Digital Applesoft

Online supplements to make your programs healthier

By Ivan Drucker

In the article "Hardcore Applesoft" in the March 2010 issue of *Juiced.GS*, there is a bug in the TRY structure (Figure 4 in that issue's page 21) which makes it unsafe to use GOSUB or FOR...NEXT from a TRY block inside a subroutine. To fix this problem, change the following lines as shown. Every TRY structure in your program will need its own unique variable name, not used elsewhere, in place of XX.

10101 XX = PEEK(248) : ONERR GOTO 10150

10151 POKE 216,0 : ER = PEEK (222) : POKE 223,XX : CALL -3288

This makes Applesoft forget about any returning to any GOSUB or FOR which came after the start of the TRY block, so RETURN from your subroutine works correctly, rather than jumping back inside the TRY block. In technical terms, the PEEK gets the stack index at the outset of the TRY, and the POKE/CALL restores it, thus erasing anything put on the stack by GOSUB or FOR after the TRY started.

SWITCH...CASE

Our last installment examined IF...ELSE IF...ELSE structures for decision making. Languages such as C and Java provide an additional decision making structure, SWITCH...CASE, which, after the first true "case" (evaluation), will "fall through" all remaining cases as through they were also true,

```
10100 REM SWITCH
10101 ZZ = 0
    10110 IF (p1=1) THEN ? 1 : ZZ = 1 : REM CASE
    10120 IF ( ZZ OR (p1=2) ) THEN ? 2 : ZZ = 1 : REM CASE
    10130 IF ( ZZ OR (p1=3) ) THEN ? 3 : ZZ = 1 : REM CASE
    10140 ? 0 : REM DEFAULT
10150 REM END SWITCH
```

***Figure 1**. A Structured Applesoft single-line (per case) SWITCH structure.*

```
10100 REM SWITCH
10101 ZZ = 0
10150 REM CASE
10151 ON (condition) GOTO 10152 : GOTO 10200
10152 ZZ = 1
    10160 statement
    10170 etc
10200 REM CASE
10201 ON ( (condition) OR ZZ ) GOTO 10202 : GOTO 10250
10202 ZZ = 1
    10210 statement
    10220 etc
10250 REM CASE
10251 ON ( (condition) OR ZZ ) GOTO 10252 : GOTO 10300
10252 ZZ = 1
    10260 statement
    10270 etc
10300 REM DEFAULT
    10310 statement
    10320 etc
10350 REM END SWITCH
```

***Figure 2**. A Structured Applesoft multiple-line (per case) SWITCH structure.*

```
10000 REM SUB^OpenFile:10000-10499

    10010 LET D$ = CHR$(4) : REM D$->ctrl-D

    10100 REM TRY
    10101 XX = PEEK(248) : ONERR GOTO 10150
        10110 PRINT D$;"VERIFY ";P1$
        10120 PRINT D$;"OPEN ";P1$
        10130 PRINT D$;"CLOSE"
        10135 PRINT "opened and closed file successfully"
    10140 POKE 216,0 : GOTO 10350

    10150 REM CATCH
    10151 POKE 216,0 : ER = PEEK(222) : POKE 223,XX : CALL
          -3288

    10160 REM CATCH ERROR: FILE TYPE MISMATCH
    10161 ON (ER=13) GOTO 10170 : GOTO 10200
        10170 PRINT "ERROR: not a text file"
    10180 GOTO 10350

    10200 REM CATCH ERROR: FILE NOT FOUND, SYNTAX ERROR
          [DOS 3.3]
    10201 ON ( (ER=6) OR (ER=11) ) GOTO 10210 : GOTO 10250
        10210 PRINT "ERROR: invalid filename"
    10220 GOTO 10350

    10250 REM CATCH ERROR: BREAK (CTRL-C)
        10251 ON (ER=255) GOTO 10260 : GOTO 10300
        10260 GOSUB 10360 : REM DO FINALLY
        10270 POKE 117,PEEK(218) : POKE 118,PEEK(219) :
              STOP : RUN : REM THROW
    10280 GOTO 10350

    10300 REM CATCH OTHERS
        10310 GOSUB 10360 : REM DO FINALLY
        10320 RESUME : REM THROW
    10330 GOTO 10350

    10350 REM FINALLY
    10351 GOSUB 10360 : GOTO 10400
        10360 PRINT D$;"CLOSE"
    10399 RETURN

    10400 REM END TRY

10499 RETURN
```

***Figure 3**. A Structured Applesoft subroutine containing a TRY structure with multiple CATCH ERROR blocks, a FINALLY block,*

executing their code blocks until a BREAK (if present) is reached.

The SWITCH structure can also include an optional DEFAULT block at the end which always executes unless there is a BREAK which precedes it. (Other languages permit DEFAULT anywhere in the SWITCH structure, where it will execute only if subsequent cases aren't true, but this style is rarely used and not easily accomplished in Applesoft.)

SWITCH...CASE...DEFAULT is far less commonly used than IF...ELSE IF...ELSE, but for those who wish to use it, it can be structured in Applesoft as shown in Figures 1 and 2 (page 1). To BREAK out of the SWITCH structure from any CASE, GOTO the END SWITCH line and conclude with REM BREAK. Example:

10290 GOTO 10350 : REM BREAK

TRY, TRY AGAIN

Our last issue also included a look at how to handle errors in an Applesoft program with a TRY structure. With this method, your program "tries" to execute code in a TRY block, and if an error occurs, it is "thrown" to a subsequent CATCH block, which handles it. (This supplement assumes you've read the article.)

Here, we'll discuss more options for your TRY structure. In addition to TRY and CATCH, you can use one or more CATCH ERROR blocks to act on specific errors, a FINALLY block for concluding code which always executes whether or not an error occurs, and you can THROW an error if you want the user to see it.

To check for specific errors, start with a standard CATCH as discussed in the main article, shown

in lines 10150-10151 of Figure 3 (page 2). This is followed with one or more CATCH ERROR blocks which act on specific errors by evaluating ER. These are followed by a CATCH OTHERS block, which is executed if the error is not anticipated by the previous CATCH ERROR blocks. (This is essentially the same structure as IF...ELSE IF...ELSE.) As shown in Figure 3, each ON line should be preceded with a separate line which says REM CATCH ERROR: followed by the error name(s) being checked for, and each catch block should conclude with a GOTO to FINALLY (discussed next), if present, or END TRY. The full list of errors and their corresponding codes is shown in Figure 4.

If you wish to have concluding code that always executes regardless of whether or not the TRY block throws an error, create a FINALLY block after your CATCH or CATCH OTHERS block. A common use of FINALLY is to close an open file or perform some other "cleanup" operation you want to have happen no matter what. To make this happen, you need a REM FINALLY line, followed by another line which GOSUBs to the subsequent line, and then on the same line (after the GOSUB), GOTOs the END TRY line. Your code then follows, and must conclude with RETURN. You can see this in lines 10350-10399 of Figure 3. The GOTO which concludes the TRY block and any catch blocks should jump to the FINALLY line rather than END TRY. If you want to RETURN or THROW (discussed next) from a catch block, you must GOSUB to the first line of your FINALLY code and append REM DO FINALLY, as shown in lines 10260 and 10310. This ensures there's no way out of your TRY structure without FINALLY executing.

What happens if the code that you tried threw an error, and your CATCH block caught it, but you want the user to see the error, as though it hadn't been caught? No problem; you deliberately THROW it back to the offending statement, which executes it again, and this time, the error will be displayed and the program will stop. To do this, use the RESUME command, followed by REM THROW, as shown in line 10310. Don't use RESUME for any other purpose, because it is a kind of GOTO, which we don't want in our structured program. In fact, because we want to show the error and stop the program, you should only THROW in scenarios where you are reasonably sure that the offending statement will behave the same way it did when it originally threw the error. If you wish to throw a CTRL-C (BREAK), don't use RESUME; instead, use POKE 117,PEEK(218) : POKE 118,PEEK (219) : STOP : RUN, as shown in line 10270 of Figure 3, which will print a BREAK statement with the correct line number, and restart the program if the user types CONT.

A	0	NEXT WITHOUT FOR
DP	1	LANGUAGE NOT AVAILABLE
DP	2	RANGE ERROR
D	3	RANGE ERROR
P	3	NO DEVICE CONNECTED
DP	4	WRITE PROTECTED
DP	5	END OF DATA
D	6	FILE NOT FOUND
P	6	PATH NOT FOUND
D	7	VOLUME MISMATCH
DP	8	I/O ERROR
DP	9	DISK FULL
DP	10	FILE LOCKED
D	11	SYNTAX ERROR
P	11	INVALID PARAMETER
DP	12	NO BUFFERS AVAILABLE
DP	13	FILE TYPE MISMATCH
DP	14	PROGRAM TOO LARGE
DP	15	NOT DIRECT COMMAND
AP	16	SYNTAX ERROR
P	17	DIRECTORY FULL
P	18	FILE NOT OPEN
P	19	DUPLICATE FILE NAME
P	20	FILE BUSY
P	21	FILE(S) STILL OPEN
A	22	RETURN WITHOUT GOSUB
A	42	OUT OF DATA
A	53	ILLEGAL QUANTITY
A	69	OVERFLOW
A	77	OUT OF MEMORY
A	90	UNDEFINED STATEMENT
A	107	BAD SUBSCRIPT
A	120	REDIMENSIONED ARRAY
A	133	DIVIDE BY ZERO
A	163	TYPE MISMATCH
A	176	STRING TOO LONG
A	191	FORMULA TOO COMPLEX
A	224	UNDEFINED FUNCTION
A	254	REENTER
A	255	BREAK (CTRL-C)

***Figure 4**. Table of PEEK(222) codes and corresponding errors. A=Applesoft, D=DOS 3.3, P=ProDOS.*

Juiced.GS: Order form

2010 Subscription (USA)

Juiced.GS is the last remaining print publication dedicated to the Apple II. Continue reading news, reviews, interviews, and how-to's in 2010 with this subscription for four more quarterly issues.

___ x $19 = $_____

2010 Subscription (International)

Juiced.GS is the last remaining print publication dedicated to the Apple II. Continue reading news, reviews, interviews, and how-to's in 2010 with this subscription for four more quarterly issues.

___ x $26 = $_____

Volume 14 (2009)

Includes interviews with Bob Bishop and Michael J. Mahon; secrets of the IIGS monitor and Integer BASIC; how to use rSounds and ProDOS volumes on a Mac or Unix; and coverage of KansasFest 2009 and Mt. Keira Fest.

___ x $16 = $_____

Volume 13 (2008)

Includes interviews with IIGS laptop designer Ben Heckendorn, ReactiveMicro.com's Henry Courbis, and IIc unboxer Dan Budiac; reviews of the MicroDrive/Turbo and CFFA cards, ADTPro, and the Replica I; and an analysis of disk protection methods.

___ x $16 = $_____

Volume 12 (2007)

Includes interviews with podcaster Carrington Vanston, music group 8 Bit Weapon, and Brøderbund veteran Roland Gustafsson; reviews of SAFE 2, and CiderPress; Ewen Wannop's TCP/IP for Dummies series; and Michael J. Mahon's NadaNet network tutorial.

___ x $16 = $_____

Volume 11 (2006)

Includes a *BBS Documentary* review and director interview; a guide to text adventures; a history of the IIGS; reviews of DiskMaker 8 and the Semi-Virtual Disk; and interviews with Mike Westerfield of The Byte Works and Mike Harvey of *Nibble*.

___ x $16 = $_____

Volume 10 (2005)

Includes how to teach VisiCalc to today's students; a look at Google alternatives; reviews of Contiki, the Uther Ethernet card, Opus][, Silver Platter, and the Mockingboard v1; an interview with Kelvin Sherlock; and KansasFest 2005 coverage.

___ x $16 = $_____

Volume 9 (2004)

Includes an inside look at the SideClick contextual menu manager; a review of VNCView GS; interviews with Replica I creator Vince Briel and Silvern Castle programmer Jeff Fink; Richard Bennett's continuing emulation series; and a memorial tribute to Gary Utter.

___ x $16 = $_____

Volume 8 (2003)

Includes reviews of Adventure Alive and SAFE; a history of Apple II networking; PERL for the Apple II; how to start a user group; Richard Bennett's continuing emulation series; and interviews with Apple historian Steve Weyhrich and hardware geek Jeri Ellsworth.

___ x $16 = $_____

Volume 7 (2002)

Includes reviews of Wings and Virtual GS emulator; comparisons of the best games, word processors, telecom programs, and DAs; how to connect a 4 GB hard drive to an Apple II; a history of Apple II storage; and Richard Bennett's continuing Outback Emulation series.

___ x $16 = $_____

Name: _____

Address: _____ **City:** _____

State/Province: _____ **Zip:** _____ **Country:** _____

Email: _____ **Total order: $**_____

All prices include shipping. Mail your check, made out to Gamebits, to Juiced.GS, P.O. Box 703, Leominster, MA 01453-0703

Or use your credit card to order online at http://www.juiced.gs/