```
                    1   ************************************************************
                    2   *                                                          *
                    3   *                    Fast Arc Generator                    *
                    4   *           by Michael J. Mahon, Copyright 2022            *
                    5   *                                                          *
                    6   * FASTARC can be BLOADed in any unused memory at least     *
                    7   * 492 bytes long.  On the first CALL it relocates itself  *
                    8   * to run at its load address, then performs its function.*
                    9   *                                                          *
                   10   * FASTARC's main entry point is at its load address.       *
                   11   * It plots an aspect ratio corrected, clipped arc with     *
                   12   * radius 'r' centered at 'xc,yc' starting at 'start'       *
                   13   * angle and going counterclockwise for 'length' units      *
                   14   * of 360/64 (or 5.625) degrees.  (If a clockwise arc is    *
                   15   * desired, 128 should be added to the 'start' parameter.)* 
                   16   * The 'start' angle is expressed as a value from 0..64,    *
                   17   * which maps to 0..360 degrees in units of 5.625 degrees.*
                   18   * The time required to draw an arc is approximately        *
                   19   * 0.75*r*(length/64)+23 milliseconds.                      *
                   20   *                                                          *
                   21   * The arc will be drawn unconnected to the previous plot   *
                   22   * unless 128 is added to 'length' (128..192), in which     *
                   23   * case the final X,Y of the preceding plot will be         *
                   24   * connected by a line to the start of the arc.             *
                   25   *                                                          *
                   26   * POKE input parameters 'xclo', 'xchi', 'yc', and 'r'      *
                   27   * into locations 6, 7, 8, and 9, and 'start' and 'length'* 
                   28   * into locations 74 and 75 before calling.                 *
                   29   *                                                          *
                   30   * Its secondary entry point, FASTCIRC, is at the load      *
                   31   * address plus 14. It plots a circle without needing to    *
                   32   * set 74 and 75.  The time required to draw a full circle*
                   33   * is approximately 0.75*r+23 milliseconds.                 *
                   34   *                                                          *
                   35   ************************************************************
                   36
                   37   *    Apple II ROM subroutines
                   38   ERROR    equ   $D412      ; Print error message.
                   39   HPLOT0   equ   $F457      ; Plot dot at (X,Y),(A)
                   40   HGLIN    equ   $F53A      ; Plot line to (A,X),(Y)
                   41   IORTS    equ   $FF58      ; Location of monitor RTS
                   42
                   43   *    Input parameters
                   44            dum   $06
0006: 00           45   xc       db    0          ; Center x   (0..255)
0007: 00           46   xchi     db    0          ; Center xhi (0..1)
0008: 00           47   yc       db    0          ; Center y   (0..255)
0009: 00           48   r        db    0          ; Radius     (0..214) <== NOTE!
                   49            dend
                   50
                   51            dum   $4A        ; (74 & 75)
004A: 00           52   start    db    0          ; Arc start  (0..63) (or +128)
004B: 00           53   leng     db    0          ; Arc length (0..64) (or +128)
                   54            dend
                   55
                   56   *    Page zero variables (swapped with Applesoft)
                   57            dum   $34
                   58   pz       equ   *          ; Start of page zero space.
0034: 00           59   x        db    0          ; X coordinate lo
0035: 00           60   xhi      db    0          ; X coordinate hi
0036: 00           61   prevr    db    0          ; Last call 'r' (init = $FF)
0037: 00           62   y        db    0          ; Y coordinate
0038: 00           63   yhi      db    0          ; Sign of y ($FF, 0, or 1)
0039: 00 00        64   t        dw    0          ; Temp word
003B: 00           65   length   db    0          ; Arc length (0..64)
003C: 00           66   clipping db    0          ; >0 = clipping, 0 = plotting.
003D: 00           67   savex    db    0
                   68   pzsize   equ   *-pz       ; Size of 'pz' space.
                   69            dend
```

```
                        70
                        71            org   $1C00       ; Chosen because $1C and $1D only
                        72                                ; occur between 'fastarc' and
                        73                                ; 'relocate' as hi bytes of add-
                        74                                ; resses that require relocation.
                        75                                ; NOTE: Origin page *MUST* be even.
                        76
1C00: 38                77   fastarc  sec                ; Initial branch to relocation
1C01: B0 78             78            bcs   :rel1         ;  code (patches SEC to CLC).
1C03: A5 09             79            lda   r
1C05: C9 D7             80            cmp   #215          ; Is r < 215?
1C07: 90 0F             81            bcc   :rok          ; -Yes, r is OK.
1C09: A2 35             82            ldx   #$35          ; -No, "ILLEGAL QUANTITY ERROR"
1C0B: 4C 12 D4          83            jmp   ERROR
                        84
1C0E: A9 00             85   :fstcirc lda   #0            ; "Draw a circle" entry point.
1C10: 85 4A             86            sta   start         ; 'start' = 0
1C12: A9 40             87            lda   #64
1C14: 85 4B             88            sta   leng          ; 'leng' = 64
1C16: D0 E8             89            bne   fastarc       ; (Always)
                        90
1C18: 20 CB 1C          91   :rok     jsr   :swappz       ; Swap pz space with 'mypz'.
1C1B: A5 4B             92            lda   leng          ; Copy 'leng' to 'length'
1C1D: 29 7F             93            and   #$7F          ;  turning off "connected" bit.
1C1F: 85 3B             94            sta   length
1C21: A5 4B             95            lda   leng          ; Hi bit on means "connected",
1C23: 29 80             96            and   #$80          ;  so isolate it,
1C25: 49 80             97            eor   #$80          ;   invert it, and
1C27: 85 3C             98            sta   clipping      ;    set initial clipping state.
1C29: A5 09             99            lda   r
1C2B: C5 36             100           cmp   prevr         ; Is 'r' same as last call?
1C2D: F0 29             101           beq   :arc          ; -Yes, use cached tables.
                        102
                        103  * Construct first quadrant sine/cosine tables scaled by
                        104  * radius, based on 8-bit 1st quadrant ':costbl'.
                        105
1C2F: 85 36             106           sta   prevr         ; Remember cached table 'r'.
1C31: A2 0F             107           ldx   #15           ; Gen dy,x & dx,x for x = 15..0
1C33: BD 7D 1C          108  :genloop lda   :costbl,x
1C36: A4 09             109           ldy   r
1C38: 20 DB 1C          110           jsr   :mpyay        ; A.t = r * costbl,x
1C3B: 06 39             111           asl   t             ; Round dy.
1C3D: 69 00             112           adc   #0
1C3F: BC 8D 1C          113           ldy   :comp,x       ; Y = complement X.
1C42: 99 A6 1D          114           sta   dy+1,y        ; dy+1,y = int(r*COSTBL,x + 0.5)
1C45: 85 3A             115           sta   t+1           ; t+1 = int(r*COSTBL,x + 0.5)
                        116
                        117  * Adjust 'dx' for Apple II pixel aspect ratio, computed by
                        118  * Sather in "Understanding the Apple II" as X:Y = 1.19:1
                        119  * on page 8-28.  Here this is approximated by:
                        120  *          1 + (1/8 + 1/16) = 1.1875
                        121
1C47: A0 30             122  :xfactor ldy   #$30         ; Y = 0.1875 (can be POKEd for
1C49: 20 DB 1C          123           jsr   :mpyay        ;  different pixel aspect ratio)
1C4C: 65 3A             124           adc   t+1           ; Add in original dy value.
1C4E: 9D 95 1D          125           sta   dx,x          ; dx = int(dy * 1.1875 + .5)
1C51: CA                126           dex
1C52: 10 DF             127           bpl   :genloop
1C54: E8                128           inx                 ; X = 0
1C55: 8E A5 1D          129           stx   dy            ; dy+0 = sine(0) = 0.
```

```
                       131  * Generate 'length' arc starting at angle 'start'
                       132
1C58: A5 4A            133  :arc     lda   start      ; Is "clockwise" bit set?
1C5A: 10 02            134           bpl   :ccw       ; -No, draw counterclockwise.
1C5C: 49 FF            135           eor   #$FF       ; -Yes, complement start.
1C5E: AA               136  :ccw     tax              ; Save start in X.
1C5F: 4A               137           lsr
1C60: 4A               138           lsr
1C61: 4A               139           lsr
1C62: 4A               140           lsr
1C63: 29 03            141           and   #$03       ; Isolate quadrant (0..3).
1C65: A8               142           tay
1C66: 8A               143           txa              ; Recover start value.
1C67: 29 0F            144           and   #$0F
1C69: 59 77 1C         145           eor   :flipvec,y
1C6C: AA               146           tax              ; Starting offset in quadrant
1C6D: 88               147           dey
1C6E: 30 2D            148           bmi   :q1        ; Start in first quadrant.
1C70: F0 37            149           beq   :q2        ; Start in second quadrant.
1C72: 88               150           dey
1C73: F0 3E            151           beq   :q3        ; Start in third quadrant.
1C75: D0 48            152           bne   :q4        ; Start in fourth quadrant.
                       153
1C77: 00 0F 00         154  :flipvec db    0,$F,0,$F
1C7A: 0F
                       155
1C7B: B0 75            156  :rel1    bcs   :rel2      ; Relay branch
                       157
1C7D: FF FE FB         158  :costbl  db    $FF,$FE,$FB,$F4,$EC,$E1,$D4,$C5
1C80: F4 EC E1 D4 C5
1C85: B5 A2 8E         159           db    $B5,$A2,$8E,$78,$61,$4A,$31,$19
1C88: 78 61 4A 31 19
                       160
1C8D: 0F 0E 0D         161  :comp    db    $F,$E,$D,$C,$B,$A,$9,$8
1C90: 0C 0B 0A 09 08
1C95: 07 06 05         162           db    $7,$6,$5,$4,$3,$2,$1,$0
1C98: 04 03 02 01 00
                       163
1C9D: 20 F4 1C         164  :q1      jsr   :adddx
1CA0: 20 18 1D         165           jsr   :subaddy
1CA3: E8               166           inx
1CA4: E0 10            167           cpx   #16        ; Done with 1st quadrant?
1CA6: D0 F5            168           bne   :q1        ; -No, continue.
1CA8: CA               169           dex              ; -Yes, reset X to 15.
1CA9: 20 03 1D         170  :q2      jsr   :subdx
1CAC: 20 18 1D         171           jsr   :subaddy
1CAF: CA               172           dex              ; Done with 2nd quadrant?
1CB0: 10 F7            173           bpl   :q2        ; -No, continue.
1CB2: E8               174           inx              ; -Yes, reset X to 0.
1CB3: 20 03 1D         175  :q3      jsr   :subdx
1CB6: 20 12 1D         176           jsr   :addsuby
1CB9: E8               177           inx
1CBA: E0 10            178           cpx   #16        ; Done with 3rd quadrant?
1CBC: D0 F5            179           bne   :q3        ; -No, continue.
1CBE: CA               180           dex              ; -Yes, reset X to 15.
1CBF: 20 F4 1C         181  :q4      jsr   :adddx
1CC2: 20 12 1D         182           jsr   :addsuby
1CC5: CA               183           dex              ; Done with 4th quadrant?
1CC6: 10 F7            184           bpl   :q4        ; -No, continue.
1CC8: E8               185           inx              ; -Yes, reset X to 0.
1CC9: F0 D2            186           beq   :q1        ; (always) Wrap to 1st quad.
                       187
1CCB: A2 09            188  :swappz  ldx   #pzsize-1  ; Swap 'pz' with 'mypz'.
1CCD: B5 34            189  :swap    lda   pz,x
1CCF: BC 8B 1D         190           ldy   mypz,x
1CD2: 94 34            191           sty   pz,x
1CD4: 9D 8B 1D         192           sta   mypz,x
```

```
1CD7: CA        193           dex
1CD8: 10 F3     194           bpl   :swap
1CDA: 60        195           rts
                196
                197  * 8x8 multiply subroutine
                198
1CDB: 85 39     199  :mpyay   sta   t          ; On entry: A = multiplicand
1CDD: 8C EA 1C  200           sty   :plier+1   ;           Y = multiplier
1CE0: A9 00     201           lda   #0         ; On exit:  A = hi product
1CE2: A0 08     202           ldy   #8         ;           t = lo product
1CE4: 46 39     203           lsr   t
1CE6: 90 03     204  :mloop   bcc   :noadd
1CE8: 18        205           clc
1CE9: 69 00     206  :plier   adc   #0-0       ; (Multiplier stored here)
1CEB: 6A        207  :noadd   ror
1CEC: 66 39     208           ror   t
1CEE: 88        209           dey
1CEF: D0 F5     210           bne   :mloop
1CF1: 60        211           rts
                212
1CF2: B0 7D     213  :rel2    bcs   :rel3      ; Relay branch
                214
1CF4: 18        215  :adddx   clc              ; Add dx to xc.
1CF5: A5 06     216           lda   xc
1CF7: 7D 95 1D  217           adc   dx,x
1CFA: 85 34     218           sta   x
1CFC: A5 07     219           lda   xchi
1CFE: 69 00     220           adc   #0
1D00: 85 35     221           sta   xhi
1D02: 60        222           rts
                223
1D03: 38        224  :subdx   sec              ; Subtract dx from xc.
1D04: A5 06     225           lda   xc
1D06: FD 95 1D  226           sbc   dx,x
1D09: 85 34     227           sta   x
1D0B: A5 07     228           lda   xchi
1D0D: E9 00     229           sbc   #0
1D0F: 85 35     230           sta   xhi
1D11: 60        231           rts
                232
1D12: A5 4A     233  :addsuby lda   start      ; Is "clockwise" bit on?
1D14: 10 08     234           bpl   :adddy     ; -No, go counterclockwise.
1D16: 30 19     235           bmi   :subdy     ; -Yes, go clockwise.
                236
1D18: A5 4A     237  :subaddy lda   start      ; Is "clockwise" bit on?
1D1A: 10 15     238           bpl   :subdy     ; -No, go counterclockwise.
1D1C: 30 00     239           bmi   :adddy     ; -Yes, go clockwise.
                240
1D1E: 86 3D     241  :adddy   stx   savex
1D20: 18        242           clc              ; y = yc + dy
1D21: A5 08     243           lda   yc
1D23: 7D A5 1D  244           adc   dy,x
1D26: 85 37     245           sta   y
1D28: A9 00     246           lda   #0
1D2A: 69 00     247           adc   #0
1D2C: 85 38     248           sta   yhi
1D2E: 18        249           clc
1D2F: 90 10     250           bcc   :plot      ; (always)
                251
1D31: 86 3D     252  :subdy   stx   savex
1D33: 38        253           sec              ; y = yc - dy
1D34: A5 08     254           lda   yc
1D36: FD A5 1D  255           sbc   dy,x
1D39: 85 37     256           sta   y
1D3B: A9 00     257           lda   #0
1D3D: E9 00     258           sbc   #0
1D3F: 85 38     259           sta   yhi        ; Fall into ':plot'.
```

```
                        260
1D41: A4 35     261  :plot    ldy   xhi        ; Plot (with clipping)
1D43: F0 09     262          beq   :noclipx   ; No clip if X < 256.
1D45: 88        263          dey
1D46: D0 10     264          bne   :clip      ; Clip if xhi <> 1.
1D48: A0 17     265          ldy   #280-1-256
1D4A: C4 34     266          cpy   x
1D4C: 90 0A     267          bcc   :clip      ; Clip if X > 279.
1D4E: A4 38     268  :noclipx ldy   yhi        ; y < 0 or y > 255?
1D50: D0 06     269          bne   :clip      ; -Yes, clip.
1D52: A0 BF     270          ldy   #192-1     ; -No, is Y in range?
1D54: C4 37     271          cpy   y
1D56: B0 05     272          bcs   :noclip
1D58: 84 3C     273  :clip    sty   clipping   ; Remember we're clipping,
1D5A: 18        274          clc              ;  and continue.
1D5B: 90 1F     275          bcc   :chkend    ;   (Always)
                276
1D5D: A0 00     277  :noclip  ldy   #0
1D5F: A5 3C     278          lda   clipping   ; Were we clipping?
1D61: 84 3C     279          sty   clipping   ; (We aren't any more!)
1D63: D0 0E     280          bne   :point     ; -Yes, HPLOT X,Y.
1D65: A6 35     281          ldx   xhi        ; -No, HPLOT TO X,Y.
1D67: A5 34     282          lda   x
1D69: A4 37     283          ldy   y
1D6B: 20 3A F5  284          jsr   HGLIN      ; Plot line to X,Y,
1D6E: 18        285          clc              ;  and continue.
1D6F: 90 0B     286          bcc   :chkend    ;   (Always)
                287
1D71: B0 18     288  :rel3    bcs   relocate   ; Relay branch.
                289
1D73: A4 35     290  :point   ldy   xhi        ; -Yes, HPLOT X,Y.
1D75: A6 34     291          ldx   x
1D77: A5 37     292          lda   y
1D79: 20 57 F4  293          jsr   HPLOT0     ; Plot point at X,Y,
1D7C: C6 3B     294  :chkend  dec   length
1D7E: A6 3B     295          ldx   length
1D80: E8        296          inx              ; Are we finished?
1D81: D0 05     297          bne   :return    ; -No, continue.
1D83: 68        298          pla              ; -Yes, skip return
1D84: 68        299          pla              ;   to loop and exit
1D85: 4C CB 1C  300          jmp   :swappz    ;    through ':swappz'.
                301
1D88: A6 3D     302  :return  ldx   savex      ;  restore X reg,
1D8A: 60        303          rts              ;   and return to loop.
                304
                305          dum   *          ; (Overwrites relocation code)
1D8B: 00 00 00  306  mypz     ds    pzsize     ; Page zero swap space
1D8E: 00 00 00 00 00 00 00
1D95: 00 00 00  307  dx       ds    16         ; Cosine table scaled by radius
1D98: 00 00 00 00 00 00 00 00
1DA0: 00 00 00 00 00
1DA5: 00 00 00  308  dy       ds    17         ; Sine table scaled by radius
1DA8: 00 00 00 00 00 00 00 00
1DB0: 00 00 00 00 00 00
                309          dend
```

```
                  311  ************************************************************
                  312  *                                                          *
                  313  *     Self-relocation code (executed on first call)        *
                  314  *                                                          *
                  315  * To use this relocator, as many consecutive byte values   *
                  316  * as there are pages to be relocated must be found (e.g.   *
                  317  * by histogramming byte values in the object code).        *
                  318  *                                                          *
                  319  * Then the code must be reassembled with the origin set    *
                  320  * to the lowest byte value page.  This ensures that the    *
                  321  * chosen byte values only occur as the high bytes of       *
                  322  * relocatable absolute addresses.  Note that this is       *
                  323  * trivially possible if the relocatable code fits on one   *
                  324  * page.  It is very likely possible if the code fits on    *
                  325  * two to four pages (though the probability of finding     *
                  326  * several consecutive unused values decreases as the       *
                  327  * size of the relocatable code block increases).           *
                  328  *                                                          *
                  329  * Note that the code to recognize the flag bytes for       *
                  330  * relocatable addresses in this version of the relocater   *
                  331  * assumes that the relocatable block fits on 2 pages,      *
                  332  * and must be rewritten for the number of pages required.  *
                  333  *                                                          *
                  334  * Also, this relocation code won't handle the case of a    *
                  335  * single byte (like data or an immediate) which requires   *
                  336  * relocation.  It allows only 2-byte absolute addresses.   *
                  337  *                                                          *
                  338  ************************************************************
                  339
                  340  offset   equ   $FB        ; Page zero unused by Applesoft
                  341  loadadr  equ   $FD        ;  so safe before page 0 swap.
                  342  CLC_op   equ   $18        ; CLC to turn off relocation.
                  343
1D8B: 20 58 FF    344  relocate jsr   IORTS      ; Get load address
1D8E: BA          345  :asm     tsx             ; ($FF in IORTS inits 'prevr'!)
1D8F: BD FF 00    346           ldaa  $100-1,x   ; Lo byte of return address
1D92: 38          347           sec
1D93: E9 8D       348           sbc   #<:asm-1   ; 'offset' is the actual
1D95: 85 FB       349           sta   offset     ;   address minus the
1D97: BD 00 01    350           lda   $100,x     ;    assembled address.
1D9A: E9 1D       351           sbc   #>:asm-1
1D9C: 85 FC       352           sta   offset+1
1D9E: 18          353           clc
1D9F: A9 8A       354           lda   #<mypz-1   ; Set 'loadadr' to the
1DA1: 65 FB       355           adc   offset     ;  actual address of 'mypz'-1,
1DA3: 85 FD       356           sta   loadadr    ;   which is initially the
1DA5: A9 1D       357           lda   #>mypz     ;    pointer to the end of the
1DA7: 65 FC       358           adc   offset+1   ;     relocatable code block.
1DA9: 85 FE       359           sta   loadadr+1
1DAB: 18          360           clc
1DAC: A9 1C       361           lda   #>fastarc
1DAE: 65 FC       362           adc   offset+1
1DB0: AA          363           tax             ; X = Load page of 'fastarc'.
1DB1: A0 01       364           ldy   #1        ; (Makes it easy to back up 1)
1DB3: A5 FD       365  :reloc   lda   loadadr   ; Decrement scan pointer.
1DB5: D0 02       366           bne   :samepag  ; Page transition?
1DB7: C6 FE       367           dec   loadadr+1 ; -Yes, decrement page too.
1DB9: C6 FD       368  :samepag dec   loadadr   ; -No, just decrement lo byte.
1DBB: B1 FD       369           lda   (loadadr),y ; Get next byte.
1DBD: 29 FE       370           and   #$FE      ; Ignore low bit of flag bytes.
1DBF: C9 1C       371           cmp   #>fastarc ; Unique abs addr hi bytes?
1DC1: D0 17       372           bne   :noreloc  ; -No, don't relocate.
1DC3: 88          373           dey             ; -Yes, point to lo addr byte
1DC4: 18          374           clc             ;        and relocate address.
1DC5: B1 FD       375           lda   (loadadr),y
1DC7: 65 FB       376           adc   offset
1DC9: 91 FD       377           sta   (loadadr),y
```

```
1DCB: C8        378           iny
1DCC: B1 FD     379           lda   (loadadr),y
1DCE: 65 FC     380           adc   offset+1
1DD0: 91 FD     381           sta   (loadadr),y
1DD2: A5 FD     382           lda   loadadr    ; Skip past lo addr byte.
1DD4: D0 02     383           bne   :sampage   ; Is it on same page?
1DD6: C6 FE     384           dec   loadadr+1  ; -No, decrement page.
1DD8: C6 FD     385  :sampage dec   loadadr    ; -Yes, just dec lo byte.
1DDA: A5 FD     386  :noreloc lda   loadadr    ; Is scan ptr = load addr?
1DDC: C5 FB     387           cmp   offset     ; Is lo byte equal?
1DDE: D0 D3     388           bne   :reloc     ; -No, continue scan.
1DE0: E4 FE     389           cpx   loadadr+1  ; -Yes, is hi byte equal?
1DE2: D0 CF     390           bne   :reloc     ; -No, continue scan.
1DE4: A9 18     391           lda   #CLC_op    ; -Yes, close relocate door.
1DE6: 88        392           dey              ; Y = 0
1DE7: 91 FD     393           sta   (loadadr),y
1DE9: 6C FD 00  394           jmp   (loadadr)  ;  and draw first arc!
```

```
--End assembly, 492 bytes, Errors: 0


Symbol table - alphabetical order:

   CLC_op  =$18        ERROR   =$D412      HGLIN   =$F53A      HPLOT0  =$F457
   IORTS   =$FF58      clipping=$3C        dx      =$1D95      dy      =$1DA5
   fastarc =$1C00      leng    =$4B        length  =$3B        loadadr =$FD
   mypz    =$1D8B      offset  =$FB        prevr   =$36        pz      =$34
   pzsize  =$0A        r       =$09        relocate=$1D8B      savex   =$3D
   start   =$4A        t       =$39        x       =$34        xc      =$06
   xchi    =$07        xhi     =$35        y       =$37        yc      =$08
   yhi     =$38


Symbol table - numerical order:

   xc      =$06        xchi    =$07        yc      =$08        r       =$09
   pzsize  =$0A        CLC_op  =$18        pz      =$34        x       =$34
   xhi     =$35        prevr   =$36        y       =$37        yhi     =$38
   t       =$39        length  =$3B        clipping=$3C        savex   =$3D
   start   =$4A        leng    =$4B        offset  =$FB        loadadr =$FD
   fastarc =$1C00      mypz    =$1D8B      relocate=$1D8B      dx      =$1D95
   dy      =$1DA5      ERROR   =$D412      HPLOT0  =$F457      HGLIN   =$F53A
   IORTS   =$FF58
```