THE GS-APPLESOFT CONNECTION

An Applesoft Time and Date Utility for the IIGS

The IIGS Toolbox is a collection of powerful software routines locked inside the ROM memory chips. Applesoft, the Apple II BASIC language that lives in the ROM of all Apple II's since the Apple II Plus, does not know about the Toolbox. With a few bytes of machine language, though, most of the ROM-resident tools can be used by your Applesoft programs.

GSTIME is a short utility that reads the time and date from the IIGS's internal clock/calendar into an Applesoft string variable. The program relies on the Toolbox to do the dirty work of reading the clock and converting the date and time data into an ASCII string. It could have been written to let the ProDOS MLI (Machine Language Interface) read the time, but this approach would not work under DOS 3.3 and the conversion of the binary time/date data into readable ASCII form is something I'd just as soon let the Toolbox do for me.

USING THE PROGRAM

To use the program, you must first BLOAD GSTIME. You can BLOAD it at any free memory location; the default is 768 ($300) and that location shouldn't be a problem unless you have some other machine language routine installed there. You can read the time into any string variable, *STRING$*, with the command

```
CALL 768,STRING$
```

After the call, *STRING$* becomes a 20-character string containing the date and time. The format of the date and time is controlled by the clock settings in the Control Panel. To change the display formats (e.g., 24-hour time versus a.m./p.m. notation) and to change the time and date, open the Classic Desk Accessory menu (simultaneously press Control-Open-Apple-Escape), choose the Control Panel CDA, and from there, choose the clock menu.

If you want your string to hold just the time and not the date, use the commands

```
CALL 768,STRING$:TIME$ = RIGHT$ (STRING$,11)
```

If you need just the date, use

```
CALL 768,STRING$:DATE$ = LEFT$ (STRING$,9)
```

ENTERING THE PROGRAM

If you have an assembler that supports the 65816 (Merlin 16, ORCA/M, or APW, to name a few) enter the source code from Listing 1 and save the object code as GSTIME. If you don't have an assembler, type in the hex codes in Listing 2 and save the object code with the command

```
BSAVE GSTIME,A$300,L$35
```

For help with entering *Nibble* listings, see the Typing Tips section.

HOW IT WORKS

The first thing the program does is use the Applesoft ROM routines CHKCOM, FINDVAR, and CNFRMSTR to ensure that the CALL is followed by a comma and then a string variable. These routines will fall into the appropriate Applesoft error routine if not. After CNFRMSTR is called, a pointer to the string descriptor is placed in zero page locations $85,$86. The string descriptor is three bytes: a length byte and the two byte address of the string data.

The next thing the program does is push the address of the buffer that the Toolbox will use onto the stack. $220 is used as the buffer for two reasons. By using part of Applesoft's input buffer, GSTIME needs no extra memory to hold the data. By putting the buffer at $220 instead of $200, Applesoft will not be fooled into thinking there is something to INPUT after GSTIME returns to Applesoft.

Before any program calls the Toolbox, it must first push all necessary parameters onto the stack. In this case, just one parameter is needed: the address of the data buffer, which we have already placed on the stack. Then the appropriate Toolbox function number is loaded into the 16-bit X-register and the Toolbox call is made (**line 43**).

Lines 48-53 move the data from the buffer into the string variable. The MOVESTR routine ($E3E9) moves the data to the string storage space and MOVEPTR ($DA9A) adjusts the string's descriptor.

LISTING 1: GSTIME Source Code

```
 1 ********************************
 2 * GSTIME Source Code           *
 3 * By Paul MacMillan            *
 4 * Copyright(c) 1988            *
 5 * MicroSPARC, Inc.             *
 6 * Concord, MA 01742            *
 7 * Assembler: Merlin 16         *
 8 ********************************
 9
10 CHKCOM   =    $DEBE      ;ROM ROUTINES: CHECK COMMA
11 FINDVAR  =    $DFE3      ;  SET UP POINTERS TO VARIABLE
12 CNFRMSTR =    $DD6C      ;  CONFIRM ARGUMENT IS STRING
13 MOVESTR  =    $E3E9      ;  MOVE STRING TO STRING SPACE
14 MOVEPTR  =    $DA9A      ;  FIX POINTERS AFTER MOVE
15 VARADR   =    $85        ;POINTER TO STRING DESCRIPTOR
16 INBUF    =    $220       ;INPUT BUFFER
17 ANYWHERE =    $300       ;ARBITRARY ORG LOCATION
18 TOOLBOX  =    $E10000    ;TOOLBOX
19
20 START    ORG  ANYWHERE
21          XC               ;TELL MERLIN TO ALLOW
22          XC               ; 65816 OPCODES
23
24 PARSE    JSR  CHKCOM      ;CONFIRM COMMA FOLLOWS CALL
25          JSR  FINDVAR     ;GET POINTER TO VARIABLE
26          JSR  CNFRMSTR    ;IS IT A STRING?
27          STA  VARADR      ;STORE POINTER
28          STY  VARADR+1
29
30 STACK    LDA  #0          ;PUT POINTER TO BUFFER ON STACK
31          PHA              ; FOR TOOLBOX CALL
32          PHA
33          LDA  #>INBUF
34          PHA
35          LDA  #<INBUF
36          PHA
37
38 TOOL     CLC              ;SET NATIVE MODE
39          XCE
40          REP  #$30        ;ALLOW 16 BIT REGISTERS AND
41          MX   %00         ; TELL MERLIN ABOUT IT
42          LDX  #$0F03      ;READASCIITIME FUNCTION
43          JSL  TOOLBOX     ;MAKE TOOLBOX CALL
44          SEC              ;RETURN TO
45          XCE              ; EMULATION MODE
46          MX   %11         ; AND 8-BIT MERLIN MODE
47
48 FIXVAR   LDA  #<INBUF     ;SET UP POINTERS
49          LDY  #>INBUF
50          LDX  #0
51          STX  INBUF-2
52          JSR  MOVESTR     ;MOVE STRING TO STRING SPACE
53          JSR  MOVEPTR     ; AND FIX POINTERS TO VARIABLE
54
55          RTS
56          CHK              ;MERLIN CHK CODE IS E0
```

END OF LISTING 1

LISTING 2: GSTIME

```
Start: 300          Length: 35

C7|0300:20 BE DE 20 E3 DF 20 6C
95|0308:DD 85 85 84 86 A9 00 48
17|0310:48 A9 02 48 A9 20 48 18
8E|0318:FB C2 30 A2 03 0F 22 00
1F|0320:00 E1 38 FB A9 20 A0 02
9A|0328:A2 00 8E 34 02 20 E9 E3
27|0330:20 9A DA 60 E0

TOTAL: 0E4C
```

END OF LISTING 2