Subject: (no subject)
Posted by keith on Fri, 24 May 2013 03:40:46 GMT
View Forum Message <> Reply to Message

Message-ID:
Date: Fri, 14-Sep-84 20:24:49 EDT
Article-I.D.: utzoo.4318
Posted: Fri Sep 14 20:24:49 1984
Date-Received: Fri, 14-Sep-84 20:24:49 EDT
Organization: U of Toronto Zoology
Lines: 56

I have read some of the recent news on net.micro.ti and have noticed that some discussion of CALL PEEKs and CALL LOADs has taken place. I know of a few memory locations that may be useful to anyone reading the news for this news-group. There is a two-byte address which can give you information about how much memory you still have available, when you have XBASIC, without memory expansion. I do not know if it will report memory available when memory expansion is attached, as I do not have this. The advantage of this command over SIZE is that it can be incorporated into a program to report memory free during the actual run of a program. I obtained this information from an old issue of COMPUTE magazine.

CALL PEEK(-31974,A,B)::PRINT A*256+B-2455 should work.

The 2455 refers to the number of bytes of VDP RAM used by the machine when XBASIC is attached (i.e. 16,384 bytes - 2455 = 13,929). The addresses are in the CPU Scratch Pad RAM.

Someone was interested in the fact that you could use a number, by itself, in the IF statement. In case no one else comments on this, I thought I would tell what I know. When the condition following the IF is evaluated, a -1 is returned for a true condition and a 0 is returned for a false one. If a single variable follows the IF, its value is examined. If the value is -1 (or not 0) then the instruction after the THEN is followed. If the value is 0 the instruction after the ELSE is followed (or if there is no ELSE, the instruc-tion in the next program line is followed).

Another useful thing to know if you are interested in compacting your BASIC or XBASIC program is that relational expressions can be incorporated into equations. An example that I have used this for is the rounding of numbers. You've probably heard of the rounding method in which you shift the decimal point to the right of the digit at which you wish rounding to occur, then add .5 to the number and take the INT of that number, and then shift back the decimal point to its correct position. This method is fine except for the fact that numbers ending in exactly .5 (after the decimal point is shifted) are always rounded up. If you had a large number of such values, you would

bias later calculations based on those numbers. I have incorporated relational expressions into an equation that permits the computer to round up values ending in .5, if they have an odd digit to the left of the .5, and to round down such values, if they have an even digit to the left of the .5. The same equation permits the normal rounding up or down of other numbers as well. All of this is achieved in a single program line. The line looks like this:

```
IR=IR-(R-IR>.5)+(R-IR=.5)*(IR/2-INT(IR/2)0)
```

The R variable is assigned the value of your number after the decimal point has been shifted, the IR variable is previously assigned the INT of R. In this equation, IR is assigned the value of the number after it has been rounded off, but the decimal point has still to be shifted back to its original position.

There are probably plenty of other uses for such a technique, which might save you a line or two of BASIC code here and there. If anyone has a simpler method of rounding numbers, I'd like to hear about it. This method may not be the best and I'd appreciate any good suggestions. Now to send this before it becomes far too long.