```
ROGRAM_NAME='PESA RM4000 (3300 CONTROLLER), RS-232,WA'
(*   DATE:07/30/96    TIME:15:07:44    *)
(***********************************************************)
(* The following code block is provided as a guide to      *)
(* programming the device(s) listed above. This is a       *)
(* sample only, and will most likely require modification  *)
(* to be integrated into a master program.                 *)
(*                                                         *)
(* Device-specific protocols should be obtained directly   *)
(* from the equipment manufacturer to ensure compliance    *)
(* with the most current versions. Within the limits       *)
(* imposed by any non-disclosure agreements, AMX will      *)
(* provide protocols upon request.                         *)
(*                                                         *)
(* If further programming assistance is required, please   *)
(* contact your AMX customer support team.                 *)
(*                                                         *)
(***********************************************************)
(* GENERAL FUNCTION LIST -                                 *)
(* DISPLAY, CHANGE, AND DELET SALVO ENTRY, CHANGE SWITCHER,*)
(* ALL DISPLAY/SEND SWITCHER STATUS FUNCTIONALITY, CHANGE  *)
(* LOCK AND PROTECT, RESTORE SYSTEM FROM ALL CALL AND ALL  *)
(* CALL, AND DISPLAY LOCK STATUS.                          *)
(*                                                         *)
(* IMPORTANT:  THE PESA RM4000 WHICH WAS IN HOUSE WAS CON- *)
(* FIGURED WITH TWO LEVELS; THUS CODE REFLECTS THIS.       *)
(* CONTROLLER: 3300 SYSTEM.                                *)
(*                                                         *)
(* GENERAL INFORMATION:                                    *)
(* THE PROTCOL FOR INTERFACING TO THE PESA ROUTING SWITCH  *)
(* CONTROLLER IS OVER A CPU LINK, WHICH IS A RS-232 SERIAL *)
(* INTERFACE.  DATA TRANSFERED OVER THE CPU LINK IS CON-   *)
(* TROLLED BY THE STATE OF THE RTS AND CTS LINES AND OPER- *)
(* ATE INDEPENDENTLY OF ONE ANOTHER.  THE 3300 REQUIRES AN *)
(* ADDITIONAL CONNECTION TO THE DSR INPUT PIN.  THIS SIGNAL*)
(* MUST BE ACTIVE TO INDICATE TO THE CONTROLLER THE PRE-   *)
(* SENCE OF AN ACTIVE CPU LINK CONNECTION.                 *)
(*                                                         *)
(* CPU LINK MESSAGES ARE CONSTRUCTED IN ASCII CHARACTERS.  *)
(* THE CHARACTERS ARE STANDARD 7-BIT ASCII WITH THE 8th BIT*)
(* SET TO 0.  THE COMMUNICATIONS BETWEEN THE CONTROLLER AND*)
(* THE EXTERNAL COMPUTER CONSIST OF A VARIABLE LENGTH      *)
(* BUFFER OF CHARACTERS CONTAINING THE DESIRED COMMAND, A  *)
(* STRING OF DATA BYTES, A CHECKSUM, AND A TERMINATOR.     *)
(*                                                         *)
(* MESSAGE FORMAT:  COMMAND, <DATA>, CHECKSUM, TERMINATOR  *)
(*                                                         *)
(* THERE ARE NO TIMING REQUIREMENTS ON THE TRANSMISSION OF *)
(* CHARACTERS INTO AND OUT OF THE CONTROLLER EXCEPT FOR THE*)
(* OBEYANCE OF THE RTS-CTS HANDSHAKE.  THE PESA CONTROLLERS*)
(* LOOK FOR THE TERMINATION CHARACTERS IN A MESSAGE STRING *)
(* AND PROCESS ALL INFORMATION THAT HAS BEEN SENT SINCE THE*)
(* LAST TERMINATOR WAS RECEIVED OR SINCE INITIALIZATION OF *)
(* THE CPU LINK COMMUNICATIONS PORT.  THIS INFORMATION IS  *)
(* HANDLED AS ONE COMMUNICATIONS BUFFER.                   *)
(*                                                         *)
(* PROTOCOL #1 COMMANDS ARE ONE LETTER IN LENGTH AND ARE   *)
(* THE FIRST CHARACTER ENCOUNTERED IN TH COMMAND STRING.   *)
(*                                                         *)
(* THE CHECKSUM IS A NUMBER DERIVED FROM EACH DATA BYTE FOR*)
```

```
(* THE PURPOSE OF VERIFYING DATA TRANSMISSION ON BOTH SIDES*)
(* OF THE TRANSMISSION LINK.  A DATA STREAM BEING TRANS-   *)
(* MITTED COMPUTES A CHECKSUM WHICH IS SENT WITH THE DATA  *)
(* AND THE TERMINATION CHARACTERS.  THE RECEIVING EQUIPMENT*)
(* GENERATES A CHECKSUM FROM THE RECEIVED DATA AND COMPARES*)
(* THE TWO CHECKSUMS.  THE CHECKSUM IS CALCULATED AS       *)
(* FOLLOW:                                                 *)
(* 1.  Cumulatively add the bytes, ignoring any overflow.  *)
(*     Save this eight bit number.                         *)
(* 2.  Create two ASCII characters for the checksum by     *)
(*     dividing the saved number into two fields, the upper*)
(*     four bits and lower four bits.  Add $30 to each 4-  *)
(*     bit field.  The upper four bits becomes the "TENS"  *)
(*     digit; the lower four bits becomes the "ONES" ditit.*)
(*     The checksum and the terminator characters are not  *)
(*     included when adding the incoming data to compute a *)
(*     checksum.                                           *)
(*                                                         *)
(* THE TERMINATOR IS COMPRISED OF AN ASCII CARRIAGE RETURN *)
(* (CR/$0D) FOLLOWED BY AN ASCII LINE FEED (LF/$0A).  WHEN *)
(* THESE TWO CHARACTERS HAVE BEEN RECEIVED IN CORRECT ORDER*)
(* THE CONTROLLER INITIATES PROCESSING OF THE CPU LINK     *)
(* COMMAND.                                                *)
(*                                                         *)
(***********************************************************)
(*          DEVICE NUMBER DEFINITIONS GO BELOW             *)
(***********************************************************)
DEFINE_DEVICE

PESA            = 2             (* AXB-EM232: 9600,8,2 *)
PANEL           = 128           (* AMX PANEL *)
(***********************************************************)
(*               CONSTANT DEFINITIONS GO BELOW             *)
(***********************************************************)
DEFINE_CONSTANT

B = $42                     (* DISPLAY SALVO COMMAND *)
C = $43                     (* CHANGE SALVO ENTRY *)
D = $44                     (* DELETE SALVO ENTRY *)
H = $48                     (* CHANGE SWITCHER COMMAND *)
J = $4A                     (* SWITCHER STATUS COMMAND (NO ERROR INFO) *)
L = $4C                     (* CHANGE LOCK COMMAND *)
P = $50                     (* CHANGE PROTECT COMMAND *)
R = $52                     (* RESTORE SYSTEM FROM ALL CALL *)
S = $53                     (* SWITHCER STATUS CHANGE COMMAND *)
T = $54                     (* ALL CALL *)
W = $57                     (* DISPLAY LOCK STATUS *)
Y = $59                     (* SWITCHER STATUS (SINGLE DESTINATION) *)
Z = $5A                     (* SWITCHER STATUS (ALL) *)

Q_MAX           = 30
(***********************************************************)
(*               VARIABLE DEFINITIONS GO BELOW             *)
(***********************************************************)
DEFINE_VARIABLE

PESA_BUFFER[255]            (* PESA BUFFER *)
TEMP_BUFFER[255]            (* TEMP BUFFER *)

INPUT                       (* INPUT *)
OUTPUT                      (* OUTPUT *)
```

```
L1[3]                          (* LEVEL 1 SOURCE/INPUT # *)
L2[3]                          (* LEVEL 2 SOURCE/INPUT # *)

DEST_GROUP[40][16]             (* SWITCHER DESTINATION GROUP *)
SLV[2]                         (* SALVO GROUP NUMBER *)
DEST[3]                        (* OUTPUT/DESTINATION NUMBER *)
STAT[2]                        (* STATUS FIELD *)
JUNK[10]                       (* JUNK STRING *)
LEN                            (* LENGHT OF RESPONSE *)
NUM_OF_GROUPS                  (* NUMBER OF SWITCHER DESTINATION GROUPS *)
COMMAND_STR[50]                (* COMMAND STRING W/PARAMETERS *)
DST[3]                         (* OUTPUT/DESTINATION NUMBER RESPONSE *)

QUEUE[Q_MAX][20]               (* QUEUED COMMANDS *)
Q_HEAD                         (* QUEUE HEAD POINTER *)
Q_TAIL                         (* QUEUE TAIL POINTER *)
Q_HAS_ITEMS                    (* 1 IF ANY ITEMS ARE IN THE QUEUE *)
Q_READY                        (* 1 IF READY TO SEND THE NEXT CMD *)
(*********************************************************)
(*               LATCHING DEFINITIONS GO BELOW           *)
(*********************************************************)
DEFINE_LATCHING

(*********************************************************)
(*       MUTUALLY EXCLUSIVE DEFINITIONS GO BELOW         *)
(*********************************************************)
DEFINE_MUTUALLY_EXCLUSIVE

(*********************************************************)
(*           SUBROUTINE DEFINITIONS GO BELOW             *)
(*********************************************************)
(***************************************)
(* CALL NAME: CHECK AV QUE             *)
(* FUNCTION:  CHECK THE PESA AV QUE    *)
(* FOR COMMANDS                        *)
(***************************************)
DEFINE_CALL 'CHECK QUE'
{
  IF (Q_HAS_ITEMS AND Q_READY)     (* IF READY AND MORE COMMANDS *)
  {
    OFF[Q_READY]
    IF (Q_TAIL = Q_MAX)            (* ROLLOVER *)
      Q_TAIL = 1
    ELSE
      Q_TAIL = Q_TAIL + 1          (* INCREMENT *)
    IF (Q_TAIL = Q_HEAD)           (* IF CAUGHT UP *)
      OFF[Q_HAS_ITEMS]             (* THEN NO ITEMS *)
    CLEAR_BUFFER PESA_BUFFER
    SEND_STRING PESA,QUEUE[Q_TAIL] (* SEND COMMAND *)
    SEND_STRING 0,QUEUE[Q_TAIL]    (* SEND COMMAND *)
    WAIT 10 'QUEUE'                (* TIMEOUT *)
      ON[Q_READY]
  }
}
(***************************************)
(* CALL NAME: ADD TO AV Q              *)
(* FUNCTION:  ADD COMMANDS TO THE      *)
(* PESA AV QUE                         *)
(***************************************)
DEFINE_CALL 'ADD TO Q' (CMD[20])
```

```
{
  IF (Q_HEAD = Q_MAX)              (* IF HEAD IS AT END *)
  {
    IF (Q_TAIL <> 1)               (* DON'T OVERWRITE IF FULL *)
    {
      Q_HEAD = 1
      QUEUE[Q_HEAD] = CMD          (* ADD *)
      ON[Q_HAS_ITEMS]
    }
  }
  ELSE IF (Q_TAIL <> Q_HEAD + 1)   (* DON'T OVERWRITE IF FULL *)
  {
    Q_HEAD = Q_HEAD + 1
    QUEUE[Q_HEAD] = CMD            (* ADD *)
    ON[Q_HAS_ITEMS]
  }
  ELSE
    SEND_STRING 0,"'SWITCHER QUE FULL!',$0D,$0A"
}
(************************************)
(* CALL NAME: CALC CHECKSUM         *)
(* FUNCTION:  CALCULATE CHECKSUM FOR  *)
(* PESA COMMAND STRING              *)
(************************************)
DEFINE_CALL 'CALC CHECKSUM'(STR[50])
LOCAL_VAR CS LOOP
{
  LOOP = 1
  CS = 0
  WHILE (LOOP <= LENGTH_STRING(STR)) (* DO CHECKSUM *)
  {
    CS = CS + STR[LOOP]
    LOOP = LOOP + 1
  }
  CS = CS & $FF                  (* CS IS SINGLE BYTE *)
  CALL 'ADD TO Q' ("STR,($30+(CS/$10)),($30+(CS&$0F)),$0D,$0A")
}
(************************************)
(* CALL NAME: SWITCH AV             *)
(* FUNCTION:  SWITCH IN SWITCHING     *)
(* MATRIX                           *)
(************************************)
DEFINE_CALL 'SWITCH AV' (IN,OUT)
LOCAL_VAR SWITCH_STR[20] INPUT_STR[10]
{
  SWITCH_STR = "H,'0',ITOA(OUT/10),ITOA(OUT%10)"
  INPUT_STR = "'0',ITOA(IN/10),ITOA(IN%10)"
  SWITCH_STR = "SWITCH_STR,INPUT_STR,INPUT_STR"

  CLEAR_BUFFER PESA_BUFFER
  CALL 'CALC CHECKSUM'(SWITCH_STR)
}
(************************************************************)
(*                STARTUP CODE GOES BELOW                   *)
(************************************************************)
DEFINE_START

(* INITIALIZE VARIABLES *)
INPUT = 1
OUTPUT = 1
Q_HEAD = 1
```

```
Q_TAIL = 1
ON[Q_READY]
OFF[Q_HAS_ITEMS]

CREATE_BUFFER PESA,PESA_BUFFER (* CREATE PESA BUFFER *)
(*********************************************************)
(*            THE ACTUAL PROGRAM GOES BELOW              *)
(*********************************************************)
DEFINE_PROGRAM

CALL 'CHECK QUE'

PUSH[PANEL,10]               (* DISPLAY SALVO NUMBER 2 *)
{
  SLV = "$30,$32"
  COMMAND_STR = "B,SLV"
  CLEAR_BUFFER PESA_BUFFER
  CALL 'CALC CHECKSUM' (COMMAND_STR)
  WAIT_UNTIL(LENGTH_STRING(PESA_BUFFER) AND FIND_STRING(PESA_BUFFER,'S',1))
  {
    JUNK = REMOVE_STRING(PESA_BUFFER,'S',1)
    TEMP_BUFFER = REMOVE_STRING(PESA_BUFFER,"$0D",1)
    WAIT_UNTIL(LENGTH_STRING(TEMP_BUFFER))
    {
      DST = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
             GET_BUFFER_CHAR(TEMP_BUFFER)"
      L1 = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
            GET_BUFFER_CHAR(TEMP_BUFFER)"
      L2 = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
            GET_BUFFER_CHAR(TEMP_BUFFER)"
      SEND_STRING 0, "'L1',':',L1,' ','L2',':',L2,' ',10,13"
    }
  }
}

PUSH[PANEL,11]               (* CHANGE SALVO ENTRY *)
{                            (* NUMBER 2 *)
  SLV = "$30,$32"
  DEST= "$30,$30,$32"
  L1  = "$30,$30,$31"
  L2  = "$30,$30,$32"
  COMMAND_STR = "C,SLV,S,DEST,L1,L2"
  CLEAR_BUFFER PESA_BUFFER
  CALL 'CALC CHECKSUM' (COMMAND_STR)
}

PUSH[PANEL,12]               (* DELETE SALVO ENTRY *)
{                            (* NUMBER 2 *)
  SLV = "$30,$32"
  DEST = "$30,$30,$32"
  COMMAND_STR = "D,SLV,S,DEST"
  CLEAR_BUFFER PESA_BUFFER
  CALL 'CALC CHECKSUM' (COMMAND_STR)
}

PUSH[PANEL,1]                (* INPUT UP *)
  IF (INPUT < 48)  INPUT = INPUT + 1
PUSH[PANEL,2]                (* INPUT DN *)
  IF (INPUT > 1)  INPUT = INPUT - 1
PUSH[PANEL,3]                (* OUTPUT UP *)
  IF (OUTPUT < 40)  OUTPUT = OUTPUT + 1
```

```
PUSH[PANEL,4]                (* OUTPUT DN *)
  IF (OUTPUT > 1)  OUTPUT = OUTPUT - 1

PUSH[PANEL,13]               (* CHANGE SWITCHER *)
  CALL 'SWITCH AV' (INPUT,OUTPUT)

PUSH[PANEL,14]               (* DISPLAY SWITCHER STATUS *)
{                            (* NO ERROR INFO *)
  CLEAR_BUFFER PESA_BUFFER
  SEND_STRING PESA,"J,$34,$3A,$0D,$0A"
  SEND_STRING 0,"J,$34,$3A,$0D,$0A"
  WAIT_UNTIL(FIND_STRING(PESA_BUFFER,"$0D",1))
  {
    TEMP_BUFFER = REMOVE_STRING(PESA_BUFFER,"$0A",1)

    L1 = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
          GET_BUFFER_CHAR(TEMP_BUFFER)"
    L2 = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
          GET_BUFFER_CHAR(TEMP_BUFFER)"
    SEND_STRING 0, "'L1',':',L1,' ','L2',':',L2,' ',10,13"
  }
}

PUSH[PANEL,15]               (* TRANSMIT SALVO GROUP *)
{
  CLEAR_BUFFER PESA_BUFFER
  SEND_STRING PESA,"$56,$30,$31,$3B,$37,$0D,$0A"
}

PUSH[PANEL,16]               (* SEND SWITCHER STATUS (SINGLE DEST) *)
{
  DEST = "$30,$30,($30 + OUTPUT)"
  COMMAND_STR = "Y,DEST"
  CLEAR_BUFFER PESA_BUFFER
  CALL 'CALC CHECKSUM' (COMMAND_STR)
  WAIT_UNTIL(LENGTH_STRING(PESA_BUFFER))
  {
     TEMP_BUFFER = REMOVE_STRING(PESA_BUFFER,"$0A",1)

     DST = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
            GET_BUFFER_CHAR(TEMP_BUFFER)"

     SEND_STRING 0,"'DESTINATION',':'DST,10,13"

     STAT = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER)"
     SEND_STRING 0, "'STATUS FIELD',':',STAT,10,13"

     L1 = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
           GET_BUFFER_CHAR(TEMP_BUFFER)"
     L2 = "GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
           GET_BUFFER_CHAR(TEMP_BUFFER)"
     SEND_STRING 0, "'L1',':',L1,' ','L2',':',L2,' ',10,13"
  }
}

PUSH[PANEL,17]               (* DISPLAY SWITCHING MATRIX (ALL) *)
{
  CLEAR_BUFFER PESA_BUFFER
  SEND_STRING PESA,"Z,$35,$3A,$0D,$0A"
  WAIT_UNTIL (LENGTH_STRING(PESA_BUFFER) AND FIND_STRING(PESA_BUFFER,"$0D",1))
  {
```

```
    TEMP_BUFFER = REMOVE_STRING(PESA_BUFFER,"$0D",1)
    WAIT_UNTIL (LENGTH_STRING(TEMP_BUFFER))
    {
      LEN = LENGTH_STRING(TEMP_BUFFER)
      NUM_OF_GROUPS = LEN / 8
      WHILE (NUM_OF_GROUPS > 0)
      { (* STARTS AT END OF ARRAY *)
        DEST_GROUP[NUM_OF_GROUPS] = "GET_BUFFER_CHAR(TEMP_BUFFER),
        GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
        GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
        GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
        GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER),
        GET_BUFFER_CHAR(TEMP_BUFFER),GET_BUFFER_CHAR(TEMP_BUFFER)"
        SEND_STRING 0,"'DESTINATION GROUP',' ',DEST_GROUP[NUM_OF_GROUPS],10,13"
        NUM_OF_GROUPS = NUM_OF_GROUPS - 1
      }
    }
  }
}

PUSH[PANEL,18]              (* CHANGE LOCK (toggle function) *)
{
  CLEAR_BUFFER PESA_BUFFER
  DEST = "$30,$30,$31"
  COMMAND_STR = "L,S,DEST"
  CALL 'CALC CHECKSUM' (COMMAND_STR)
}

PUSH[PANEL,19]              (* CHANGE PROTECT (toggle functon) *)
{
  CLEAR_BUFFER PESA_BUFFER
  DEST = "$30,$30,$31"
  COMMAND_STR = "P,S,DEST"
  CALL 'CALC CHECKSUM' (COMMAND_STR)
}

PUSH[PANEL,20]              (* RESTORE SYSTEM FROM ALL CALL *)
{
  CLEAR_BUFFER PESA_BUFFER
  COMMAND_STR = "R"
  CALL 'CALC CHECKSUM' (COMMAND_STR)
}

PUSH[PANEL,21]              (* ALL CALL *)
{
  CLEAR_BUFFER PESA_BUFFER
  L1  = "$30,$30,$36"
  L2  = "$30,$30,$37"
  COMMAND_STR = "T,L1,L2"
  CALL 'CALC CHECKSUM' (COMMAND_STR)
}

PUSH[PANEL,22]              (* DISPLAY LOCK STATUS *)
{
  CLEAR_BUFFER PESA_BUFFER
  COMMAND_STR = "W,S"
 CALL 'CALC CHECKSUM' (COMMAND_STR)
  WAIT_UNTIL (LENGTH_STRING(PESA_BUFFER))
  { (* FIRST DESTINATION *)
     DST = "GET_BUFFER_CHAR(PESA_BUFFER),GET_BUFFER_CHAR(PESA_BUFFER),
            GET_BUFFER_CHAR(PESA_BUFFER)"
```

```
  }
}


IF (LENGTH_STRING(PESA_BUFFER))  (* RESPONSE FROM PESA *)
{
  CANCEL_WAIT 'QUEUE'
  ON[Q_READY]
  IF (FIND_STRING(PESA_BUFFER,'G',1)) (* GOOD RESPONSE *)
  {  (* DO NOTHING! *)
  }
  ELSE IF (FIND_STRING(PESA_BUFFER,'E',1)) (* ERROR RESPONSE *)
    SEND_STRING 0,"'SWITCHER CHECKSUM INCORRECT',$0D,$0A"
  ELSE IF (FIND_STRING(PESA_BUFFER,'L',1)) (* ERROR RESPONSE *)
    SEND_STRING 0,"'SWITCHER OUTPUT LOCKED',$0D,$0A"
  ELSE IF (FIND_STRING(PESA_BUFFER,'N',1)) (* ERROR RESPONSE *)
  {
    SEND_STRING 0,"'SWITCHER FUNCTION NOT ALLOWED/'"
    SEND_STRING 0,"'SWITCHER MALFUNCTION',$0D,$0A"
  }
}
(***********************************************************)
(*                     END OF PROGRAM                      *)
(*         DO NOT PUT ANY CODE BELOW THIS COMMENT          *)
(***********************************************************)
```