```
$REGOVERLAY DEBUG FAST(130) OPTIMIZE(0) PAGEWIDTH(80)
/*-------------------------------------------------------------*/
$TITLE ('EO.PLM')
/*=============================================================

  NAME:  Electro-optic camera program

  HISTORY: --REV--  --DATE--  --AUTHOR--
              -      4-26-88      JEM

=============================================================*/
$EJECT
EO: DO;

/* Standard literals */

DECLARE DCL       LITERALLY 'DECLARE';
DECLARE FALSE     LITERALLY '000H';
DECLARE FOREVER   LITERALLY 'WHILE 1';
DECLARE LIT       LITERALLY 'LITERALLY';
DECLARE TRUE      LITERALLY '0FFH';
DECLARE DO_CASE   LITERALLY 'DO';
DECLARE END_DO    LITERALLY 'GOTO CASE_END; END';
DECLARE END_CASE  LITERALLY 'CASE_END: END';

/* Variables */

DCL  MODE                  BYTE FAST;
DCL  DISK_MODE             BYTE FAST;
DCL  CHK_KEY               DWORD SLOW;
DCL  ANALYZING_IMAGE       BYTE SLOW;
DCL  ARCHIVE_FAILED        BYTE SLOW;
DCL  ARCHIVE_ABORT         BYTE SLOW;
DCL  AUTO_MODE             BYTE FAST;
DCL  AUTO_IMAGES           BYTE SLOW;
DCL  AUTO_HOUR             BYTE SLOW;
DCL  AUTO_MINUTE           BYTE SLOW;
DCL  AUTO_PERIOD           WORD SLOW;

DCL  SCREEN                BYTE FAST;
DCL  OLD_SCREEN            BYTE SLOW;
DCL  SCREEN_CHANGE         BYTE FAST;
DCL  TOGGLE                BYTE FAST;
DCL  OLD_TOGGLE            BYTE SLOW;
DCL  TOGGLE_CHANGE         BYTE FAST;
DCL  CURSOR_CTL            BYTE SLOW;
DCL  DISP (32)             BYTE SLOW;
DCL  PANEL_ACTIVE          BYTE SLOW;
DCL  POS                   BYTE SLOW;
DCL  MESSAGE               WORD SLOW;
DCL  ADDRS                 WORD SLOW;

DCL  BFR_IMAGES            BYTE FAST;
DCL  BFR_IMAGE_NO (6)      WORD SLOW;
DCL  BFR_IMAGE_TIME (6)    DWORD SLOW;
DCL  BFR_READ_PTR          BYTE FAST;
DCL  BFR_WRITE_PTR         BYTE FAST;
DCL  COPY_COUNT            WORD SLOW;

DCL  ENABLE_IMAGE_POWER    BYTE SLOW;
```

```
DCL  ENABLE_MICRO_POWER    BYTE SLOW;
DCL  IMAGE_PWR_ON          BYTE SLOW;
DCL  CCD_CLOCK_MODE        BYTE FAST;

DCL  BUCKET (16)           BYTE SLOW;
DCL  ANALYZER_ADDRESS      WORD FAST;
DCL  ANALYZER_COUNT        BYTE FAST;

DCL  BATTERY               BYTE SLOW;
DCL  BATTERY_VOLTS         INTEGER SLOW;
DCL  CAMERA_TEMP           INTEGER SLOW;
DCL  CCD_TEMP              INTEGER SLOW;
DCL  COOLER_VOLTS          INTEGER SLOW;
DCL  DEW_TEMP              INTEGER SLOW;
DCL  HUMIDITY              INTEGER SLOW;
DCL  RECORDER_TEMP         INTEGER SLOW;
DCL  PRE_LOG_ZERO          INTEGER SLOW;
DCL  POST_LOG_ZERO         INTEGER SLOW;
DCL  VIDEO                 INTEGER SLOW;

DCL  LAST_PORT1_STATUS     BYTE FAST;
DCL  LAST_SECOND           BYTE FAST;
DCL  LED_CTL               BYTE FAST;
DCL  PORT1_CHANGING        BYTE FAST;
DCL  PORT1_STATUS          BYTE FAST;
DCL  NEXT_TICK_TIME        WORD FAST;
DCL  LED_TICKS             BYTE FAST;
DCL  TOGGLE_TICK           WORD FAST;
DCL  REFRESH_TICK          WORD FAST;
DCL  TICK                  WORD FAST;
DCL  TIME (8)              BYTE SLOW;
DCL  CAMERA_BODY_ACTIVE    BYTE FAST;
DCL  CAMERA_STATE          BYTE FAST;
DCL  CAMERA_TIME           WORD FAST;
DCL  EVENT                 BYTE FAST;
DCL  HI_TIME               WORD FAST;
DCL  TIMER_START_TIME (8)  DWORD SLOW;
DCL  TIMER_TIME (8)        DWORD SLOW;

DCL  COMMAND (10)          BYTE FAST;
DCL  COMMAND_ID            BYTE FAST;
DCL  ID                    BYTE FAST;
DCL  CURRENT_COMMAND       BYTE FAST;
DCL  DMA_INTERRUPTED       BYTE SLOW;
DCL  BUFFER_ADDRESS        WORD FAST;
DCL  BUFFER (200)          BYTE SLOW;
DCL  FIFO_COUNT            BYTE SLOW;
DCL  POINTER               BYTE SLOW;
DCL  SCSI_DISK_READY       BYTE FAST;
DCL  SCSI_TAPE_READY       BYTE FAST;
DCL  DISK_READY_COUNT      BYTE FAST;
DCL  TAPE_READY_COUNT      BYTE FAST;
DCL  RESIDUAL_COUNT        BYTE SLOW;
DCL  SCSI_BUSY             BYTE FAST;
DCL  SCSI_DMA_XFER         BYTE FAST;
DCL  SCSI_ERROR            BYTE FAST;
DCL  STATUS_IN_FIFO        BYTE FAST;
DCL  TARGET_MESSAGE        BYTE FAST;
DCL  TARGET_STATUS         BYTE FAST;
DCL  TEN_BIT               BYTE FAST;
```

```
DCL  TOTAL_XFER_COUNT      DWORD FAST;

DCL  START_BUFFER_FILL     BYTE FAST;
DCL  START_IMAGE_XFER      BYTE FAST;
DCL  START_MICRO_XFER      BYTE FAST;
DCL  XFER_BUFFER_ADDRESS   WORD FAST;
DCL  XFER_LENGTH           BYTE FAST;
DCL  XFER_POINTER          WORD FAST;
DCL  XFER_READ             BYTE FAST;

DCL  CLOCK_CHANGE          BYTE FAST;
DCL  DATE                  BYTE FAST;
DCL  HOUR                  BYTE FAST;
DCL  MINUTE                BYTE FAST;
DCL  MONTH                 BYTE FAST;
DCL  SECOND                BYTE FAST;
DCL  YEAR                  BYTE FAST;

DCL CODE (*) BYTE DATA (3,4,6,1);

/* CCD Clock Modes */
DCL  DARK          LIT '0';
DCL  HOR_VER_DARK  LIT '1';
DCL  HOR_DARK      LIT '2';
DCL  HOR           LIT '3';
DCL  HOR_VER       LIT '4';

/* Non volatile variables */

DCL  DATA_AREA (512)       BYTE AT (08400H);

DCL  DISK_IMAGES           BYTE AT (08600H);
DCL  AUTO_PWR_OFF          BYTE AT (08601H);
DCL  ENABLE_COOLER         BYTE AT (08602H);
DCL  OLD_DISK_READ_PTR     BYTE AT (08603H);
DCL  OLD_DISK_IMAGES       BYTE AT (08604H);
DCL  DISK_WRITE_PTR        BYTE AT (08605H);
DCL  DISK_READ_PTR         BYTE AT (08606H);
DCL  CLOCK_CAL             SHORTINT AT (08607H);

DCL  IMAGE_NO              WORD AT (08680H);
DCL  POWER_CHECK           WORD AT (08684H);
DCL  X                     WORD AT (08686H);
DCL  Y                     WORD AT (08688H);

DCL  DISK_IMAGE_TIME (60)  DWORD AT (08690H);
DCL  DISK_IMAGE_NO (60)    WORD  AT (08780H);

/* Processor I/O declarations */

DCL  AD_COMMAND        BYTE      AT(00002H);  /*  W  */
DCL  AD_RESULT_HI      BYTE      AT(00003H);  /*  R  */
DCL  AD_RESULT_LO      BYTE      AT(00002H);  /*  R  */
DCL  BAUD_RATE         BYTE      AT(0000EH);  /*  W  */
DCL  HSI_MODE          BYTE      AT(00003H);  /*  W  */
DCL  HSI_STAT          BYTE      AT(00006H);  /*  R  */
DCL  HSI_TIME          WORD      AT(00004H);  /*  R  */
DCL  HSO_COMMAND       BYTE      AT(00006H);  /*  W  */
DCL  HSO_TIME          WORD      AT(00004H);  /*  W  */
DCL  INT_MASK          BYTE      AT(00008H);  /* R/W */
```

```
DCL  INT_MASK_1        BYTE      AT(00013H);  /* R/W */
DCL  INT_PENDING       BYTE      AT(00009H);  /* R/W */
DCL  INT_PENDING_1     BYTE      AT(00012H);  /* R/W */
DCL  IOC0              BYTE      AT(00015H);  /*  W  */
DCL  IOC1              BYTE      AT(00016H);  /*  W  */
DCL  IOC2              BYTE      AT(0000BH);  /*  W  */
DCL  IOPORT0           BYTE      AT(0000EH);  /*  R  */
DCL  IOPORT1           BYTE      AT(0000FH);  /* R/W */
DCL  IOPORT2           BYTE      AT(00010H);  /* R/W */
DCL  IOS0              BYTE      AT(00015H);  /*  R  */
DCL  IOS1              BYTE      AT(00016H);  /*  R  */
DCL  IOS2              BYTE      AT(00017H);  /*  R  */
DCL  PWM_CONTROL       BYTE      AT(00017H);  /*  W  */
DCL  SBUF              BYTE      AT(00007H);  /* R/W */
DCL  SP                WORD      AT(00018H);  /* R/W */
DCL  SP_CON            BYTE      AT(00011H);  /*  W  */
DCL  SP_STAT           BYTE      AT(00011H);  /*  R  */
DCL  TIMER1            WORD      AT(0000AH);  /*  R  */
DCL  TIMER2            WORD      AT(0000CH);  /*  R  */
DCL  WATCHDOG          BYTE      AT(0000AH);  /*  W  */
DCL  WSR               BYTE      AT(00014H);  /* R/W */
DCL  ZERO              WORD      AT(00000H);  /*  R  */

DECLARE CLEAR_WATCHDOG LIT 'WATCHDOG = 01EH; WATCHDOG = 0E1H;';

/* Control codes */

DCL  HSI_MODE_INIT     LIT '00000000B';
DCL  WSR_INIT          LIT '00000000B';
DCL  SP_CON_INIT       LIT '00000000B';
/*
   Bit 0-1: Mode select
   Bit 2:   Parity enable
   Bit 3:   Receive enable
   Bit 4:   Ninth data bit   */

DCL  IOC0_INIT         LIT '00001000B';
/*
   Bit 0:   HSI0 input enable
   Bit 1:   Timer 2 reset each write
   Bit 2:   HSI1 input enable
   Bit 3:   Timer 2 external reset enable
   Bit 4:   HSI2 input enable
   Bit 5:   Timer 2 reset from HSI0
   Bit 6:   HSI3 input enable
   Bit 7:   Timer 2 clock from HSI1  */

DCL  IOC1_INIT         LIT '00000111B';
/*
   Bit 0:   PWM output enable
   Bit 1:   External interrupt P0.7 enable
   Bit 2:   Timer 1 overflow interrupt enable
   Bit 3:   Timer 2 overflow interrupt enable
   Bit 4:   HSO4 output enable
   Bit 5:   TXD output enable
   Bit 6:   HSO5 output enable
   Bit 7:   HSI interrupt on FIFO full  */

DCL  IOC2_INIT         LIT '11000000B';
/*
```

```
   Bit 0:   Timer 2 fast increment enable
   Bit 1:   Timer 2 up/down enable
   Bit 2:   PWM prescale enable
   Bit 3:   Sample and hold disable
   Bit 4:   A/D converter prescale disable
   Bit 5:   Timer 2 overflow at 8000H
   Bit 6:   HSO CAM lock enable
   Bit 7:   HSO CAM clear each write */

DCL  INT_MASK_INIT     LIT '00100001B';
/*
   Bit 0:  Timer 1 overflow interrupt
   Bit 1:  Conversion complete interrupt
   Bit 2:  HSI data available interrupt
   Bit 3:  HSO pin interrupt
   Bit 4:  HSI0 pin interrupt
   Bit 5:  Software timer interrupt
   Bit 6:  Serial port interrupt
   Bit 7:  External interrupt     */

DCL  INT_MASK_1_INIT   LIT '00000000B';
/*
   Bit 0:  Serial transmit interrupt
   Bit 1:  Serial receive interrupt
   Bit 2:  HSI FIFO half interrupt
   Bit 3:  Timer 2 capture interrupt
   Bit 4:  Timer 2 overflow interrupt
   Bit 5:  EXTINT pin interrupt
   Bit 6:  HSI FIFO full interrupt     */

/* Interrupt vector */
DCL  INT_3  WORD AT (02006H) DATA (.INVALID_INTERRUPT);
DCL  INT_4  WORD AT (02008H) DATA (.INVALID_INTERRUPT);
DCL  INT_6  WORD AT (0200CH) DATA (.INVALID_INTERRUPT);
DCL  INT_7  WORD AT (0200EH) DATA (.INVALID_INTERRUPT);
DCL  INT_8  WORD AT (02030H) DATA (.INVALID_INTERRUPT);
DCL  INT_9  WORD AT (02032H) DATA (.INVALID_INTERRUPT);
DCL  INT_10 WORD AT (02034H) DATA (.INVALID_INTERRUPT);
DCL  INT_11 WORD AT (02036H) DATA (.INVALID_INTERRUPT);
DCL  INT_12 WORD AT (02038H) DATA (.INVALID_INTERRUPT);
DCL  INT_13 WORD AT (0203AH) DATA (.INVALID_INTERRUPT);
DCL  INT_14 WORD AT (0203CH) DATA (.INVALID_INTERRUPT);
DCL  INT_15 WORD AT (0203EH) DATA (.NONMASKABLE_INTERRUPT);
DCL  INT_16 WORD AT (02010H) DATA (.INVALID_INTERRUPT);
DCL  INT_17 WORD AT (02012H) DATA (.INVALID_INTERRUPT);

DCL  IOPORT1_INIT      LIT '11111111B';
/*
   Bit 0:
   Bit 1:
   Bit 2:
   Bit 3:
   Bit 4:
   Bit 5:
   Bit 6:
   Bit 7:    */

DCL  IOPORT2_INIT      LIT '11000000B';
/*
   Bit 0:   TXD output
```

```
   Bit 1:   RXD input
   Bit 3:   T2CLK input
   Bit 4:   T2RST input
   Bit 5:   PWM output
   Bit 6:   Timer 2 up/down input
   Bit 7:   Timer 2 capture input   */

/* Chip configuration byte */
DCL  CCR   BYTE  AT(02018H)  DATA (11001101B);
/*
   Bit 0:   1 = Enable powerdown
   Bit 1:   0 = 8 bit bus width
   Bit 2:   1 = WR strobes
   Bit 3:   1 = ALE strobe
   Bit 4-5: 00 = 1 wait state
   Bit 6-7: 11 = No program lock     */


DCL  CONFIG_INFO(16) BYTE   AT(00100H)
     DATA (0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH);

/* Areas reserved by Intel */

DCL  RESERVE_1(*)   BYTE  AT(02014H)
     DATA (0FFH,0FFH,0FFH,0FFH);
DCL  RESERVE_2(*)   BYTE  AT(02019H)
     DATA (0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH);
DCL  RESERVE_3(*)   BYTE  AT(02020H)
     DATA (0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH);
DCL  RESERVE_4(*)   BYTE  AT(02040H)
     DATA (0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,
           0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH);

/* External I/O ports */

DCL  CLOCK STRUCTURE (CONTROL  BYTE,
                      SECOND   BYTE,
                      MINUTE   BYTE,
                      HOUR     BYTE,
                      DAY      BYTE,
                      DATE     BYTE,
                      MONTH    BYTE,
                      YEAR     BYTE) AT (087F8H);

DCL  ADDRESS_COUNTER    WORD  AT(08800H);
DCL  SCSI_WRITE (16)    BYTE  AT(08810H);
DCL  SCSI_READ  (16)    BYTE  AT(08830H);
DCL  BUFFER_READ_PORT   BYTE  AT(0887FH);
DCL  BUFFER_WRITE_PORT  BYTE  AT(0885FH);
DCL  DISPLAY_DTA        BYTE  AT(09801H);
DCL  DISPLAY_CTL        BYTE  AT(09800H);
```

```
/* Bit ports */

DCL  SEQUENCE_0         BYTE  AT(09000H);
DCL  SEQUENCE_1         BYTE  AT(09001H);
DCL  SEQUENCE_2         BYTE  AT(09002H);
DCL  MICRO_PWR_ENABLE   BYTE  AT(09003H);
DCL  SEQUENCE_HOLD      BYTE  AT(09004H);
DCL  IMAGE_PWR_DISABLE  BYTE  AT(09005H);
DCL  CAMERA_PWR_DISABLE BYTE  AT(09006H);
DCL  SEQUENCE_ENABLE    BYTE  AT(09007H);

DCL  CAMERA_BODY_CTL    BYTE  AT(0A000H);
DCL  YELLOW_LED_ON        LIT '00101110B';
DCL  GREEN_LED_ON         LIT '00101101B';
DCL  RED_LED_ON           LIT '00100011B';
DCL  SENSE_CAMERA_TEMP    LIT '00001001B';
DCL  SENSE_CCD_TEMP       LIT '00001010B';
DCL  SENSE_CAMERA_HUM     LIT '00001100B';
DCL  SENSE_RECORDER_TEMP  LIT '00011000B';
DCL  REV_HUMIDITY_SENSOR  LIT '00101011B';
DCL  ALL_OFF              LIT '00000000B';

/*-------------------------------------------------------------
------- MODULES ----------------------------------------------*/

$TITLE (EOMAIN) INCLUDE (EOMAIN.INC)
$TITLE (EOSCSI) INCLUDE (EOSCSI.INC)
$TITLE (EODISP) INCLUDE (EODISP.INC)
$TITLE (EOHW)   INCLUDE (EOHW.INC)

/*-------------------------------------------------------------*/

/* Startup delay */

DO;
  DCL COUNT WORD FAST;
  DO COUNT = 1 TO 20000;
  END;
END;

CALL MAIN;

/*-------------------------------------------------------------*/

END;
```

```
/*================================================================
  EOMAIN.INC
----------------------------------------------------------------

  NAME:  Electro-optic camera main module

  HISTORY: --REV--  --DATE--  --AUTHOR--
              -      5-26-88      JEM

================================================================*/

MAIN: PROCEDURE;

  /* Modes */
  DCL  OFF              BYTE DATA (' ');
  DCL  STANDBY          BYTE DATA ('S');
  DCL  FLUSH            BYTE DATA ('F');
  DCL  EXPOSE           BYTE DATA ('E');
  DCL  START_ARCHIVE    BYTE DATA ('A');
  DCL  OPEN             BYTE DATA ('O');
  DCL  LOAD             BYTE DATA ('L');
  DCL  START_IMAGE      BYTE DATA ('I');
  DCL  WRITE_HEADER     BYTE DATA ('H');
  DCL  FROM_DISK        BYTE DATA ('D');
  DCL  TO_TAPE          BYTE DATA ('T');
  DCL  END_OF_FILE      BYTE DATA ('Z');
  DCL  UNLOAD           BYTE DATA ('U');
  DCL  ARCHIVE_DONE     BYTE DATA ('Q');

  /* Disk modes */
  DCL  IDLE           LIT '0';
  DCL  RECORDING      LIT '1';
  DCL  CHECKING       LIT '2';
  DCL  STOPPED        LIT '3';

  /* Auto modes */
  DCL  A_OFF          LIT '0';
  DCL  A_ON           LIT '1';
  DCL  A_ARMED        LIT '2';
  DCL  A_EXPOSE       LIT '3';
  DCL  A_MANUAL       LIT '4';

  /* Screens */
  DCL  POWER_ON      LIT '0';
  DCL  POWER_OFF     LIT '1';
  DCL  MAIN_STATUS   LIT '2';
  DCL  TIMER_SET     LIT '3';
  DCL  CLOCK         LIT '4';
  DCL  SYSTEM_MENU   LIT '10';
  DCL  SYSTEM_1      LIT '11';
  DCL  SYSTEM_2      LIT '12';
  DCL  SYSTEM_3      LIT '13';
  DCL  SYSTEM_9      LIT '19';
  DCL  DISK_MENU     LIT '20';
  DCL  DISK_1        LIT '21';
  DCL  DISK_2        LIT '22';
  DCL  DISK_3        LIT '23';
  DCL  DISK_9        LIT '29';
  DCL  TAPE_MENU     LIT '30';
  DCL  TAPE_1        LIT '31';
```

```
  DCL  TAPE_2        LIT '32';
  DCL  TAPE_3        LIT '33';
  DCL  TAPE_9        LIT '39';
  DCL  SPECIAL_MENU  LIT '40';
  DCL  SPECIAL_1     LIT '41';
  DCL  SPECIAL_2     LIT '42';
  DCL  SPECIAL_3     LIT '43';
  DCL  SPECIAL_4     LIT '44';
  DCL  SPECIAL_5     LIT '45';
  DCL  SPECIAL_9     LIT '49';
  DCL  COOLING_MENU  LIT '50';
  DCL  COOLING_1     LIT '51';
  DCL  COOLING_2     LIT '52';
  DCL  COOLING_3     LIT '53';
  DCL  COOLING_9     LIT '59';
  DCL  IMAGING_MENU  LIT '60';
  DCL  IMAGING_1     LIT '61';
  DCL  IMAGING_2     LIT '62';
  DCL  IMAGING_3     LIT '63';
  DCL  IMAGING_4     LIT '64';
  DCL  IMAGING_5     LIT '65';
  DCL  IMAGING_9     LIT '69';

  DCL  DISPLAY_TIMER    LIT '0';
  DCL  STANDBY_TIMER    LIT '1';
  DCL  TAPE_TIMER       LIT '2';
  DCL  AUTO_TIMER       LIT '3';

  CALL INIT_HW;
  CALL DISPLAY_INIT;

  IF POWER_CHECK = 12345
    THEN DO;
      SCREEN = POWER_ON;
      POWER_CHECK = 0;
    END;
    ELSE DO;
      SCREEN = MAIN_STATUS;
      CALL WAIT (500000);
    END;

  MESSAGE = .('          ');
  MODE = OFF;
  OLD_SCREEN = SCREEN;
  SCREEN_CHANGE, PANEL_ACTIVE = TRUE;
  TOGGLE, OLD_TOGGLE = TOGGLE_SWITCH;
  POS, BFR_IMAGES, BFR_READ_PTR, BFR_WRITE_PTR = 0;
  DISK_MODE = IDLE;
  ANALYZING_IMAGE = FALSE;
  ARCHIVE_FAILED, ARCHIVE_ABORT = FALSE;
  AUTO_MODE = A_OFF;
  AUTO_IMAGES = 0;
  BATTERY = 1;

DO FOREVER;

  /*------- Battery monitor section -------------*/

  DO;
    DCL BATT INTEGER;
```

```
  BATT = BATTERY_VOLTS;
  IF MODE <> STANDBY THEN BATT = BATT + 50;
  DO CASE BATTERY;
    IF BATT > 1220
      THEN BATTERY = 1;
    IF BATT < 1080
      THEN BATTERY = 0;
      ELSE IF BATT > 1200
        THEN BATTERY = 2;
    IF BATT < 1150
      THEN BATTERY = 1;
      ELSE IF BATT > 1270
        THEN BATTERY = 3;
    IF BATT < 1220
      THEN BATTERY = 2;
  END;
  IF BATTERY_VOLTS < 800
    THEN DO;
      POWER_CHECK = 12345;
      CALL DISPLAY_CLEAR;
      ENABLE_MICRO_POWER = FALSE;
    END;
END;

/*-------- Auto timer section -----------*/

IF AUTO_MODE = A_ON
  THEN DO;
    IF (AUTO_HOUR = UNSIGN (VALUE (HOUR))) AND
        (AUTO_MINUTE = UNSIGN (VALUE (MINUTE)))
      THEN DO;
        AUTO_MODE = A_ARMED;
        CALL TIMER (AUTO_TIMER,1);
      END;
  END;

IF AUTO_MODE = A_ARMED
  THEN DO;
    IF AUTO_IMAGES > 0
      THEN DO;
        IF TIMEOUT (AUTO_TIMER)
          THEN DO;
            AUTO_MODE = A_EXPOSE;
            CALL TIMER (AUTO_TIMER,AUTO_PERIOD * 1000000);
          END;
      END;
      ELSE AUTO_MODE = A_ON;
  END;

/*------- Camera LED control section --------------*/

IF ARCHIVE_PRESENT
  THEN CALL LED_OFF;
  ELSE DO;
    IF (BFR_IMAGES + DISK_IMAGES < 50) AND
        (BFR_IMAGES < 6) AND
         (BATTERY >= 2)
      THEN CALL LED_GREEN_BLINK;
      ELSE IF (BFR_IMAGES + DISK_IMAGES < 60) AND
                (BFR_IMAGES < 6) AND
```

```
                (BATTERY >= 1)
          THEN CALL LED_YELLOW_BLINK;
          ELSE CALL LED_RED_BLINK;
    END;

  /*--------- Display control section -------*/

  DO;
    TOGGLE = TOGGLE_SWITCH;
    IF TOGGLE <> OLD_TOGGLE
      THEN DO;
        TOGGLE_CHANGE = TRUE;
        OLD_TOGGLE = TOGGLE;
      END;
      ELSE TOGGLE_CHANGE = FALSE;
  END;

$TITLE (EOSCREEN) INCLUDE (EOSCREEN.INC)

  DO;
    IF SCREEN_CHANGE
      THEN SCREEN_CHANGE, PANEL_ACTIVE = FALSE;
    IF SCREEN <> OLD_SCREEN
      THEN DO;
        SCREEN_CHANGE, PANEL_ACTIVE = TRUE;
        OLD_SCREEN = SCREEN;
        CALL DISPLAY_CLEAR;
        CALL DISPLAY_CURSOR (32);
      END;
  END;

  /*-------- Main control section -----------*/

  DO;

    DO_CASE;

      IF MODE = OFF
        THEN DO;
          IF SCREEN = MAIN_STATUS THEN MODE = STANDBY;
        END_DO;

      IF MODE = STANDBY
        THEN DO;
          IF NOT PANEL_ACTIVE
            THEN CALL WAIT_EVENT;
          IF ARCHIVE_PRESENT
            THEN DO;
              IF (DISK_IMAGES > 0) AND (BATTERY > 0) AND
                  NOT (ARCHIVE_FAILED OR ARCHIVE_ABORT)
                THEN DO;
                  MODE = START_ARCHIVE;
                  ENABLE_IMAGE_POWER = TRUE;
                  CCD_CLOCK_MODE = DARK;
                  DISK_MODE = IDLE;
                  ANALYZING_IMAGE = FALSE;
                  CALL WAIT (50000);
                  CALL TIMER (TAPE_TIMER,70000000);
                END;
            END;
```

```
      ELSE DO;
        IF CAMERA_ON OR (NOT AUTO_PWR_OFF)
          THEN DO;
            MODE = FLUSH;
            ENABLE_IMAGE_POWER = TRUE;
            CCD_CLOCK_MODE = HOR_VER_DARK;
            CAMERA_PWR_DISABLE = TRUE;
            CALL WAIT (250000);
            BFR_IMAGE_NO(0),BFR_IMAGE_NO(1),
             BFR_IMAGE_NO(2),BFR_IMAGE_NO(3),
              BFR_IMAGE_NO(4),BFR_IMAGE_NO(5) = 0;
            BFR_WRITE_PTR, BFR_READ_PTR = 0;
            DISK_MODE = IDLE;
            ANALYZING_IMAGE = FALSE;
            CALL TIMER (STANDBY_TIMER,10000000);
          END;
        IF (BATTERY > 0) AND
            (DISK_IMAGES < 60) AND
             (NOT ARCHIVE_PRESENT) AND
              ((AUTO_MODE = A_OFF) OR
               (AUTO_MODE = A_EXPOSE) OR
                (AUTO_MODE = A_MANUAL))
          THEN CAMERA_PWR_DISABLE = FALSE;
          ELSE CAMERA_PWR_DISABLE = TRUE;
      END;
  END_DO;

IF MODE = FLUSH
  THEN DO;
    IF CAMERA_SHUTTER_OPEN
      THEN DO;
        MODE = EXPOSE;
        ANALYZING_IMAGE = FALSE;
        CALL CAMERA_QUIET;
        CCD_CLOCK_MODE = HOR_DARK;
        CALL WAIT (3000);
        CCD_CLOCK_MODE = HOR;
      END;
    IF SCSI_DISK_READY AND NOT SCSI_BUSY
      THEN DO;
        IF SCSI_ERROR <> 0
          THEN DISK_MODE = IDLE;
        IF DISK_MODE = CHECKING
          THEN DO;
            DISK_MODE = IDLE;
            CALL BUFFER_READ (.DATA_AREA,128,23);
            IF (DATA_AREA(37) = HIGH (HIGH (CHK_KEY))) AND
                (DATA_AREA(38) = LOW (HIGH (CHK_KEY))) AND
                 (DATA_AREA(39) = HIGH (LOW (CHK_KEY))) AND
                  (DATA_AREA(40) = LOW (LOW (CHK_KEY)))
              THEN DO;
                DISK_IMAGES,OLD_DISK_IMAGES =
                   DISK_IMAGES + 1;
                DISK_IMAGE_NO(DISK_WRITE_PTR) =
                   BFR_IMAGE_NO(BFR_READ_PTR);
                DISK_IMAGE_TIME(DISK_WRITE_PTR) =
                   BFR_IMAGE_TIME(BFR_READ_PTR);
                DISK_WRITE_PTR =
                   (DISK_WRITE_PTR + 1) MOD 60;
                OLD_DISK_READ_PTR = DISK_READ_PTR;
```

```
            BFR_IMAGES = BFR_IMAGES - 1;
            BFR_READ_PTR = (BFR_READ_PTR + 1) MOD 6;
          END;
      END;
    IF DISK_MODE = RECORDING
      THEN DO;
        DISK_MODE = CHECKING;
        CALL SCSI_DISK_READ (4002 +
         (DOUBLE (DOUBLE (DISK_WRITE_PTR)) * 3403),
          1,20);
      END;
    IF (DISK_MODE = IDLE) AND (BFR_IMAGES > 0)
      THEN DO;
        DISK_MODE = RECORDING;
        CHK_KEY = REAL_TIME;
        CALL BUFFER_WRITE
         (.CHK_KEY,8,11079 + (BFR_READ_PTR * 10890));
        CALL SCSI_DISK_WRITE_10 (600 +
         (DOUBLE (DOUBLE (DISK_WRITE_PTR)) * 3403),
          3403,190 + (BFR_READ_PTR * 10890));
      END;
  END;
IF BATTERY = 0
  THEN AUTO_PWR_OFF = TRUE;
IF (BATTERY > 0) AND
    (BFR_IMAGES < 6) AND
     (BFR_IMAGES + DISK_IMAGES < 60) AND
      ((AUTO_MODE = A_OFF) OR
       (AUTO_MODE = A_EXPOSE) OR
        (AUTO_MODE = A_MANUAL))
  THEN CAMERA_PWR_DISABLE = FALSE;
  ELSE CAMERA_PWR_DISABLE = TRUE;
IF CAMERA_ON OR
    (BFR_IMAGES > 0) OR
     (NOT AUTO_PWR_OFF)
  THEN DO;
    CALL TIMER (STANDBY_TIMER,10000000);
    IF (DISK_MODE = STOPPED) AND NOT SCSI_BUSY
      THEN DO;
        CALL SCSI_DISK_START;
        DISK_MODE = IDLE;
      END;
  END;
  ELSE DO;
    IF TIMEOUT (STANDBY_TIMER) AND NOT SCSI_BUSY
      THEN IF DISK_MODE = STOPPED
        THEN DO;
          MODE = STANDBY;
          ENABLE_IMAGE_POWER = FALSE;
          CAMERA_PWR_DISABLE = TRUE;
          CALL WAIT (500000);
          BFR_IMAGE_NO(0),BFR_IMAGE_NO(1),
           BFR_IMAGE_NO(2),BFR_IMAGE_NO(3),
            BFR_IMAGE_NO(4),BFR_IMAGE_NO(5) = 0;
          BFR_WRITE_PTR, BFR_READ_PTR = 0;
        END;
        ELSE DO;
          CALL SCSI_DISK_STOP;
          DISK_MODE = STOPPED;
        END;
```

```
      END;
  END_DO;

IF MODE = EXPOSE
  THEN DO;
    IF NOT CAMERA_SHUTTER_OPEN
      THEN DO;
        MODE = FLUSH;
        BFR_IMAGES = BFR_IMAGES + 1;
        IF (BFR_IMAGES + DISK_IMAGES >= 60) OR
            (AUTO_MODE = A_EXPOSE) OR
             (AUTO_MODE = A_MANUAL) OR
              (BFR_IMAGES >= 6) OR
               (BATTERY = 0)
          THEN CAMERA_PWR_DISABLE = TRUE;
        IF AUTO_MODE = A_EXPOSE
          THEN DO;
            AUTO_MODE = A_ARMED;
            IF AUTO_IMAGES > 0
              THEN AUTO_IMAGES = AUTO_IMAGES - 1;
          END;
          ELSE IF AUTO_MODE = A_MANUAL
                 THEN AUTO_MODE = A_ON;
        BFR_IMAGE_NO(BFR_WRITE_PTR) = IMAGE_NO;
        CALL CLOCK_TIME (.TIME);
        BFR_IMAGE_TIME(BFR_WRITE_PTR) = UNIX_TIME (.TIME);
        IMAGE_NO = IMAGE_NO + 1;
        CALL IMAGE_XFER_START
          (190 + (BFR_WRITE_PTR * 10890));
        ANALYZER_ADDRESS = 900 + (BFR_WRITE_PTR * 10890);
        DO ANALYZER_COUNT = 0 TO 15;
          BUCKET(ANALYZER_COUNT) = 0;
        END;
        ANALYZER_COUNT = 0;
        ANALYZING_IMAGE = TRUE;
        BFR_WRITE_PTR = (BFR_WRITE_PTR + 1) MOD 6;
      END;
  END_DO;

IF MODE = START_ARCHIVE
  THEN DO;
    IF (NOT SCSI_BUSY) AND TIMEOUT (TAPE_TIMER)
      THEN DO;
        MODE = OPEN;
        CALL SCSI_TAPE_UNLOAD;
      END;
    IF ARCHIVE_ABORT
      THEN DO;
        MODE = ARCHIVE_DONE;
        CALL TIMER (STANDBY_TIMER,5000000);
      END;
  END_DO;

IF MODE = OPEN
  THEN DO;
    IF NOT SCSI_BUSY
      THEN DO;
        IF SCSI_ERROR <> 0
          THEN CALL SCSI_TAPE_UNLOAD;
          ELSE DO;
```

```
            MODE = LOAD;
            CALL TIMER (STANDBY_TIMER,300000000);
          END;
      END;
  END_DO;

IF MODE = LOAD
  THEN DO;
    IF SCSI_TAPE_READY AND SCSI_DISK_READY AND
        NOT SCSI_BUSY
      THEN DO;
        MODE = START_IMAGE;
        OLD_DISK_READ_PTR = DISK_READ_PTR;
        OLD_DISK_IMAGES = DISK_IMAGES;
        CALL SCSI_DISK_READ (600 +
          (DOUBLE (DOUBLE (DISK_READ_PTR)) * 3403),1,24);
      END;
    IF TIMEOUT (STANDBY_TIMER)
      THEN ARCHIVE_ABORT = TRUE;
    IF ARCHIVE_ABORT
      THEN DO;
        MODE = ARCHIVE_DONE;
        CALL TIMER (STANDBY_TIMER,5000000);
      END;
  END_DO;

IF MODE = START_IMAGE
  THEN DO;
    IF SCSI_TAPE_READY AND NOT SCSI_BUSY
      THEN DO;
        DCL COUNT BYTE, CHECKSUM WORD;
        MODE = WRITE_HEADER;
        DO COUNT = 0 TO 255;
          DATA_AREA(COUNT) = 0;
        END;
        CALL MOVE (5,.('eo'),.DATA_AREA(0));
        CALL TO_DECIMAL
          (.DATA_AREA(2),DISK_IMAGE_NO(DISK_READ_PTR));
        CALL MOVE (7,.('   777 '),.DATA_AREA(100));
        CALL MOVE (7,.('     0 '),.DATA_AREA(108));
        CALL MOVE (7,.('     0 '),.DATA_AREA(116));
        CALL MOVE (12,.('    6517000 '),.DATA_AREA(124));
        DATA_AREA(136) = ' ';
        CALL TO_OCTAL
          (.DATA_AREA(137),DISK_IMAGE_TIME(DISK_READ_PTR));
        DATA_AREA(147) = ' ';
        CALL MOVE (8,.('        '),.DATA_AREA(148));
        DATA_AREA(156) = 0;
        CHECKSUM = 0;
        DO COUNT = 0 TO 156;
          CHECKSUM = CHECKSUM + DATA_AREA(COUNT);
        END;
        DATA_AREA(154) = 0;
        CALL TO_OCTAL (.DATA_AREA(400),CHECKSUM);
        DO COUNT = 6 TO 9;
          DATA_AREA(144 + COUNT) = DATA_AREA(400 + COUNT);
        END;
        CALL BUFFER_WRITE (.DATA_AREA(0),128,20);
        CALL BUFFER_WRITE (.DATA_AREA(128),128,21);
        CALL BUFFER_FILL (0,128,22);
```

```
        CALL BUFFER_FILL (0,128,23);
        CALL SCSI_TAPE_WRITE (1,20);
      END;
  END_DO;

IF MODE = WRITE_HEADER
  THEN DO;
    IF SCSI_TAPE_READY AND NOT SCSI_BUSY
      THEN DO;
        IF SCSI_ERROR <> 0
          THEN DO;
            MODE = UNLOAD;
            CALL SCSI_TAPE_UNLOAD;
            ARCHIVE_FAILED = TRUE;
          END;
          ELSE DO;
            MODE = FROM_DISK;
            COPY_COUNT = 0;
            CALL SCSI_DISK_READ (601 +
              (DOUBLE (DOUBLE (DISK_READ_PTR)) * 3403),
                32,50);
          END;
      END;
  END_DO;

IF MODE = FROM_DISK
  THEN DO;
    IF SCSI_TAPE_READY AND NOT SCSI_BUSY
      THEN DO;
        IF SCSI_ERROR <> 0
          THEN DO;
            MODE = UNLOAD;
            CALL SCSI_TAPE_UNLOAD;
            ARCHIVE_FAILED = TRUE;
          END;
          ELSE DO;
            MODE = TO_TAPE;
            IF COPY_COUNT = 106
              THEN DO;
                DISK_IMAGES = DISK_IMAGES - 1;
                DISK_READ_PTR = (DISK_READ_PTR + 1) MOD 60;
                CALL SCSI_TAPE_WRITE (7,50);
              END;
              ELSE CALL SCSI_TAPE_WRITE (16,50);
          END;
      END;
  END_DO;

IF MODE = TO_TAPE
  THEN DO;
    IF SCSI_TAPE_READY AND NOT SCSI_BUSY
      THEN DO;
        IF SCSI_ERROR <> 0
          THEN DO;
            MODE = UNLOAD;
            CALL SCSI_TAPE_UNLOAD;
            ARCHIVE_FAILED = TRUE;
          END;
          ELSE DO;
            IF COPY_COUNT = 106
```

```
              THEN DO;
                IF (DISK_IMAGES = 0) OR ARCHIVE_ABORT
                  THEN DO;
                    DCL COUNT BYTE;
                    MODE = END_OF_FILE;
                    DO COUNT = 0 TO 16;
                      CALL BUFFER_FILL (0,128,30 + COUNT);
                    END;
                    COPY_COUNT = 0;
                    CALL SCSI_TAPE_WRITE (2,30);
                  END;
                  ELSE DO;
                    MODE = START_IMAGE;
                    CALL SCSI_DISK_READ (600 +
                      (DOUBLE (DOUBLE (DISK_READ_PTR)) *
                        3403),1,24);
                  END;
              END;
              ELSE DO;
                MODE = FROM_DISK;
                COPY_COUNT = COPY_COUNT + 1;
                CALL SCSI_DISK_READ (601 +
                  (DOUBLE (DOUBLE (DISK_READ_PTR)) *
                   3403) + (COPY_COUNT * 32),32,50);
              END;
          END;
      END;
  END_DO;

IF MODE = END_OF_FILE
  THEN DO;
    IF SCSI_TAPE_READY AND NOT SCSI_BUSY
      THEN DO;
        IF SCSI_ERROR <> 0
          THEN DO;
            MODE = UNLOAD;
            CALL SCSI_TAPE_UNLOAD;
            ARCHIVE_FAILED = TRUE;
          END;
          ELSE DO;
            IF COPY_COUNT < 12
              THEN DO;
                COPY_COUNT = COPY_COUNT + 1;
                CALL SCSI_TAPE_WRITE (2,30);
              END;
              ELSE DO;
                MODE = UNLOAD;
                CALL SCSI_TAPE_UNLOAD;
              END;
          END;
      END;
  END_DO;

IF MODE = UNLOAD
  THEN DO;
    IF NOT SCSI_BUSY
      THEN DO;
        MODE = ARCHIVE_DONE;
        CALL TIMER (STANDBY_TIMER,5000000);
      END;
```

```
        END_DO;

      IF MODE = ARCHIVE_DONE
        THEN DO;
          IF TIMEOUT (STANDBY_TIMER) AND NOT SCSI_BUSY
            THEN DO;
              MODE = STANDBY;
              ENABLE_IMAGE_POWER = FALSE;
            END;
        END_DO;

    END_CASE;
  END;

END;

END MAIN;

/*---------------------------------------------------------------*/
```

```
/*===============================================================
  EOSCREEN.INC
---------------------------------------------------------------

  NAME:  Electro-optic camera screen module

  HISTORY: --REV--  --DATE--  --AUTHOR--
              -      5-26-88      JEM

===============================================================*/

  DO;
    IF BATTERY = 0
      THEN MESSAGE = .('LOW   BATT ');
      ELSE IF ARCHIVE_PRESENT
        THEN DO;
          IF ARCHIVE_FAILED
            THEN MESSAGE = .('ERROR     ');
            ELSE IF MODE = START_ARCHIVE
              THEN MESSAGE = .('Test Test ');
              ELSE IF MODE = LOAD
                THEN MESSAGE = .('LOAD  TAPE');
                ELSE IF MODE = START_IMAGE
                  THEN MESSAGE = .('Busy Busy ');
                  ELSE IF ARCHIVE_ABORT
                    THEN MESSAGE = .('ABORT     ');
                    ELSE IF MODE = UNLOAD
                      THEN MESSAGE = .('Done Done ');
        END;
        ELSE DO;
          IF (CAMERA_TEMP < -20) AND (CCD_TEMP < -20)
            THEN MESSAGE = .('CHECKCABLE');
            ELSE IF DISK_IMAGES + BFR_IMAGES >= 60
              THEN MESSAGE = .('DISK  FULL');
              ELSE IF MODE = EXPOSE
                THEN MESSAGE = .('Exp  Exp  ');
                ELSE IF AUTO_MODE = A_ON
                  THEN MESSAGE = .('TimerTimer');
                  ELSE IF AUTO_MODE = A_ARMED
                    THEN MESSAGE = .('Timer  ON ');
                    ELSE MESSAGE = .('ReadyReady');
        END;
  END;

  DO_CASE;

    IF SCREEN = MAIN_STATUS
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_SPECIAL (1);
            CALL DISPLAY_WORDS (0,32,.('  :   [     ]   ',
                                       '               '));
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
            CALL DISPLAY (29,2); CALL DISPLAY (30,3);
          END;
        IF HOUR < 16
          THEN CALL DISPLAY (0,' ');
          ELSE CALL DISPLAY (0,MS_DIGIT(HOUR));
        CALL DISPLAY (1,LS_DIGIT(HOUR));
```

```
    CALL DISPLAY (3,MS_DIGIT(MINUTE));
    CALL DISPLAY (4,LS_DIGIT(MINUTE));
    IF (SECOND AND 00000001B) = 0
      THEN CALL DISPLAY (5,161);
      ELSE CALL DISPLAY (5,223);
    CALL DISPLAY_NUMBER (7,5,0,IMAGE_NO);
    CALL DISPLAY (13,4 + BATTERY);
    CALL DISPLAY (14,MODE);
    IF MODE = STANDBY
      THEN CALL DISPLAY (16,1);
      ELSE CALL DISPLAY (16,' ');
    IF NOT CAMERA_ON
      THEN CALL DISPLAY (17,'x');
      ELSE IF CAMERA_SHUTTER_OPEN
        THEN CALL DISPLAY (17,186);
        ELSE CALL DISPLAY (17,219);
    CALL DISPLAY_NUMBER (18,1,0,BFR_IMAGES);
    IF NOT SCSI_DISK_READY
      THEN CALL DISPLAY (19,47);
      ELSE IF DISK_MODE = RECORDING
        THEN CALL DISPLAY (19,126);
        ELSE CALL DISPLAY (19,45);
    CALL DISPLAY_NUMBER (20,2,0,DISK_IMAGES);
    IF AUTO_MODE = A_ON
      THEN CALL DISPLAY (22,0);
      ELSE CALL DISPLAY (22,' ');
    IF (SECOND AND 00000001B) = 0
      THEN CALL DISPLAY_WORDS (23,5,MESSAGE);
      ELSE CALL DISPLAY_WORDS (23,5,MESSAGE + 5);
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      IF AUTO_MODE = A_ON
        THEN AUTO_MODE = A_MANUAL;
      IF MODE = STANDBY
        THEN SCREEN = POWER_OFF;
      SCREEN = IMAGING_MENU;
      SCREEN = SYSTEM_MENU;
      SCREEN = CLOCK;
      SCREEN = TIMER_SET;
    END;
  END_DO;

IF SCREEN = POWER_ON
  THEN DO;
    PANEL_ACTIVE = TRUE;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' 1 Enter Code 5 ',
                                   ' 2    3  4    6 '));
        CALL DISPLAY (0,0); CALL DISPLAY (15,0);
        CALL DISPLAY (16,1); CALL DISPLAY (31,1);
        CALL DISPLAY (21,2); CALL DISPLAY (26,3);
        CALL TIMER (DISPLAY_TIMER,5000000);
        POS = 0;
      END;
    IF TIMEOUT (DISPLAY_TIMER) AND TOGGLE = 0
      THEN DO;
        POWER_CHECK = 12345;
        ENABLE_MICRO_POWER = FALSE;
      END;
```

```
      IF TOGGLE_CHANGE AND TOGGLE > 0
        THEN DO;
          CALL TIMER (DISPLAY_TIMER,5000000);
          IF CODE (POS) = TOGGLE
            THEN DO;
              POS = POS + 1;
              IF POS = 4 THEN SCREEN = MAIN_STATUS;
            END;
            ELSE POS = 0;
        END;
    END;

  IF SCREEN = POWER_OFF
    THEN DO;
      PANEL_ACTIVE = TRUE;
      IF SCREEN_CHANGE
        THEN DO;
          CALL DISPLAY_WORDS (0,32,.('Power Off       ',
                                     ' HOLD           '));
          CALL DISPLAY (16,1);
          CALL TIMER (DISPLAY_TIMER,800000);
        END;
      IF TOGGLE_CHANGE
        THEN SCREEN = MAIN_STATUS;
      IF TIMEOUT (DISPLAY_TIMER)
        THEN DO;
          POWER_CHECK = 12345;
          CALL DISPLAY_CLEAR;
          ENABLE_MICRO_POWER = FALSE;
        END;
    END;

/*-------------------------------------------------------------*/

  IF SCREEN = TIMER_SET
    THEN DO;
      DCL CURS_POS (*) BYTE DATA (10,13,21,26);
      PANEL_ACTIVE = TRUE;
      IF SCREEN_CHANGE
        THEN DO;
          CALL DISPLAY_WORDS (0,32,.(' + Timer 00:00  ',
                                     ' -         s    '));
          CALL DISPLAY (0,0); CALL DISPLAY (16,1);
          CALL DISPLAY (15,0); CALL DISPLAY (31,1);
          CALL DISPLAY (29,2); CALL DISPLAY (30,3);
          POS = 2;
          CALL TIMER (DISPLAY_TIMER,50000000);
        END;
      IF TOGGLE_CHANGE OR TIMEOUT(DISPLAY_TIMER)
        THEN DO CASE TOGGLE;
          ;
          DO;
            DO CASE POS;
              IF AUTO_HOUR < 23
                THEN AUTO_HOUR = AUTO_HOUR + 1;
                ELSE AUTO_HOUR = 0;
              IF AUTO_MINUTE < 59
                THEN AUTO_MINUTE = AUTO_MINUTE + 1;
                ELSE AUTO_MINUTE = 0;
              IF AUTO_MODE = A_OFF
```

```
          THEN AUTO_MODE = A_ON;
          ELSE IF AUTO_IMAGES < 60
            THEN AUTO_IMAGES = AUTO_IMAGES + 1;
            ELSE IF AUTO_MODE = A_ON
              THEN DO;
                AUTO_MODE = A_OFF;
                AUTO_IMAGES = 0;
              END;
        IF AUTO_PERIOD < 60
          THEN AUTO_PERIOD = AUTO_PERIOD + 5;
          ELSE IF AUTO_PERIOD < 600
            THEN AUTO_PERIOD = AUTO_PERIOD + 30;
            ELSE AUTO_PERIOD = 5;
      END;
      CALL TIMER (DISPLAY_TIMER,100000);
    END;
    DO;
      DO CASE POS;
        IF AUTO_HOUR >0
          THEN AUTO_HOUR = AUTO_HOUR - 1;
          ELSE AUTO_HOUR = 23;
        IF AUTO_MINUTE > 0
          THEN AUTO_MINUTE = AUTO_MINUTE - 1;
          ELSE AUTO_MINUTE = 59;
        IF AUTO_MODE = A_OFF
          THEN DO;
            AUTO_MODE = A_ON;
            AUTO_IMAGES = 60;
          END;
          ELSE IF AUTO_IMAGES > 0
            THEN AUTO_IMAGES = AUTO_IMAGES - 1;
            ELSE IF AUTO_MODE = A_ON
              THEN AUTO_MODE = A_OFF;
        IF AUTO_PERIOD > 60
          THEN AUTO_PERIOD = AUTO_PERIOD - 30;
          ELSE IF AUTO_PERIOD > 5
            THEN AUTO_PERIOD = AUTO_PERIOD - 5;
            ELSE AUTO_PERIOD = 600;
      END;
      CALL TIMER (DISPLAY_TIMER,100000);
    END;
    IF POS = 0
      THEN POS = 3;
      ELSE POS = POS - 1;
    IF POS = 3
      THEN POS = 0;
      ELSE POS = POS + 1;
    SCREEN = MAIN_STATUS;
    SCREEN = CLOCK;
  END;
IF TOGGLE_CHANGE
  THEN CALL TIMER (DISPLAY_TIMER,400000);
CALL DISPLAY_CURSOR (CURS_POS(POS));
CALL DISPLAY_NUMBER (9,2,0,AUTO_HOUR);
CALL DISPLAY_NUMBER (12,2,0,AUTO_MINUTE);
CALL DISPLAY_NUMBER (24,3,0,AUTO_PERIOD);
IF AUTO_MODE = A_OFF
  THEN CALL DISPLAY_WORDS (20,3,.('OFF'));
  ELSE DO;
    CALL DISPLAY (22,'i ');
```

```
        CALL DISPLAY_NUMBER (20,2,0,AUTO_IMAGES);
      END;
  END_DO;

IF SCREEN = CLOCK
  THEN DO;
    DCL CURS_POS (*) BYTE DATA (32,8,11,14,24,27,30);
    PANEL_ACTIVE = TRUE;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' + Clk   /  /   ',
                                   ' -       :  :   '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
        CALL DISPLAY (20,2); CALL DISPLAY (21,3);
        POS = 1;
        CALL DISPLAY_CURSOR (CURS_POS(POS));
      END;
    CALL DISPLAY (7,MS_DIGIT(YEAR));
    CALL DISPLAY (8,LS_DIGIT(YEAR));
    CALL DISPLAY (10,MS_DIGIT(MONTH));
    CALL DISPLAY (11,LS_DIGIT(MONTH));
    CALL DISPLAY (13,MS_DIGIT(DATE));
    CALL DISPLAY (14,LS_DIGIT(DATE));
    CALL DISPLAY (23,MS_DIGIT(HOUR));
    CALL DISPLAY (24,LS_DIGIT(HOUR));
    CALL DISPLAY (26,MS_DIGIT(MINUTE));
    CALL DISPLAY (27,LS_DIGIT(MINUTE));
    CALL DISPLAY (29,MS_DIGIT(SECOND));
    CALL DISPLAY (30,LS_DIGIT(SECOND));
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      DO;
        CALL CLOCK_SET (POS,+1);
        CALL TIMER (DISPLAY_TIMER,400000);
      END;
      DO;
        CALL CLOCK_SET (POS,-1);
        CALL TIMER (DISPLAY_TIMER,400000);
      END;
      DO;
        IF POS <= 1
          THEN POS = 6;
          ELSE POS = POS - 1;
        CALL DISPLAY_CURSOR (CURS_POS(POS));
      END;
      DO;
        IF POS = 6
          THEN POS = 1;
          ELSE POS = POS + 1;
        CALL DISPLAY_CURSOR (CURS_POS(POS));
      END;
      SCREEN = TIMER_SET;
      SCREEN = MAIN_STATUS;
    END;
    IF TIMEOUT (DISPLAY_TIMER) THEN DO CASE TOGGLE;
      ;
      DO;
        CALL CLOCK_SET (POS,+1);
        CALL TIMER (DISPLAY_TIMER,100000);
```

```
          END;
          DO;
            CALL CLOCK_SET (POS,-1);
            CALL TIMER (DISPLAY_TIMER,100000);
          END;
          ;;;;
        END;
      END_DO;

/*-------------------------------------------------------------*/

    IF SCREEN = SYSTEM_MENU
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_WORDS (0,32,.('System          ',
                                       '  Functions     '));
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
            CALL DISPLAY (28,2); CALL DISPLAY (29,3);
          END;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;;;
          SCREEN = MAIN_STATUS;
          SCREEN = DISK_MENU;
          SCREEN = SYSTEM_9;
          SCREEN = SYSTEM_1;
        END;
      END_DO;

    IF SCREEN = SYSTEM_1
      THEN DO;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;
          AUTO_PWR_OFF = TRUE;
          AUTO_PWR_OFF = FALSE;
          ;;
          SCREEN = SYSTEM_MENU;
          SCREEN = SYSTEM_2;
        END;
        IF SCREEN_CHANGE OR TOGGLE_CHANGE
          THEN IF AUTO_PWR_OFF
            THEN CALL DISPLAY_WORDS
              (0,32,.(' AUTO OFF       ',
                      ' Power On       '));
            ELSE CALL DISPLAY_WORDS
              (0,32,.(' Auto Off       ',
                      ' POWER ON       '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END_DO;

    IF SCREEN = SYSTEM_2
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_WORDS (0,32,.('Rec Temp     c  ',
                                       'Battery      v  '));
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
          END;
        CALL DISPLAY_INTEGER (8,5,1,RECORDER_TEMP);
```

```
      CALL DISPLAY_INTEGER (23,6,2,BATTERY_VOLTS);
      IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
        ;;;;;
        SCREEN = SYSTEM_1;
        SCREEN = SYSTEM_3;
      END;
    END_DO;

  IF SCREEN = SYSTEM_3
    THEN DO;
      IF SCREEN_CHANGE
        THEN DO;
          CALL DISPLAY_WORDS (0,32,.(' Clear Buffer   ',
                                     ' Reset Micro    '));
          CALL DISPLAY (0,0); CALL DISPLAY (16,1);
          CALL DISPLAY (15,0); CALL DISPLAY (31,1);
        END;
      IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
        ;
        BFR_IMAGES, BFR_READ_PTR, BFR_WRITE_PTR = 0;
        CALL RESET;
        ;;
        SCREEN = SYSTEM_2;
        SCREEN = SYSTEM_9;
      END;
    END_DO;

  IF SCREEN = SYSTEM_9
    THEN DO;
      IF SCREEN_CHANGE
        THEN DO;
          CALL DISPLAY_WORDS (0,32,.(' Abort Archive  ',
                                     ' Restore Disk   '));
          CALL DISPLAY (0,0); CALL DISPLAY (16,1);
          CALL DISPLAY (15,0); CALL DISPLAY (31,1);
        END;
      IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
        ;
        ARCHIVE_ABORT = TRUE;
        DO;
          DISK_IMAGES = OLD_DISK_IMAGES;
          DISK_READ_PTR = OLD_DISK_READ_PTR;
        END;
        ;;
        SCREEN = SYSTEM_3;
        SCREEN = SYSTEM_MENU;
      END;
    END_DO;

/*-------------------------------------------------------------*/

  IF SCREEN = DISK_MENU
    THEN DO;
      IF SCREEN_CHANGE
        THEN DO;
          CALL DISPLAY_WORDS (0,32,.('Disk            ',
                                     '  Functions     '));
          CALL DISPLAY (15,0); CALL DISPLAY (31,1);
          CALL DISPLAY (28,2); CALL DISPLAY (29,3);
        END;
```

```
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;;;
      SCREEN = SYSTEM_MENU;
      SCREEN = TAPE_MENU;
      SCREEN = DISK_9;
      SCREEN = DISK_1;
    END;
  END_DO;

IF SCREEN = DISK_1
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' Clear Disk     ',
                                   ' Clr Last Img   '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      DISK_IMAGES, DISK_READ_PTR, DISK_WRITE_PTR = 0;
      IF DISK_IMAGES > 0
        THEN DO;
          DISK_IMAGES = DISK_IMAGES - 1;
          IF DISK_WRITE_PTR = 0
            THEN DISK_WRITE_PTR = 59;
            ELSE DISK_WRITE_PTR = DISK_WRITE_PTR - 1;
        END;
      ;;
      SCREEN = DISK_MENU;
      SCREEN = DISK_2;
    END;
  END_DO;

IF SCREEN = DISK_2
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' Test Disk      ',
                                   ' Format Disk    '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_SCSI_STATUS;
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      CALL SCSI_DISK_TEST;
      CALL SCSI_DISK_FORMAT;
      ;;
      SCREEN = DISK_1;
      SCREEN = DISK_3;
    END;
  END_DO;

IF SCREEN = DISK_3
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' Start Disk     ',
                                   ' Stop Disk      '));
```

```
          CALL DISPLAY (0,0); CALL DISPLAY (16,1);
          CALL DISPLAY (15,0); CALL DISPLAY (31,1);
        END;
      CALL DISPLAY_SCSI_STATUS;
      IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
        ;
        CALL SCSI_DISK_START;
        CALL SCSI_DISK_STOP;
        ;;
        SCREEN = DISK_2;
        SCREEN = DISK_9;
      END;
    END_DO;

  IF SCREEN = DISK_9
    THEN DO;
      IF SCREEN_CHANGE
        THEN DO;
          CALL DISPLAY_WORDS (0,32,.(' Write Disk     ',
                                     ' Read Disk      '));
          CALL DISPLAY (0,0); CALL DISPLAY (16,1);
          CALL DISPLAY (15,0); CALL DISPLAY (31,1);
        END;
      CALL DISPLAY_SCSI_STATUS;
      IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
        ;
        CALL SCSI_DISK_WRITE (0,1,0);
        CALL SCSI_DISK_READ (0,1,0);
        ;;
        SCREEN = DISK_3;
        SCREEN = DISK_MENU;
      END;
    END_DO;

/*-------------------------------------------------------------*/

  IF SCREEN = TAPE_MENU
    THEN DO;
      IF SCREEN_CHANGE
        THEN DO;
          CALL DISPLAY_WORDS (0,32,.('Tape            ',
                                     '  Functions     '));
          CALL DISPLAY (15,0); CALL DISPLAY (31,1);
          CALL DISPLAY (28,2); CALL DISPLAY (29,3);
        END;
      IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
        ;;;
        SCREEN = DISK_MENU;
        SCREEN = SPECIAL_MENU;
        SCREEN = TAPE_9;
        SCREEN = TAPE_1;
      END;
    END_DO;

  IF SCREEN = TAPE_1
    THEN DO;
      IF SCREEN_CHANGE
        THEN DO;
          CALL DISPLAY_WORDS (0,32,.(' Erase Tape     ',
                                     ' Unload Tape    '));
```

```
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_SCSI_STATUS;
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      CALL SCSI_TAPE_ERASE;
      CALL SCSI_TAPE_UNLOAD;
      ;;
      SCREEN = TAPE_MENU;
      SCREEN = TAPE_2;
    END;
  END_DO;

IF SCREEN = TAPE_2
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' Filemark       ',
                                   ' Rewind Tape    '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_SCSI_STATUS;
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      CALL SCSI_TAPE_WRITE_FILEMARKS (1);
      CALL SCSI_TAPE_REWIND;
      ;;
      SCREEN = TAPE_1;
      SCREEN = TAPE_3;
    END;
  END_DO;

IF SCREEN = TAPE_3
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' Space +1       ',
                                   ' Space -1       '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_SCSI_STATUS;
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      CALL SCSI_TAPE_SPACE (1,1);
      CALL SCSI_TAPE_SPACE (1,-1);
      ;;
      SCREEN = TAPE_2;
      SCREEN = TAPE_9;
    END;
  END_DO;

IF SCREEN = TAPE_9
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' Write Tape     ',
                                   ' Read Tape      '));
```

```
            CALL DISPLAY (0,0); CALL DISPLAY (16,1);
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
          END;
        CALL DISPLAY_SCSI_STATUS;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;
          CALL SCSI_TAPE_WRITE (1,0);
          CALL SCSI_TAPE_READ (1,0);
          ;;
          SCREEN = TAPE_3;
          SCREEN = TAPE_MENU;
        END;
      END_DO;

/*-------------------------------------------------------------*/

    IF SCREEN = SPECIAL_MENU
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_WORDS (0,32,.('Special         ',
                                       '  Functions     '));
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
            CALL DISPLAY (28,2); CALL DISPLAY (29,3);
          END;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;;;
          SCREEN = TAPE_MENU;
          SCREEN = COOLING_MENU;
          SCREEN = SPECIAL_9;
          SCREEN = SPECIAL_1;
        END;
      END_DO;

    IF SCREEN = SPECIAL_1
      THEN DO;
        DCL CURS_POS (*) BYTE DATA (0,0,8,9,3,4,5,6);
        DCL STRING (8) BYTE;
        DCL (VALUE BASED ADDRS) BYTE;
        PANEL_ACTIVE = TRUE;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_WORDS (0,32,.(' + 0000:00 Hex  ',
                                       ' -              '));
            CALL DISPLAY (0,0); CALL DISPLAY (16,1);
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
            CALL DISPLAY (20,2); CALL DISPLAY (21,3);
            POS = 7;
          END;
        CALL DISPLAY_CURSOR (CURS_POS(POS));
        CALL TO_HEX (.STRING(4),ADDRS);
        CALL DISPLAY_WORDS (3,4,.STRING(4));
        IF ADDRS > 25
          THEN DO;
            CALL TO_HEX (.STRING(0),DOUBLE (VALUE));
            CALL DISPLAY_WORDS (8,2,.STRING(2));
          END;
          ELSE CALL DISPLAY_WORDS (8,2,.('--'));
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;
```

```
      DO;
        IF STRING(POS) = 57
          THEN STRING(POS) = 65;
          ELSE IF STRING(POS) = 70
            THEN STRING(POS) = 48;
            ELSE STRING(POS) = STRING(POS) + 1;
        IF POS > 3
          THEN ADDRS = FROM_HEX (.STRING(4));
          ELSE IF ADDRS > 25
            THEN VALUE = LOW (FROM_HEX (.STRING(0)));
      END;
      DO;
        IF STRING(POS) = 48
          THEN STRING(POS) = 70;
          ELSE IF STRING(POS) = 65
            THEN STRING(POS) = 57;
            ELSE STRING(POS) = STRING(POS) - 1;
        IF POS > 3
          THEN ADDRS = FROM_HEX (.STRING(4));
          ELSE IF ADDRS > 25
            THEN VALUE = LOW (FROM_HEX (.STRING(0)));
      END;
      DO;
        IF POS = 2
          THEN POS = 7;
          ELSE POS = POS - 1;
        CALL DISPLAY_CURSOR (CURS_POS(POS));
      END;
      DO;
        IF POS = 7
          THEN POS = 2;
          ELSE POS = POS + 1;
        CALL DISPLAY_CURSOR (CURS_POS(POS));
      END;
      SCREEN = SPECIAL_MENU;
      SCREEN = SPECIAL_2;
    END;
  END_DO;

IF SCREEN = SPECIAL_2
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' + Disk Image   ',
                                   ' -              '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
        POS = DISK_READ_PTR;
      END;
    CALL DISPLAY_NUMBER (20,2,0,POS);
    CALL DISPLAY_NUMBER (24,5,0,DISK_IMAGE_NO(POS));
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      IF POS = 59
        THEN POS = 0;
        ELSE POS = POS + 1;
      IF POS = 0
        THEN POS = 59;
        ELSE POS = POS - 1;
      ;;
```

```
      SCREEN = SPECIAL_1;
      SCREEN = SPECIAL_3;
    END;
  END_DO;

IF SCREEN = SPECIAL_3
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' + Bfr  Image   ',
                                   ' -             '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
        POS = BFR_READ_PTR;
      END;
    CALL DISPLAY_NUMBER (20,1,0,POS);
    CALL DISPLAY_NUMBER (24,5,0,BFR_IMAGE_NO(POS));
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      IF POS = 5
        THEN POS = 0;
        ELSE POS = POS + 1;
      IF POS = 0
        THEN POS = 5;
        ELSE POS = POS - 1;
      ;;
      SCREEN = SPECIAL_2;
      SCREEN = SPECIAL_4;
    END;
  END_DO;

IF SCREEN = SPECIAL_4
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' Start Clock    ',
                                   ' Stop Clock     '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      CALL CLOCK_START;
      CALL CLOCK_STOP;
      ;;
      SCREEN = SPECIAL_3;
      SCREEN = SPECIAL_5;
    END;
  END_DO;

IF SCREEN = SPECIAL_5
  THEN DO;
    PANEL_ACTIVE = TRUE;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' +  Clock adj   ',
                                   ' -      sec/mo  '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
```

```
        CALL DISPLAY_INTEGER (19,4,0,CLOCK_CAL * 5);
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;
          IF CLOCK_CAL < 25
            THEN CLOCK_CAL = CLOCK_CAL + 1;
          IF CLOCK_CAL > -25
            THEN CLOCK_CAL = CLOCK_CAL - 1;
          ;;
          SCREEN = SPECIAL_4;
          SCREEN = SPECIAL_9;
        END;
      END_DO;

    IF SCREEN = SPECIAL_9
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_WORDS (0,32,.(' Write Bfr      ',
                                       ' Read Bfr       '));
            CALL DISPLAY (0,0); CALL DISPLAY (16,1);
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
          END;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;
          DO;
            CALL CLOCK_TIME (.DATA_AREA);
            CALL BUFFER_WRITE (.DATA_AREA,128,0);
          END;
          CALL BUFFER_READ (.DATA_AREA,128,0);
          ;;
          SCREEN = SPECIAL_5;
          SCREEN = SPECIAL_MENU;
        END;
      END_DO;

/*-------------------------------------------------------------*/

    IF SCREEN = COOLING_MENU
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_WORDS (0,32,.('Cooling         ',
                                       '  Functions     '));
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
            CALL DISPLAY (28,2); CALL DISPLAY (29,3);
          END;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;;;
          SCREEN = SPECIAL_MENU;
          SCREEN = IMAGING_MENU;
          SCREEN = COOLING_9;
          SCREEN = COOLING_1;
        END;
      END_DO;

    IF SCREEN = COOLING_1
      THEN DO;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;
          ENABLE_COOLER = TRUE;
```

```
      ENABLE_COOLER = FALSE;
      ;;
      SCREEN = COOLING_MENU;
      SCREEN = COOLING_2;
    END;
    IF SCREEN_CHANGE OR TOGGLE_CHANGE
      THEN IF ENABLE_COOLER
        THEN CALL DISPLAY_WORDS
          (0,32,.(' COOLING ON     ',
                  ' Cooling Off    '));
        ELSE CALL DISPLAY_WORDS
          (0,32,.(' Cooling On     ',
                  ' COOLING OFF    '));
    CALL DISPLAY (0,0); CALL DISPLAY (16,1);
    CALL DISPLAY (15,0); CALL DISPLAY (31,1);
  END_DO;

IF SCREEN = COOLING_2
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.('Cam Temp     c  ',
                                   'Cam Hum    %rh  '));
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_INTEGER (8,5,1,CAMERA_TEMP);
    CALL DISPLAY_INTEGER (24,3,0,HUMIDITY);
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;;;;;
      SCREEN = COOLING_1;
      SCREEN = COOLING_3;
    END;
  END_DO;

IF SCREEN = COOLING_3
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.('CCD Temp     c  ',
                                   'Dew Temp     c  '));
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_INTEGER (8,5,1,CCD_TEMP);
    CALL DISPLAY_INTEGER (24,5,1,DEW_TEMP);
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;;;;;
      SCREEN = COOLING_2;
      SCREEN = COOLING_9;
    END;
  END_DO;

IF SCREEN = COOLING_9
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.('Cooler       v  ',
                                   '                  '));
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_INTEGER (7,6,2,COOLER_VOLTS);
```

```
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;;;;;
          SCREEN = COOLING_3;
          SCREEN = COOLING_MENU;
        END;
      END_DO;

/*-------------------------------------------------------------*/

    IF SCREEN = IMAGING_MENU
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_WORDS (0,32,.('Imaging         ',
                                       '  Functions     '));
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
            CALL DISPLAY (28,2); CALL DISPLAY (29,3);
          END;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;;;
          SCREEN = COOLING_MENU;
          SCREEN = MAIN_STATUS;
          SCREEN = IMAGING_9;
          SCREEN = IMAGING_1;
        END;
      END_DO;

    IF SCREEN = IMAGING_1
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_SPECIAL (2);
            CALL DISPLAY_WORDS (0,32,.('                ',
                                       ' Histogram      '));
            CALL DISPLAY (15,0); CALL DISPLAY (31,1);
          END;
        DO POS = 0 TO 11;
          IF BUCKET(POS) < 2
            THEN CALL DISPLAY (POS,95);
            ELSE IF BUCKET(POS) < 25
              THEN CALL DISPLAY (POS,4);
              ELSE IF BUCKET(POS) < 50
                THEN CALL DISPLAY (POS,5);
                ELSE IF BUCKET(POS) < 100
                  THEN CALL DISPLAY (POS,6);
                  ELSE CALL DISPLAY (POS,7);
        END;
        IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
          ;;;;;
          SCREEN = IMAGING_MENU;
          SCREEN = IMAGING_2;
        END;
      END_DO;

    IF SCREEN = IMAGING_2
      THEN DO;
        IF SCREEN_CHANGE
          THEN DO;
            CALL DISPLAY_WORDS (0,32,.(' V:0000 Pixel:  ',
                                       '   H:0000  000  '));
```

```
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
        CALL DISPLAY (17,2); CALL DISPLAY (18,3);
        CALL TIMER (DISPLAY_TIMER,10);
      END;
    IF TIMEOUT (DISPLAY_TIMER) OR TOGGLE_CHANGE
      THEN CALL DISPLAY_NUMBER (27,3,0,PIXEL
        ((BFR_WRITE_PTR + 5) MOD 6,X,Y));
    IF TOGGLE_CHANGE
      THEN DO;
        CALL TIMER (DISPLAY_TIMER,500000);
        PANEL_ACTIVE = TRUE;
        DO CASE TOGGLE;
          PANEL_ACTIVE = FALSE;
          IF Y = 0     THEN Y = 1037; ELSE Y = Y - 1;
          IF Y >= 1037 THEN Y = 0;    ELSE Y = Y + 1;
          IF X = 0     THEN X = 1343; ELSE X = X - 1;
          IF X >= 1343 THEN X = 0;    ELSE X = X + 1;
          SCREEN = IMAGING_1;
          SCREEN = IMAGING_3;
        END;
      END;
    IF TIMEOUT (DISPLAY_TIMER)
      THEN DO;
        CALL TIMER (DISPLAY_TIMER,20000);
        DO CASE TOGGLE;
          PANEL_ACTIVE = FALSE;
          IF Y = 0     THEN Y = 1037; ELSE Y = Y - 1;
          IF Y >= 1037 THEN Y = 0;    ELSE Y = Y + 1;
          IF X = 0     THEN X = 1343; ELSE X = X - 1;
          IF X >= 1343 THEN X = 0;    ELSE X = X + 1;
          ;;
        END;
      END;
    CALL DISPLAY_NUMBER (3,4,0,Y);
    CALL DISPLAY_NUMBER (21,4,0,X);
  END_DO;

IF SCREEN = IMAGING_3
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.('Pre          v  ',
                                   'Post         v  '));
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_INTEGER (7,6,2,PRE_LOG_ZERO);
    CALL DISPLAY_INTEGER (23,6,2,POST_LOG_ZERO);
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;;;;;
      SCREEN = IMAGING_2;
      SCREEN = IMAGING_4;
    END;
  END_DO;

IF SCREEN = IMAGING_4
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.('Video        v  ',
```

```
                                    '                '));
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    CALL DISPLAY_INTEGER (8,5,2,VIDEO);
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;;;;;
      SCREEN = IMAGING_3;
      SCREEN = IMAGING_5;
    END;
  END_DO;

IF SCREEN = IMAGING_5
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' Capture Image  ',
                                    '                '));
        CALL DISPLAY (0,0);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      IF BFR_IMAGES = 0
        THEN DO;
          CALL IMAGE_XFER_START
           (190 + (BFR_WRITE_PTR * 10890));
          BFR_WRITE_PTR = (BFR_WRITE_PTR + 1) MOD 6;
        END;
      ;;;
      SCREEN = IMAGING_4;
      SCREEN = IMAGING_9;
    END;
  END_DO;

IF SCREEN = IMAGING_9
  THEN DO;
    IF SCREEN_CHANGE
      THEN DO;
        CALL DISPLAY_WORDS (0,32,.(' CCD Clk Mode:  ',
                                    '                '));
        CALL DISPLAY (0,0); CALL DISPLAY (16,1);
        CALL DISPLAY (15,0); CALL DISPLAY (31,1);
      END;
    IF CCD_CLOCK_MODE = DARK
      THEN CALL DISPLAY_WORDS (17,12,.('        Dark'));
      ELSE IF CCD_CLOCK_MODE = HOR_DARK
        THEN CALL DISPLAY_WORDS (17,12,.('Hor     Dark'));
        ELSE IF CCD_CLOCK_MODE = HOR_VER
          THEN CALL DISPLAY_WORDS (17,12,.('Hor Ver     '));
          ELSE IF CCD_CLOCK_MODE = HOR_VER_DARK
            THEN CALL DISPLAY_WORDS (17,12,.('Hor Ver Dark'));
            ELSE CALL DISPLAY_WORDS (17,12,.('Hor         '));
    IF TOGGLE_CHANGE THEN DO CASE TOGGLE;
      ;
      IF CCD_CLOCK_MODE = DARK
        THEN CCD_CLOCK_MODE = HOR_VER_DARK;
        ELSE IF CCD_CLOCK_MODE = HOR_VER_DARK
          THEN CCD_CLOCK_MODE = HOR_DARK;
          ELSE IF CCD_CLOCK_MODE = HOR_DARK
            THEN CCD_CLOCK_MODE = HOR;
```

```
            ELSE IF CCD_CLOCK_MODE = HOR
              THEN CCD_CLOCK_MODE = HOR_VER;
              ELSE CCD_CLOCK_MODE = DARK;
        IF CCD_CLOCK_MODE = DARK
          THEN CCD_CLOCK_MODE = HOR_VER;
          ELSE IF CCD_CLOCK_MODE = HOR_VER
            THEN CCD_CLOCK_MODE = HOR;
            ELSE IF CCD_CLOCK_MODE = HOR
              THEN CCD_CLOCK_MODE = HOR_DARK;
              ELSE IF CCD_CLOCK_MODE = HOR_DARK
                THEN CCD_CLOCK_MODE = HOR_VER_DARK;
                ELSE CCD_CLOCK_MODE = DARK;
        ;;
        SCREEN = IMAGING_5;
        SCREEN = IMAGING_MENU;
      END;
    END_DO;

END_CASE;
```

```
/*===============================================================
  EOSCSI.INC
---------------------------------------------------------------

  NAME:  Electro-optic camera SCSI module

  HISTORY: --REV--  --DATE--  --AUTHOR--
              -      5-31-88      JEM

===============================================================*/

DCL  SCSI_RD_XFER_COUNT_LO  BYTE AT (.SCSI_READ + 0);
DCL  SCSI_RD_XFER_COUNT_HI  BYTE AT (.SCSI_READ + 1);
DCL  SCSI_RD_FIFO           BYTE AT (.SCSI_READ + 2);
DCL  SCSI_RD_COMMAND        BYTE AT (.SCSI_READ + 3);
DCL  SCSI_RD_STATUS         BYTE AT (.SCSI_READ + 4);
DCL  SCSI_RD_INT_STATUS     BYTE AT (.SCSI_READ + 5);
DCL  SCSI_RD_SEQ_STEP       BYTE AT (.SCSI_READ + 6);
DCL  SCSI_RD_FIFO_FLAGS     BYTE AT (.SCSI_READ + 7);
DCL  SCSI_RD_CONFIGURATION  BYTE AT (.SCSI_READ + 8);

DCL  SCSI_WR_XFER_COUNT_LO  BYTE AT (.SCSI_WRITE + 0);
DCL  SCSI_WR_XFER_COUNT_HI  BYTE AT (.SCSI_WRITE + 1);
DCL  SCSI_WR_FIFO           BYTE AT (.SCSI_WRITE + 2);
DCL  SCSI_WR_COMMAND        BYTE AT (.SCSI_WRITE + 3);
DCL  SCSI_WR_BUS_ID         BYTE AT (.SCSI_WRITE + 4);
DCL  SCSI_WR_TIMEOUT        BYTE AT (.SCSI_WRITE + 5);
DCL  SCSI_WR_SYNC_PERIOD    BYTE AT (.SCSI_WRITE + 6);
DCL  SCSI_WR_SYNC_OFFSET    BYTE AT (.SCSI_WRITE + 7);
DCL  SCSI_WR_CONFIGURATION  BYTE AT (.SCSI_WRITE + 8);
DCL  SCSI_WR_CLOCK_FACTOR   BYTE AT (.SCSI_WRITE + 9);
DCL  SCSI_WR_TEST           BYTE AT (.SCSI_WRITE + 10);

DCL  CTL_CMD_COMPLETE  LIT '00010001B';
DCL  CTL_FLUSH_FIFO    LIT '00000001B';
DCL  CTL_MSG_ACCEPTED  LIT '00010010B';
DCL  CTL_NOP           LIT '00000000B';
DCL  CTL_RESET_BUS     LIT '00000011B';
DCL  CTL_RESET_CHIP    LIT '00000010B';
DCL  CTL_SELECT_ATN    LIT '01000010B';
DCL  CTL_SET_ATN       LIT '00011010B';
DCL  CTL_XFER_DMA      LIT '10010000B';
DCL  CTL_XFER_PAD      LIT '10011000B';
DCL  CTL_XFER_INFO     LIT '00010000B';

DCL  MSG_ABORT         LIT '006H';
DCL  MSG_RESET         LIT '00CH';
DCL  MSG_COMPLETE      LIT '000H';
DCL  MSG_ERROR         LIT '005H';
DCL  MSG_IDENTIFY      LIT '080H';
DCL  MSG_INVALID       LIT '0FFH';
DCL  MSG_NOP           LIT '008H';
DCL  MSG_SAVE_PTR      LIT '002H';

DCL  STS_BUSY          LIT '008H';
DCL  STS_CHECK         LIT '002H';
DCL  STS_GOOD          LIT '000H';

DCL  CMD_ERASE         LIT '019H';
DCL  CMD_FORMAT        LIT '004H';
```

```
DCL  CMD_LOAD_UNLOAD   LIT '01BH';
DCL  CMD_RD_DEFECT     LIT '037H';
DCL  CMD_READ          LIT '008H';
DCL  CMD_READ_EXT      LIT '028H';
DCL  CMD_REQ_SENSE     LIT '003H';
DCL  CMD_REZERO        LIT '001H';
DCL  CMD_REWIND        LIT '001H';
DCL  CMD_SPACE         LIT '011H';
DCL  CMD_SEEK          LIT '00BH';
DCL  CMD_SND_DIAG      LIT '01DH';
DCL  CMD_START_STOP    LIT '01BH';
DCL  CMD_TST_READY     LIT '000H';
DCL  CMD_WRITE         LIT '00AH';
DCL  CMD_WRITE_EXT     LIT '02AH';
DCL  CMD_WRITE_MARKS   LIT '010H';
DCL  CMD_MODE_SELECT   LIT '015H';

DCL  DISK  LIT '0';
DCL  TAPE  LIT '1';

/*-------------------------------------------------------------*/

SCSI_INIT: PROCEDURE;

  DCL COUNT BYTE;

  /* Initialize controller */

  SCSI_WR_COMMAND = CTL_RESET_CHIP;
  SCSI_WR_COMMAND = CTL_NOP;
  SCSI_WR_CLOCK_FACTOR = 3;
  SCSI_WR_CONFIGURATION = 10000010B;
  SCSI_WR_SYNC_PERIOD = 5;
  SCSI_WR_SYNC_OFFSET = 0;
  SCSI_WR_TIMEOUT = 2;

  /* Initialize variables */

  DO COUNT = 0 TO 9;
    COMMAND(COUNT) = 0;
  END;
  DO COUNT = 0 TO 199;
    BUFFER(COUNT) = 0;
  END;

  SCSI_DMA_XFER = FALSE;
  DMA_INTERRUPTED = FALSE;
  FIFO_COUNT, RESIDUAL_COUNT = 0;
  STATUS_IN_FIFO = FALSE;
  TARGET_STATUS = STS_GOOD;
  TARGET_MESSAGE = MSG_INVALID;
  POINTER = 0;
  TOTAL_XFER_COUNT = 0;
  SCSI_DISK_READY, SCSI_TAPE_READY = FALSE;
  DISK_READY_COUNT, TAPE_READY_COUNT = 0;
  SCSI_BUSY = FALSE;
  SCSI_ERROR = 0;
  TEN_BIT = FALSE;

  /* Issue first command */
```

```
  SCSI_WR_COMMAND = CTL_FLUSH_FIFO;
  SCSI_WR_BUS_ID, ID = DISK;
  SCSI_WR_FIFO = MSG_IDENTIFY;
  SCSI_WR_FIFO, CURRENT_COMMAND = CMD_TST_READY;
  DO COUNT = 1 TO 5;
    SCSI_WR_FIFO = 0;
  END;
  SCSI_WR_COMMAND = CTL_SELECT_ATN;

END SCSI_INIT;

/*-------------------------------------------------------------*/

SCSI_DISK_READ: PROCEDURE (BLOCK,LEN,ADDR);

  DCL BLOCK DWORD, LEN WORD, ADDR WORD;

  IF SCSI_BUSY THEN RETURN;
  BUFFER_ADDRESS = ADDR;
  TOTAL_XFER_COUNT = DOUBLE(LEN) * 512;
  COMMAND(0) = CMD_READ_EXT;
  COMMAND(1) = 0;
  COMMAND(2) = HIGH (HIGH (BLOCK));
  COMMAND(3) = LOW (HIGH (BLOCK));
  COMMAND(4) = HIGH (LOW (BLOCK));
  COMMAND(5) = LOW (LOW (BLOCK));
  COMMAND(6) = 0;
  COMMAND(7) = HIGH (LEN);
  COMMAND(8) = LOW (LEN);
  COMMAND(9) = 0;
  COMMAND_ID = DISK;
  SCSI_BUSY = TRUE;

END SCSI_DISK_READ;

/*-------------------------------------------------------------*/

SCSI_DISK_WRITE: PROCEDURE (BLOCK,LEN,ADDR);

  DCL BLOCK DWORD, LEN WORD, ADDR WORD;

  IF SCSI_BUSY THEN RETURN;
  BUFFER_ADDRESS = ADDR;
  TEN_BIT = FALSE;
  TOTAL_XFER_COUNT = DOUBLE(LEN) * 512;
  COMMAND(0) = CMD_WRITE_EXT;
  COMMAND(1) = 0;
  COMMAND(2) = HIGH (HIGH (BLOCK));
  COMMAND(3) = LOW (HIGH (BLOCK));
  COMMAND(4) = HIGH (LOW (BLOCK));
  COMMAND(5) = LOW (LOW (BLOCK));
  COMMAND(6) = 0;
  COMMAND(7) = HIGH (LEN);
  COMMAND(8) = LOW (LEN);
  COMMAND(9) = 0;
  COMMAND_ID = DISK;
  SCSI_BUSY = TRUE;

END SCSI_DISK_WRITE;
```

```
/*-------------------------------------------------------------*/

SCSI_DISK_WRITE_10: PROCEDURE (BLOCK,LEN,ADDR);

  DCL BLOCK DWORD, LEN WORD, ADDR WORD;

  IF SCSI_BUSY THEN RETURN;
  BUFFER_ADDRESS = ADDR;
  TEN_BIT = TRUE;
  TOTAL_XFER_COUNT = DOUBLE(LEN) * 512;
  COMMAND(0) = CMD_WRITE_EXT;
  COMMAND(1) = 0;
  COMMAND(2) = HIGH (HIGH (BLOCK));
  COMMAND(3) = LOW (HIGH (BLOCK));
  COMMAND(4) = HIGH (LOW (BLOCK));
  COMMAND(5) = LOW (LOW (BLOCK));
  COMMAND(6) = 0;
  COMMAND(7) = HIGH (LEN);
  COMMAND(8) = LOW (LEN);
  COMMAND(9) = 0;
  COMMAND_ID = DISK;
  SCSI_BUSY = TRUE;

END SCSI_DISK_WRITE_10;

/*-------------------------------------------------------------*/

SCSI_DISK_START: PROCEDURE;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_START_STOP;
  COMMAND(1) = 0;
  COMMAND(2) = 0;
  COMMAND(3) = 0;
  COMMAND(4) = 1;
  COMMAND(5) = 0;
  COMMAND_ID = DISK;
  SCSI_BUSY = TRUE;

END SCSI_DISK_START;

/*-------------------------------------------------------------*/

SCSI_DISK_STOP: PROCEDURE;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_START_STOP;
  COMMAND(1) = 0;
  COMMAND(2) = 0;
  COMMAND(3) = 0;
  COMMAND(4) = 0;
  COMMAND(5) = 0;
  COMMAND_ID = DISK;
  SCSI_BUSY = TRUE;

END SCSI_DISK_STOP;

/*-------------------------------------------------------------*/
```

```
SCSI_DISK_FORMAT: PROCEDURE;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_FORMAT;
  COMMAND(1) = 0;
  COMMAND(2) = 0;
  COMMAND(3) = 0;
  COMMAND(4) = 0;
  COMMAND(5) = 0;
  COMMAND_ID = DISK;
  SCSI_BUSY = TRUE;

END SCSI_DISK_FORMAT;

/*-------------------------------------------------------------*/

SCSI_DISK_TEST: PROCEDURE;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_SND_DIAG;
  COMMAND(1) = 00000100B;
  COMMAND(2) = 0;
  COMMAND(3) = 0;
  COMMAND(4) = 0;
  COMMAND(5) = 0;
  COMMAND_ID = DISK;
  SCSI_BUSY = TRUE;

END SCSI_DISK_TEST;

/*-------------------------------------------------------------*/

SCSI_TAPE_READ: PROCEDURE (LEN,ADDR);

  DCL LEN WORD, ADDR WORD;

  IF SCSI_BUSY THEN RETURN;
  BUFFER_ADDRESS = ADDR;
  TOTAL_XFER_COUNT = DOUBLE(LEN) * 1024;
  COMMAND(0) = CMD_READ;
  COMMAND(1) = 1;
  COMMAND(2) = 0;
  COMMAND(3) = HIGH (LEN);
  COMMAND(4) = LOW (LEN);
  COMMAND(5) = 0;
  COMMAND_ID = TAPE;
  SCSI_BUSY = TRUE;

END SCSI_TAPE_READ;

/*-------------------------------------------------------------*/

SCSI_TAPE_WRITE: PROCEDURE (LEN,ADDR);

  DCL LEN WORD, ADDR WORD;

  IF SCSI_BUSY THEN RETURN;
  BUFFER_ADDRESS = ADDR;
  TEN_BIT = FALSE;
  TOTAL_XFER_COUNT = DOUBLE(LEN) * 1024;
```

```
  COMMAND(0) = CMD_WRITE;
  COMMAND(1) = 1;
  COMMAND(2) = 0;
  COMMAND(3) = HIGH (LEN);
  COMMAND(4) = LOW (LEN);
  COMMAND(5) = 0;
  COMMAND_ID = TAPE;
  SCSI_BUSY = TRUE;

END SCSI_TAPE_WRITE;

/*-------------------------------------------------------------*/

SCSI_TAPE_WRITE_FILEMARKS: PROCEDURE (NUM);

  DCL NUM BYTE;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_WRITE_MARKS;
  COMMAND(1) = 0;
  COMMAND(2) = 0;
  COMMAND(3) = 0;
  COMMAND(4) = NUM;
  COMMAND(5) = 0;
  COMMAND_ID = TAPE;
  SCSI_BUSY = TRUE;

END SCSI_TAPE_WRITE_FILEMARKS;

/*-------------------------------------------------------------*/

SCSI_TAPE_SPACE: PROCEDURE (MODE,NUM);

  DCL MODE BYTE, NUM SHORTINT;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_SPACE;
  COMMAND(1) = MODE AND 00000001B;
  IF NUM >= 0
    THEN DO;
      COMMAND(2) = 0;
      COMMAND(3) = 0;
    END;
    ELSE DO;
      COMMAND(2) = 0FFH;
      COMMAND(3) = 0FFH;
    END;
  COMMAND(4) = UNSIGN(NUM);
  COMMAND(5) = 0;
  COMMAND_ID = TAPE;
  SCSI_BUSY = TRUE;

END SCSI_TAPE_SPACE;

/*-------------------------------------------------------------*/

SCSI_TAPE_UNLOAD: PROCEDURE;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_LOAD_UNLOAD;
```

```
  COMMAND(1) = 0;
  COMMAND(2) = 0;
  COMMAND(3) = 0;
  COMMAND(4) = 0;
  COMMAND(5) = 0;
  COMMAND_ID = TAPE;
  SCSI_BUSY = TRUE;

END SCSI_TAPE_UNLOAD;

/*-------------------------------------------------------------*/

SCSI_TAPE_REWIND: PROCEDURE;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_REWIND;
  COMMAND(1) = 0;
  COMMAND(2) = 0;
  COMMAND(3) = 0;
  COMMAND(4) = 0;
  COMMAND(5) = 0;
  COMMAND_ID = TAPE;
  SCSI_BUSY = TRUE;

END SCSI_TAPE_REWIND;

/*-------------------------------------------------------------*/

SCSI_TAPE_ERASE: PROCEDURE;

  IF SCSI_BUSY THEN RETURN;
  COMMAND(0) = CMD_ERASE;
  COMMAND(1) = 1;
  COMMAND(2) = 0;
  COMMAND(3) = 0;
  COMMAND(4) = 0;
  COMMAND(5) = 0;
  COMMAND_ID = TAPE;
  SCSI_BUSY = TRUE;

END SCSI_TAPE_ERASE;

/*-------------------------------------------------------------*/

SCSI_CONTROL: PROCEDURE;

  DCL COUNT BYTE;
  DCL STATUS WORD;
  DCL LO_STATUS BYTE AT (.STATUS);
  DCL HI_STATUS BYTE AT (.STATUS + 1);

    HI_STATUS = (SCSI_RD_STATUS AND 01100111B);
    LO_STATUS = (SCSI_RD_INT_STATUS AND 11111000B);

    /* Data phase */

    IF ((STATUS = 0118H) OR (STATUS = 0018H) OR
       (STATUS = 0110H) OR (STATUS = 0010H)) AND
       TOTAL_XFER_COUNT = 0
      THEN SCSI_WR_COMMAND = CTL_SET_ATN;
```

```
IF (STATUS = 0010H) AND DMA_INTERRUPTED
  THEN DO;
    SCSI_WR_XFER_COUNT_LO = RESIDUAL_COUNT;
    SCSI_WR_XFER_COUNT_HI = 0;
    SCSI_WR_COMMAND = CTL_FLUSH_FIFO;
    SCSI_WR_COMMAND = CTL_XFER_DMA;
    SCSI_DMA_XFER = TRUE;
    CALL SCSI_FROM_BUFFER (1);
    DMA_INTERRUPTED = FALSE;
    RETURN;
  END;

IF ((STATUS = 0010H) OR (STATUS = 0018H)) AND TEN_BIT
  THEN DO;
    IF TOTAL_XFER_COUNT >= 7680
      THEN DO;
        SCSI_WR_XFER_COUNT_LO = LOW(7680);
        SCSI_WR_XFER_COUNT_HI = HIGH(7680);
        TOTAL_XFER_COUNT = TOTAL_XFER_COUNT - 7680;
      END;
      ELSE DO;
        SCSI_WR_XFER_COUNT_LO = LOW(LOW(TOTAL_XFER_COUNT));
        SCSI_WR_XFER_COUNT_HI = HIGH(LOW(TOTAL_XFER_COUNT));
        TOTAL_XFER_COUNT = 0;
      END;
    SCSI_WR_COMMAND = CTL_XFER_DMA;
    SCSI_DMA_XFER = TRUE;
    CALL SCSI_FROM_BUFFER_10 (BUFFER_ADDRESS);
    BUFFER_ADDRESS = BUFFER_ADDRESS + 48;
    RETURN;
  END;

IF (STATUS = 0010H) OR (STATUS = 0018H)
  THEN DO;
    IF TOTAL_XFER_COUNT >= 8192
      THEN DO;
        SCSI_WR_XFER_COUNT_LO = LOW(8192);
        SCSI_WR_XFER_COUNT_HI = HIGH(8192);
        TOTAL_XFER_COUNT = TOTAL_XFER_COUNT - 8192;
      END;
      ELSE DO;
        SCSI_WR_XFER_COUNT_LO = LOW(LOW(TOTAL_XFER_COUNT));
        SCSI_WR_XFER_COUNT_HI = HIGH(LOW(TOTAL_XFER_COUNT));
        TOTAL_XFER_COUNT = 0;
      END;
    SCSI_WR_COMMAND = CTL_XFER_DMA;
    SCSI_DMA_XFER = TRUE;
    CALL SCSI_FROM_BUFFER (BUFFER_ADDRESS);
    BUFFER_ADDRESS = BUFFER_ADDRESS + 64;
    RETURN;
  END;

IF (STATUS = 0110H) OR (STATUS = 0118H)
  THEN DO;
    IF TOTAL_XFER_COUNT >= 8192
      THEN DO;
        SCSI_WR_XFER_COUNT_LO = LOW(8192);
        SCSI_WR_XFER_COUNT_HI = HIGH(8192);
        TOTAL_XFER_COUNT = TOTAL_XFER_COUNT - 8192;
```

```
      END;
      ELSE DO;
        SCSI_WR_XFER_COUNT_LO = LOW(LOW(TOTAL_XFER_COUNT));
        SCSI_WR_XFER_COUNT_HI = HIGH(LOW(TOTAL_XFER_COUNT));
        TOTAL_XFER_COUNT = 0;
      END;
    SCSI_WR_COMMAND = CTL_XFER_DMA;
    SCSI_DMA_XFER = TRUE;
    CALL SCSI_TO_BUFFER (BUFFER_ADDRESS);
    BUFFER_ADDRESS = BUFFER_ADDRESS + 64;
    RETURN;
  END;

/* Status phase */

IF (STATUS = 0310H)
  THEN DO;
    SCSI_WR_COMMAND = CTL_CMD_COMPLETE;
    STATUS_IN_FIFO = TRUE;
    SCSI_DMA_XFER = FALSE;
    RETURN;
  END;

IF (STATUS = 0318H)
  THEN DO;
    SCSI_WR_COMMAND = CTL_CMD_COMPLETE;
    STATUS_IN_FIFO = TRUE;
    SCSI_DMA_XFER = FALSE;
    RETURN;
  END;

/* Message in phase */

IF (STATUS = 0708H) AND STATUS_IN_FIFO
  THEN DO;
    TARGET_STATUS = SCSI_RD_FIFO;
    STATUS_IN_FIFO = FALSE;
    TARGET_MESSAGE = SCSI_RD_FIFO;
    SCSI_WR_COMMAND = CTL_MSG_ACCEPTED;
    SCSI_DMA_XFER = FALSE;
    RETURN;
  END;

IF (STATUS = 0710H)
  THEN DO;
    FIFO_COUNT = SCSI_RD_FIFO_FLAGS;
    SCSI_WR_COMMAND = CTL_FLUSH_FIFO;
    SCSI_WR_COMMAND = CTL_XFER_INFO;
    SCSI_DMA_XFER = FALSE;
    RETURN;
  END;

IF (STATUS = 0708H)
  THEN DO;
    TARGET_MESSAGE = SCSI_RD_FIFO;
    IF TARGET_MESSAGE = MSG_SAVE_PTR
      THEN DO;
        DCL XFER_COUNT WORD;
        DCL XFER_COUNT_LO BYTE AT (.XFER_COUNT);
        DCL XFER_COUNT_HI BYTE AT (.XFER_COUNT + 1);
```

```
        XFER_COUNT_LO = SCSI_RD_XFER_COUNT_LO;
        XFER_COUNT_HI = SCSI_RD_XFER_COUNT_HI;
        RESIDUAL_COUNT =
          (XFER_COUNT MOD 128) + FIFO_COUNT;
        DMA_INTERRUPTED = TRUE;
        BUFFER_ADDRESS =
          BUFFER_ADDRESS - (XFER_COUNT / 128);
        CALL MICRO_FROM_BUFFER
          (.BUFFER,128,BUFFER_ADDRESS - 1);
        CALL MICRO_TO_BUFFER
          (.BUFFER + 128 - RESIDUAL_COUNT,
           RESIDUAL_COUNT,1);
        TOTAL_XFER_COUNT = TOTAL_XFER_COUNT +
          XFER_COUNT - RESIDUAL_COUNT + FIFO_COUNT;
      END;
    SCSI_WR_COMMAND = CTL_MSG_ACCEPTED;
    SCSI_DMA_XFER = FALSE;
    RETURN;
  END;

IF (STATUS = 0718H)
  THEN DO;
    SCSI_WR_COMMAND = CTL_XFER_INFO;
    SCSI_DMA_XFER = FALSE;
    RETURN;
  END;

/* Message out phase */

IF (STATUS = 0610H)
  THEN DO;
    SCSI_WR_COMMAND = CTL_FLUSH_FIFO;
    SCSI_WR_FIFO = MSG_RESET;
    SCSI_WR_COMMAND = CTL_XFER_INFO;
    SCSI_DMA_XFER = FALSE;
    RETURN;
  END;

/* Disconnect */

IF (STATUS = 0020H)
  THEN DO;
    IF CURRENT_COMMAND <> CMD_TST_READY
      THEN SCSI_BUSY = FALSE;
    IF TARGET_MESSAGE = MSG_INVALID
      THEN DO;
        IF CURRENT_COMMAND <> CMD_TST_READY
          THEN SCSI_ERROR = 16;
          ELSE IF ID = DISK
            THEN DO;
              SCSI_DISK_READY = FALSE;
              DISK_READY_COUNT = 0;
            END;
            ELSE DO;
              SCSI_TAPE_READY = FALSE;
              TAPE_READY_COUNT = 0;
            END;
      END;
    IF TARGET_MESSAGE = MSG_COMPLETE
      THEN DO;
```

```
    IF CURRENT_COMMAND = CMD_TST_READY
      THEN DO;
        IF TARGET_STATUS = STS_GOOD
          THEN IF ID = DISK
            THEN DO;
              IF DISK_READY_COUNT > 5
                THEN SCSI_DISK_READY = TRUE;
                ELSE DISK_READY_COUNT =
                         DISK_READY_COUNT + 1;
            END;
            ELSE DO;
              IF TAPE_READY_COUNT > 5
                THEN SCSI_TAPE_READY = TRUE;
                ELSE TAPE_READY_COUNT =
                         TAPE_READY_COUNT + 1;
            END;
      END;
      ELSE DO;
        IF TARGET_STATUS = STS_GOOD
          THEN DO;
            IF CURRENT_COMMAND = CMD_REQ_SENSE
              THEN DO;
                CALL MICRO_FROM_BUFFER (.BUFFER,22,2);
                SCSI_ERROR = BUFFER(2);
              END;
              ELSE SCSI_ERROR = 0;
          END;
        IF TARGET_STATUS = STS_BUSY
          THEN DO;
            SCSI_BUSY = TRUE;
          END;
        IF TARGET_STATUS = STS_CHECK
          THEN DO;
            SCSI_BUSY = TRUE;
            COMMAND(0) = CMD_REQ_SENSE;
            COMMAND(1) = 0;
            COMMAND(2) = 0;
            COMMAND(3) = 0;
            COMMAND(4) = 22;
            COMMAND(5) = 0;
            BUFFER_ADDRESS = 2;
            TOTAL_XFER_COUNT = 22;
          END;
      END;
  END;
SCSI_WR_COMMAND = CTL_FLUSH_FIFO;
SCSI_WR_FIFO = MSG_IDENTIFY;
SCSI_DMA_XFER = FALSE;
IF SCSI_BUSY
  THEN DO;
    ID = COMMAND_ID;
    SCSI_ERROR = 0;
    DMA_INTERRUPTED = FALSE;
    IF (COMMAND(0) AND 11100000B) = 00100000B
      THEN DO COUNT = 0 TO 9;
        SCSI_WR_FIFO = COMMAND(COUNT);
      END;
      ELSE DO COUNT = 0 TO 5;
        SCSI_WR_FIFO = COMMAND(COUNT);
      END;
```

```
            CURRENT_COMMAND = COMMAND(0);
          END;
          ELSE DO;
            IF (ID = DISK) AND ARCHIVE_PRESENT
              THEN ID = TAPE;
              ELSE ID = DISK;
            SCSI_WR_FIFO, CURRENT_COMMAND = CMD_TST_READY;
            DO COUNT = 1 TO 5;
              SCSI_WR_FIFO = 0;
            END;
          END;
        STATUS_IN_FIFO = FALSE;
        TARGET_MESSAGE = MSG_INVALID;
        POINTER = 0;
        SCSI_WR_BUS_ID = ID;
        SCSI_WR_COMMAND = CTL_SELECT_ATN;
        RETURN;
      END;

    /* SCSI bus reset */

    IF (STATUS = 0080H)
      THEN SCSI_BUSY = FALSE;

END SCSI_CONTROL;

/*-------------------------------------------------------------*/
```

```
/*================================================================
  EODISP.INC
----------------------------------------------------------------

  NAME:  Electro-optic camera display module

  HISTORY: --REV--  --DATE--  --AUTHOR--
              -      4-04-88      JEM

================================================================*/

DCL  CHAR (*) BYTE  DATA
   (00000B, /* Char 0 */
    00000B,
    00100B,
    01110B,
    11111B,
    00000B,
    00000B,
    00000B,

    00000B, /* Char 1 */
    00000B,
    11111B,
    01110B,
    00100B,
    00000B,
    00000B,
    00000B,

    00000B, /* Char 2 */
    00010B,
    00110B,
    01110B,
    00110B,
    00010B,
    00000B,
    00000B,

    00000B, /* Char 3 */
    01000B,
    01100B,
    01110B,
    01100B,
    01000B,
    00000B,
    00000B,

    /* Set 1 */

    00000B, /* Char 4 */
    00000B,
    00000B,
    00000B,
    00000B,
    00000B,
    11111B,
    01110B,

    00000B, /* Char 5 */
```

```
00000B,
00000B,
00000B,
11111B,
01110B,
11111B,
01110B,

00000B, /* Char 6 */
00000B,
11111B,
01110B,
11111B,
01110B,
11111B,
01110B,

11111B, /* Char 7 */
01110B,
11111B,
01110B,
11111B,
01110B,
11111B,
01110B,

/* Set 2 */

00000B, /* Char 4 */
00000B,
00000B,
00000B,
11111B,
10001B,
11111B,
00000B,

00000B, /* Char 5 */
00000B,
00000B,
11111B,
10001B,
10001B,
11111B,
00000B,

00000B, /* Char 6 */
00000B,
11111B,
10001B,
10001B,
10001B,
11111B,
00000B,

11111B, /* Char 7 */
10001B,
10001B,
10001B,
10001B,
```

```
    10001B,
    11111B,
    00000B);

/*-------------------------------------------------------------*/

DISPLAY_INIT: PROCEDURE;

  DCL ADDRS BYTE;

  CALL WAIT (20000);
  DISPLAY_CTL = 00111000B;
  CALL WAIT (10000);
  DISPLAY_CTL = 00111000B;
  CALL WAIT (1000);
  DISPLAY_CTL = 00111000B;
  CALL WAIT (40);
  DISPLAY_CTL = 00111000B;
  CALL WAIT (40);
  DISPLAY_CTL = 00000001B;
  CALL WAIT (1700);
  DISPLAY_CTL = 00000110B;
  CALL WAIT (40);
  DISPLAY_CTL = 01000000B;
  CALL WAIT (40);
  DO ADDRS = 0 TO 63;
    DISPLAY_DTA = CHAR(ADDRS);
    CALL WAIT (40);
  END;
  DISPLAY_CTL = 00001110B;
  CALL WAIT (40);
  CURSOR_CTL = 208;
  DO ADDRS = 0 TO 31;
    DISP(ADDRS) = ' ';
  END;

END DISPLAY_INIT;

/*-------------------------------------------------------------*/

DISPLAY_SPECIAL: PROCEDURE (SET);

  DCL SET BYTE;

  SET = SET * 32;
  DISPLAY_CTL = 01100000B;
  CALL WAIT (40);
  DO ADDRS = 0 TO 31;
    DISPLAY_DTA = CHAR(ADDRS + SET);
    CALL WAIT (40);
  END;

END DISPLAY_SPECIAL;

/*-------------------------------------------------------------*/

DISPLAY_CLEAR: PROCEDURE;

  DCL POS BYTE;
```

```
  DO POS = 0 TO 31;
    IF DISP(POS) <> ' '
      THEN DO;
        IF POS > 15
          THEN DISPLAY_CTL = POS + 176;
          ELSE DISPLAY_CTL = POS + 128;
        CALL WAIT (40);
        DISP(POS),DISPLAY_DTA = ' ';
        CALL WAIT (40);
      END;
    DISPLAY_CTL = 208;
    CALL WAIT (40);
  END;

END DISPLAY_CLEAR;

/*-------------------------------------------------------------*/

DISPLAY_CURSOR: PROCEDURE (POS);

  DCL POS BYTE;

  IF POS > 15
    THEN CURSOR_CTL = POS + 176;
    ELSE CURSOR_CTL = POS + 128;

END DISPLAY_CURSOR;

/*-------------------------------------------------------------*/

DISPLAY: PROCEDURE (POS,CHAR);

  DCL POS BYTE, CHAR BYTE;

  IF DISP(POS) <> CHAR
    THEN DO;
      IF POS > 15
        THEN DISPLAY_CTL = POS + 176;
        ELSE DISPLAY_CTL = POS + 128;
      CALL WAIT (40);
      DISP(POS),DISPLAY_DTA = CHAR;
      CALL WAIT (40);
    END;
  DISPLAY_CTL = CURSOR_CTL;
  CALL WAIT (40);

END DISPLAY;

/*-------------------------------------------------------------*/

DISPLAY_WORDS: PROCEDURE (POS,LEN,WRD);

  DCL POS BYTE;
  DCL WRD WORD;
  DCL LEN BYTE;
  DCL (WORDS BASED WRD) (*) BYTE;
  DCL CHR BYTE;

  DO CHR = 0 TO LEN - 1;
    IF DISP(POS) <> WORDS(CHR)
```

```
      THEN DO;
        IF POS > 15
          THEN DISPLAY_CTL = POS + 176;
          ELSE DISPLAY_CTL = POS + 128;
        CALL WAIT (40);
        DISP(POS),DISPLAY_DTA = WORDS(CHR);
        CALL WAIT (40);
      END;
    POS = POS + 1;
  END;
  DISPLAY_CTL = CURSOR_CTL;
  CALL WAIT (40);

END DISPLAY_WORDS;

/*-------------------------------------------------------------*/

DISPLAY_SCSI_STATUS: PROCEDURE;

  IF SCSI_BUSY
    THEN CALL DISPLAY_WORDS (12,3,.('Bsy'));
    ELSE IF SCSI_ERROR = 0
      THEN CALL DISPLAY_WORDS (12,3,.('OK '));
      ELSE CALL DISPLAY_NUMBER (12,3,0,SCSI_ERROR);

END DISPLAY_SCSI_STATUS;

/*-------------------------------------------------------------*/

DISPLAY_NUMBER: PROCEDURE (POS,LEN,DEC,VAL);

  DCL POS BYTE, VAL WORD, LEN BYTE, DEC BYTE;
  DCL POSITION BYTE, DIGIT BYTE;
  DCL STRING (5) BYTE;

  CALL TO_DECIMAL (.STRING,VAL);
  IF (DEC > 0) AND (LEN > DEC)
    THEN DO;
      DEC = LEN - DEC - 1;
      DIGIT = 6 - LEN;
      CALL DISPLAY ((POS + DEC),'.');
    END;
    ELSE DO;
      DEC = 10;
      DIGIT = 5 - LEN;
    END;
  DO POSITION = POS TO (POS + LEN - 1);
    IF POSITION <> POS + DEC
      THEN DO;
        CALL DISPLAY (POSITION,STRING(DIGIT));
        DIGIT = DIGIT + 1;
      END;
  END;

END DISPLAY_NUMBER;

/*-------------------------------------------------------------*/

DISPLAY_INTEGER: PROCEDURE (POS,LEN,DEC,IVAL);
```

```
  DCL IVAL INTEGER;
  DCL POS BYTE, LEN BYTE, DEC BYTE;
  DCL POSITION BYTE, DIGIT BYTE;
  DCL STRING (5) BYTE;

  IF IVAL < 0
    THEN CALL DISPLAY (POS,'-');
    ELSE CALL DISPLAY (POS,' ');
  CALL TO_DECIMAL (.STRING,UNSIGN (IABS (IVAL)));
  IF (DEC > 0) AND (LEN > DEC + 1)
    THEN DO;
      DEC = LEN - DEC - 1;
      DIGIT = 7 - LEN;
      CALL DISPLAY ((POS + DEC),'.');
    END;
    ELSE DO;
      DEC = 10;
      DIGIT = 6 - LEN;
    END;
  DO POSITION = (POS + 1) TO (POS + LEN - 1);
    IF POSITION <> POS + DEC
      THEN DO;
        CALL DISPLAY (POSITION,STRING(DIGIT));
        DIGIT = DIGIT + 1;
      END;
  END;

END DISPLAY_INTEGER;

/*-------------------------------------------------------------*/

TO_DECIMAL: PROCEDURE (STR,VAL);

  DCL VAL WORD;
  DCL STR WORD;
  DCL (STRING BASED STR) (*) BYTE;

  STRING(0) = LOW ((VAL / 10000) + 48);
  STRING(1) = LOW (((VAL MOD 10000) / 1000) + 48);
  STRING(2) = LOW (((VAL MOD 1000) / 100) + 48);
  STRING(3) = LOW (((VAL MOD 100) / 10) + 48);
  STRING(4) = LOW ((VAL MOD 10) + 48);

END TO_DECIMAL;

/*-------------------------------------------------------------*/

TO_OCTAL: PROCEDURE (STR,VAL);

  DCL VAL DWORD;
  DCL STR WORD;
  DCL (STRING BASED STR) (*) BYTE;

  STRING(0) = LOW (LOW ((VAL / 134217728) + 48));
  STRING(1) = LOW (LOW (((VAL MOD 134217728) / 16777216) + 48));
  STRING(2) = LOW (LOW (((VAL MOD 16777216) / 2097152) + 48));
  STRING(3) = LOW (LOW (((VAL MOD 2097152) / 262144) + 48));
  STRING(4) = LOW (LOW (((VAL MOD 262144) / 32768) + 48));
  STRING(5) = LOW (LOW (((VAL MOD 32768) / 4096) + 48));
  STRING(6) = LOW (LOW (((VAL MOD 4096) / 512) + 48));
```

```
  STRING(7) = LOW (LOW (((VAL MOD 512) / 64) + 48));
  STRING(8) = LOW (LOW (((VAL MOD 64) / 8) + 48));
  STRING(9) = LOW (LOW ((VAL MOD 8) + 48));

END TO_OCTAL;

/*-------------------------------------------------------------*/

TO_HEX: PROCEDURE (STR,VAL);

  DCL VAL WORD;
  DCL DIGIT BYTE;
  DCL STR WORD;
  DCL (STRING BASED STR) (*) BYTE;

  DIGIT = LOW (VAL / 4096);
  IF DIGIT <= 9
    THEN STRING(0) = DIGIT + 48;
    ELSE STRING(0) = DIGIT + 55;
  DIGIT = LOW ((VAL MOD 4096) / 256);
  IF DIGIT <= 9
    THEN STRING(1) = DIGIT + 48;
    ELSE STRING(1) = DIGIT + 55;
  DIGIT = LOW ((VAL MOD 256) / 16);
  IF DIGIT <= 9
    THEN STRING(2) = DIGIT + 48;
    ELSE STRING(2) = DIGIT + 55;
  DIGIT = LOW (VAL MOD 16);
  IF DIGIT <= 9
    THEN STRING(3) = DIGIT + 48;
    ELSE STRING(3) = DIGIT + 55;

END TO_HEX;

/*-------------------------------------------------------------*/

FROM_HEX: PROCEDURE (STR) WORD;

  DCL VAL WORD;
  DCL STR WORD;
  DCL (STRING BASED STR) (*) BYTE;

  IF STRING(3) <= 57
    THEN VAL = STRING(3) - 48;
    ELSE VAL = STRING(3) - 55;
  IF STRING(2) <= 57
    THEN VAL = VAL + ((STRING(2) - 48) * 16);
    ELSE VAL = VAL + ((STRING(2) - 55) * 16);
  IF STRING(1) <= 57
    THEN VAL = VAL + ((STRING(1) - 48) * 256);
    ELSE VAL = VAL + ((STRING(1) - 55) * 256);
  IF STRING(0) <= 57
    THEN VAL = VAL + ((STRING(0) - 48) * 4096);
    ELSE VAL = VAL + ((STRING(0) - 55) * 4096);
  RETURN VAL;

END FROM_HEX;

/*-------------------------------------------------------------*/
```

```
MS_DIGIT: PROCEDURE (PACK) BYTE;

  DCL PACK BYTE;
  RETURN SHR(PACK,4) + 48;

END MS_DIGIT;

/*-------------------------------------------------------------*/

LS_DIGIT: PROCEDURE (PACK) BYTE;

  DCL PACK BYTE;
  RETURN (PACK AND 00001111B) + 48;

END LS_DIGIT;

/*-------------------------------------------------------------*/
```