```
1    ************************************************************
2    *                                                          *
3    *                SQR (X)  as  USR (X)                      *
4    *                                                          *
5    *                  Michael J. Mahon                        *
6    *                   April 03, 2009                         *
7    *                   Copyright 2009                         *
8    *                                                          *
9    *  Called by USR(X), computes square root of X using a     *
10   *  direct non-restoring algorithm requiring about 5.2      *
11   *  milliseconds on a 1MHz processor, or about nine times   *
12   *  faster than the Applesoft SQR function.                 *
13   *                                                          *
14   *  USR.SQR is location-independent, and so may be BRUN     *
15   *  at any location to install it as the USR function.      *
16   *                                                          *
17   ************************************************************
18
19   * Applesoft Page Zero Definitions
20
21   USR      equ   $0A        ; JMP to USR function
22   FAC      equ   $9D        ; Floating-Point Accumulator
23   FACSGN   equ   FAC+5      ; FAC unpacked sign
24   EXTRAFAC equ   $AC        ; FAC guard byte
25   TEMP1    equ   $93        ; Temp FP storage
26   ARG      equ   $A5        ; Aux FAC
27
28   * Apple ROM routines
29
30   IQERR    equ   $E199      ; "Illegal Quantity Error"
31   MOVAF    equ   $EB63      ; Move rounded FAC to ARG
32   IORTS    equ   $FF58      ; Location of RTS
33
34   * Macro Definitions
35
36   rol32    mac               ; Rotate 32 bit int left
37            rol   ]1+3
38            rol   ]1+2
39            rol   ]1+1
40            rol   ]1
41            eom
42
43   asl32    mac               ; Shift 32 bit int left
44            asl   ]1+3
45            rol   ]1+2
46            rol   ]1+1
47            rol   ]1
48            eom
```

```
                  50            org   $310
                  51
                  52   * Set up USR jump to go to sqr
                  53
0310: 20 58 FF    54   entry    jsr   IORTS      ; Get entry+2 on stack
0313: BA          55            tsx
0314: 18          56            clc
0315: BD FF 00    57            ldaa  $100-1,x   ; Lo byte of entry+2
0318: 69 12       58            adc   #<]offset  ; Lo byte of sqr
031A: 85 0B       59            sta   USR+1
031C: BD 00 01    60            ldaa  $100,x     ; Hi byte of entry+2
031F: 69 00       61            adc   #>]offset  ; Hi byte of sqr
0321: 85 0C       62            sta   USR+2
0323: 60          63   rts      rts
```

```
                65   * Square Root function
                66
0324: A5 9D     67   sqr      lda   FAC         ; Is FAC zero?
0326: F0 FB     68            beq   rts         ; -Yes, sqrt(0) = 0.
0328: 24 A2     69            bit   FACSGN      ; -No, is argument negative?
032A: 10 03     70            bpl   :ok         ; -No, go ahead.
032C: 4C 99 E1  71            jmp   IQERR       ; -Yes, Illegal Quantity!
032F: A5 9D     72   :ok      lda   FAC         ; Divide exponent by 2
0331: 18        73            clc               ;  for initial approximation.
0332: 69 80     74            adc   #$80        ; Move hi bit to C & clear it.
0334: 6A        75            ror               ; Divide by 2, restore hi bit.
0335: 85 9D     76            sta   FAC
0337: 90 0C     77            bcc   :even       ; If exp even, no adjust,
0339: E6 9D     78            inc   FAC         ; If odd, inc exponent
033B: 46 9E     79            lsr   FAC+1       ;  and shift it right.
033D: 66 9F     80            ror   FAC+2
033F: 66 A0     81            ror   FAC+3
0341: 66 A1     82            ror   FAC+4
0343: 66 AC     83            ror   EXTRAFAC    ; (save low bit)
0345: 20 63 EB  84   :even    jsr   MOVAF       ; Move rounded FAC to ARG
0348: A2 03     85            ldx   #3
034A: A9 00     86            lda   #0          ; Zero FAC & TEMP1 mants
034C: 95 9E     87   :clear   sta   FAC+1,x
034E: 95 94     88            sta   TEMP1+1,x
0350: CA        89            dex
0351: 10 F9     90            bpl   :clear
0353: A2 20     91            ldx   #32         ; Gen 32 bits of sqrt
0355: D0 18     92            bne   :start      ; (always)
                93
0357: A5 94     94   :loop    lda   TEMP1+1     ; Compare 40 bits of X
0359: C5 9E     95            cmp   FAC+1       ;  with current sqrt.
035B: D0 16     96            bne   :check
035D: A5 95     97            lda   TEMP1+2     ; Compare next byte
035F: C5 9F     98            cmp   FAC+2
0361: D0 10     99            bne   :check
0363: A5 96     100           lda   TEMP1+3     ; Compare next byte
0365: C5 A0     101           cmp   FAC+3
0367: D0 0A     102           bne   :check
0369: A5 97     103           lda   TEMP1+4     ; Compare next byte
036B: C5 A1     104           cmp   FAC+4
036D: D0 04     105           bne   :check
036F: A5 A6     106  :start   lda   ARG+1       ; Compare final byte
0371: C9 40     107           cmp   #%01000000
0373: 90 1E     108  :check   bcc   :shift0     ; X > sqrt
0375: A5 A6     109           lda   ARG+1       ; X <= sqrt, so subtract
0377: E9 40     110           sbc   #%01000000
0379: 85 A6     111           sta   ARG+1
037B: A5 97     112           lda   TEMP1+4
037D: E5 A1     113           sbc   FAC+4
037F: 85 97     114           sta   TEMP1+4
0381: A5 96     115           lda   TEMP1+3
0383: E5 A0     116           sbc   FAC+3
0385: 85 96     117           sta   TEMP1+3
```

```
0387: A5 95     118           lda   TEMP1+2
0389: E5 9F     119           sbc   FAC+2
038B: 85 95     120           sta   TEMP1+2
038D: A5 94     121           lda   TEMP1+1
038F: E5 9E     122           sbc   FAC+1
0391: 85 94     123           sta   TEMP1+1
                124  :shift0  rol32 FAC+1      ; Rotate C into sqrt
0393: 26 A1     124           rol   FAC+1+3
0395: 26 A0     124           rol   FAC+1+2
0397: 26 9F     124           rol   FAC+1+1
0399: 26 9E     124           rol   FAC+1
                124           eom
                125           asl32 ARG+1      ; Left shift X
039B: 06 A9     125           asl   ARG+1+3
039D: 26 A8     125           rol   ARG+1+2
039F: 26 A7     125           rol   ARG+1+1
03A1: 26 A6     125           rol   ARG+1
                125           eom
                126           rol32 TEMP1+1
03A3: 26 97     126           rol   TEMP1+1+3
03A5: 26 96     126           rol   TEMP1+1+2
03A7: 26 95     126           rol   TEMP1+1+1
03A9: 26 94     126           rol   TEMP1+1
                126           eom
                127           asl32 ARG+1      ;  by 2 bits.
03AB: 06 A9     127           asl   ARG+1+3
03AD: 26 A8     127           rol   ARG+1+2
03AF: 26 A7     127           rol   ARG+1+1
03B1: 26 A6     127           rol   ARG+1
                127           eom
                128           rol32 TEMP1+1
03B3: 26 97     128           rol   TEMP1+1+3
03B5: 26 96     128           rol   TEMP1+1+2
03B7: 26 95     128           rol   TEMP1+1+1
03B9: 26 94     128           rol   TEMP1+1
                128           eom
03BB: CA        129           dex              ; Done?
03BC: D0 99     130           bne   :loop      ; -No, keep looping.
03BE: A5 94     131           lda   TEMP1+1    ; -Yes, compute final
03C0: C5 9E     132           cmp   FAC+1      ;   bit of sqrt
03C2: 66 AC     133           ror   EXTRAFAC   ;    in FAC extension.
03C4: 60        134           rts
                135
                136  ]offset  equ   sqr-entry-2 ; Offset = sqr - (entry+2)


--End assembly, 181 bytes, Errors: 0


Symbol table - alphabetical order:

   ARG     =$A5        EXTRAFAC=$AC        FAC     =$9D        FACSGN  =$A2
   IORTS   =$FF58      IQERR   =$E199      MOVAF   =$EB63      TEMP1   =$93
```

```
   USR     =$0A     V  ]offset =$12     MD asl32   =$8000      entry   =$0310
MD rol32   =$8000      rts     =$0323       sqr     =$0324
```

Symbol table - numerical order:

```
   USR     =$0A     V  ]offset =$12        TEMP1   =$93        FAC     =$9D
   FACSGN  =$A2        ARG     =$A5        EXTRAFAC=$AC        entry   =$0310
   rts     =$0323      sqr     =$0324   MD rol32   =$8000   MD asl32   =$8000
   IQERR   =$E199      MOVAF   =$EB63      IORTS   =$FF58
```