COMMANDING CODES

Nobody could call the Commodore's version of the BASIC language advanced, but — like Commodore machines themselves — it has a sturdy simplicity and logic. There may be glaring deficiencies in, for example, its commands but these can usually be compensated for, and its screen editor is still one of the best available.

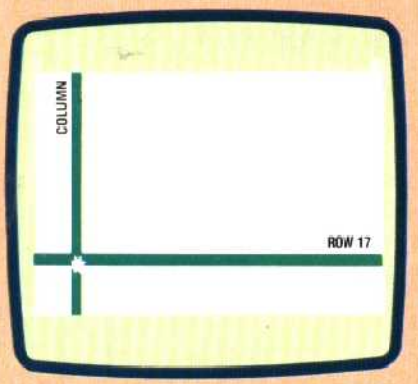
All CBM machines allow long variable names, but only the first two characters of a name are scanned by the interpreter, so variables such as FUJIYAMA and FUTILITY are allowable, but equivalent. Thus the output of this fragment:

```
100 FUJIYAMA=17:FUTILITY=2*FUJIYAMA
200 PRINT FUJIYAMA,FUTILITY
```

is:

```
 34        34
```

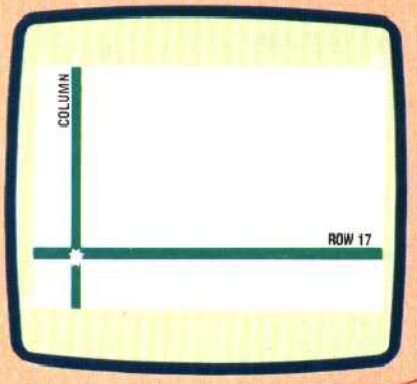
Positioning The Cursor



The ability to include cursor commands in a string quantity can make graphic design on the Commodores easy — especially when coupled with the powerful screen editing command

```
100 PRINT AT(17,4)"*"
```

If Commodore BASIC supported the PRINT AT command, then positioning the cursor would be simple.



Since it doesn't permit this, we must use the programmable cursor feature:

Initialise POSITIONS with a HOME and 24 CRSR DOWNs. Then put the row and column parameters in this expression.

```
50 POSITION$="
100 PRINT LEFT$(POSITION$,17)TAB(4-1)"*"
```

If you have a lot of screen formatting to do, it might be worth putting the cursor positioning command into a subroutine, and then initialising the variables ROW and COLUMN with the screen position required before calling the subroutine

```
50 POSITION$="
100 ROW=17:COLUMN=4:GOSUB 1000:PRINT "*"
500 END
1000 PRINT LEFT$(POSITION$,ROW)TAB(COLUMN-1); :RETURN
```

LIZ DIXON

This applies to all variable types: floating point (e.g. NUMBER), integer (e.g. NUMBER%), string (e.g. NUMBER$), and array (e.g. NUMBER$(62,47)). The variable types are themselves conventional, but on the Vic-20 integer variables are unusable because the machine does not support integer arithmetic; the integer type was retained simply for compatibility with other CBM machines that support integer arithmetic.

An annoying consequence of the variable name rules is that a valid-looking name may be illegal because its first two letters make a reserved word — START, for example, is equivalent to ST as a variable name, and ST is a reserved word.

Array variables may have up to 225 dimensions and are limited in extent only by the amount of RAM available. The first element of any array is element(0), so DIM EX(6) creates an array of seven elements: EX(0), EX(1), EX(2). . .EX(6). The DIM statement is actually unnecessary here, for if the interpreter encounters a single-dimension array variable for which no DIM statement has been executed, a default dimension of 10 is assumed; if the subscript of such an array is greater than 10, then a BAD SUBSCRIPT error will result. This is a convenient facility but it does not encourage good programming practice: the interpreter has to rearrange memory whenever it encounters a DIM statement (or the first reference to an unDIMmed array), so all arrays should be DIMmed at the same time at the start of a program, before any simple variables are employed. No great calamity will occur if this isn't done, but speed of execution will suffer slightly.

Because of the way most BASIC interpreters work, program execution can be speeded up by initialising the most commonly used program variables in their order of importance; this can be done with assignment statements or with the DIM statement. A line such as:

```
10 DIM A$(10,24),K,L,SCORE
```

will have no obvious effect but is a quick way of placing the variables K,L, and SCORE high in the symbol table, thus making them more readily accessible to the interpreter, therefore increasing program execution speed.

A look through the list of BASIC keywords in a CBM user manual (of which more later) reveals a few omissions from, and additions to, the full Microsoft set. The most important omission is probably INKEY$ and the most significant additions are TIMES and STATUS.

INKEY$, the keyboard-scanning function, is replaced by the statement GET. Like INKEY$, it causes the first character of the keyboard buffer to