INSTANT GS CONTROL

Control your IIGS without the Control Panel

The Apple IIGS provides some exciting programming features that are, unfortunately, difficult to access from Applesoft. If you know where to look and you know the procedure, you can change the text color and the border color. You can slow down the processor, and speed it back up. You can even display characters from different languages, such as Danish, Swedish, or Spanish. If these features appeal to you, read on! There's more!

The Helper is an ampersand (&) routine that allows you to read the current state of several IIGS features and modify them. There are 16 text, background, and border colors from which to choose. You may select from eight different languages to display on the screen. If you're programming time-critical routines that require the slower system clock speed, you can slow the system down in a flash. Of course, you can also instantly speed it up.

In addition, you can check the current status of the Caps Lock key, Shift key, Control key, and the numeric keypad. A master restore function that resets all IIGS features back to their original state so you can leave the system with the original speed, color, and language.

The Helper doesn't affect the values shown on your Control Panel. It modifies things only temporarily; opening the Control Panel resets all the system parameters to the panel's settings.

```
To set the system speed to slow (1 MHz), use:
10 &_SPEED= 0

To set the system speed to fast (2.8 MHz), use:
10 &_SPEED= 1

To read the caps lock key, use
10 C$ = "C"

20 GET A$ : &_READ= C$,C : IF C = 1 THEN ?"CAPS LOCK DOWN"

To set the text color to orange (color # 9), use
10 &_TEXT= 9
```

Example 1: Using The Helper

USING THE PROGRAM

To install the Helper, use the following command in the beginning of your program:

```
PRINT CHR$(4);"BRUN HELPER"
```

The Helper uses the ampersand hook to enhance Applesoft by adding seven new commands. To distinguish these commands from any other ampersand utilities you may have also installed, I've separated each command from the ampersand with an underscore character (_). This allows the Helper to quickly pick out the commands intended for it. If it encounters a command intended for another ampersand program, it transfers control to that program.

Table 1 gives you a brief description of each Helper command. Note that you can send these commands values using numbers, real variables, or real variable formulas. Example 1 shows examples of how to use the commands. The only command that doesn't exclusively use real variables is the READ command. The READ command has to decide what values you're looking for, so it looks at the first character in an additional string variable for direction.

There is a relationship between the 80 column firmware and the Control Panel's System Speed setting that users of the Helper need to know about. When the 80 column firmware is active, the system speed always reverts to the Control Panel speed setting in immediate mode. Thus, the speed as set by The Helper is only valid while your program is running.

ENTERING THE PROGRAM

Assemble the program shown in Listing 1 at $7000 (28672 decimal). If you don't have an assembler, type in the hexadecimal codes in Listing 2. Save the program with this command:

```
BSAVE HELPER,A$7000,L$320
```

Then enter the demonstration program from **Listing 3** and save it by typing

```
SAVE HELPER.DEMO
```

HOW THE PROGRAM WORKS

The Helper modifies special IIGS memory locations known as soft switches. These soft switches are really hardware interfaces under software control, and they allow a programmer to determine what the IIGS is doing and make program decisions based on this determination.

Surprisingly, the Helper affects very few of the many soft switches. It changes the values of those it works with by reading the current value, modifying the returned value, and storing the modified value back where it came from. When you use this "read and modify" technique, some original information residing at a given soft switch is not affected when other information there is changed.

Lines 10-13 tell the assembler to assemble the code at $7000, use the 65816 opcodes, and use 8 bit processor registers. **Lines 19-40** tell the assembler which IIGS memory locations and internal subroutines should be used. Note that **lines 35-40** are the memory locations for the soft switches I mentioned before.

Lines 60-74 save the current system configuration. The values found here will be used by the RESTORE command to return the IIGS to its original state.

Next, **lines 75** and **76** decide whether you're using ProDOS or DOS 3.3. If you booted DOS 3.3, **lines 77-83** request 1190 bytes of free memory for the Helper. If there is enough memory, the Helper is moved to its new memory space.

Table 1: The Helper Functions

Command	Function	Variables
&__LOAD	Loads a new display language usage: &__LOAD= 1	0 = English (USA) 1 = English (UK) 2 = French 3 = Danish 4 = Spanish 5 = Italian 6 = German 7 = Swedish
&__RESTORE	Restores system to original state usage: &__RESTORE	
&__SPEED	Sets system speed usage: &__SPEED= 1	0 = slow 1 - 255 = fast
&__COLOR	Sets background color usage: &__COLOR= 1	0 = black 1 = deep red 2 = dark blue 3 = purple 4 = dark green 5 = dark gray 6 = medium blue (default background color) 7 = light blue 8 = brown 9 = orange 10 = light gray 11 = pink 12 = green 13 = yellow 14 = aquamarine 15 = white (default text color)
&__HCOLOR	Sets border color usage: &__HCOLOR= 1	See &__COLOR
&__TEXT	Sets text color usage: &__TEXT= 1	See &__COLOR
&__READ	Reads current system values usage: A$="B" : &__READ= A$,A (use A$="B", "C" or "H" etc.)	B: 0-15 backgnd color C: 0 = Caps Lock up C: 1 = Caps Lock down H: 0 -15 border color K: 0 = Control key up K: 1 = Control key down L: 0-7 = Language N: 0 = keyboard key pressed N: 1 = numeric keypad key pressed S: 0 = Shift key up S: 1 = Shift key down T: 0 - 15 Text color V: 0 = Slow speed V: 1 = Fast speed

The Helper uses the ampersand hook to enhance Applesoft by adding seven new commands. To distinguish these commands I've separated each command from the ampersand with an underscore character (_).

If you're using ProDOS, **lines 84-88** request 768 free bytes from BASIC.SYSTEM, after which **lines 84-103** relocate the program if enough free memory is available. **Lines 104-114** save the old ampersand hook for later use, reset the hook to run the program, and return control to Applesoft.

The rest of the code in **Listing 1** is the actual ampersand program, and it's entirely relocatable. That is, it can be moved anywhere in memory without modification.

Line 121 checks for the underscore character (_). Many, many ampersand routines use the same commands as the Helper, so I've decided to use the underscore to help distinguish the Helper from these others. If it's found, the Helper continues processing the command. If not, **line 123** sends control to another ampersand program, if there is one.

Lines 124-128 tell the computer to use an eight-bit Accumulator when in native mode. **Line 129** gets the command token.

From there, the program processes the command, determines the value to write to the soft switch, enters native 65816 processor mode, reads and modifies the soft switch, returns to 6502 processor emulation mode, and returns control to Applesoft.

LISTING 1: HELPER Source Code

```
*
*  HELPER Source Code
*  By Kenn Scribner
*  Copyright (C) 1988
*  MicroSparc Inc.
*  Concord,  MA  01742
*
*  ORCA/M 4.1 Assembler
*
          ORG    $7000       ;assemble at $7000 (28672 decimal)
          65816  ON          ;enable 65816 opcodes
          LONGA  OFF         ;use 8 bit accumulator
          LONGI  OFF         ;use 8 bit X and Y registers

HELPER    START
*
* Local EQUates
*
A1L       EQU    $3C             ;generic temprorary registers for MOVE
A2L       EQU    $3E
A4L       EQU    $42
LINNUM    EQU    $50             ;general purpose 16 bit register
VARPNT    EQU    $83             ;last used variable pointer
FACLO     EQU    $A1             ;temporary holding register
CHRGET    EQU    $B1             ;advance TXTPTR to next token/character
AMPVECT   EQU    $3F5            ;ampersand routine vector (pointer)
ERROR     EQU    $D412           ;ASoft error handler
CHKCOM    EQU    $DEBE           ;check for comma in program, err if not
PTRGET    EQU    $DFE3           ;locate/create variable at TXTPTR
GIVAYF    EQU    $E2F2           ;float A,Y into FAC for storage
GETBYTC   EQU    $E6F5           ;evaluate formula, return integer value
MOVMF     EQU    $EB2B           ;store packed FAC, pointed to by Y,X
SETHMEM1  EQU    $F28E           ;calculate and set HIMEM
MOVE      EQU    $FE2C           ;Monitor memory move routine
IORTS     EQU    $FF58           ;Default DOS 3.3 & return address
TBCOLOR   EQU    $E0C022         ;Text/Background color register
KYMODREG  EQU    $E0C025         ;special key register
LANGSEL   EQU    $E0C02B         ;language select (display) register
CLOCKCTL  EQU    $E0C034         ;border color register
CYAREG    EQU    $E0C036         ;system speed register

;.....................................................................
;
; ProDOS EQUates
;.....................................................................

GETBUFR   EQU    $BEF5           ;reserve A pages (256 bytes) above HIMEM
KVERSION  EQU    $BFFF           ;ProDOS ID byte (+ ProDOS, - DOS 3.3)

;.....................................................................
;
; DOS 3.3 EQUates
;.....................................................................

MAXFILES  EQU    $A2B8           ;DOS 3.3 MAXFILES subroutine

*
* Beginning of Program Text.
*
          CLC                   ;begin by storing original values
          XCE                   ;enter native mode
          LDA    >CYAREG        ;find the current speed
          AND    #%10000000     ;mask out all but speed bit
          STA    SPDMSK+1       ;store for restore
          LDA    >LANGSEL       ;find current displayed language
          AND    #%11101000     ;mask out all but language/primary bits
          STA    LANGMSK+1      ;store for restore
          LDA    >TBCOLOR       ;find current text/background colors
          STA    TBMSK+1        ;store for restore
          LDA    >CLOCKCTL      ;find border color
          AND    #%00001111     ;mask out all but border color bits
          STA    BGMSK+1        ;store for restore
          SEC                   ;return to emulation mode
          XCE
          LDA    KVERSION       ;check for ProDOS
          BPL    PRODOS         ;positive value means ProDOS loaded
DOS33     LDA    #$01           ;DOS 3.3 loaded, set MAXFILES = 1
          JSR    MAXFILES
          STZ    LINNUM         ;set HIMEM = $9600
          LDA    #$96
          STA    LINNUM+1
          JSR    SETHMEM1
          BNE    GOTMEM         ;move program (branch always taken)
PRODOS    LDA    #$03           ;reserve 3 pages (3 X 256 bytes)
          JSR    GETBUFR        ;request memory from BASIC.SYSTEM
          BCC    GOTMEM         ;memory available, no errors
MEMERR    LDX    #$4D           ;?OUT OF MEMORY error
          JMP    ERROR
GOTMEM    PHA                   ;save returned page number on stack...
          PHA                   ;...for later use
          LDA    #<BEGIN        ;move program...
          STA    A1L
          LDA    #>BEGIN
          STA    A1L+1
          LDA    #<END
          STA    A2L
          LDA    #>END
          STA    A2L+1
          STZ    A4L
          PLA
          STA    A4L+1
          LDY    #$00
          JSR    MOVE           ;...move completed
          LDA    #$4C           ;"JMP" opcode (making sure it's there)
          STA    AMPVECT
          LDA    AMPVECT+1      ;save old vector (in case you're...
          STA    EXTAMPR+1      ;...using another & utility)
          LDA    AMPVECT+2
          STA    EXTAMPR+2
          LDA    #$00           ;install new vector
```

```
111             STA   AMPVECT+1
112             PLA
113             STA   AMPVECT+2
114             RTS                   ;installed!
115
116 ;-----------------------------------------------------------
117 ;
118 ; Beginning of real & program (to be moved into operating location).
119 ;-----------------------------------------------------------
120
121 BEGIN   CMP   #$5F                ;'_' delimiter
122             BEQ   PARSE           ;'_' found (must be HELPER routine?)
123 EXTAMPR JMP   IORTS               ;not HELPER routine, try another & pgm.
124 PARSE   CLC                       ;enter native mode
125             XCE
126             SEP   #$20            ;set up 8 bit registers
127             SEC                   ;return to emulation mode
128             XCE
129             JSR   CHRGET          ;read next token
130
131 ;-----------------------------------------------------------
132 ;
133 ; LOAD Subroutine (sets new display language).
134 ;-----------------------------------------------------------
135 LDCHK   CMP   #$B6                ;"LOAD" token value
136             BNE   RECHK           ;no, check for RESTORE token
137 LOAD    JSR   CHRGET              ;get another token
138             CMP   #$D0            ;"=" token value
139             BEQ   L1              ;found it, continue
140             LDX   #$10            ;not there, generate syntax error
141             JMP   ERROR
142 L1      JSR   GETBYTC             ;return integer language value in X
143             TXA
144             CMP   #$08            ;language greater than 8?
145             BCC   LVALID          ;no, valid language number
146             LDX   #$35            ;yes, generate illegal quantity error
147             JMP   ERROR
148 LVALID  ASL   A                   ;move language to upper three bits
149             ASL   A               ; (xxxxxYYY to YYYxxxxx, Y = language)
150             ASL   A
151             ASL   A
152             ASL   A
153             ORA   #%00001000      ;set bit for primary language (YYYxPxxx)
154             STA   FACLO           ;save for later masking
155             CLC                   ;enter native mode
156             XCE
157             LDA   >LANGSEL        ;load in current language register
158             AND   #%00010000      ;clear all mode bits but NTSC/PAL video
159             ORA   FACLO           ;set new language
160             STA   >LANGSEL        ;store new language
161             SEC                   ;return to emulation mode
162             XCE
163             RTS                   ;done!
164
```

```
165 ;-----------------------------------------------------------
166 ;
167 ; RESTORE Subroutine.  Values shown are default values (according to
168 ;  the Control Panel).  These will be modified to reflect what's
169 ;  being used by the system at the moment HELPER is first executed.
170 ;-----------------------------------------------------------
171 RECHK   CMP   #$AE                ;"RESTORE" token value
172             BNE   SPCHK           ;no, check for SPEED= token
173 RESTORE JSR   CHRGET              ;yes, point TXTPTR to formula/variable
174             CLC                   ;enter native mode
175             XCE
176             LDA   >CYAREG         ;find current speed
177             AND   #%01111111      ;set to slow (speed bit = 0)
178 SPDMSK  ORA   #%10000000          ;speed mask (self-modified before MOVE)
179             STA   >CYAREG         ;restore system speed
180             LDA   >LANGSEL        ;find current language
181             AND   #%00010000      ;mask out language (leave video bit)
182 LANGMSK ORA   #%00011000          ;language mask (self-mod'd before MOVE)
183             STA   >LANGSEL        ;restore display language
184 TBMSK   LDA   #%11110110          ;T/B mask (self-modified before MOVE)
185             STA   >TBCOLOR        ;restore Text/Background colors
186             LDA   >CLOCKCTL       ;find current border color
187             AND   #%11110000      ;clear border color bits
188 BGMSK   ORA   #%00000110          ;border color mask (self-mod'd)
189             STA   >CLOCKCTL       ;restore border color
190             SEC                   ;return to emulation mode
191             XCE
192             RTS                   ;done!
193
194 ;-----------------------------------------------------------
195 ;
196 ; SPEED= Subroutine (sets system speed).
197 ;-----------------------------------------------------------
198 SPCHK   CMP   #$A9                ;"SPEED=" token
199             BNE   CLCHK
200 SPEED   JSR   GETBYTC
201             LDA   FACLO           ;retrieve current speed
202             BEQ   SETSPD          ;zero, SLOW speed
203             LDA   #$80            ;non-zero, set FAST speed
204             STA   FACLO           ;save for later masking
205 SETSPD   CLC                     ;enter native mode
206             XCE
207             LDA   >CYAREG         ;find current speed
208             AND   #%01111111      ;zero bit for masking
209             ORA   FACLO           ;mask bit (sets to "1" if FAST selected)
210             STA   >CYAREG         ;save new system speed
211             SEC                   ;return to emulation mode
212             XCE
213             RTS                   ;done!
214
215 ;-----------------------------------------------------------
216 ;
217 ; COLOR= Subroutine (sets background color).
218 ;-----------------------------------------------------------
219 CLCHK   CMP   #$A0                ;"COLOR=" token value
220             BNE   HCCHK
221 COLOR   JSR   GETBYTC
222             TXA
```

LISTING 1: HELPER Source Code *continued from page 94*

```
223           CMP   #$10            ;color greater than 15 selected?
224           BCC   CVALID          ;no, valid color
225           LDX   #$35            ;yes, illegal quantity error
226           JMP   ERROR
227 CVALID    STA   FACLO           ;save color for masking
228           CLC                   ;enter native mode
229           XCE
230           LDA   >TBCOLOR        ;find current background color
231           AND   #%11110000      ;clear color bits for masking
232           ORA   FACLO           ;set color bits
233           STA   >TBCOLOR        ;save new background color
234           SEC                   ;return to emulation mode
235           XCE
236           RTS                   ;done!
237
238 ;.....................................................
239 ;
240 ; HCOLOR= Subroutine (Sets border color).
241 ;.....................................................
242 HCCHK     CMP   #$92            ;"HCOLOR=" token value
243           BNE   TECHK
244 HCOLOR    JSR   GETBYTC
245           TXA
246           CMP   #$10            ;color greater than 15 selected?
247           BCC   HVALID          ;no, valid color
248           LDX   #$35            ;yes, generate illegal quantity error
249           JMP   ERROR
250 HVALID    STA   FACLO           ;save color for masking
251           CLC                   ;enter native mode
252           XCE
253           LDA   >CLOCKCTL       ;find current border color
254           AND   #%11110000      ;clear color bits for masking
255           ORA   FACLO           ;set color bits
256           STA   >CLOCKCTL       ;save new border color
257           SEC                   ;return to emulation mode
258           XCE
259           RTS                   ;done!
260 ;.....................................................
261 ;
262 ; TEXT Subroutine (Sets text color).
263 ;.....................................................
264 TECHK     CMP   #$89            ;"TEXT" token value
265           BNE   RDCHK
266 TEXT      JSR   CHRGET          ;advance TXTPTR to next token
267           CMP   #$D0            ;"=" token value
268           BEQ   T1              ;found it, continue
269           LDX   #$10            ;not there, generate syntax error
270           JMP   ERROR
271 T1        JSR   GETBYTC         ;return integer text color in X
272           TXA
273           CMP   #$10            ;is it greater than 15?
274           BCC   TVALID          ;no, continue
275           LDX   #$35            ;yes, generate illegal quantity error
276           JMP   ERROR
277 TVALID    ASL   A               ;move 4 bits to high nibble
278           ASL   A               ; (xxxxYYYY to YYYYxxxx, Y = color bits)
279           ASL   A
280           ASL   A
281           STA   FACLO           ;save for later masking
282           CLC                   ;enter native mode
283           XCE
284           LDA   >TBCOLOR        ;find current text color
285           AND   #%00001111      ;clear text color bits for masking
286           ORA   FACLO           ;set text color bits
287           STA   >TBCOLOR        ;save new text color
288           SEC                   ;return to emulation mode
289           XCE
290           RTS                   ;done!
291
292 ;.....................................................
293 ;
294 ; READ Subroutine (Reads current system values).
295 ;.....................................................
296 RDCHK     CMP   #$87            ;"READ" token value
297           BEQ   READ
298 SYNERR    LDX   #$10            ;illegal token, generate syntax error
299           JMP   ERROR
300 READ      JSR   CHRGET          ;advance TXTPTR to next token
301           CMP   #$D0            ;"=" token value
302           BNE   SYNERR
303           JSR   CHRGET          ;advance TXTPTR to string variable
304           JSR   PTRGET          ;find string text pointer
305           LDY   #$01            ;retrieve pointer low byte
306           LDA   (VARPNT),Y
307           TAX                   ;save low byte
308           INY                   ;retrieve pointer high byte
309           LDA   (VARPNT),Y
310           STX   VARPNT          ;reset VARPNT to point to text data
311           STA   VARPNT+1
312           LDY   #$00            ;read first character of string's text
313           LDA   (VARPNT),Y
314           AND   #%11011111      ;turn character into upper-case letter
315 RDCOLR    CMP   #'B'            ;want background color?
316           BNE   RDCAPS          ;no, check for Caps Lock status
317           JSR   CHKCOM          ;yes, now look for comma separator
318           JSR   PTRGET          ;find/create return status variable
319           CLC                   ;enter native mode
320           XCE
321           LDA   >TBCOLOR        ;find current text/background color
322           SEC                   ;return to native mode
323           XCE
324           AND   #%00001111      ;clear text color
325           TAY                   ;prepare to stuff result into FAC
326           LDA   #$00
327           JSR   GIVAYF          ;float A,Y into FAC
328           LDX   VARPNT          ;prepare to store result
329           LDY   VARPNT+1
330           JSR   MOVMF           ;store result into variable
331           RTS                   ;done!
332 RDCAPS    CMP   #'C'            ;want Caps Lock status?
333           BNE   RDHCOL          ;no, check for border color value
334           JSR   CHKCOM          ;yes, now look for comma separator
335           JSR   PTRGET          ;find/create return status variable
336           CLC                   ;enter native mode
337           XCE
338           LDA   >KYMODREG       ;find current special key register
339           SEC                   ;return to emulation mode
340           XCE
341           AND   #%00000100      ;mask out all bits but Caps Lock
342           BEQ   NOCAPS          ;if zero, Caps Lock was UP
343 CAPS      LDY   #$01            ;not zero, Caps Lock was DOWN
344           BNE   STUFCAPS        ; (forced branch always taken)
345 NOCAPS    LDY   #$00            ;return zero, Caps Lock was UP
346 STUFCAPS  LDA   #$00            ;prepare to float result into FAC
347           JSR   GIVAYF          ;float into FAC
348           LDX   VARPNT          ;prepare to store result
349           LDY   VARPNT+1
350           JSR   MOVMF           ;store result
351           RTS                   ;done!
352 RDHCOL    CMP   #'H'            ;want border color status?
353           BNE   RDCTRL          ;no, check for Control Key status
354           JSR   CHKCOM          ;yes, check for comma separator
355           JSR   PTRGET          ;find/create return status variable
356           CLC                   ;enter native mode
357           XCE
358           LDA   >CLOCKCTL       ;find current border color
359           SEC                   ;return to emulation mode
360           XCE
361           AND   #%00001111      ;clear all bits but border color
362           TAY                   ;prepare to float result into FAC
363           LDA   #$00
364           JSR   GIVAYF          ;float into FAC
365           LDX   VARPNT          ;prepare to store result
366           LDY   VARPNT+1
367           JSR   MOVMF           ;store result
368           RTS                   ;done!
369 RDCTRL    CMP   #'K'            ;want Control Key status?
370           BNE   RDLANG          ;no, check for Language
371           JSR   CHKCOM          ;yes, check for comma separator
372           JSR   PTRGET          ;find/create return status variable
373           CLC                   ;enter native mode
374           XCE
375           LDA   >KYMODREG       ;find current special key register
376           SEC                   ;return to emulation mode
377           XCE
378           AND   #%00000010      ;clear out all bits but Control Key
379           BEQ   NOCTRL          ;if zero, Ctrl Key is presently UP
380 CTRL      LDY   #$01            ;not zero, Ctrl Key is presently DOWN
381           BNE   STUFCTRL        ; (forced branch always taken)
382 NOCTRL    LDY   #$00            ;return zero, Ctrl Key was UP
383 STUFCTRL  LDA   #$00            ;prepare to float result into FAC
384           JSR   GIVAYF          ;float into FAC
385           LDX   VARPNT          ;prepare to store result
386           LDY   VARPNT+1
387           JSR   MOVMF           ;store result
388           RTS                   ;done!
389 RDLANG    CMP   #'L'            ;want current language?
390           BNE   RDNKPD          ;no check for Numeric Keypad keypress
391           JSR   CHKCOM          ;yes, check for comma separator
392           JSR   PTRGET          ;find/create return status variable
393           CLC                   ;enter native mode
394           XCE
395           LDA   >LANGSEL        ;find current language
396           SEC                   ;return to emulation mode
397           XCE
398           AND   #%11100000      ;mask out NTSC/PAL and primary bits
399           LSR   A               ;move 3 bits to low nibble
400           LSR   A               ; (YYYxxxxx to xxxxxYYY, Y = language)
401           LSR   A
402           LSR   A
403           LSR   A
404           TAY                   ;prepare to float result into FAC
405           LDA   #$00
406           JSR   GIVAYF          ;float into FAC
407           LDX   VARPNT          ;prepare to store result
408           LDY   VARPNT+1
409           JSR   MOVMF           ;store result
410           RTS                   ;done!
411 RDNKPD    CMP   #'N'            ;want Numeric Keypad keypress status
412           BNE   RDSHFT          ;no, check for Shift Key status
413           JSR   CHKCOM          ;yes, check for comma separator
414           JSR   PTRGET          ;find/create return status variable
415           CLC                   ;enter native mode
416           XCE
417           LDA   >KYMODREG       ;find current special key register
418           SEC                   ;return to emulation mode
419           XCE
420           AND   #%00010000      ;clear out all bits but Numeric Keypad
421           BEQ   NONKPD          ;if zero, no Numeric Keypad key pressed
422 NKPD      LDY   #$01            ;not zero, Numeric Keypad key pressed
423           BNE   STUFNKPD        ; (forced barnch always taken)
424 NONKPD    LDY   #$00            ;return zero, not a Numeric Keypad key
425 STUFNKPD  LDA   #$00            ;prepare to float result into FAC
426           JSR   GIVAYF          ;float into FAC
427           LDX   VARPNT          ;prepare to store result
428           LDY   VARPNT+1
429           JSR   MOVMF           ;store result
430           RTS                   ;done!
431 RDSHFT    CMP   #'S'            ;want Shift Key status?
432           BNE   RDTEXT          ;no, check for text color
433           JSR   CHKCOM          ;yes, check for comma separator
434           JSR   PTRGET          ;find/create return status variable
435           CLC                   ;enter native mode
436           XCE
437           LDA   >KYMODREG       ;find current special key register
438           SEC                   ;return to emulation mode
439           XCE
440           AND   #%00000001      ;clear out all bits but Shift key status
441           BEQ   NOSHFT          ;if zero, Shift Key is not being pressed
442 SHFT      LDY   #$01            ;not zero, Shift Key is being pressed
443           BNE   STUFSHFT        ; (forced branch always taken)
444 NOSHFT    LDY   #$00            ;return zero, Shift Key not pressed
445 STUFSHFT  LDA   #$00            ;prepare to float result into FAC
```

LISTING 1: HELPER Source Code *continued*

```
446             JSR   GIVAYF          ;float into FAC
447             LDX   VARPNT          ;prepare to store result
448             LDY   VARPNT+1
449             JSR   MOVMF           ;store result
450             RTS                   ;done!
451 RDTEXT      CMP   #'T'            ;want current text color?
452             BNE   RDSPD           ;no check for system speed
453             JSR   CHKCOM          ;yes, check for comma separator
454             JSR   PTRGET          ;find/create return status variable
455             CLC                   ;enter native mode
456             XCE
457             LDA   >TBCOLOR        ;find current text/background color
458             SEC                   ;return to emulation mode
459             XCE
460             AND   #%11110000      ;mask out background color bits
461             LSR   A               ;move 4 bits to low nibble
462             LSR   A               ; (YYYYxxxx to xxxxYYYY, Y = color bits)
463             LSR   A
464             LSR   A
465             TAY                   ;prepare to float result into FAC
466             LDA   #$00
467             JSR   GIVAYF          ;float into FAC
468             LDX   VARPNT          ;prepare to store result
469             LDY   VARPNT+1
470             JSR   MOVMF           ;store result
471             RTS                   ;done!
472 RDSPD       CMP   #'V'            ;want current system speed?
473             BEQ   V1              ;yes
474             LDX   #$35            ;no, illegal quantity error
475             JMP   ERROR
476 V1          JSR   CHKCOM          ;check for comma separator
477             JSR   PTRGET          ;find/create return status variable
478             CLC                   ;enter native mode
479             XCE
480             LDA   >CYAREG         ;find current system speed
481             BPL   SLOW            ;MSB low, speed set to SLOW
482 FAST        LDY   #$01            ;speed set to FAST
483             BNE   DONE            ; (forced branch always taken)
484 SLOW        LDY   #$00            ;speed set to SLOW
485 DONE        SEC                   ;return to emulation mode
486             XCE
487             LDA   #$00            ;prepare to float result into FAC
488             JSR   GIVAYF          ;float into FAC
489             LDX   VARPNT          ;prepare to store result
490             LDY   VARPNT+1
491             JSR   MOVMF           ;store result
492 END         RTS                   ;done!
493             END
```

END OF LISTING 1

LISTING 2: HELPER

Start: 7000 Length:320

```
A8 7000:18 FB AF 36 C0 E0 29 00
AF 7008:8D D5 70 AF 2B C0 E0 29
CA 7010:E8 8D E1 70 AF 22 C0 E0
7E 7018:8D E7 70 AF 34 C0 E0 29
8B 7020:0F 8D F3 70 38 FB AD FF
35 7028:BF 10 10 A9 01 20 58 A2
90 7030:64 50 A9 96 85 51 20 8E
4B 7038:F2 D0 0C A9 03 20 F5 BE
FD 7040:90 05 A2 4D 4C 12 D4 48
24 7048:48 A9 7E 85 3C A9 70 85
E6 7050:3D A9 1F 85 3E A9 73 85
D4 7058:3F 64 42 68 85 43 A0 00
B6 7060:20 2C FE A9 4C 8D F5 03
8E 7068:AD F6 03 8D 83 70 AD F7
B9 7070:03 8D 84 70 A9 00 8D F6
30 7078:03 68 8D F7 03 60 C9 5F
B6 7080:F0 03 4C 58 FF 18 FB E2
0F 7088:20 38 FB 20 B1 00 C9 B6
C2 7090:D0 33 20 B1 00 C9 D0 F0
D2 7098:05 A2 10 4C 12 D4 20 F5
8B 70A0:E6 8A C9 08 90 05 A2 35
16 70A8:4C 12 D4 0A 0A 0A 0A 0A
5E 70B0:09 08 85 A1 18 FB AF 2B
10 70B8:C0 E0 29 10 05 A1 8F 2B
23 70C0:C0 E0 38 FB 60 C9 AE D0
4F 70C8:32 20 B1 00 18 FB AF 36
67 70D0:C0 E0 29 7F 09 80 8F 36
C6 70D8:C0 E0 AF 2B C0 E0 29 10
53 70E0:09 18 8F 2B C0 E0 A9 F6
77 70E8:8F 22 C0 E0 AF 34 C0 E0
13 70F0:29 F0 09 06 8F 34 C0 E0
D0 70F8:38 FB 60 C9 A9 D0 1C 20
3B 7100:F5 E6 A5 A1 F0 04 A9 80
D5 7108:85 A1 18 FB AF 36 C0 E0
E5 7110:29 7F 05 A1 8F 36 C0 E0
```

```
3A | 7118:38 FB 60 C9 A0 D0 20 20
B3 | 7120:F5 E6 8A C9 10 90 05 A2
2F | 7128:35 4C 12 D4 85 A1 18 FB
44 | 7130:AF 22 C0 E0 29 F0 05 A1
15 | 7138:8F 22 C0 E0 38 FB 60 C9
4A | 7140:92 D0 20 20 F5 E6 8A C9
7F | 7148:10 90 05 A2 35 4C 12 D4
9B | 7150:85 A1 18 FB AF 34 C0 E0
C2 | 7158:29 F0 05 A1 8F 34 C0 E0
DC | 7160:38 FB 60 C9 89 D0 30 20
D5 | 7168:B1 00 C9 D0 F0 05 A2 10
BB | 7170:4C 12 D4 20 F5 E6 8A C9
A9 | 7178:10 90 05 A2 35 4C 12 D4
BE | 7180:0A 0A 0A 0A 85 A1 18 FB
4C | 7188:AF 22 C0 E0 29 0F 05 A1
D3 | 7190:8F 22 C0 E0 38 FB 60 C9
CA | 7198:87 F0 05 A2 10 4C 12 D4
48 | 71A0:20 B1 00 C9 D0 D0 F4 20
13 | 71A8:B1 00 20 E3 DF A0 01 B1
3B | 71B0:83 AA C8 B1 83 86 83 85
FE | 71B8:84 A0 00 B1 83 29 DF C9
D6 | 71C0:42 D0 1E 20 BE DE 20 E3
7B | 71C8:DF 18 FB AF 22 C0 E0 38
88 | 71D0:FB 29 0F A8 A9 00 20 F2
50 | 71D8:E2 A6 83 A4 84 20 2B EB
D2 | 71E0:60 C9 43 D0 25 20 BE DE
24 | 71E8:20 E3 DF 18 FB AF 25 C0
69 | 71F0:E0 38 FB 29 04 F0 04 A0
D9 | 71F8:01 D0 02 A0 00 A9 00 20
67 | 7200:F2 E2 A6 83 A4 84 20 2B
13 | 7208:EB 60 C9 48 D0 1E 20 BE
90 | 7210:DE 20 E3 DF 18 FB AF 34
26 | 7218:C0 E0 38 FB 29 0F A8 A9
FE | 7220:00 20 F2 E2 A6 83 A4 84
37 | 7228:20 2B EB 60 C9 4B D0 25
95 | 7230:20 BE DE 20 E3 DF 18 FB
CD | 7238:AF 25 C0 E0 38 FB 29 02
AE | 7240:F0 04 A0 01 D0 02 A0 00
BC | 7248:A9 00 20 F2 E2 A6 83 A4
D9 | 7250:84 20 2B EB 60 C9 4C D0
8E | 7258:23 20 BE DE 20 E3 DF 18
A4 | 7260:FB AF 2B C0 E0 38 FB 29
49 | 7268:E0 4A 4A 4A 4A 4A A8 A9
A6 | 7270:00 20 F2 E2 A6 83 A4 84
1A | 7278:20 2B EB 60 C9 4E D0 25
B0 | 7280:20 BE DE 20 E3 DF 18 FB
6F | 7288:AF 25 C0 E0 38 FB 29 10
FD | 7290:F0 04 A0 01 D0 02 A0 00
0F | 7298:A9 00 20 F2 E2 A6 83 A4
66 | 72A0:84 20 2B EB 60 C9 53 D0
70 | 72A8:25 20 BE DE 20 E3 DF 18
94 | 72B0:FB AF 25 C0 E0 38 FB 29
41 | 72B8:01 F0 04 A0 01 D0 02 A0
6C | 72C0:00 A9 00 20 F2 E2 A6 83
A3 | 72C8:A4 84 20 2B EB 60 C9 54
C1 | 72D0:D0 22 20 BE DE 20 E3 DF
70 | 72D8:18 FB AF 22 C0 E0 38 FB
5D | 72E0:29 F0 4A 4A 4A 4A A8 A9
DC | 72E8:00 20 F2 E2 A6 83 A4 84
02 | 72F0:20 2B EB 60 C9 56 F0 05
AF | 72F8:A2 35 4C 12 D4 20 BE DE
78 | 7300:20 E3 DF 18 FB AF 36 C0
14 | 7308:E0 10 04 A0 01 D0 02 A0
B0 | 7310:00 38 FB A9 00 20 F2 E2
B7 | 7318:A6 83 A4 84 20 2B EB 60

TOTAL: 87AF

END OF LISTING 2
```

LISTING 3: HELPER.DEMO

```
37 | 10  REM ************************
C0 | 20  REM *    HELPER.DEMO       *
B9 | 30  REM * By  Kenn Scribner    *
AE | 40  REM * Copyright (C) 1988   *
CB | 50  REM *  MicroSPARC, Inc.    *
24 | 60  REM * Concord, MA 01742    *
45 | 70  REM ************************
FA | 80 D$ =  CHR$ (4)
82 | 90  IF  PEEK ( - 1) <  > 192 THEN  HOME : PRINT
   |       "NOT AN APPLE IIGS!": END
42 | 100  PRINT D$;"PR#3"
F4 | 110  PRINT D$;"BRUN HELPER"
84 | 120  PRINT : PRINT  TAB( 26);"HELPER Demonstrat
   |      ion Program": PRINT : PRINT
50 | 130  PRINT  TAB( 9);"With HELPER, you may perfo
   |      rm several new tasks with your IIgs!": PRIN
   |      T
```

LISTING 3: HELPER.DEMO

```
F8 | 140  POKE 34,7: HOME : PRINT  TAB( 5);"Such as.
     ..changing text colors!"
D5 | 150  PRINT : PRINT : FOR I = 1 TO 13: PRINT
     TAB( 13);"ABCDEFGHIJKLMNOPQRSTUVWXYZ abcde
     fghijklmnopqrstuvwxyz": NEXT I
D3 | 160  VTAB 8: HTAB 38: PRINT "(Press Return to c
     ontinue)";: POKE  - 16368,0: GET A$
28 | 170 CL$ = "T": & _ READ  = CL$,TC: FOR I = 0 TO
      15: & _ TEXT  = I: FOR J = 1 TO 800: NEXT
     J: NEXT I: & _ TEXT  = TC
77 | 180  PRINT : VTAB 8: HTAB 1: CALL  - 868: PRINT
      TAB( 5);"Or...the background colors!  (Pr
     ess Return to continue";: POKE  - 16368,0:
     GET A$
D3 | 190 CL$ = "B": & _ READ  = CL$,BC: FOR I = 0 TO
      15: & _ COLOR= I: FOR J = 1 TO 800: NEXT J
     : NEXT I: & _ COLOR= BC
03 | 200  HOME : PRINT  TAB( 5);"You may even change
      the border colors!  (Press return to conti
     nue)";: POKE  - 16368,0: GET A$
A0 | 210 CL$ = "H": & _ READ  = CL$,HC: FOR I = 0 TO
      15: & _ HCOLOR= I: FOR J = 1 TO 800: NEXT
     J: NEXT I: & _ HCOLOR= HC
DB | 220  HOME : PRINT " The IIGS is also capable of
      printing characters found in other languag
     es..."
11 | 230  PRINT : PRINT "  These characters are repl
     aced when a different language is selected.
     .."
82 | 240  PRINT : PRINT  TAB( 26);"#  @  [  ]  \  0
      {  }  |  ~"
BF | 250 CL$ = "L": & _ READ  = CL$,LA: PRINT : PRIN
     T  TAB( 5);"They look like this for English
      in the USA  (Press Return to continue)  ";
     : & _ LOAD  = 0: POKE  - 16368,0: GET A$
45 | 260  VTAB 14: HTAB 44: CALL  - 868: PRINT "UK
     (Press Return to continue)  ";: & _ LOAD  =
      1: POKE  - 16368,0: GET A$
0C | 270  VTAB 14: HTAB 25: CALL  - 868: PRINT "in F
     rench  (Press Return to continue)  ";:
     & _ LOAD  = 2: POKE  - 16368,0: GET A$
CC | 280  VTAB 14: HTAB 28: CALL  - 868: PRINT "Dani
     sh  (Press Return to continue)  ";: & _ LOA
     D  = 3: POKE  - 16368,0: GET A$
2A | 290  VTAB 14: HTAB 28: CALL  - 868: PRINT "Span
     ish  (Press Return to continue)  ";: & _
     LOAD  = 4: POKE  - 16368,0: GET A$
AA | 300  VTAB 14: HTAB 28: CALL  - 868: PRINT "Ital
     ian  (Press Return to continue)  ";: & _
     LOAD  = 5: POKE  - 16368,0: GET A$
DF | 310  VTAB 14: HTAB 28: CALL  - 868: PRINT "Germ
     an  (Press Return to continue)  ";: & _ LOA
     D  = 6: POKE  - 16368,0: GET A$
90 | 320  VTAB 14: HTAB 28: CALL  - 868: PRINT "Swed
     ish  (Press Return to continue)  ";: & _
     LOAD  = 7: POKE  - 16368,0: GET A$
0A | 330  & _ LOAD  = LA: POKE 34,4: HOME
A0 | 340  PRINT  TAB( 5);"You may read values for th
     e text, background, and border colors, as":
     PRINT : PRINT  TAB( 5);"well as read the c
     urrent display language, the status of the"

2C | 350  PRINT : PRINT  TAB( 5);"Caps Lock key, Con
     trol key, Shift key, and a Numeric Keypad k
     ey.": PRINT : PRINT  TAB( 5);"You also have
      the capability of reading and changing the
      system's speed."
77 | 360  PRINT : PRINT  TAB( 5);"If you get totally
      lost, you have a master restore capability
      that": PRINT : PRINT  TAB( 5);"will reset
     the system to the way it was when you start
     ed!"
91 | 370  VTAB 24: HTAB 20: PRINT "Press Return to r
     eturn to Applesoft...";: POKE  - 16368,0:
     GET A$: POKE 34,0: & _ RESTORE : HOME : EN
     D
```

TOTAL: CE08

END OF LISTING 3