be scanned and returns its ASCII value. GET is most often used in statements such as this:

```
150 GET GT$:IF GT$="" THEN 150
```

which should have the effect of halting program execution until a key is pressed, in which case GT$ will contain the character corresponding to the keypress. This may not always be the case because GET scans the keyboard buffer rather than the keyboard itself, so if the buffer contains some characters when GET is performed then program execution will not wait for the user's keypress. That can be demonstrated with this program:

```
50 FOR K=1 to 100:PRINT K:NEXT K
60 PRINT "PRESS ANY KEY"
150 GET GT$:IF GT$="" THEN 150
200 PRINT "YOU PRESSED KEY ";GT$
```

If you run this program you will see the numbers from 1 to 100 appear on the screen, followed by the input message, and then nothing until you press a key. If, however, you press a key while the numbers are being printed, then that keypress will go into the buffer and be found there by the GET statement, so that there will be no pause in program execution. This can be annoying — possibly disastrous in games, for example, where much frantic key-pressing occurs. The answer is to reset the buffer just before the GET statement is executed, by inserting in the program:

```
149 POKE KBPTR,0
```

where KBPTR is the address of the keyboard queue counter (198 in the Commodore 64).

GET does not normally generate a blinking cursor on the screen, but POKE FLASH,0 — where FLASH is the address of the cursor blink enable flag (204 on the Commodore 64) — will supply one.

TIME$ (usually abbreviated to TI$) is the system clock; at power-on it is initialised to '000000' and thereafter indicates the time in hours, minutes and seconds up to '235959' — 23 hours 59 minutes and 59 seconds since initialisation — when it resets to '000000'. It is available to the user like any other variable, and can be set to any legal time such as: TI$="000000" or TI$="084503".

Associated with TI$ is TI, its numeric counterpart. TI contains the time since initialisation in 60ths of a second (called 'jiffys'), so it ranges in value from 0 to 5183999 (60*60*60*24-1). TI is dependent upon TI$ and cannot itself be initialised — you initialise only TI$.

STATUS (abbreviated to ST) is a system-defined variable. When an error is detected at an input/output device the value of ST will be a number indicating the type of error encountered. Typically this would be written as:

```
330 IF ST > 0 THEN GOSUB 30000: REM I/O FILE
    HANDLING ERROR
:
30000 REM ERROR MESSAGES
30100 IF ST=16 PRINT "UNRECOVERABLE READ
      ERROR"
```

CMD is a useful member of the Commodore instruction set. Its effect is to divert output from the screen to a selected output channel. This has many useful applications such as:

```
OPEN 4,4:CMD 4:LIST
```

This LISTs the current program to the printer rather than on the screen; when the LIST is complete you should execute:

```
PRINT#4:CLOSE 4
```

CMD can be used in a program to copy the screen to the printer. Suppose GOSUB 3000 in your program causes a message or some data to be PRINTed on the screen (rather than POKEd into screen memory). In order to copy that display to the printer, type in:

```
OPEN4,4:CMD 4:GOSUB3000:PRINT#4:CLOSE4
```

Although a question mark (?) is acceptable as an abbreviation of the keyword PRINT, you cannot abbreviate the keyword PRINT# to ?# — you must use pR (p followed by shifted r) to do this.

The Commodore keyword abbreviations are as old as the Commodore machines, but to listen to Spectrum owners you would think that Sinclair invented the idea. Almost all keywords can be abbreviated to their initial letter followed by the shifted second letter. When two or more keywords have the same initial two letters then the abbreviation will be the first two lettters and shifted third: READ, RESTORE and RETURN, for example, are abbreviated to rE, reS and reT.

The screen editor, and the robust, user-friendly operating system that underlies it, is the most significant single feature of the Commodore machines. It has been in use since the PET was released and is still the best micro screen editor available. To edit a program line, for example, LIST the line, move the cursor directly to any point in the line, edit the text, and press return. It doesn't matter where the text is on the screen or where the cursor is in the line — when you press return the text on the screen line containing the cursor enters the system as if you'd just typed it.

One subtlety of this editing system is in the copying facility it permits. Suppose you have to enter these two lines:

```
100 IF INT(NUMBER/INDEX—RATE)=5 THEN 3000
200 IF INT (NUMBER/INDEX—RATE)=7 THEN 3800
```

You need type in only the first line and press return; then, with that text on the screen, move the cursor up, change the line number from 100 to 200, change 5 to 7, change 3000 to 3800, and press return again. Line 100 remains untouched in memory, and its edited text on the screen becomes line 200. This process may be repeated as often and as much as you wish. That's only one of the fascinating tricks you can play with the editor, but it demonstrates its ease and directness of use. After the Commodore editor, using other systems feels like playing the piano with mittens on.

SUPER SWINDLER

Jerry Schneider swindled $1,000,000 worth of equipment from the Los Angeles Pacific Telephone and Telegraph company in 1971. Salvaging manuals and equipment from PT & T's waste bins, he posed as a reporter and acquired access codes for PT & T's IBM 360 computer. Discreet orders were made and goods re-sold by Schneider. He served 40 days in prison and then took up as a computer security consultant.