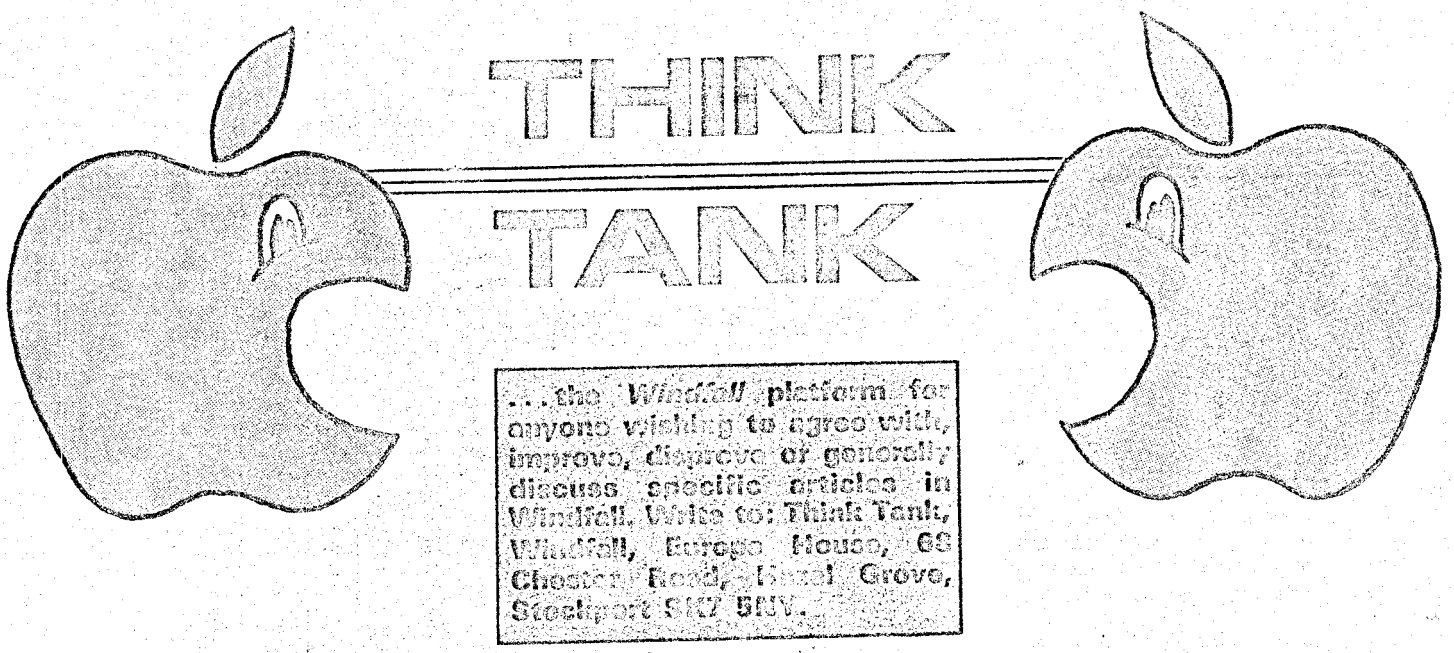


Get HELP while on the run...

LOOKING through a few back issues of *Windfall* recently, I came across some correspondence about getting "help" pages on view while a program was running, writes **J.P. Lewis.**

The point being discussed was the danger of loading binary file images of a previously saved screen from disc.

Since this is a tediously slow process anyway, irrespective of any problems it may cause, I thought that I would share my method of generating help information (Figure I) with other *Windfall* readers.

The idea is simple. Since the 16k language card is not normally available to Applesoft programs, use it to hold copies of information screens. These screens can be copied onto the $400-$7FF memory map very rapidly and the original screen can be stored elsewhere as a temporary measure until the information has been read.

My method is to save up to 15 pages on the language card, and to use the last 1k as a buffer for the video memory map. To choose which of the 15 pages I want to look at, I make use of Applesoft's USR function, which passes its parameter to the floating point accumulator (FAC).

This can be picked up by the machine code and converted into the corresponding memory locations. There is a little problem in this however, as there are two sections of memory in the language card addressed as $D000–$DFFF. This requires a little extra handling, but once a suitable system has been devised for numbering the help pages, the coding is fairly straightforward.

I think the program is self-explanatory, but I apologise to any purists for all the self-modifying code. Points to note are:

- HELP pages 0 to 3 are stored in bank 2 of $D000–$DFFF, pages 4 to 10 are stored in $E000–$FFFF, and pages 11 to 14 are in bank 1 of $D000–$DFFF.
- A HELP page stays on view until the

```
;Generating HELP pages from 16k RAM

;J.P.Lewis    1/5/83

;Pie 2 editor with ASM/65 assembler

        .OPT NOL,NOS
        .OFS=$60

VIDEO   =$400
KBD     =$C000
KBDSTB  =$C010
RAMON1  =$C08B  ;Select Ram read on language card, bank1
RAMON2  =$C083  ;ditto, bank 2
RCMON   =$C082  ;Deselect Ram read, (enable ROM)
IQERR   =$E199  ;Illegal quantity error message.
CONINT  =$E6FB  ;Convert contents of FAC into an integer
                ;in the X register.

        *=$300

        JSR CONINT
        CPX #$F         ;Check that a valid page
        BCC SAFE        ;number has been chosen.
        JMP IQERR

SAFE    CPX #$B         ;Pages 0 to 10 use bank 2,
        BCC BANK2       ;pages 11 to 14 use bank 1
        BIT RAMON1      ;Switch to Page 1 of RAM
        BIT RAMON1
        TXA             ;To use the arithmetic below
        SEC             ;to calculate the memory locations
        SBC #$B         ;used, subtract 11 from the page
        JMP SWAP        ;number, and work from there.

BANK2   BIT RAMON2      ;Switch to page 2 of RAM
        BIT RAMON2
        TXA             ;Convert the chosen page number into
SWAP    ASL A           ;a memory reference: multiply by 4
        ASL A           ;then add $D0.  Pages zero and 11
        CLC             ;are at $D000-$D3FF, and so on.
        ADC #$D0
        STA DUMP2+2     ;In-line modification of the
        LDA #$FC        ;code used for copying the pages
        STA DUMP1+2     ;from the video to the language
        LDY #$4         ;card.
        STY VID1+2
        STY VID2+2

        LDX #0
VID1    LDA VIDEO,X     ;Save the video screen at
DUMP1   STA $FC00,X     ;$FC00 to $FFFF.
DUMP2   LDA $FFFF,X     ;Dummy address.  Load the chosen page
VID2    STA VIDEO,X     ;from the language card to the screen.
        INX
        BNE VID1
        INC VID1+2      ;Move on to the next 256 of data.
        INC VID2+2
        INC DUMP1+2
        INC DUMP2+2
        DEY             ;Check to see if 4 blocks of 256 bytes
        BNE VID1        ;have been shifted.
WAIT    LDA KBD         ;Wait for the <return> key
        BPL WAIT        ;to be pressed before
        CMP #$8D        ;loading the original page
        BNE WAIT        ;back onto the screen.
        BIT KBDSTB

        LDY #$FC        ;Reset the transfer code back to
        STY BACK1+2     ;the correct starting values.
        LDY #4
        STY BACK2+2
        LDX #0
BACK1   LDA $FC00,X     ;Shift 256 bytes.
BACK2   STA VIDEO,X
        INX
        BNE BACK1
        INC BACK1+2     ;Move to next block of 256 bytes
        INC BACK2+2
        DEY
        BNE BACK1       ;Check for all 4 blocks moved.

        BIT ROMON       ;Switch ROM back on, and write-protect
        RTS             ;the language card before returning
        .END
```

Figure I: The HELP program

THINK TANK

return key is pressed. This should be pointed out either at the start of your Applesoft program or on each help page.

☐ After the original screen is copied back onto the video, the ramcard is write-protected.

☐ Calls for pages above 14 result in an ILLEGAL QUANTITY error message appearing. If you have fewer than 15 pages this can be modified in the machine code.

To use this routine, you should BLOAD the code (given as a hex dump in Figure II) at $300, then change the JMP command at $A to point to $300.

You also need to put in a lot of preparation for all the HELP pages you want to have available. To keep this as simple as possible I have a short Applesoft program that inputs 24 strings, then clears the screen, outputs them, and ends with a BSAVE of the screen memory map.

When I have all the screens I want, I load them in order from $1000 onwards, and save them in one single block. Figure III is a fragment of a program (using only the first 11 Help pages) that shows how I implement the system.

○ Mr Lewis' approach to using the language card to print screen HELP pages is similar to one that appeared in the June 1983 issue of Windfall, but it allows more pages and passes the page number in a more convenient way.

```
*300.384

0300- 20 FB E6 E0 0F 90 03 4C
0308- 99 E1 E0 0B 90 0D 2C 8B
0310- C0 2C 8B C0 8A 38 E9 0B
0318- 4C 22 03 2C 83 C0 2C 83
0320- C0 8A 0A 0A 18 69 D0 8D
0328- 41 03 A9 FC 8D 3E 03 A0
0330- 04 8C 3B 03 8C 44 03 A2
0338- 00 BD 00 08 9D 00 00 BD
0340- 00 E0 9D 00 08 E8 D0 F1
0348- EE 3B 03 EE 44 03 EE 3E
0350- 03 EE 41 03 88 D0 E2 AD
0358- 00 C0 10 FB C9 8D D0 F7
0360- 2C 10 C0 A0 FC 8C 71 03
0368- A0 04 8C 74 03 A2 00 BD
0370- 00 00 9D 00 08 E8 D0 F7
0378- EE 71 03 EE 74 03 88 D0
0380- EE 2C 82 C0 60
```

Figure II: Hex dump of the HELP code

```
10 RMSWTCH = 12 * 4096 + 8 * 16 +
     1: REM  This gives $C081.
20  POKE RMSWTCH,1: POKE RMSWTCH,
     1: REM  Write enable the lan
     guage card.
30  PRINT  CHR$ (4)"BLOAD HELPDAT
     A,A$D000"
40  PRINT  CHR$ (4)"BLOAD Help.co
     de"
50  POKE 11,0: POKE 12,3: REM   S
     et up the USR pointers
60  REM

Rest of program


300  REM  Sample HELP call, which
      assumes that the user has b
     een warned to respond with a
      '?' if he needs help
310 HELP = 7: REM   Page seven of
      help is appropriate here
320  INPUT "Next action ? ";A$
330  IF A$ = "?" THEN HELP =  USR
     (HELP): GOTO 310
340  REM  Rest of program.
```

Figure III: How to make use of HELP in Applesoft programs

DOS stands in danger from altered reset vectors

WITH reference to Dave Miller's article on disabling the autostart reset *(Windfall, March 1983)* users should be aware that the use of altered reset vectors can cause trouble with DOS, writes **M.F. Sheppard.**

As explained in the Apple Reference Manual, when RESET is pressed the I/O vectors (CSW and KSW) are set to point to the standard routines (for screen and keyboard respectively), which disconnects DOS.

Normally when DOS is in operation the reset vectors point to the DOS Warm-start entry at $9DBF, so that DOS can re-connect itself. If however they point anywhere else (including any Basic entry), then DOS may not be able to get back in.

Try the following short program:

```
10 POKE 1010,102:POKE 1011,213:CALL -1169
20 INPUT A$:REM ALLOWS 'RESET' TO BE PRESSED
40 PRINT CHR$(4);"CATALOG"
```

When RUN and any normal input made in response to the "?", the CATALOG is displayed normally. If however you press RESET at the pause, although the program runs again you will just see the word CATALOG displayed after the input, because the CTRL/D has had no effect.

Now the discussion on Pages 101-105 of the DOS Manual shows that DOS can be re-connected by the routine at (hex) 3EA, (decimal) 1002. Adding the following line to the program above does it, and the complete program will re-run correctly after RESET.

```
30 CALL 1002: PRINT
```

This does not need any "PR in" statement because the I/O vectors are not already pointing to DOS. However, the first character after this CALL is not printed, so a blank PRINT is needed to ensure that the CTRL/D is effective.

I discovered this effect in a TASC compiled program, but it is also applicable to normal Applesoft. With the amendment above, it is perfectly possible to make a compiled program restart on RESET, or jump to any specific line, by pointing the Reset vectors to the appropriate location. This may be found by a trial compilation using dummy values for the reset vectors.

Incidentally, K. Williamson has asked me to point out that the answer to the loss of TASC COMMON strings described by C.A.G. Webster and himself *(Windfall, September 1982)* is given in the TASC manual. Use REM!CLEARCHAIN before the command to BRUN the next program. I have verified that this moves all COMMON strings to the top of memory and resets the string pointers correctly. ¢

Applelip

A text file can be used to save the variables and arrays etc of a crashed Basic program by typing the following from the keyboard (assuming that D$ was defined as usual in the program).

```
POKE 51,0 : POKE 118,254
PRINT D$"OPEN FILENAME"
PRINT D$"WRITE FILENAME"
PRINT (variables, arrays)
PRINT D$"CLOSE FILENAME"
```

Printing error messages

In the June 1982 issue of Windfall an Appletip presented a way of printing the DOS error messages from within an ONERR GOTO subroutine without having to incorporate the text of them within the host program.

The following subroutine will do the same, but also print any appropriate Applesoft error messages. The short piece of code (to clear up some stack errors) from the Applesoft manual has also been incorporated.

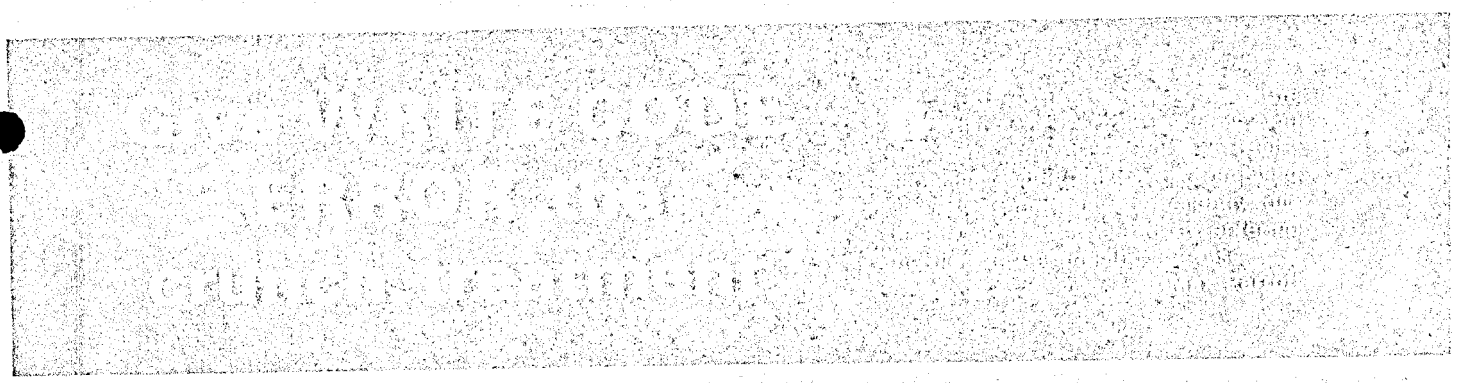
The subroutine is relocatable, but for convenience has been assembled for the popular $300 (768) address. It is best, I think, to type it in and save it to disc, **BLOAD**ing it whenever and wherever it is required. Top save the subroutine type **BSAVE ERRORPRINTER,A$300,L$3C.**

Max Parrott

```
*DISASSEMBLY AND HEXADECIMAL DUMP

*300LL

0300-  20 FB DA   JSR   $DAFB
0303-  A6 DE      LDX   $DE
0305-  F0 09      BEQ   $0310
0307-  E0 10      CPX   #$10
0309-  B0 05      BCS   $0310
030B-  20 02 A7   JSR   $A702
030E-  30 12      BMI   $0322
0310-  BD 60 D2   LDA   $D260,X
0313-  48         PHA
0314-  20 5C DB   JSR   $DB5C
0317-  E8         INX
0318-  68         PLA
0319-  10 F5      BPL   $0310
031B-  A9 50      LDA   #$50
031D-  A0 D3      LDY   #$D3
031F-  20 3A DB   JSR   $DB3A
0322-  A9 58      LDA   #$58
0324-  A0 D3      LDY   #$D3
0326-  20 3A DB   JSR   $DB3A
0329-  A5 DB      LDA   $DB
032B-  A6 DA      LDX   $DA
032D-  20 24 ED   JSR   $ED24
0330-  68         PLA
0331-  A8         TAY
0332-  68         PLA
0333-  A6 DF      LDX   $DF
0335-  9A         TXS
0336-  48         PHA
0337-  98         TYA
0338-  48         PHA
0339-  4C FB DA   JMP   $DAFB
033C-  00         BRK
033D-  00         BRK
033E-  00         BRK
033F-  00         BRK
0340-  00         BRK
0341-  00         BRK
0342-  00         BRK
0343-  00         BRK
0344-  00         BRK
*300.33B
0300- 20 FB DA A6 DE F0 09 E0
0308- 10 B0 05 20 02 A7 30 12
0310- BD 60 D2 48 20 5C DB E8
0318- 68 10 F5 A9 50 A0 D3 20
0320- 3A DB A9 58 A0 D3 20 3A
0328- DB A5 DB A6 DA 20 24 ED
0330- 68 A8 68 A6 DF 9A 48 98
0338- 48 4C FB DA
```



Easy access to the assembler

I read on Page 22 of the February 1982 issue of Windfall how to save the Integer Basic mini-assembler to disc and then access it with an Applesoft program. I have found a simpler method of accessing the assembler direct from the keyboard if you have a language card.

Method

Load DOS 3.3 System Master
Type INT
Type CALL–151
Type F666G

Screen

]
]INT
>CALL–151
*F666G
!.

To run programs return to the monitor mode by typing $FF69G. To exit from the assembler to Integer Basic type $E003G.

Graham Shields