```
Apple 1 Wozmon ROM Listing - with own comments
author Neil Franklin, last modification 2018.07.08

derived from: Apple-1 Operation Manual, pages 7..9, 6502 Hex Monitor Listing


******************************   RESET entry point

RESET:
FF00   D8         CLD            6502 switch off decimal arithmetic mode
FF01   58         CLI            6502 switch off interrupt lockout
FF02   A0 7F      LDY #$7F       6820 Pin directions, Pins0..6 OUT, Pin7 IN
FF04   8C 12 D0   STY $D012      6820 Port B DDR (mode after reset), Port A IN
FF07   A9 A7      LDA #$A7       6820 Ctrl Word, CA1/CB1 pos edge handshake
FF09   8D 11 D0   STA $D011      6820 Port A Ctrl, input with handshake
FF0C   8D 13 D0   STA $D013      6820 Port B Ctrl, output with handshake

******************************   Edit Command Line Input

NOTCR:
FF0F   C9 DF      CMP #$5F+$80   if key ASCII "_", with Bit7 set
FF11   F0 13      BEQ $FF26        yes, drop previous character from buffer
FF13   C9 9B      CMP #$1B+$80   if key ASCII Escape, with Bit7 set
FF15   F0 03      BEQ $FF1A        yes, abort line, "\" and Y = 0
FF17   C8         INY            next position in line buffer, reset #$7F+1
FF18   10 0F      BPL $FF29      if Y < $80, still space in buffer, continue

ESCAPE:
FF1A   A9 DC      LDA #$5C+$80   abort line: ASCII "\" to show this happening
FF1C   20 EF FF   JSR $FFEF      output character

GETLINE:
FF1F   A9 8D      LDA #$0D+$80   start line: ASCII Carriage Return new line
FF21   20 EF FF   JSR $FFEF      output character
FF24   A0 01      LDY #0+1       begin line buffer, Y = 0, +1 to compensate DEY

BACKSPACE:
FF26   88         DEY            backspace: reduce number of characters in Y
FF27   30 F6      BMI $FF1F      if Y < 0, ran out of line buffer, abort

NEXTCHAR:
FF29   AD 11 D0   LDA $D011      6820 Port A Status, test if ready for input
FF2C   10 FB      BPL $FF29      Bit7 = 0 is not ready, wait until becomes 1
FF2E   AD 10 D0   LDA $D010      6820 Port A, read character from terminal
FF31   99 00 02   STA $0200,Y    to line buffer, $0200..$027F, Y = position
FF34   20 EF FF   JSR $FFEF      output character
FF37   C9 8D      CMP #$0D+$80   if key ASCII Carriage Return, with Bit7 set
FF39   D0 D4      BNE $FF0F        no, continue editing line buffer

******************************   Parse Command Line

FF3B   A0 FF      LDY #0-1       begin line buffer, Y = 0, -1 to compensate INY
FF3D   A9 00      LDA #0         A = 0 for setting X and initial hex mode
FF3F   AA         TAX            X = 0 for starting hex number, later

SETSTOR:
FF40   0A         ASL A          no effect when A = 0, 2*$BA=$74 for ":"

SETMODE:
FF41   85 2B      STA $2B        hex number processing mode, 0 = examine addr

BLKSKIP:
FF43   C8         INY            process next character, -1 becomes 0 = first
```

```
NEXTITEM:
FF44   B9 00 02   LDA $0200,Y    from line buffer, $0200..$027F, Y = position
FF47   C9 8D      CMP #$0D+$80   if key ASCII Carriage Return, with Bit7 set
FF49   F0 D4      BEQ $FF1F        yes, output Carriage Return, new line
FF4B   C9 AE      CMP #$2E+$80   if key ASCII ".", with Bit7 set
FF4D   90 F4      BCC $FF43        below, may be space, ignore, process next
FF4F   F0 F0      BEQ $FF41        yes, hex mode new, $AE = block examine end
FF51   C9 BA      CMP #$3A+$80   if key ASCII ":", with Bit7 set
FF53   F0 EB      BEQ $FF40        yes, hex mode new, 2*$BA=$74 = store data
FF55   C9 D2      CMP #$52+$80   if key ASCII "R", with Bit7 set
FF57   F0 3B      BEQ $FF94        yes, Run program at address

******************************   Parse Hexadecimal Number into 16bit

FF59   86 28      STX $28        set hex number low half = 0
FF5B   86 29      STX $29        set hex number high half = 0
FF5D   84 2A      STY $2A        note begin Y position of number in buffer

NEXTHEX:
FF5F   B9 00 02   LDA $0200,Y    from line buffer, $0200..$027F, Y = position
FF62   49 B0      EOR #$30+$80   non-SBC "SUB" of ASCII "0", with Bit7 set
FF64   C9 0A      CMP #9+1       if result <= 9, and so a decimal digit?
FF66   90 06      BCC $FF6E        yes, add this digit without correction
FF68   69 88      ADC #$FA-$71-1 if "A".."F" were $C1..$C6, EOR made $71..$76
                                   shift $71.. to $FA.., but ADC with Carry=1
FF6A   C9 FA      CMP #$FA        if result < $FA..$FF, so not digits A..F
FF6C   90 11      BCC $FF7F        yes, not a hex digit, exit number conversion

DIG:
FF6E   0A         ASL A          move right/LSB hex digit to left/MSB
FF6F   0A         ASL A            needs 4 times 1 bit shift
FF70   0A         ASL A            also deletes left/MSB $F0 from A..F digits
FF71   0A         ASL A
FF72   A2 04      LDX #4         set up loop, for 4 bits per hex digit

HEXSHIFT:
FF74   0A         ASL A
FF75   26 28      ROL $28        hex number low half, new = old *16 + A
FF77   26 29      ROL $29        hex number high half, new = old *16 + A
FF79   CA         DEX
FF7A   D0 F8      BNE $FF74      loop 4 times
FF7C   C8         INY            digit has been processed, next one
FF7D   D0 E0      BNE $FF5F      loop always, get next char from line buffer

******************************   How to Process this Hexadecimal Number

NOTHEX:
FF7F   C4 2A      CPY $2A        compare Y with begin position in buffer
FF81   F0 97      BEQ $FF1A      no hex digits processed, was no number, abort
FF83   24 2B      BIT $2B        test hex number processing mode, $74 = store
FF85   50 10      BVC $FF97        no, $00 = examine, or $AE = block examine

******************************   Hex is Data to be Stored

FF87   A5 28      LDA $28        only use low byte of 16bit hex
FF89   81 26      STA ($26,X)    byte to store address (X=0)
FF8B   E6 26      INC $26        next store address low
FF8D   D0 B5      BNE $FF44      get next character from line buffer
FF8F   E6 27      INC $27        next store address high

TONEXTITEM:
FF91   4C 44 FF   JMP $FF44      get next character from line buffer

******************************   Execute Run command
```

```
RUN:
FF94   6C 24 00   JMP ($0024)    Run program at examine address
                                   it must terminate with JMP $FF1F, GETLINE

******************************

NOTSTOR:
FF97   30 2B      BMI $FFC4      hex mode, $00 = set addr, $AE = block exam

******************************   Hex is Examine Address

FF99   A2 02      LDX #2         set up loop, for 2 bytes per address

SETADR:
FF9B   B5 27      LDA $28-1,X    use hex number, both halves
FF9D   95 25      STA $26-1,X    set store address
FF9F   95 23      STA $24-1,X    set examine address
FFA1   CA         DEX
FFA2   D0 F7      BNE $FF9B      loop 2 times

NXTPRINT:
FFA4   D0 14      BNE $FFBA      not just set, dont output address, only data
FFA6   A9 8D      LDA #$0D+$80   ASCII Carriage Return, with Bit7 set
FFA8   20 EF FF   JSR $FFEF      output character, to start new picture line
FFAB   A5 25      LDA $25        examine address high half
FFAD   20 DC FF   JSR $FFDC      output hexadecimal byte
FFB0   A5 24      LDA $24        examine address low half
FFB2   20 DC FF   JSR $FFDC      output hexadecimal byte
FFB5   A9 BA      LDA #$3A+$80   ASCII ":", with Bit7 set
FFB7   20 EF FF   JSR $FFEF      output character

PRDATA:
FFBA   A9 A0      LDA #$20+$80   ASCII " ", with Bit7 set
FFBC   20 EF FF   JSR $FFEF      output character
FFBF   A1 24      LDA ($24,X)    byte from examine address (X=0)
FFC1   20 DC FF   JSR $FFDC      output hexadecimal byte

******************************   Hex is Block Examine End Address

XAMNEXT:
FFC4   86 2B      STX $2B        hex number processing mode, reset to 0
FFC6   A5 24      LDA $24        compare examine address with hex number, = end
FFC8   C5 28      CMP $28
FFCA   A5 25      LDA $25
FFCC   E5 29      SBC $29
FFCE   B0 C1      BCS $FF91      end address reached, go back to parser $FF44
FFD0   E6 24      INC $24        next examine address low
FFD2   D0 02      BNE $FFD6      no page border crossed
FFD4   E6 25      INC $25        next examine address high

MOD8CHK:
FFD6   A5 24      LDA $24
FFD8   29 07      AND #$07       test output first=0 or other=1..7 byte of line
FFDA   10 C8      BPL $FFA4      loop always, first output address, others not

******************************   Output Hexadecimal Byte (JSR $FFDC, PRBYTE)

PRBYTE:
FFDC   48         PHA            save byte for right/LSB hex digit
FFDD   4A         LSR A          shift left/MSB hex digit to right/LSB
FFDE   4A         LSR A            needs 4 times 1 bit shift
FFDF   4A         LSR A
FFE0   4A         LSR A
FFE1   20 E5 FF   JSR $FFE5      output left/MSB hex digit
FFE4   68         PLA            restore byte for right/LSB hex digit
```

```
 FFE5   !!! fallthrough           output right/LSB hex digit

 ******************************   Output Hexadecimal Digit (JSR $FFE5, PRHEX)

 PRHEX:
 FFE5   29 0F      AND #$0F       delete any upper bits from digit
 FFE7   09 B0      ORA #$30+$80   non-ADC "ADD" of ASCII "0", with Bit7 set
 FFE9   C9 BA      CMP #$39+$80+1 is result <= "9", and so a decimal digit?
 FFEB   90 02      BCC $FFEF         yes, output this digit without correction
 FFED   69 06      ADC #$41-$39-1 ASCII "A"-"9" = 7, but ADC with Carry=1 so 6
 FFEF   !!! fallthrough           instead of JSR+RTS or JMP

 ******************************   Output Character (JSR $FFEF, ECHO)

 ECHO:
 FFEF   2C 12 D0   BIT $D012      6820 Port B, test if ready to accept character
 FFF2   30 FB      BMI $FFEF      Bit7 = 1 is not ready, wait until becomes 0
 FFF4   8D 12 D0   STA $D012      6820 Port B, write character to terminal
 FFF7   60         RTS

 ******************************   unused space

 FFF8   00 00      .DB $00 $00    2 bytes

 ******************************   Hardware Vectors

 FFFA   00 0F      .DW $0F00      6502 NMI Vector, 6820 /IRQB can be jumpered
 FFFC   00 FF      .DW $FF00      6502 RESET Vector, start at $FF00
 FFFE   00 00      .DW $0000      6502 IRQ Vector, 6820 /IRQA can be jumpered
```