN TOUCH (USE ELEMENT) (RETOUCH KEY IN CONFLICT)
                                STACK(PTR).IFLAG := >FF
                        FI
                        PTR := PTR + 1
                OD
        FI
        IF
                PREVIEW.FLAG = ON
        THEN
                RUN PREVUE
        FI
CORP
```

```
PROC STDFLT              [SET MASTER MENU DEFAULTS]
        DATA
                SCALAR SENTENCE, I, J, PTR: INTEGER
                SCALAR FOUND, MORE: LOGICAL
        ATAD
        SENTENCE := 0
        MORE := TRUE
        WHILE
                MORE
        DO
                SENTENCE := SENTENCE + 10
                I := SENTENCE
                J := 0
                FOUND := FALSE
                WHILE
                        (J <= 9) OR (NOT FOUND)
                DO
                        I = I + J
                        PTR := 0
                        WHILE
                                (PTR <= ESTENT) AND (NOT FOUND)
                        DO
                                IF
                                        EST(PTR).INIT = I
                                THEN
                                        RUN FINLIT (USE EST(PTR).ID)
                                        IF
                                                FOUND
                                        THEN
                                                <STACK ELEMENT>
                                                STACK(TOP).IFLAG := ON
                                                RUN EXECIT
                                        FI
                                ELSE
                                        PTR := PTR + 1
                                FI
                        OD
                        J := J + 1
                OD
                IF
                        J > 1
                THEN
                        RUN FINDIT (USE 0)
                        RUN EXECIT
                ELSE
                        MORE := FALSE
                FI
        OD
CORP
```

A-46

```
PROC LSENDS          [DISCONNECT LOOSE ENDS]
        DATA
                SCALAR SOURCE: INTEGER
        ATAD
        ESADDR := 0
        WHILE
                ESADDR <= ESTENT  [THIS LOOP DISCONNECTS ALL SINKS]
                                  [WHOSE SOURCE HAS BEEN DISCONNECTED, AND]
                                  [DISCONNECTS ALL SOURCES WHICH HAVE NO SINKS]
        DO
                IF
                        EST(ESADDR).TYPE = 1
                THEN
                        SOURCE := EST(ESADDR).ID
                        IF
                                EST(ESADDR).FCN = <ON>
                        THEN
                                <LOOP THROUGH THE EST LOOKING FOR A SINK
                                WHICH IS CONTROLLED BY THIS SOURCE.  IF NO
                                SINKS ARE FOUND, THEN SET EST(ESADDR).FCN
                                TO <OFF>>
                        FI
                        IF
                                EST(ESADDR).FCN = <OFF>
                        THEN
                                <LOOP THROUGH THE EST LOOKING FOR ANY SINK
                                WHICH IS CONTROLLED BY THIS SOURCE.  WHEN
                                FOUND, SET EST FIELDS FCN, CB, COLOR TO OFF
                                AND SEND THE DISCONNECT MESSAGES>
                        FI
                FI
                ESADDR := ESADDR + 1
        OD
CORP
```

```
PROC BACK                    [BACK KEY, ALL MENUS]
        DATA
                SCALAR SAVE: INTEGER
        ATAD
        RUN RESET
        RQMENU := MDT(CURRMA).PM                    [RQMENU = PREVIOUS MENU]
        RUN GMADDR (USE RQMENU, ALT MENUAD)         [GET NEW MENU MDT ADDRESS]
        SAVE := MDT(MENUAD).PM                      [SAVE NEW MENU PREVIOUS MENU]

        <TO PRE      DRMENU ALWAYS SETS THE NEW MENUS "PREVIOUS MENU" FIELD TO
        <BE THE MENU ID OF THE LAST MENU DRAWN.  TO AVOID GETTING>
        <INTO A MENU LOOP WHEN THE BACK KEY IS USED, WE LET THE    >
        <MENU WHICH WE ARE GOING BACK TO KEEP ITS EXISTING PREVIOUS>
        <MENU FIELD.>

        RUN DRMENU
        MDT(MENUAD).PM := SAVE
CORP
```

```
PROC HELP                    [HELP KEY, ALL MENUS]
        DATA
        ATAD
        <POP THE STACK>
        IF
                ERRMSG <> 0     AND
                ERRMSG <> ERRFLG
        THEN
                RUN ERRDSP
        ELSE
                IF
                        <STACK IS EMPTY>
                THEN
                        PDMENU := MDT(CURRMA).HELP
                ELSE
                        PDMENU := MCT(MEADDR).HELP
                FI
                IF
                        PDMENU = 0
                THEN
                        <DISPLAY NO HELP AVAILABLE MSG>
                ELSE
                        RUN GRMST (USE 0)
                FI
        FI
CORP
```

A-49

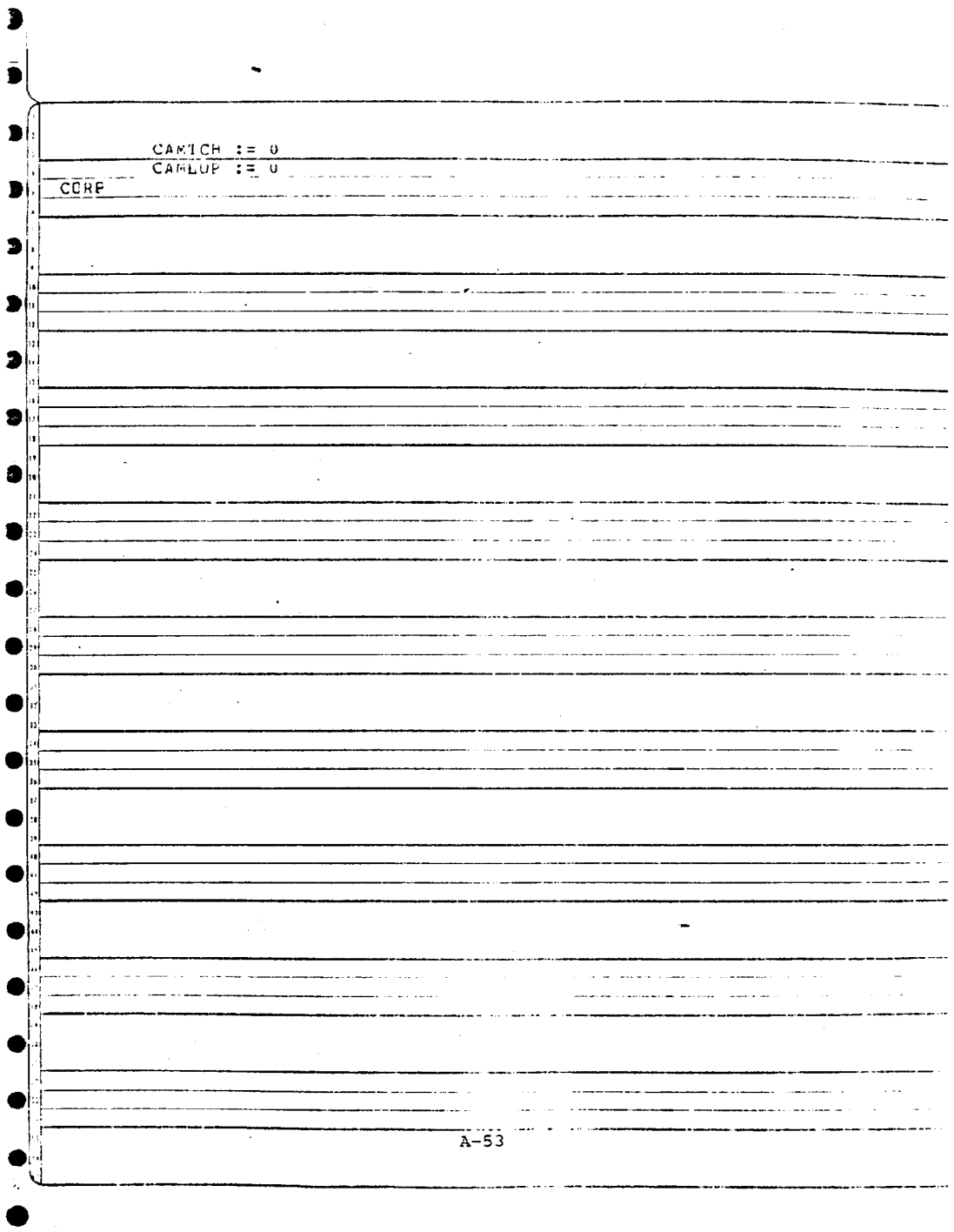
```
PROC FINDIT (USE ID)    [FIND MCT ELEMENT WITH ESID OF ID]
        DATA
                SCALAR FOUND: LOGICAL
                SCALAR ID: INTEGER
        ATAD
        MEADDR := MDT(C03MNO).MCT
        FOUND := FALSE
        WHILE
                <MORE ELEMENTS> AND (NOT FOUND)
        DO
                IF
                        ID = 0
                THEN
                        IF
                                MCT(MEADDR).DEL = <GO KEY>
                        THEN
                                FOUND := TRUE
                        ELSE
                                MEADDR := MEADDR + MCTLEN
                        FI
                ELSE
                        IF
                                MCT(MEADDR).ESID = ID
                        THEN
                                FOUND := TRUE
                        FI
                FI
        OD
CORP
```

```
PROC EXECIT                [EXECUTE DEFAULT STACK]
        DATA
                SCALAR MORE: LOGICAL
        ATAD
        MORE := TRUE
        WHILE
                MORE
        DO
                MORE := FALSE
                IF
                        STACK(TOP).IFLAG = <ON>
                THEN
                        RUN <IMMEDIATE ACTION ROUTINE>
                        MORE := TRUE
                ELSE
                        IF
                                STACK(TOP).DFLAG = <ON>
                        THEN
                                RUN <DELAYED ACTION ROUTINE>
                                MORE := TRUE
                        FI
                FI
        OD
CORP
```

A-51

```
PROC ZAPCAM (USE FLAG)    [ERASE A TOUCHED OR ACTIVE CAMERA]
        DATA
                SCALAR  FLAG:   LOGICAL
                SCALAR  LOOP:   INTEGER
                SCALAR  ELEM:   INTEGER
        ATAD
        IF
                CAMTCH = 0
        THEN
                <RETURN>
        CURPRE := 0                 [SET CURRENT PRESET OFF]
        IF
                CAMTCH < 5000   [CAMERA IS NOT AN ADDRESS]
        THEN
                RUN FINDIT (USE CAMTCH, MEADDR)
                <POP STACK>
        FI
        RUN COMPRS
        RUN ESTADD      (USE MEADDR, ALT ESADDR)
        IF
                FLAG = 1
        THEN
                EST(ESADDR).FON := 0
        FI
        LOOP := EST(ESADDR).FLD2        [GET NUMBER OF CAMERA PRESETS]
        ELEM := CAMTCH
        WHILE
                LOOP > 0
        DO
                IF
                        MCT(ELEM).CURRENT.BORDER = <ON>
                THEN
                        MCT(ELEM).CURRENT.BORDER := <OFF>
                        RUN REFSTK (USE ELEM)
                FI
                ELEM := ELEM + MCTLEN
                IF
                        LOOP <> 0
                THEN
                        RUN ESTADD (USE ELEM, ALT ESADDR)
                        IF
                                FLAG = 1 OR EST(ESADDR).FON = 0
                        THEN
                                EST.(ESADDR).FON := 0
                                EST.(ESADDR).FPRE := 0
                                EST.(ESADDR).CB := 0
                                EST.(ESADDR).COLOR := 0
                                MCT(ELEM).NEXT.INSENSITIVE := <ON>
                        FI
                FI
                LOOP := LOOP - 1
        OD
        RUN CAMRIN
```

A-52


CAMICH := 0
CAMLUP := 0
CORP
A-53

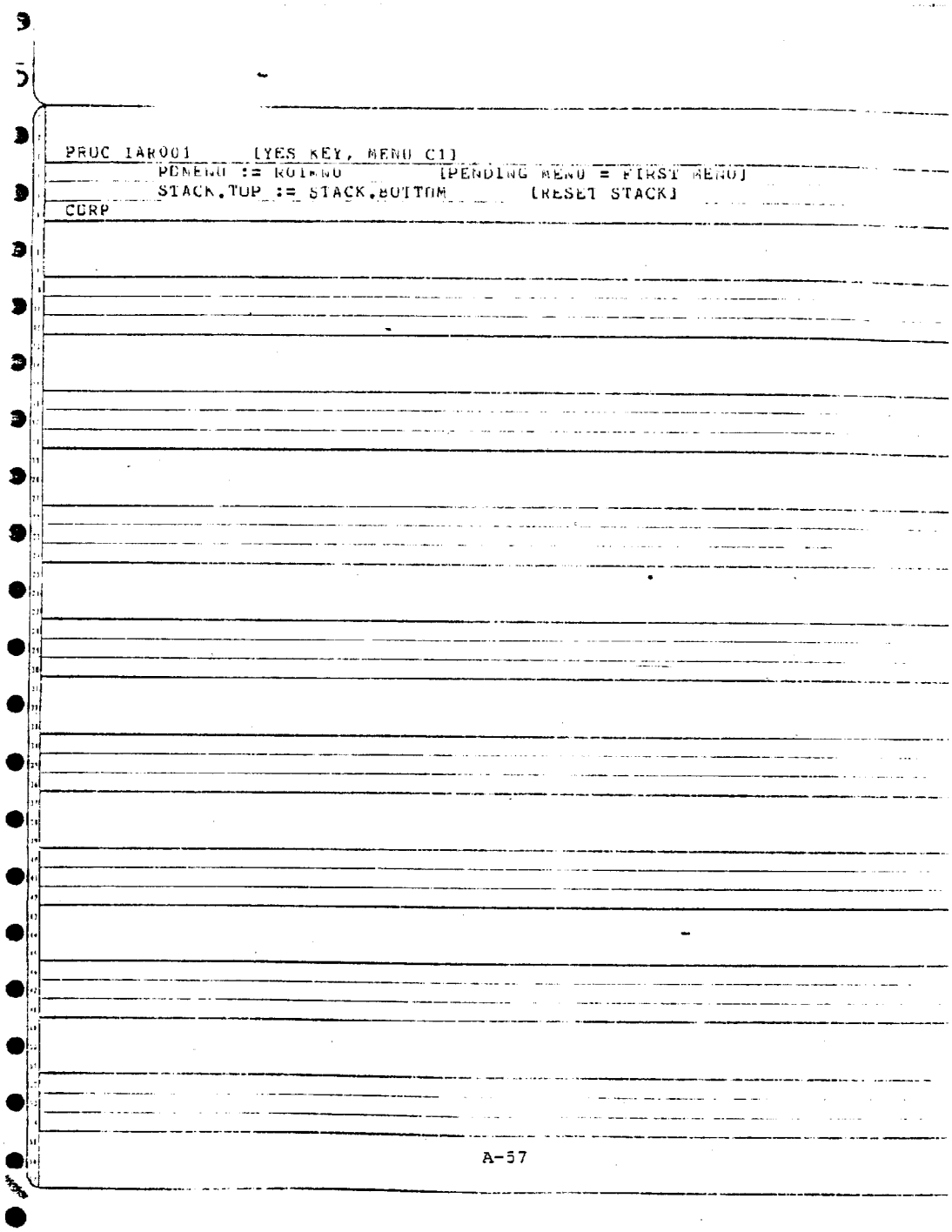
```
PROC CAMRTN     [RETURN CAMERA TO LAST ACTIVE PRESET]
        DATA
                SCALAR  LOOP: INTEGER
                SCALAR  ELEM: INTEGER
        ATAD
        IF
                CAMTCH <> 0
        THEN
                LOOP := CAMLOP
                ELEM := CAMTCH
                WHILE
                        LOOP > 0
                DO
                        ELEM := ELEM + MCTLEN
                        IF
                                MCT(ELEM).CURRENT.BODY = <OFF>
                        THEN
                                LOOP := LOOP - 1
                        ELSE
                                RUN ESTADD (USE ELEM, ALT ESADDR)
                                RQBNO := 5
                                RUN GMSGAD (USE RQBNO, ALT RQB)
                                RQB(7) := EST(ESADDR).FLD1
                                RUN ESTADD (USE CAMTCH, ALT ESADDR)
                                RQB(4) := EST(ESADDR).FLD1
                                RUN SYSREQ (USE RQB)
                        FI
                OD
        FI
CORP
```

```
PROC REFSTK (USE ELEM)     [REMOVE ELEM FROM STACK]
        DATA
                SCALAR  PTR: INTEGER
        ATAD
        PTR := STACK.BOTTOM
        WHILE
                PTR <> STACK.TOP
        DO
                PTR := PTR - 4
                IF
                        STACK(PTR).ELEMENT = ELEM
                THEN
                        STACK(PTR).IMMEDIATE := <DELETE>
                        PTR := STACK.TOP
                FI
        OD
CORP
```

A-55

```
PROC CHELEN      [CHANGE SINGLE MENU ELEMENT]
          RUN DRED2 (USE CHELM)
CORP
```

A-56


PROC 1AR001 [YES KEY, MENU C1]
PDMENU := ROTMNU [PENDING MENU = FIRST MENU]
STACK.TOP := STACK.BOTTOM [RESET STACK]
CORP
A-57

```
PROC IAR002     (NO KEY, MENU C1)
        PDMENU := CO2MNU        (PENDING MENU = SECOND CONTROL MENU)
        STACK.TOP := STACK.BOTTOM       (RESET STACK)
CORP
```

A-58

```
PROC IAR003      [BACK KEY, GENERAL MENUS]
        <POP STACK>
        RUN BACK
CORP
```

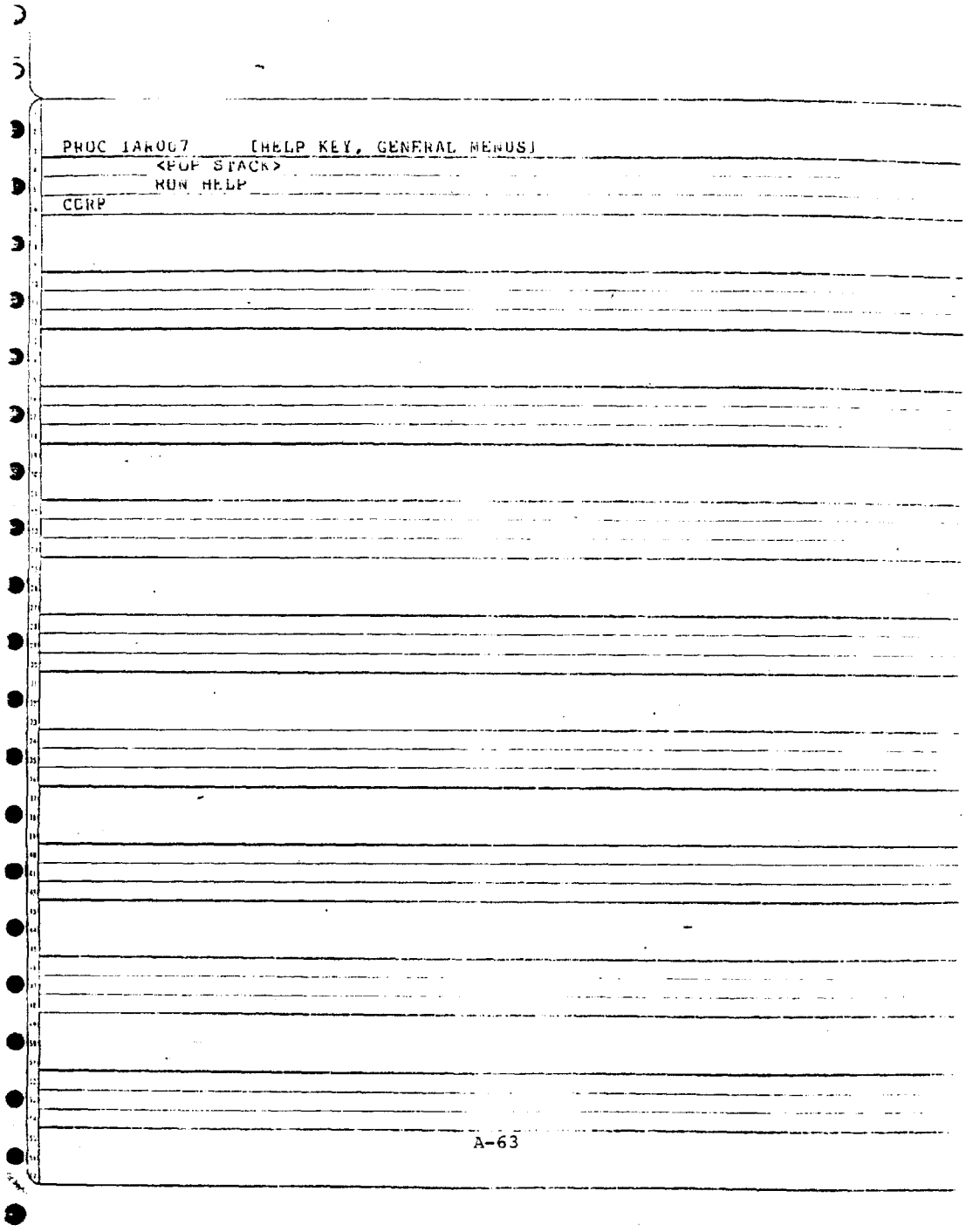
A-59

```
PROC IAR004      [GO KEY, MENUS C2,R1,R2 AND R3]
        <POP STACK>
        IF
                ERRMSG = <OR>    [ERROR PENDING]
        THEN
                RUN GRMST        [RESET ERROR]
        FI
        RUN MNEXT                [GET NEXT MENU NUMBER]
        IF
                C03DRW <> 0
        THEN
                IF
                        PDMENU = R03MNU
                THEN
                        PDMENU := C03MNU         [SET TO MASTER MENU]
                FI
        FI
CORP
```

```
PROC LAB005      [TOUCH ELEMENT KEY]
        MEADDR := STACK(STACK.TOP).ELEMENT
        RUN SLCTME          [DO MUTUAL EXCLUSIVE TEST]
        RUN TOUCH (USE MEADDR)   [TOUCH ELEMENT]
        STACK(STACK.TOP).TFLAG := <OFF>
CCRP
```

A-61

```
PROC 1AR006      [X (CANCEL) KEY, MENU C2]
     IF
                 ERRMSG = <ON>   [ERROR PENDING]
     THEN
                 <POP STACK>
     ELSE
                 PDMENU := C06MNU [GO TO MENU C6]
                 STACK.TOP := STACK.BOTTOM
     FI
CORP
```


PROC 1AR007 [HELP KEY, GENERAL MENUS]
<POP STACK>
RUN HELP
CORP
A-63

```
PROC 1AR008     [GO KEY, MENUS C3 AND C4]
        <POP STACK>
        IF
                ERRMSG = <ON>
        THEN
                RUN GRMST (USE 0)       [RESET ERROR]
        FI
        IF
                <STACK IS NOT EMPTY>
        THEN
                RUN LEXSCN
                IF
                        LEXRC = 0
                THEN
                        STACK(STACK.TOP).DFLAG := <ON>
                FI
        ELSE
                IF
                        <MENU = C03MNU>
                THEN
                        R1 := 6         [SET ERROR NUMBER]
                        R0 := 1
                        RUN GRMST (USE R0,R1)   [INDICATE ERROR]
                ELSE
                        R1 := 10        [SET ERROR NUMBER]
                        R0 := 1
                        RUN GRMST (USE R0,R1)   [INDICATE ERROR]
                FI
        FI
CORP
```

```
PROC JAK009      [X KEY, MENUS R1, R2 AND R3]
        <POP STACK>
        IF
                ERRMSG = <ON>   [ERROR PENDING]
        THEN
                RUN GRMST (USE 0)       [RESET ERROR]
        ELSE
                STACK.TOP := STACK.BOTTOM       [RESET STACK]
                IF
                        C03DRW = <OFF>  [MENU C3 HAS NOT BEEN DISPLAYED]
                THEN
                        PDMENU := C02MNU
                ELSE
                        PDMENU := C03MNU
                FI
        FI
CORP
```

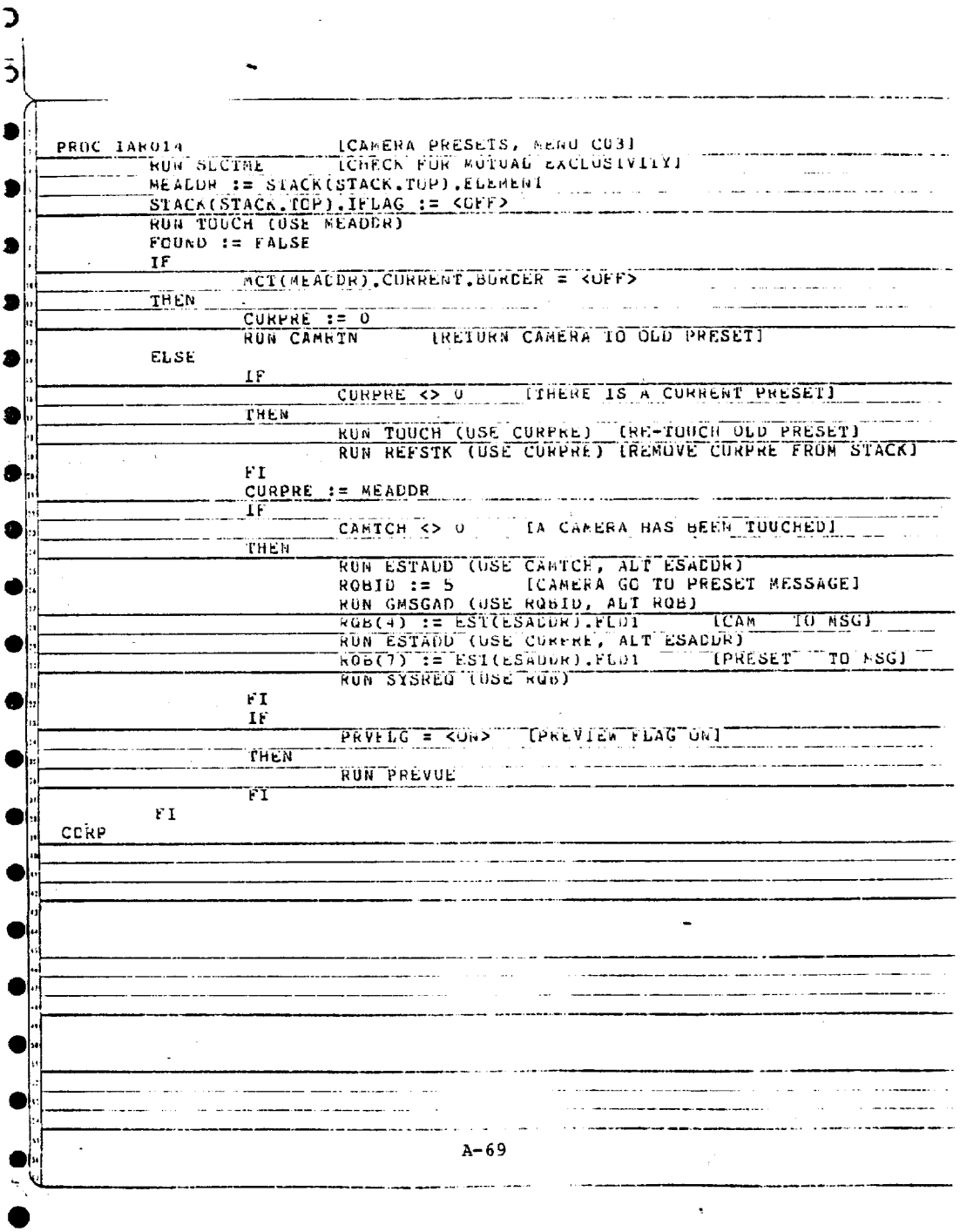
```
PROC IAR010      [X KEY, MENUS C4, C5, C6 AND ALL HELPS]
        <POP STACK>
        IF
                ERRMSG = <ON>   [ERROR IS PENDING]
        THEN
                RUN GRMST (USE 0)        [RESET ERROR]
        FI
CORP
```

A-66

```
PROC IAR011     [FRONT, LEFT AND RIGHT CAMERAS]
        DATA
                SCALAR  LOOP,
                        PRESET,
                        CONTROL: INTEGER
        ATAD
        RUN SLCTME      [ELIMINATE MUTUALLY EXCLUSIVE TOUCHES]
        MEADDR := STACK(STACK.TOP).ELEMENT
        IF
                MEADDR = CAMTCH         [A RE-TOUCH]
        THEN
                RUN ZAPCAM              [ERASE THIS CAMERA]
                RUN RDMENU              [REDRAW THE MENU]
                <RETURN>
        FI
        RUN ESTADD (USE MEADDR, ALT ESADDR)
        LOOP := EST(ESADDR).FLD1        [GET NUMBER OF CAMERA PRESETS]
        CAMLOP := LOOP          [SAVE NUMBER OF PRESETS]
        CONTROL := EST(ESADDR).ID
        RUN TOUCH (USE MEADDR)
        WHILE
                LOOP > 0        [MORE PRESETS]
        DO
                MEADDR := MEADDR + MCTLEN       [POINT TO NEXT PRESET]
                RUN ESTADD (USE MEADDR, ALT ESADDR)
                IF
                        EST(ESADDR).FLD2 <> 0   [PRESET HAS BEEN SET]
                THEN
                        EST(ESADDR).PRESENT := <ON>
                        EST(ESADDR).CB := CONTROL
                FI
                LOOP := LOOP - 1
        OD
        RUN RDMENU      [REDRAW THE MENU]
CORP
```

```
PROC 1AR013     [CCM KEY(PREVIEW), MENU C03]
        STACK(STACK.TOP).IFLAG := <OFF>
        MEADDR := STACK(STACK.TOP).ELEMENT
        RUN TOUCH (USE MEADDR)
        IF
                PRVFLG = <ON>   [PREVIEW IS ALREADY ON]
        THEN
                PRVFLG := <OFF>
        ELSE
                PRVFLG := MEADDR
                RUN PREVUE
        FI
CCRP
```

A-68

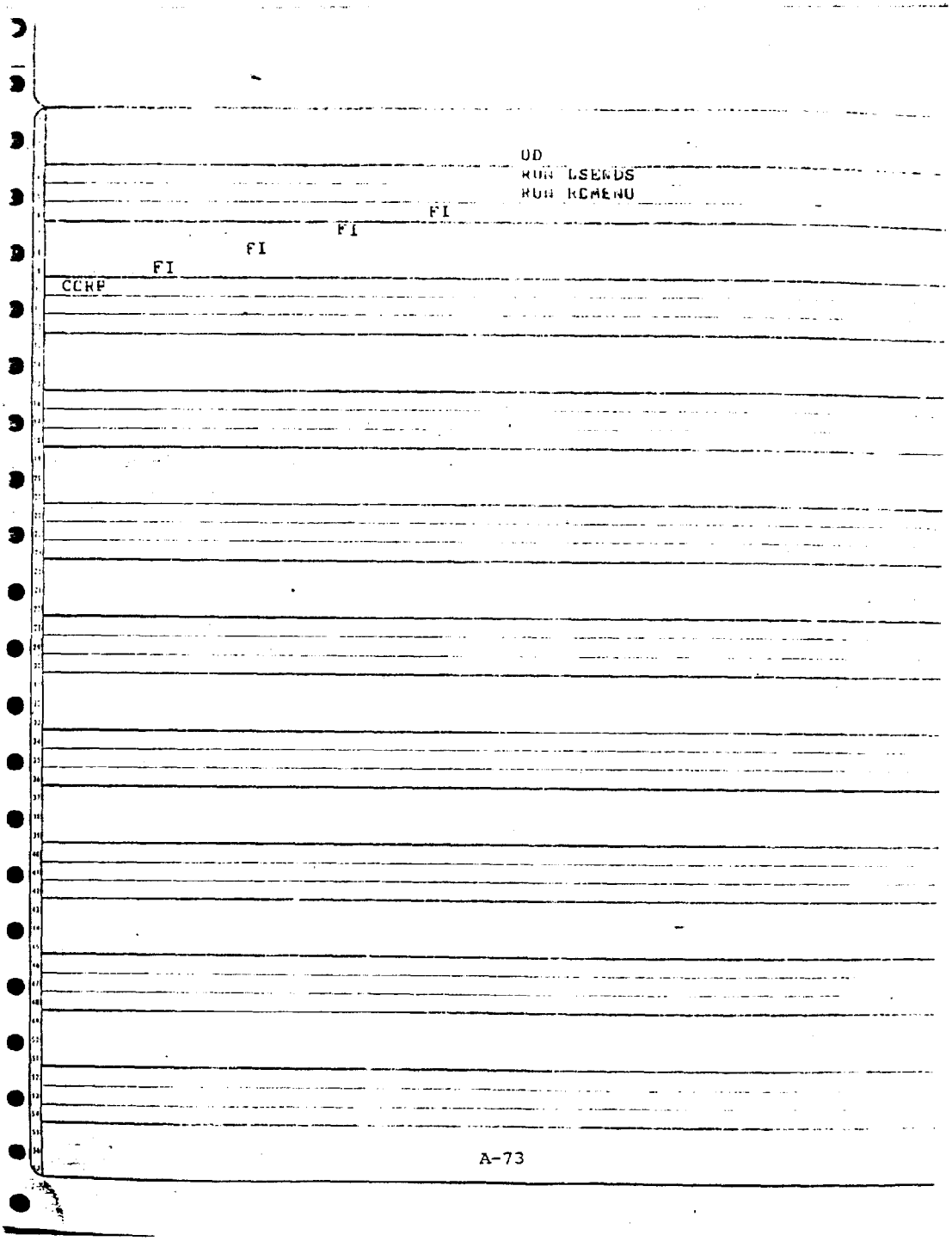


```
PROC IAK014                 [CAMERA PRESETS, MENU C03]
        RUN SLCTME          [CHECK FOR MUTUAL EXCLUSIVITY]
        MEADDR := STACK(STACK.TOP).ELEMENT
        STACK(STACK.TOP).IFLAG := <OFF>
        RUN TOUCH (USE MEADDR)
        FOUND := FALSE
        IF
                MCT(MEADDR).CURRENT.BORDER = <OFF>
        THEN
                CURPRE := 0
                RUN CAMRTN          [RETURN CAMERA TO OLD PRESET]
        ELSE
                IF
                        CURPRE <> 0         [THERE IS A CURRENT PRESET]
                THEN
                        RUN TOUCH (USE CURPRE)  [RE-TOUCH OLD PRESET]
                        RUN REFSTK (USE CURPRE) [REMOVE CURPRE FROM STACK]
                FI
                CURPRE := MEADDR
                IF
                        CAMTCH <> 0         [A CAMERA HAS BEEN TOUCHED]
                THEN
                        RUN ESTADD (USE CAMTCH, ALT ESADDR)
                        ROBID := 5          [CAMERA GO TO PRESET MESSAGE]
                        RUN GMSGAD (USE RQBID, ALT RQB)
                        RQB(4) := EST(ESADDR).FLD1          [CAM   TO MSG]
                        RUN ESTADD (USE CURPRE, ALT ESADDR)
                        RQB(7) := EST(ESADDR).FLD1          [PRESET  TO MSG]
                        RUN SYSREQ (USE RQB)
                FI
                IF
                        PRVFLG = <ON>   [PREVIEW FLAG ON]
                THEN
                        RUN PREVUE
                FI
        FI
CORP
```

```
PROC 1AR015     [OVERHEAD TV MONITORS, MENU C03]
        IF
                <STACK IS FULL>
        THEN
                RUN COMPRS
        FI
        MEADDR := STACK(STACK.TOP).ELEMENT
        STACK(STACK.TOP).IFLAG := <OFF>
        RUN TOUCH (USE MEADDR)
        IF
                MCT(MEADDR).ID <> 27    [LEFT MONITOR IS 27]
        THEN
                MEADDR := MEADDR - MCTLEN       [POINT TO LEFT TV MONITOR]
        ELSE
                MEADDR := MEADDR + MCTLEN       [POINT TO RIGHT TV MONITOR]
        FI
        RUN TOUCH (USE MEADDR)
        STACK.TOP := STACK.TOP - 4
        STACK(STACK.TOP).ELEMENT := MEADDR      [STACK OTHER MONITOR]
        STACK(STACK.TOP).IFLAG := <OFF>
        STACK(STACK.TOP).DFLAG := <OFF>
CORP
```

```
PROC IAR016                [AUDIO KEY]
     DATA
          SCALAR VALUE: INTEGER
     ATAD
     MEADDR := STACK(STACK.TOP).ELEMENT
     RQBID := 7                [AUDIO MESSAGE ID]
     RUN GMSGAD (USE RQBID, ALT RQB)          [GET AUDIO MESSAGE]
     IF
          ACT(MEADDR).CURRENT.BODY = <ON>          [AUDIO ON?]
     THEN                                          [YES]
          VALUE := 0
          RQB(RQBID).PARAMETERS := -1
     ELSE
          VALUE := 1
          RQB(RQBID).PARAMETERS := 1
     FI
     RUN SYSREQ                [SEND MESSAGE]
     RUN ESTADD (USE MEADDR, ALT ESADDR)
     EST(ESADDR).ON := VALUE
     RUN RDMENU                [RE-DRAW THE MENU]
CORP
```

```
PROC IAK018     [X (CANCEL) KEY, MENU C03]
    DATA
        SCALAR PTR: INTEGER
        SCALAR ELEM: INTEGER
    ATAD
    STACK.TOP := STACK.TOP + 4      [POP X KEY FROM STACK]
    IF
        ERRMSG = <0>                [ERROR PENDING]
    THEN
        RUN GRMST (USE 0)           [RESET ERROR]
    ELSE
        RUN COMPRS                  [COMPRESS STACK]
        IF
            <STACK IS EMPTY>
        THEN
            ERRMSG := 4
            RUN GRMST (USE ERRMSG)
        ELSE
            RUN SRTES               [SORT STACK]
            IF
                <HI-RES ELEMENT IS ON STACK>
            THEN
                ERRMSG := 2
                RUN GRMST (USE ERRMSG)
            ELSE
                ERRMSG := 0
                PTR := STACK.BOTTOM
                WHILE
                    PTR <= STACK.TOP AND ERRMSG = 0
                DO
                    ELEM := STACK(PTR).ELEMENT
                    RUN ESTADD (USE ELEM, ALT ESADDR)
                    IF
                        EST(ESADDR).ON = <OFF>
                    THEN
                        ERRMSG := 5
                        RUN GRMST (USE ERRMSG)
                    FI
                    PTR := PTR - 4  [POINT TO NEXT ELEMENT]
                OD
                IF
                    ERRMSG = 0
                THEN
                    PTR := STACK.BOTTOM
                    WHILE
                        PTR <= STACK.TOP
                    DO
                        ELEM := STACK(PTR).ELEMENT
                        RUN ESTADD (USE ELEM, ALT ESADDR)
                        RUN TOUCH (USE ELEM)
                        EST(ESADDR).ON := <OFF>
                        <SEND DISCONNECT MESSAGE>
                        PTR := PTR - 4
```


UD
RUN LSENDS
RUN RCMENU
FI
FI
FI
FI
CCRP
A-73

```
PROC 1AR017     [OTHER KEY, MENU C03]
        PUMENU := C04MNU            [PENDING MENU = MENU OF MENUS]
        RUN RESET                   [ELIMINATE ALL MENU TOUCHES]
        STACK.TOP := STACK.BOTTOM       [RESET STACK]
CORP
```

A-74

```
PROC IAR019     (CAMERA KEY, MENU CO5)
        DATA
                SCALAR  CAMERA,
                        CAM.ID,
                        CAM.NO: INTEGER
        ATAD
        STACK.TOP := STACK.BOTTOM       (RESET STACK)
        MEADDR := STACK(STACK.TOP).ELEMENT
        STACK.TOP := STACK.BOTTOM       (RESET STACK)
        IF
                <CAMERA IS ON>
        THEN
                RUN IAR027      (TURN OFF CAMERA MOTION)
                RUN IAR19A      (DISCONNECT THIS CAMERA)
        FI
        RUN ESTADD (USE MEADDR, ALT ESADDR)
        IF
                MEADDR = CAMERA         (A RE-TOUCH)
        THEN
                CAMERA := 0
        ELSE
                EST(ESADDR).ON := <ON>
                CAMERA := MEADDR
                CAM.NO := EST(ESADDR).FLD1
                CAM.ID := EST(ESADDR).FLD2
                RQBID := 0
                RUN GNSGAD (USE RQBID, ALT RQE)
                RQB(4) := CAM.NO
                RUN SYSREQ (USE RQB)
                RUN IAR19A      (CONNECT CAMERA)
        FI
        RUN RLMENU
CORP
```

```
PROC IAN19A (USE SWITCH)          [CONNECT/DISCONNECT CAMERA TO CCM]
     IF
                <CAMERA IS ON>
     THEN
                IF
                          <SWITCH = DISCONNECT>
                THEN
                          RQBID := 6          [DISCONNECT MESSAGE]
                          RUN GMSGAD (USE RQBID, ALT RQB)
                          RQB(6) := <EST ID OF CCM>
                          RUN ESTADD (USE CAMERA, ALT ESADDR)
                          EST(ESADDR).ON := <OFF>
                          RUN SYSREQ (USE RQB)
                ELSE
                          RQBID := 9          [CONNECT MESSAGE]
                          RUN GMSGAD (USE RQBID, ALT RQB)
                          RQB(2) := 6         [SET LENGTH OF MESSAGE]
                          RQB(6) := CAM.ID
                          RQB(8) := <EST ID OF CCM>
                          RUN SYSREQ (USE RQB)
                FI
     FI
CORP
```

```
PROC IAR020      (BACK KEY, MENU C05)
        STACK.TOP := STACK.BOTTOM          (RESET STACK)
        RUN IAR027                         (STOP CAMERA MOTION)
        RUN IAR19A (USE DISCONNECT)        (DISCONNECT CAMERA)
        CAMERA := 0
        RUN BACK
CORP
```

```
PROC IAR021      [HELP KEY, MENU COS]
     IF
          <ERROR PENDING AND ERROR NOT DISPLAYED>
     THEN
          STACK.TOP := STACK.TOP + 4          [POP STACK]
          RUN ERRDSP                          [DISPLAY ERROR]
     ELSE
          RUN IAR027                          [STOP CAMERA MOTION]
          RUN IAR19A (USE DISCONNECT)         [DISCONNECT CAMERA]
          CAMERA := 0
          RUN HELP
     FI
CORP
```

A-78

```
PROC IAR022                [CAMERA PRESET KEYS,CAMERA POSITIONING MENU]
        DATA
                SCALAR ELEM: INTEGER
        ATAC
        ELEM := STACK(STACK.TOP).ELEMENT
        STACK.TOP := STACK.BOTTOM          [RESET STACK]
        IF
                <CAMERA NOT ON>
        THEN
                ERRMSG := 12
                RUN GRMST (USE ERRMSG)
        ELSE
                PRESET := ELEM
                RUN IAR027          [STOP CAMERA MOTION]
                RUN ESTADD (USE ELEM, ALT ESADDR)
                EST(ESADDR).ON := 1
                RUN RDMENU
                RQBID := 4
                RUN GMSGAD (USE RQBID, ALT RQB)
                RQB(4) := CAM.NO
                RQB(6) := EST(ESADDR).FLD1
                RUN SYSREQ (USE RQB)
                EST(ESADDR).ON := <OFF>
                RUN LIGHT (USE WAIT)
                RUN RDMENU
                <DETERMINE EST ID OF TOUCHED PRESET KEY>
                EST(ESADDR).FLD2 := -1   [INDICATE THAT PRESET IS SET]
        FI
CORP
```

```
PROC TARO23                    [ZOOM KEYS, CAMERA POSITIONING MENU]
     DATA
          SCALAR   INDEX,
                   CAMERA: INTEGER
          ARRAY    ZOOM (1:4) OF ZOOM-ENTRY
          RECORD   ZOOM-ENTRY
                   .STEP,
                   .DIRECTION: INTEGER
     ATAD
     MEADDR := STACK(STACK.TOP).ELEMENT
     STACK.TOP := STACK.BOTTOM
     IF
          CAMERA  = 0
     THEN
          ERRMSG := 12
          RUN GRMST (USE ERRMSG)
     ELSE
          RQBID := 2                [ZOOM MESSAGE]
          RUN GMSGAD (USE RQBID, ALT RQB)
          RQB(RQBID).MODIFIER := CAMERA.NUMBER
          MCI(MEADDR).NEXT.BODY := <ON>
          RUN SYSREQ (USE RQB)
          RUN RDMENU
          INDEX := (MCI(MEADDR).ID - 21)
          RQB(RQBID).PARAMETERS := ZOOM(INDEX).STEP
          RQB(RQBID).PARAMETERS(2) := ZOOM(INDEX).DIRECTION
          MCI(MEADDR).NEXT.BODY := <OFF>
          RUN RDMENU
     FI
CORP
```

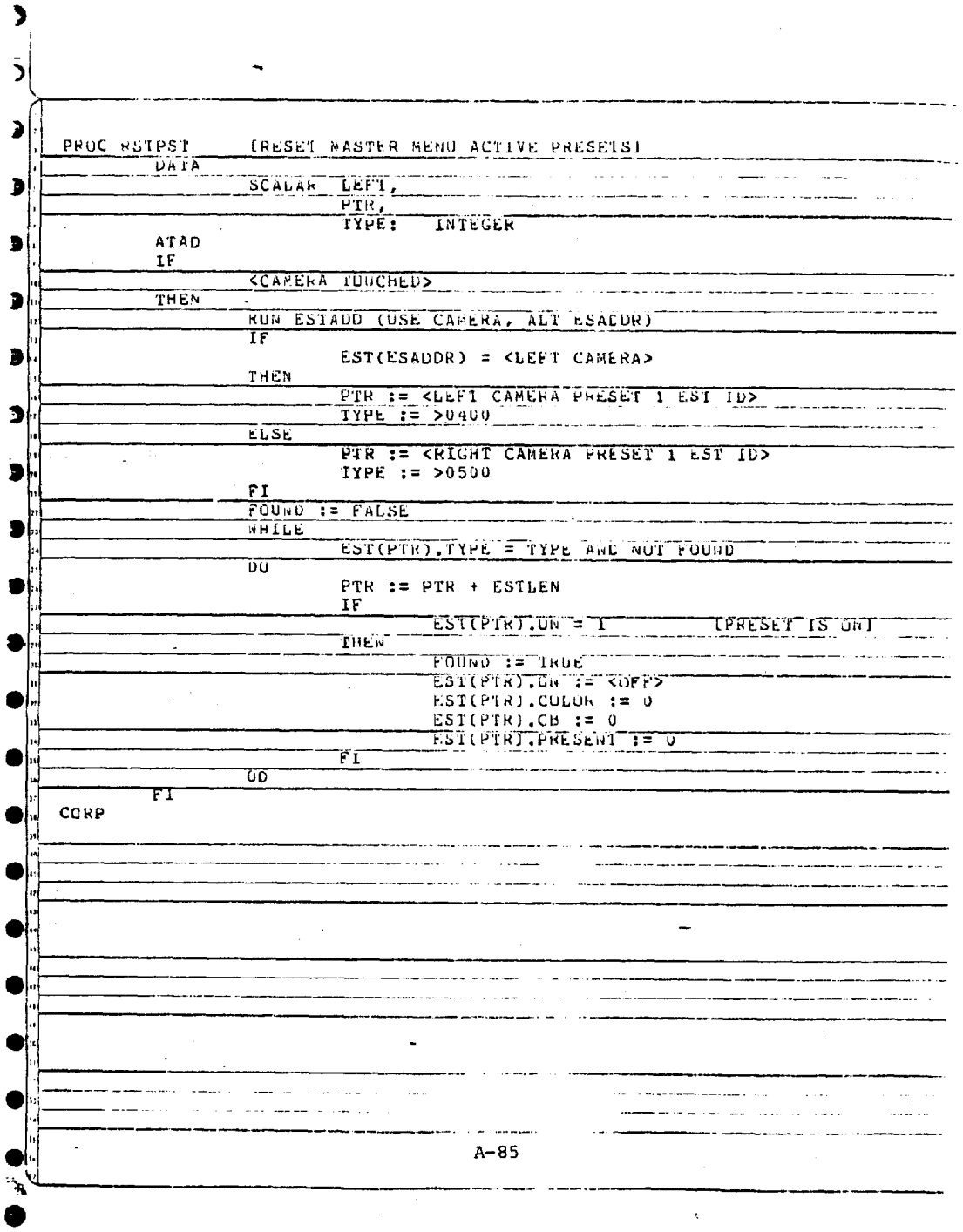
```
PROC IAR024                 (FOCUS KEYS, CAMERA POSITIONING MENU)
        DATA
                    SCALAR  INDEX,
                            CAMERA: INTEGER
                    ARRAY   FOCUS (1:4) OF FOCUS-ENTRY
                    RECORD  FOCUS-ENTRY
                            .STEP,
                            .DIRECTION: INTEGER
        ATAD
        MEADDR := STACK(STACK.TOP).ELEMENT
        STACK.TOP := STACK.BOTTOM
        IF
                    CAMERA  = 0
        THEN
                    ERRMSG := 12
                    RUN GRMST (USE ERRMSG)
        ELSE
                    RQBID := 3                  (FOCUS MESSAGE)
                    RUN GMSGAD (USE RQBID, ALT RQB)
                    RQB(RQBID).MODIFIER := CAMERA.NUMBER
                    MCT(MEADDR).NEXT.BODY := <ON>
                    RUN SYSREQ (USE RQB)
                    RUN RQMENU
                    INDEX := (MCT(MEADDR).ID - 25)
                    RQB(RQBID).PARAMETERS := FOCUS(INDEX).STEP
                    RQB(RQBID).PARAMETERS(2) := FOCUS(INDEX).DIRECTION
                    MCT(MEADDR).NEXT.BODY := <OFF>
                    RUN RQMENU
        FI
CORP
```

```
PROC IAR025                    [PAN/TILT, CAMERA POSITIONING MENU]
        DATA
                SCALAR  CAMERA,
                        SLEW.TOUCH,
                        INDEX: INTEGER
                ARRAY   SLEW (1:8) OF SLEW-ENTRY
                RECORD  SLEW-ENTRY
                        .PAN,
                        .TILT: INTEGER
        ATAD
        MEADDR := STACK(STACK.TOP).ELEMENT
        STACK.TOP := STACK.BOTTOM
        IF
                CAMERA = 0
        THEN
                ERRMSG := 12
                RUN GRMST (USE ERRMSG)
        ELSE
                RQBID := 1                      [SLEW MESSAGE]
                RUN GMSGAD (USE RQBID, ALT RQB)
                RQB(RQBID).MODIFIER := CAMERA.NUMBER
                IF
                        MEADDR = SLEW.TOUCH     [A RE-TOUCH?]
                THEN                            [YES]
                        RQB(RQBID).PARAMETERS := <OFF>          [PAN]
                        RQB(RQBID).PARAMETERS(2) := <OFF>       [TILT]
                        RUN SYSREQ (USE RQB)
                ELSE
                        IF
                                SLEW.TOUCH <> 0         [ANOTHER SLEW ON?]
                        THEN                            [YES]
                                MCT(SLEW.TOUCH).NEXT.BODY := <OFF>
                                RQB(RQBID).PARAMETERS := <OFF>
                                RQB(RQBID).PARAMETERS(2) := <OFF>
                                RUN SYSREQ (USE RQB)
                        FI
                        MCT(MEADDR).NEXT.BODY := <ON>
                        INDEX := (MCT(MEADDR).ID - 29)
                        RQB(RQBID).PARAMETERS := SLEW(INDEX).PAN
                        RQB(RQBID).PARAMETERS(2) := SLEW(INDEX).TILT
                        RUN SYSREQ (USE RQB)
                FI
        FI
        RUN RDMENU
CORP
```

```
PROC IAR02b     [GO KEY, MENU C05]
        RUN IAR027      [STOP CAMERA MOTION]
        RUN IAR19A (USE DISCONNECT)
        CAMERA := 0
        RUN MNEXT       [GET NEXT MENU NUMBER]
        RUN GRNST (USE 0)       [RESET ANY PENDING ERRORS]
CORP
```

```
PROC 1AR027      (STOP SLEW CAMERA MOTION)
        DATA
                 SCALAR  CAMERA,
                         SLEW.TOUCH,
                         PRESET: INTEGER
        ATAD
        IF
                 <CAMERA ON AND SLEW ON>
        THEN
                 MCT(SLEW.TOUCH).NEXT.BODY := <OFF>
                 RQBID := 1
                 RUN GMSGAD (USE RQBID, ALT RQB)
                 RQB(6) := <OFF>
                 RUN SYSREQ (USE RQB)
                 SLEW.TOUCH := 0
                 RQBID := 9
                 RUN GMSGAD (USE RQBID, ALT RQB)
                 RUN SYSREQ (USE RQB)    (DO A  WAIT TIME)
                 RUN RDMENU
        FI
 CORP
```

A-84



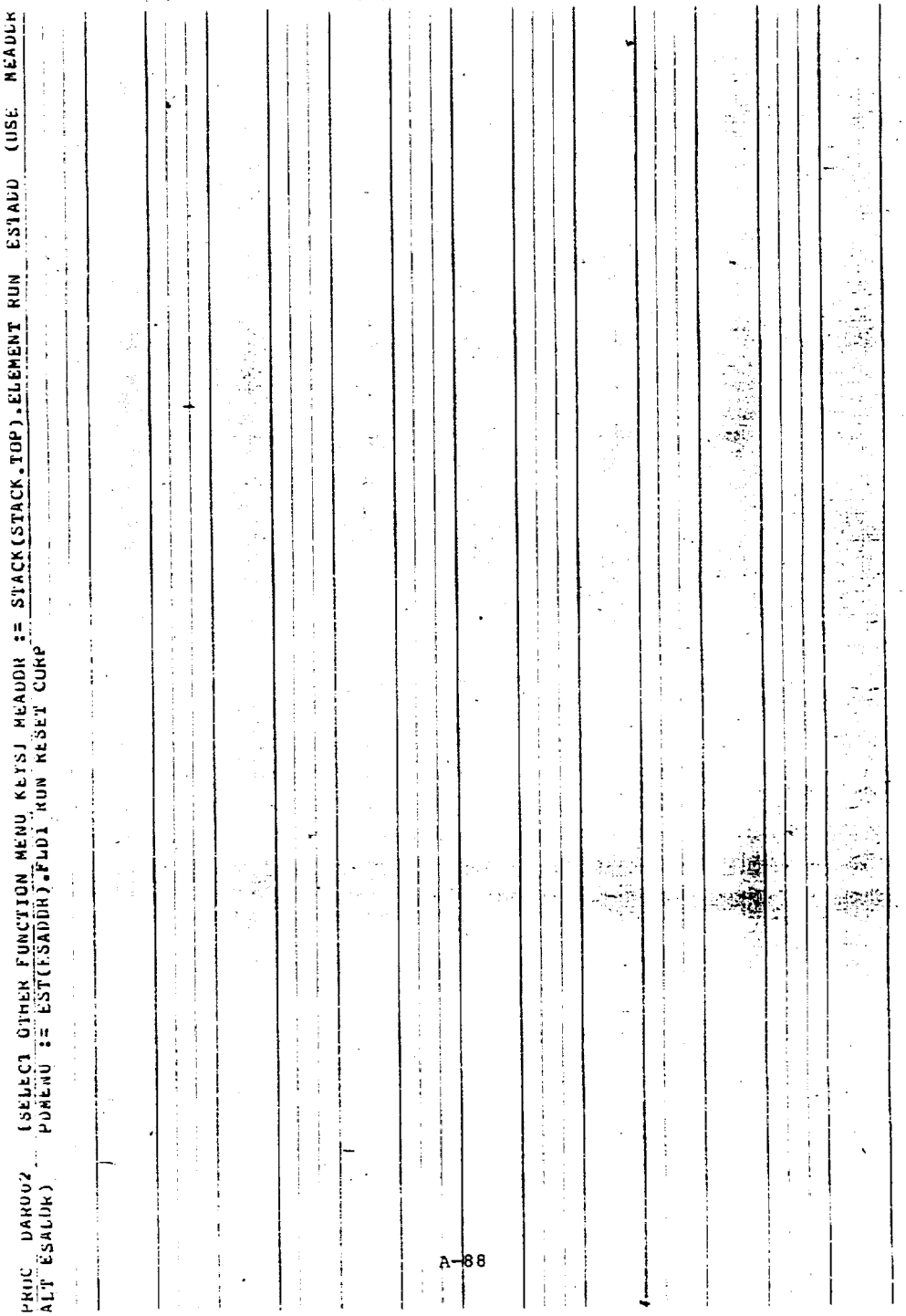
```
PROC RSTPST     (RESET MASTER MENU ACTIVE PRESETS)
        DATA
                SCALAR  LEFT,
                        PTR,
                        TYPE:   INTEGER
        ATAD
        IF
                <CAMERA TOUCHED>
        THEN
                RUN ESTADD (USE CAMERA, ALT ESADDR)
                IF
                        EST(ESADDR) = <LEFT CAMERA>
                THEN
                        PTR := <LEFT CAMERA PRESET 1 EST ID>
                        TYPE := >0400
                ELSE
                        PTR := <RIGHT CAMERA PRESET 1 EST ID>
                        TYPE := >0500
                FI
                FOUND := FALSE
                WHILE
                        EST(PTR).TYPE = TYPE AND NOT FOUND
                DO
                        PTR := PTR + ESTLEN
                        IF
                                EST(PTR).ON = 1         (PRESET IS ON)
                        THEN
                                FOUND := TRUE
                                EST(PTR).ON := <OFF>
                                EST(PTR).COLOR := 0
                                EST(PTR).CB := 0
                                EST(PTR).PRESENT := 0
                        FI
                OD
        FI
CORP
```

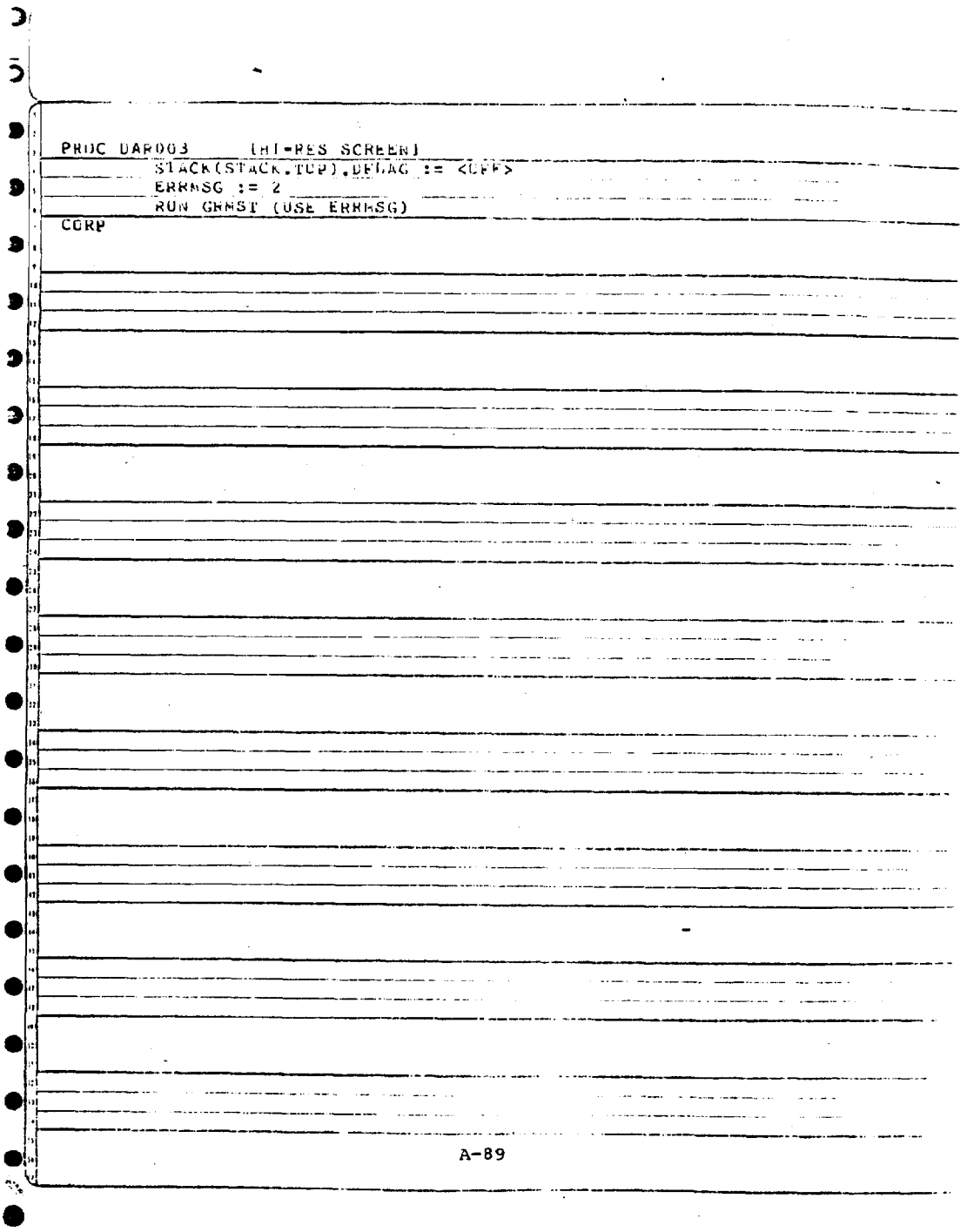
A-85

```
PROC DAR001    [SOURCE - SINK CONNECTION]
    DATA
        SCALAR  SOURCE,
                CONTROLLER,
                COLOR,
                PIR,
                LOOP: INTEGER
        SCALAR  KEEP,
                PRESET: LOGICAL
    ATAD

    RQBID := 8          [CONNECT MESSAGE]
    RUN GMSGAD (USE RQBID, ALT RQB)
    MEADDR := STACK(STACK.TOP).ELEMENT
    SOURCE := MEADDR
    MCT(MEADDR).NEXT.BORDER := <OFF>
    RUN ESTADD (USE MEADDR, ALT ESADDR)
    RQB(RQBID).PARAMETERS := SOURCE
    EST(ESADDR).FON := <ON>
    CONTROLLER := EST(ESADDR).ID
    COLOR := EST(ESADDR).COLOR
    WHILE
        <STACK NOT EMPTY>
    DO
        RUN POP-ELEM (ALT MEADDR)
        MCT(MEADDR).NEXT.BORDER := <OFF>
        RUN ESTADD (USE MEADDR, ALT ESADDR)
        IF
            MEADDR = <STILL FRAME>  OR
            MEADDR = <AUTO TRANSMIT>
        THEN
            <DETERMINE WHICH KEY WILL GO
             ON AND WHICH WILL GO OFF, BASED
             ON PREVIOUS STATE>
        FI
        RQB(RQBID).PARAMETERS(2) := EST(ESADDR).ID
        IF
            EST(ESADDR).COLOR <> COLOR  OR
            EST(ESADDR).FON = <OFF>
        THEN
            IF
                EST(ESADDR).TYPE =< 1
            THEN
                IF
                    EST(ESADDR).FON <> 0
                THEN
                    RQB(RQBID).PARAMETERS := EST(ESADDR).CB
                    RQB(RQBID).PARAMETERS(2) := EST(ESADDR).II
                    RUN SYSREQ (USE RQB)
                    RQB(RQBID).PARAMETERS := SOURCE
                    RQB(RQBID).PARAMETERS(2) := EST(ESADDR).II
                FI
                RUN SYSREQ (USE RQB)
```

```
                    FI
                    EST(ESADDR).CB := CONTROL
                    EST(ESADDR).COLOR := COLOR
                    EST(ESADDR).FON := <ON>
                    MCT(MEADDR).NEXT.BODY := <OFF>
                    MCT(MEADDR).CURRENT.BODY := <OFF>
            FI
    OD
    IF
            SOURCE = CAMTCH          [SOURCE WAS A TOUCHED CAMERA]
    THEN
            PRESET := <TRUE IF A PRESET WAS TOUCHED,
                       FALSE IF A PRESET WAS NOT TOUCHED>
            PTR := CAMTCH
            WHILE
                    CAMLOP > 0
            DO
                    PTR := PTR + MCILEN
                    KEEP := FALSE
                    IF
                            (PRESET  AND MCT(PTR).CURRENT.BORDER = <OFF>)
                            OR
                            (NOT PRESET  AND  MCT(PTR).CURRENT.BODY = <OFF>)
                    THEN
                            RUN ESTADD (USE PTR, ALT ESADDR)
                            EST(ESADDR).CB := 0
                            EST(ESADDR).FON := <OFF>
                            EST(ESADDR).FPRE := <OFF>
                            RUN ZAPOUT (USE PTR)     [ERASE PRESET]
                            MCT(PTR).CURRENT := MCT(PTR).NEXT
                    FI
                    CAMLOP := CAMLOP - 1
            OD
            CAMTCH := 0
    FI
    RUN LSENDS
CORP
```


PROC DAHOO2 (SELECT OTHER FUNCTION MENU KEYS) HEADER := STACK(STACK.TOP).ELEMENT RUN ESTADD (USE HEADER
ALT ESALOK) POMENU := ESTF(SADDR).FLD1 RUN RESET CURP



```
PROC UAR003     (HI-RES SCREEN)
        STACK(STACK.TOP).DFLAG := <OFF>
        ERRMSG := 2
        RUN GRMST (USE ERRMSG)
CORP
```

A-89

```
PROC DAR004     [PREVIEW WITH NO SOURCE]
        STACK(STACK.TOP).DFLAG := <OFF>
        ERRMSG := 3
        RUN GRMSG (USR ERRMSG)
CORP
```

A-90

```
*=============================================================
*PROC = CCTSPI
*
*   PROCEDURE NAME: CCTSPI
*
*   CCTSPI RECEIVES X AND Y TOUCH COORDINATES FROM THE
*   TOUCH SENSITIVE PANEL CONTROLLER.  IT LOCATES THE
*   ELEMENT TOUCHED (IF ANY), STACKS IT AND SETS ITS
*   IMMEDIATE FLAG ON.  THE NEXT PHASE OF THIS PROCEDURE
*   CONTINUALLY CHECKS THE TOP ELEMENT ON THE STACK TO
*   DETERMINE IF AN IMMEDIATE OR DELAYED ACTION SHOULD BE
*   CALLED.  AFTER EXECUTION OF AN ACTION ROUTINE, THE
*   TOP OF THE STACK IS AGAIN EXAMINED.  IF THE STACK IS
*   EMPTY OR BOTH FLAGS ARE OFF, THE PROCEDURE RETURNS TO
*   THE CALLER.
*
*   INPUT: R0 CONTAINS THE X COORDINATE
*          R1 CONTAINS THE Y COORDINATE
*   OUTPUT: NONE
*   RESULT: VARIABLE BASED ON ELEMENT STACKED
*   TO CALL: BL @CCTSPI
*
*=============================================================
```

```
*=============================================================
*PROC = GACTAD
*
*   PROCEDURE NAME: GACTAD
*
*   GACTAD WILL DERIVE THE ADDRESS OF A ACTION ROUTINE.
*
*   INPUT: R3 CONTAINS THE ACTION ROUTINE ADDRESS
*   OUTPUT: R3 CONTAINS THE MAPPED ADDRESS
*   RESULT: NONE
*   TO CALL: BL @GACTAD
*
*=============================================================
```

```
*=============================================================
*PROC = FINDEL
*
*   PROCEDURE NAME: FINDEL
*
*   FINDEL LOOPS THROUGH THE CURRENT MENU'S MCT TABLE TO
*   DETERMINE WHICH MENU ELEMENT WAS TOUCHED.  FIRST, THE
*   HERE COORDINATES IN THE MCT ARE SUBTRACTED FROM THE
*   COORDINATES SUPPLIED BY THE TOUCH SENSITIVE PANEL.  IF
*   EITHER X OR Y VALUE IS MADE NEGATIVE BY THE SUBTRACTION,
*   THEN THIS WAS NOT THE ELEMENT TOUCHED.  NEXT, THE
*   ELEMENT'S DISPLAY ELEMENT LIBRARY (DEL) ENTRY IS
*   FOUND.  IF THE COORDINATES ARE BOTH LESS THAN THE
*   X AND Y TOLERANCES FOUND IN THE DEL, THEN THIS IS THE
*   TOUCHED ELEMENT.  IF NOT, THE SEARCH CONTINUES UNTIL
*   ALL ELEMENTS ARE EXAMINED.  IF AN ELEMENT DOES NOT
*   HAVE A DEL ENTRY, OR ITS INSENSITIVE OR INHIBIT MCT
*   BITS ARE ON, THEN DO NOT TEST THIS ELEMENT.
*
*   INPUT: R0 CONTAINS X COORDINATE
*          R1 CONTAINS Y COORDINATE
*   OUTPUT: R2 CONTAINS THE TOUCHED ELEMENTS MCT ADDRESS,
*           OR ZERO IF NO ELEMENT IS FOUND.
*   RESULT: TOUCHED ELEMENT IS LOCATED IN THE MCT.
*   TO CALL: BL @FINDEL
*
*=============================================================
```

```
*=============================================================
*PROC = LEXSCN
*
*   PROCEDDURE NAME: LEXSCN
*
*   LEXSCN EXAMINES THE ELEMENT STACK TO DETERMINE IF ENOUGH
*   INFORMATION HAS BEEN GIVEN IN A CORRECT MANNER SO
*   THAT SUBSEQUENT ROUTINE ACTIONS WILL BE SUCCESSFUL.
*
*   BASIC REQUIRMENTS ARE SIMPLE.  EACH SENTENCE MUST
*   INCLUDE A SOURCE (CAMERA, SLIDE CAMERA, RECEIVE) AND
*   ONE OR MORE DESTINATIONS (TV SCREEN, OVERHEAD MONITORS,
*   SEND).  IF THE ELEMENT STACK DOES NOT INCLUDE A SOURCE
*   BUT DOES HAVE AN ACTIVE DESTINATION (A DESTINATION
*   WHICH IS CURRENTLY LINKED TO A SOURCE), THE SOURCE FOR
*   THAT DESTINATION IS PLACED ON THE STACK.
*
*   IF LEXSCN DETECTS AN ERROR, IT SETS LEXRC TO TWO (2)
*   AND INDICATES THE APPROPRIATE ERROR.
*
*   INPUT: ELEMENT STACK
*   OUTPUT: NONE
*   RESULT: ELEMENTS EITHER PASS OR FAIL SCAN
*   TO CALL: BL @LEXSCN
*
*=============================================================
```