```
*  Read PIA 1.01
*  -------------
*  By Adam Trionfo
*  August 5, 2010
*
*  This program display the contents of the APF MP-1000's
*  PIA Register A, which is memory location $2000.  This program
*  was written to help understand how the Joystick Routine in the
*  APF's System ROM works.
*
*  This program reads the Right and Left Joystick (and all buttons)
*  for each row and column.  The output of this program is displayed in
*  the upper-left corner of the screen.  The results of memory
*  location can be:
*
*              Right Joystick
*
*        +-----+-----+-----+-----+    Key
*  Row 0 |  1  |  0  |  4  |  7  |    ---
*        | $FE | $FD | $FB | $F7 |    S = South, or "Down"
*        +-----+-----+-----+-----+    E = East, or "Right"
*  Row 1 |  S  |  E  |  N  |  W  |    N = North, or "Up"
*        | $FE | $FD | $FB | $F7 |    W = West, or "Left"
*        +-----+-----+-----+-----+    F/E = Fire Button or En
*  Row 2 |  3  |  Cl |  6  |  9  |
*        | $FE | $FD | $FB | $F7 |
*        +-----+-----+-----+-----+
*  Row 3 |  2  | F/E |  5  |  8  |
*        | $FE | $FD | $FB | $F7 |
*        +-----+-----+-----+-----+
*
*              Left Joystick
*
*        +-----+-----+-----+-----+
*  Row 0 |  1  |  0  |  4  |  7  |
*        | $EF | $DF | $BF | $7F |
*        +-----+-----+-----+-----+
*  Row 1 |  S  |  E  |  N  |  W  |
*        | $EF | $DF | $BF | $7F |
*        +-----+-----+-----+-----+
*  Row 2 |  3  |  Cl |  6  |  9  |
*        | $EF | $DF | $BF | $7F |
*        +-----+-----+-----+-----+
*  Row 3 |  2  | F/E |  5  |  8  |
*        | $EF | $DF | $BF | $7F |
*        +-----+-----+-----+-----+
*
*  Here is an example of how it works.  If row 2 is being read, and
*  the output of $2000 is $BF, then button 6 on the left joystick is
*  being pressed.
*
*  Release History
*  ---------------
*
*  Version 1.01 (August 22, 2010)
*
*     - The only change was that a version number was added
*       to the source code.  The binary is still identical
*       to version 1.0
*
*  Version 1.0 (August 5, 2010)
*
*     - First Public Release


*  Equates (On-Board Routines)

```

```
0068 0180                HEXBYTE   EQU  $0180       ; Full Byte of Hex Code
0069 0181                HEX_LSB   EQU  $0181       ; Hex Four Least Significant Bits
0070 0182                HEX_MSB   EQU  $0182       ; Hex Four Most Significant Bits
0071 0183                ASCII_LSB EQU  $0183       ; Four Least Significant Bits as APF ASCII
0072 0184                ASCII_MSB EQU  $0184       ; Four Most Significant Bits as APF ASCII
0073 0185                STROBENUM EQU  $0185       ; Times to Strobe Zero Through $2002
0074 0186                HOLDMSB   EQU  $0186       ; Storage for temp hold of $2002 4 MSbits
0075 4296                FILLSCRN  EQU  $4296       ; Fill Screen Routine
0076 4298                ONECODE   EQU  $4298       ; Fill Screen with One APF ASCII Code
0077
0078 8000                          ORG   $8000       ; Start of APF Cartridge Area
0079
0080                     *  The BIOS uses the first five bytes of the cartridge
0081
0082 8000 bb                       FCB   $BB         ; Tell BIOS a cart is present
0083 8001 80 8b                    FDB   MENUSTR     ; Points to Menu string on cartridge
0084 8003 31                       FCB   $31         ; LSB 4 bits is # of Menu Choices
0085                     *                           Menu has one choice
0086 8004 00                       FCB   $00         ; Number of Players, ALWAYS use ZERO!
0087
0088                     *  After a menu choice is made from the APF Menu via the
0089                     *  BIOS, then control passes to the program here ($8005).
0090
0091                     *  Clear the screen with "Light Green"
0092 8005 86 60                    LDAA  #$60        ; APF ASCII for "Light Green"
0093 8007 bd 42 98                 JSR   ONECODE     ; Fill Screen with "Light Green"
0094
0095                     **********************
0096                     *                    *
0097                     *  Main Program Loop *
0098                     *                    *
0099                     **********************
0100
0101 800a bd 80 19                 JSR   INITPROG    ; Initialize Program
0102 800d bd 80 27       READPIA   JSR   STROBEROW   ; Setup which Button Row to Read
0103 8010 bd 80 52                 JSR   GETPIA      ; Read PIA $2000
0104 8013 bd 80 7e                 JSR   STOREASC    ; Store the ASCII Data on Screen
0105 8016 7e 80 0d                 JMP   READPIA     ; Continue Running Program Forever
0106
0107                     *****************
0108                     *               *
0109                     *  Subroutines  *
0110                     *               *
0111                     *****************
0112
0113                     *  Initialize Program
0114                     *  ------------------
0115                     *  1) Set default row to read to be last row (2, Fire/En, 5 and 8)
0116                     *  2) Place the dollar-sign on screen to represent a hex number
0117
0118 8019 ff 80 d2       INITPROG  STX   STROBE      ; Set Strobe Mask to Row 0
0119 801c 86 03                    LDAA  #$03        ; Number of Rows to Loop Through
0120 801e b7 01 85                 STAA  STROBENUM   ; Store Number of Row
0121 8021 86 64                    LDAA  #$64        ; Place an APF ASCII $ in Accum A
0122 8023 b7 02 00                 STAA  $0200       ; Place $ in upper-left corner
0123 8026 39                       RTS
0124
0125                     *  Strobe Joystick Rows 0-3
0126                     *  -----------------------
0127                     *  1) Get the Joystick Row to Read (Bits 0-3 of $2002)
0128                     *  2) Check which row is currently read, if it's 0, then
0129                     *     reset it to last row (three)
0130
0131 8027 b6 20 02       STROBEROW LDAA  $2002       ; Load Current Button Row to Read (4 LSbits)
0132 802a 84 f0                    ANDA  #%11110000  ; Keep 4 MSbits
0133 802c b7 01 86                 STAA  HOLDMSB     ; Storage for temp hold of $2002 4 MSbits
0134 802f b6 01 85                 LDAA  STROBENUM   ; Load the Row to Read
```

```
0135 8032 81 00                      CMPA  #$00        ; Start over at Row 3 yet?
0136 8034 26 08                      BNE   JOYROW      ; If Zero, then Four Rows Read
0137 8036 ff 80 d2                   STX   STROBE      ; Reset Strobe Mask to Last Row
0138 8039 86 04                      LDAA  #$04        ; Number of Rows to Loop Through
0139 803b b7 01 85                   STAA  STROBENUM   ; Store Number of Row to Read
0140                           *                       ; Note: STROBENUM will be 3 before routine ends
0141 803e fe 80 d2             JOYROW    LDX   STROBE      ; Four Mask addresses to Strobe Rows with
0142 8041 a6 00                      LDAA  $00,X       ; Get new byte to OR with $2002
0143 8043 ba 01 86                   ORAA  HOLDMSB     ; Update $2002 with new row to read
0144 8046 b7 20 02                   STAA  $2002       ; Place Current Row to Read into $2002
0145 8049 b6 01 85                   LDAA  STROBENUM   ; Get Number of Rows Read
0146 804c 4a                         DECA              ; Read to read next row down
0147 804d b7 01 85                   STAA  STROBENUM   ; Store Number row number to read (0-3)
0148 8050 08                         INX               ; Update to next byte to OR with $2002
0149 8051 39                         RTS
0150
0151                           *  Get PIA - Read Column of Joystick
0152                           *  ---------------------------------
0153                           *  1) Read the memory location $2000 (this hold joystick column)
0154                           *  2) Store Joystick column info in byte format that has Most
0155                           *     Significant as zero for preperation for later ASCII
0156                           *     conversion.
0157
0158 8052 b6 20 00             GETPIA    LDAA  $2000       ; Get data from PIA
0159 8055 b7 01 80                   STAA  HEXBYTE     ; Store the whole hex Byte
0160 8058 84 0f                      ANDA  #%00001111  ; MASK.  Keep 4 LSb
0161 805a b7 01 81                   STAA  HEX_LSB     ; Store Four Least Significant Bits
0162 805d b6 01 80                   LDAA  HEXBYTE     ; Get complete Hex Byte again
0163 8060 44                         LSRA              ; Right Shift Four Times to Keep
0164 8061 44                         LSRA              ; Four Most Significant Bits
0165 8062 44                         LSRA
0166 8063 44                         LSRA
0167 8064 b7 01 82                   STAA  HEX_MSB     ; Keep 4 MSbs of Hex Byte
0168 8067 8d 0c                      BSR   HEXTOASC    ; Convert 4 MSbits of Hex Byte to APF ASCII
0169 8069 b7 01 84                   STAA  ASCII_MSB   ; Store 4 MSbits as APF ASCII
0170 806c b6 01 81                   LDAA  HEX_LSB     ; Get the 4 LSbits
0171 806f 8d 04                      BSR   HEXTOASC    ; Convert 4 MSbits of Hex Byte to APF ASCII
0172 8071 b7 01 83                   STAA  ASCII_LSB   ; Store 4 MSbits as APF ASCII
0173 8074 39                         RTS
0174
0175                           *  Hex to APF ASCII Conversion
0176                           *  ---------------------------
0177                           *  1) Convert a Hex digit $00-0F already in Accum A to APF
0178                           *     ASCII.  The Hex digit's four Most Significant Bits must
0179                           *     already be zero.
0180
0181 8075 81 09                HEXTOASC  CMPA  #$09
0182 8077 22 02                      BHI   NOT_0TO9    ; Branch if $0A-$0F
0183 8079 8b 39                      ADDA  #$39        ; No, it's $0A-0F, Place offset for numbers
0184 807b 8b 37                NOT_0TO9  ADDA  #$37        ; Convert from Hex to APF ASCII
0185 807d 39                         RTS
0186
0187
0188                           *  Store APF ASCII data to Screen
0189                           *  ------------------------------
0190                           *  1) Store APF ASCII Data to Upper-Left part of Screen
0191
0192 807e b6 01 84             STOREASC  LDAA  ASCII_MSB   ;
0193 8081 b7 02 01                   STAA  $0201       ; Store after $
0194 8084 b6 01 83                   LDAA  ASCII_LSB   ;
0195 8087 b7 02 02                   STAA  $0202       ; Store after MSB
0196 808a 39                         RTS
0197
0198
0199                           *  Cartridge Menu String
0200                           *  ---------------------
0201                           *
```

```
0202                        *  The data required by the BIOS for the Main Menu is here.
0203
0204 808b ea                MENUSTR   FCB   $EA         ; 10 spaces before Program Name string
0205 808c 52 45 41 44 20 50           FCC   "READ PIA  1.0"  ; Program Name String
     49 41 20 20 31 2e
     30
0206 8099 e9                          FCB   $E9         ; 9 spaces after text (EOL)
0207 809a e9                          FCB   $E9         ; 9 spaces (Before Author)
0208 809b 42 59 20 41 44 41           FCC   "BY ADAM TRIONFO"  ; Author
     4d 20 54 52 49 4f
     4e 46 4f
0209 80aa e8                          FCB   $E8         ; 8 spaces (After Author)
0210 80ab ea                          FCB   $EA         ; 10 spaces (Before Date)
0211 80ac 41 55 47 20 20 35           FCC   "AUG  5, 2010" ; Date
     2c 20 32 30 31 30
0212 80b8 ea                          FCB   $EA         ; 10 spaces (After Date)
0213 80b9 31 2e 20 52 55 4e           FCC   "1. RUN PROGRAM"
     20 50 52 4f 47 52
     41 4d
0214 80c7 cf                          FCB   $CF         ; Control Byte - 15 fill-bytes
0215 80c8 60                          FCB   $60         ; Fill-Byte is Light Green
0216 80c9 c3                          FCB   $C3         ; Control Byte - 3 fill-bytes
0217 80ca 60                          FCB   $60         ; Fill-Byte is Light Green
0218                        *  One Full Black Line of 32 Characters
0219 80cb df                          FCB   $DF         ; Control Byte - 15 fill-bytes
0220 80cc 80                          FCB   $80         ; Fill-Byte is Black
0221 80cd df                          FCB   $DF         ; Control Byte - 15 fill-bytes
0222 80ce 80                          FCB   $80         ; Fill-Byte is Black
0223 80cf c2                          FCB   $C2         ; Control Byte - 2 fill-bytes
0224 80d0 80                          FCB   $80         ; Fill-Byte is Black
0225 80d1 ff                          FCB   $FF         ; Control Byte - End of String
0226
0227                        *  Mask for Moving Through $2002
0228 80d2 0e                STROBE    FCB   %00001110   ; Row 0 (Bit 0 is Low)
0229 80d3 0d                          FCB   %00001101   ; Row 1 (Bit 1 is Low)
0230 80d4 0b                          FCB   %00001011   ; Row 2 (Bit 2 is Low)
0231 80d5 07                          FCB   %00000111   ; Row 3 (Bit 3 is Low)
0232
0233 80d6 ff ff ff ff ff ff CARTEND   FILL  $FF,$9000-CARTEND ; Pad with $FF for a 4K Cart
     ff ff ff ff ff ff
     ff ff ff ff ff ff
     ff ff ff ff ff ff
     ff ff ff ff ff ff
     ff ff ff ff ff ff
     ff ff ff ff ff ff
     ff ff ff ff ff ff
     ff ff ff ff ff ff
     ff ff ff ff
0234
0235                        * End of Program FF,$9000-CARTEND ; Pad with $FF for a 4K Cart
```

```
ASCII_LSB  0183 *0071 0172 0194
ASCII_MSB  0184 *0072 0169 0192
CARTEND    80d6 *0233 0233
FILLSCRN   4296 *0075
GETPIA     8052 *0158 0103
HEXBYTE    0180 *0068 0159 0162
HEXTOASC   8075 *0181 0168 0171
HEX_LSB    0181 *0069 0161 0170
HEX_MSB    0182 *0070 0167
HOLDMSB    0186 *0074 0133 0143
INITPROG   8019 *0118 0101
JOYROW     803e *0141 0136
MENUSTR    808b *0204 0083
NOT_0TO9   807b *0184 0182
ONECODE    4298 *0076 0093
READPIA    800d *0102 0105
STOREASC   807e *0192 0104
STROBE     80d2 *0228 0118 0137 0141
STROBENUM  0185 *0073 0120 0134 0139 0145 0147
STROBEROW  8027 *0131 0102
```