Apple II
Technical Notes

Developer Technical Support

Apple II Miscellaneous
#10: 80-Column GetChar Routine

Revised by: Dave Lyons
September 1989
Written by: Cameron Birse
December 1986

This Technical Note presents an 80-column `GetChar` routine.
Changes since November 1988: Added discussion of single-character input on the unenhanced Apple IIe.

The following is an example of how to display a string on the 80-column screen, reposition the cursor at the beginning of the string, and use the right arrow to get characters which are already there or accept new characters in their place. The routine is a simple BASIC program which displays the string and repositions the cursor before getting incoming characters. If the character input is a right arrow, the program calls the assembly-language routine to get the character from screen memory at the current cursor location.

```
10  PRINT  CHR$ (4);"bload getchar.0": REM  first install assembly routine
20  B$ = "hello"
30  PRINT  CHR$ (4);"pr#3"
40  PRINT B$;:B$ = ""
50  A =  PEEK (1403): REM  get horiz location
60  A = A - 5: REM  move cursor to beginning of string
70  POKE 1403,A
80  GET A$: REM  get a character
90  IF A$ =  CHR$ (21) THEN  GOSUB 130: REM   if char is forward arrow,
    handle with assembly routine (GETCHAR)
100  IF A$ =  CHR$ (27) THEN 170: REM   if esc key then we're done
110  PRINT A$;:B$ = B$ + A$
120  GOTO 80
130  CALL 768: REM   GETCHAR
```

```
    140  A =  PEEK (6)
    150  A$ =  CHR$ (A)
    160  RETURN
    170  PRINT : PRINT : PRINT B$: REM  and we're done
```

An assembled listing of the assembly language `GetChar` routine follows. It works on the Apple IIe and later.

```
SOURCE   FILE #01 =>GETCHAR
----- NEXT OBJECT FILE NAME IS GETCHAR.0
0300:        0300             ORG   $300
0300:        C01F  RD80VID    EQU   $C01F         ;80 COLUMN STATE
0300:        C054  TXTPAGE1   EQU   $C054         ;TURN OFF PAGE 2 (READ)
0300:        C055  TXTPAGE2   EQU   $C055         ;TURN ON PAGE 2 (READ)
0300:        C000  CLR80COL   EQU   $C000         ;TURN OFF 80 STORE (WRITE)
0300:        C001  SET80COL   EQU   $C001         ;TURN ON 80 STORE (WRITE)
0300:        0028  BASL       EQU   $28           ;BASE ADDRESS OF SCREEN LOCATION
0300:        0029  BASH       EQU   $29
0300:        057B  OURCH      EQU   $57B          ;80 COLUMNS HORIZ. POSITION
0300:        05FB  OURCV      equ   $5fb          ;80 col vertical pos
0300:        0006  char       equ   6             ;place to hand character back to basic
0300:              *
0300:              ************************************************************
0300:              *   GETCHAR - This routine gets an ascii character from the *
0300:              *   80 column display memory of the Apple IIe. It assumes   *
0300:              *   that main memory is switched in and that the base addrs *
0300:              *   of the line has already been calculated and resides     *
0300:              *   in BASL and BASH. It is meant to be called from BASIC   *
0300:              *   as follows:                                             *
0300:              *                     CALL 768                              *
0300:              *                     A = PEEK (6)                          *
0300:              *                     A$ = CHR$(A)                          *
0300:              *   As you can see, the character is returned in location   *
0300:              *   $6 in zero page. This routine is offered as an example. *
0300:              *   No guaranties are made regarding its fitness for any    *
0300:              *   purpose.           By Cameron Birse 6/10/86             *
0300:              ************************************************************
0300:              *
0300:        0300  getchr     equ   *             ;get the char at the current cursor loc.
0300:A9 01                    lda   #$01          ;mask for horiz test
0302:2C 7B 05                 bit   OURCH         ;are we in main or aux mem?
0305:D0 17   031E             bne   main          ;if bit 0 of OURCH is set, then main mem
0307:        0307  aux        equ   *
0307:AD 7B 05                 lda   OURCH         ;get horiz pos.
030A:18                       clc                 ;clear the carry for divide
030B:6A                       ror   a             ;divide by two
030C:A8                       tay                 ;put the result in y
030D:8D 01 C0                 sta   SET80COL      ;turn on 80 store
0310:AD 55 C0                 lda   TXTPAGE2      ;flip to aux text page
0313:B1 28                    lda   (basl),y      ;get the character
0315:85 06                    sta   char
0317:AE 54 C0                 ldx   TXTPAGE1      ;turn off aux text page
031A:8D 00 C0                 sta   CLR80COL      ;turn off 80 store
031D:60                       rts
031E:        031E  main       equ   *
031E:AD 7B 05                 lda   OURCH         ;get horiz pos.
0321:18                       clc                 ;clear the carry for divide
0322:6A                       ror   a             ;divide by two
0323:A8                       tay                 ;put the result in y
0324:B1 28                    lda   (basl),y      ;get the character
0326:85 06                    sta   char
0328:60                       rts
```

Reading a Single Character

While the 80-column firmware is active (whether in 40- or 80-column mode), the `RDKEY` routine on the unenhanced Apple IIe unexpectedly allows the user to press ESC and move the cursor around the screen the same way `RDCHAR` does.

AppleSoft's `GET` statement uses `RDKEY`, so it behaves the same way. The ESC keypress is never returned, so users have problems if you use `GET` and expect them, for example, to press ESC to return to the previous menu. At this point, the cursor turns into an inverse plus sign (+) and your program is still waiting for a keypress. The user presses ESC a few more times, watching the cursor alternate between an inverse plus sign and an inverse blank, and then turns off the computer in search of a more exciting activity, like throwing darts at your disk.

If your program can run on the unenhanced IIe, either leave the 80-column firmware turned off (`PRINT CHR$(21)` to make sure it's off), or read keypresses by polling the keyboard register directly:

```
1000 IF PEEK(-16384)<128 THEN 1000      : REM Wait for a keypress
1010 A$ = CHR$(PEEK(-16384)-128)  : REM Read the key
1020 POKE -16368,0                      : REM Clear the keyboard strobe
```

or

```
0300: LDA $C000     ; check for a keypress
0303: BPL $0300     ;    keep waiting
0306: AND #$7F      ; turn off bit 7
0308: STA $C010     ; clear the keyboard strobe
```

Note that these code fragments don't display a cursor while waiting for a key.

Further Reference

- *Apple IIGS Firmware Reference*
- *Apple IIe Technical Reference Manual*
- *Apple IIc Technical Reference Manual, Second Edition*