Here's a program that helps you find your way from hex to inverted decimal and back again.

To and Fro with Apple's Inverted Decimal Code

By Don Lancaster

The Apple II, like many other microcomputers, has two ways of calling out locations in its address space.

If you are using assembly or machine language, the 65536 locations in the address space are called out with a hexadecimal code, ranging from 0000 to FFFF.

If you are using Integer BASIC or another higher-level language, these same address locations are called out as a decimal number, ranging from −32768 to +32767. Decimal numbers are more useful in BASIC because they eliminate the strings you might have to use for the hex A, B, C, D, E and F values that stand for decimal 10 through 15.

The obvious thing to do is to let each decimal location be the same numeric equivalent of each hex location. But this would take numbers that go from 0 to 65535, and Apple's Integer BASIC only allows numbers from −32767 to +32767.

To beat this problem, the Apple people invented an inverted decimal code. It uses the bottom half of the memory as direct numeric equivalents and the top half of the memory as upside down decimal negative values. This is roughly akin to Apple II's complement notation, where some of the numbers seem to be heading the wrong way at first glance. Fig. 1 shows two ways the Apple address space is mapped.

For locations hex 0 to 7FFF, the decimal code is the same as its hex value. Decimal 0 to +32767 goes with hex 0 to 7FFF. For locations hex 8001 through FFFF, the decimal code counts backwards, and with a minus sign. This way, hex FFFF is decimal −1; hex FFFE is decimal −2. The down counting continues to −32767 at hex 8002, and finally −32767 at hex 8001.

Location hex 8000 is complex. I'll return to it later. It is coded as decimal −32768, but you can't get there from here without using some tricks.

Why Convert?

Why would you ever have to worry about these two codes? Any time you write a BASIC program, if you want to turn loose the full power of the monitor, you have to use the inverted decimal code to call whatever it is you are after. If you want to change or look at a memory location in your machine, you also have the PEEK and POKE commands, also called as decimal numbers.

Some locations and sequences may make much more sense or appear more regular in hex than in inverted decimal. The screen locations for color graphics, hi-res graphics and the alphanumeric characters are much simpler to understand and much more "logical" in hex notation than in inverted decimal. So switching the coding can sometimes give you insight into what is happening, or point to a newer or better way of doing things.

Finally, if you are studying someone else's BASIC program and see a mysterious CALL 1002, how can you tell what it means? Only with a back and forth conversion can you find out

```
CALL (-32767-1)
PRINT PEEK (-32767-1)
```

Table 1. Location hex 8000 is defined as inverted decimal −32768. Here are two ways to get there without an error message.

Don Lancaster
Synergetics

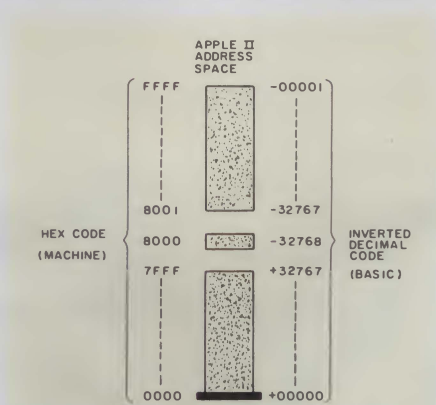

Fig. 1. Two ways of mapping Apple's address space—hex for machine language and inverted decimal for BASIC.

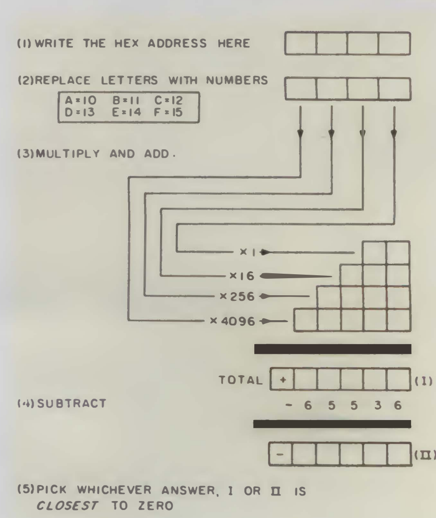

Fig. 2. Worksheet to convert hex locations to inverted decimal locations.

what the program is really up to.

For some reason, people avoid trying to convert between these two codes. Sometimes you can cheat by finding a code equivalent that may be nearby, and then counting down or up in the right direction. But, that's no fun, particularly if you go the wrong way and bomb the machine. And this avoids learning to do things the right way in the first place.

You can also generate a list and look up all the values in either direction. But this takes 65536 double entries and will produce a bunch of computer printouts. There is something better.

Scratchpad Conversions

You can look at four ways of getting from hex to inverted decimal and back again. The worksheet methods let you use a calculator and a sheet of paper. This is convenient if your Apple isn't up, or if you don't want to change what is in it, or if you just have a single location or two to convert.

Fig. 2 shows a worksheet to convert hex locations to inverted decimal locations. You can put another pad of paper along the right margin of the worksheet and work in parallel, using the same worksheet over and over again.

You use this worksheet by writing the hex address at the top and then replacing the letters with their decimal 10 through 15 equivalents. You then go through the multiply-and-add song-and-dance shown, scaling the digits by 1, 16, 256 and 4096. Total I in the figure is a direct decimal conversion of the hex address space. In other machines, this may be all you need.

But, for the Apple, you have to invert the decimal code for everything in the top half of memory. To do this, you subtract 65536 to get a new answer, II. You then pick answer I or II, the one nearest zero, as the correct conversion. Addresses 0000 to 7FFF will convert directly, while 8001 through FFFF will convert to the down-counting negative decimal values.

Fig. 3 shows two sample calculations. Here you prove that the Apple memory location "turn on the hi-res graphics" hex C057 is decimal −16297 (Fig. 3a). A second example

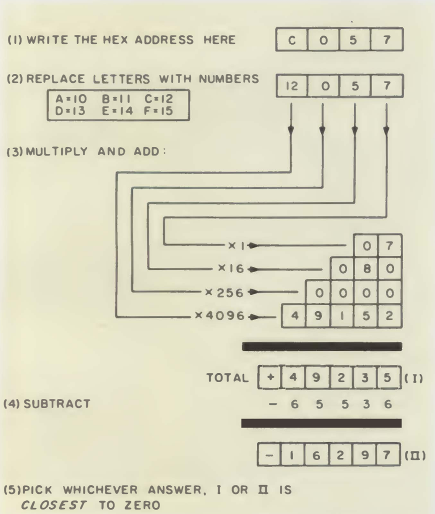

Fig. 3a. Two examples converting hex to inverted decimal.

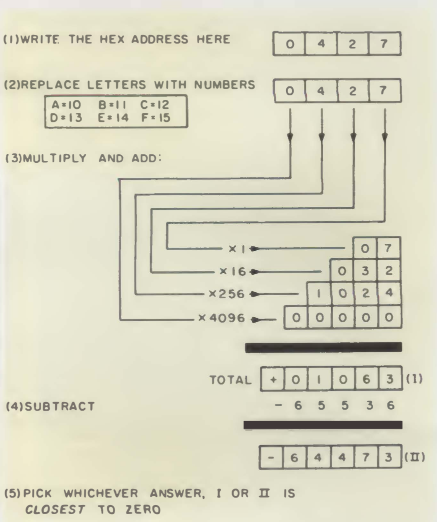

Fig. 3b. (Second example.)

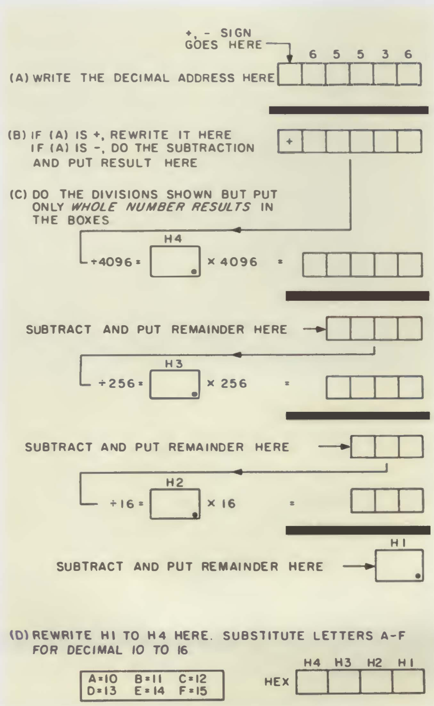

Fig. 4. Worksheet to convert decimal locations to hex locations.

See List of Advertisers on page 210

(Fig. 3b) shows that the upper right-hand character on the screen is both hex 0427 and decimal 1063.

Fig. 4 gives you a new worksheet to go the other way—from inverted decimal to hex. Here, you write the decimal address at the top. If the decimal address is positive, you do nothing to it. If the decimal address is negative, you subtract the address from 65536. Either way, you get a positive number from 0 to 32767.

You then go through a funny game called a math iteration. You try dividing by 4096 and write only the whole number result (the part of the answer to the left of the decimal point on your calculator) in box H4. Now, you multiply back only the whole number part of H4 and subtract to get a remainder.

Repeat this process for 256, 16 and finally 1. Each time, use only whole numbers. This gives you your raw hex digits. Finally, put all your H1 to H4 digits in order and replace the numbers above 9 with their A–F equivalents.

Two examples in Fig. 5 show the Apple locations that toggle the speaker and switch the screen from normal to inverse mode.

Conversion Programs

Listing 1 shows you a program to convert from hex to inverted decimal code. Steps 100–130 do some housework for you, clearing the screen and asking you to input the hex address. This address is read as a string (remember those letters!) called HEX$. You then read each piece of the string in steps 140–170. If you have a letter, step 170 changes the letter to its numeric 10–15 equal.

In step 190, test to see if you are working on the top half of memory. If you are, subroutine 400–420 complements the hex code for you and decides to print a minus sign in front of your final answer. If you are on the bottom half of memory, skip this step.

Then calculate the decimal equivalent in step 200 and print the answer in 210. Steps 220 and 230 do some formatting and repeating.

You can test this program with any of the previous examples. After your first answer, hit the carriage return to enter your next conversion. Do this till all the numbers you need are converted.

The reverse program trip of inverted decimal to hex is shown in Listing 2. It is somewhat longer and takes a fancier string. Again, start with housework in steps 100–130. Step 140 erases a question mark that messes up the display and starts you off assuming a lower half conversion. If you are working on the top half of the address space, step 150 goes to subroutine 400–410 to do repair work.

Step 160 sets up the string that will automatically convert numbers 10 to 15 to their hex letters. Lines 190 through 220 execute the conversion iteration; step 240 adds 8 to the leftmost hex digit if you started out with a negative inverted decimal number. Lines 250–260 print out your answer, while 270 and 280 close the loop to let you repeat.

Operation is about the same as the earlier program. Type RUN and enter the code to be converted. The answer appears. Then use a carriage return to start entering a new number. CTRL C ends the program and returns you to Integer BASIC.

You might like to test this program to prove that step 140 does, in fact, erase a question mark on the screen.

Which Way?

Pick the worksheet methods if you

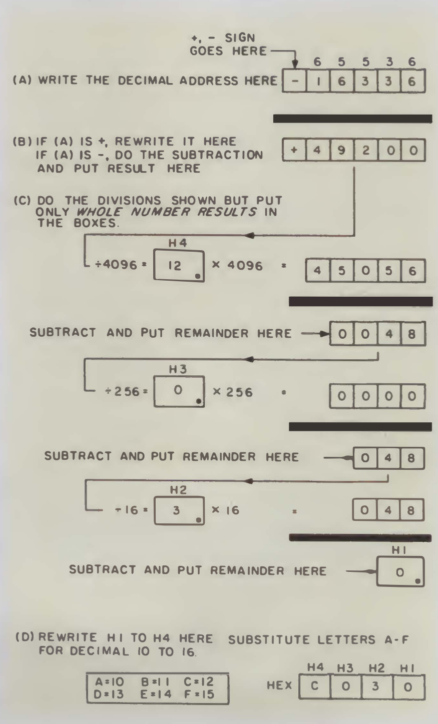

Fig. 5a. Two examples converting inverted decimal to hex.

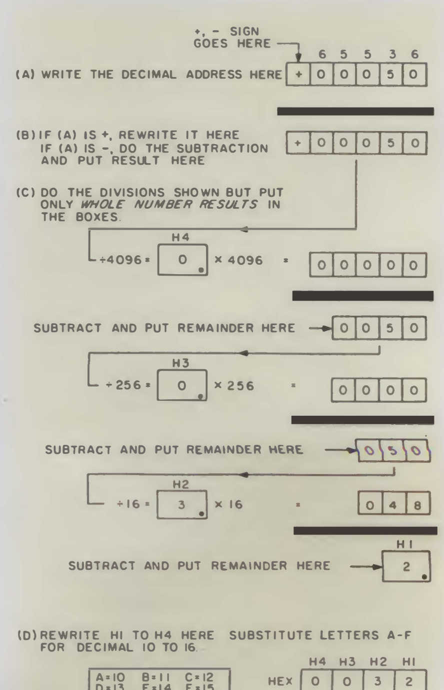

Fig. 5b. (Second example.)

have only a couple of conversions, or are away from your computer. Use the programs for everything else. You might like to combine both programs into one that goes either way. You can also use trial and error to let one program do the backwards trip for you.

The Hex 8000 Problem

Location hex 8000 looks confusing. The Apple people call this decimal −32768, but, if you try getting directly at 8000, you will probably get "> 32767 ERR". Fortunately, location 8000 rarely needs code conversion.

But there is a way. It is sneaky, but it works. Apple allows a value of −32768 *inside* the machine. It just doesn't let you put this number into the machine or take it back out any direct way.

Table 1 shows ways to get at this location. You have to give the Apple something to do that will make it internally compute a value of −32768. For instance, PEEK (−32767−1) will also work without an error message. Should you have some machine-language routine at 8000, CALL (−32767 −1) will also work without an error message.

So, don't be afraid of those funny number games involved in getting from hex to inverted decimal and back again. These codes are simple and easy to use. The two codes and their conversion are versatile tools when you learn to convert them quickly.■

```
100 REM * APPLE HEX TO DECIMAL
110 DIM HEX$(4),A(4)
120 CALL -936: VTAB 6: TAB 10
130 INPUT "HEX ADDRESS ",HEX$
140 FOR N=1 TO 4
150 A(N)=ASC(HEX$(N,N))
160 A(N)=A(N)-176
170 IF A(N)>15 THEN A(N)=A(N)-7
180 NEXT N:HEX$=" "
190 IF A(1)>7 THEN GOSUB 400
200 DECIMAL=A(1)*4096+A(2)*256+A(3)*16+A(4)
210 PRINT : TAB 10: PRINT "IS DECIMAL ";: PRINT HEX$;: PRINT DECIMAL
220 PRINT : PRINT : PRINT : PRINT : PRINT
230 INPUT HEX$: GOTO 120
400 FOR N=1 TO 4:A(N)=15-A(N): NEXT N
410 A(4)=A(4)+1
420 HEX$="-": RETURN
```

Listing 1. Program for hex to inverted decimal conversion.

```
100 REM : APPLE DECIMAL TO HEX
110 DIM HEX$(16),A(4)
120 CALL -936: VTAB 6: TAB 10
130 INPUT "DECIMAL ADDRESS ",DECIMAL
140 POKE 1689,160:FIX=0
150 IF SGN (DECIMAL)<0 THEN GOSUB 400
160 HEX$="0123456789ABCDEF"
170 A(1)=DECIMAL/4096+1
180 R=DECIMAL MOD 4096
190 A(2)=R/256+1
200 R=R MOD 256
210 A(3)=R/16+1
220 A(4)=R MOD 16+1
230 PRINT : TAB 13: PRINT "IS HEX ";
240 IF FIX=1 THEN A(1)=A(1)+8
250 FOR N=1 TO 4
260 PRINT HEX$(A(N),A(N));
270 NEXT N: PRINT : PRINT : PRINT
280 INPUT HEX$: GOTO 120
400 DECIMAL=32767+DECIMAL+1:FIX=1
410 RETURN
```

Listing 2. Program for inverted decimal to hex conversion.

See List of Advertisers on page 210