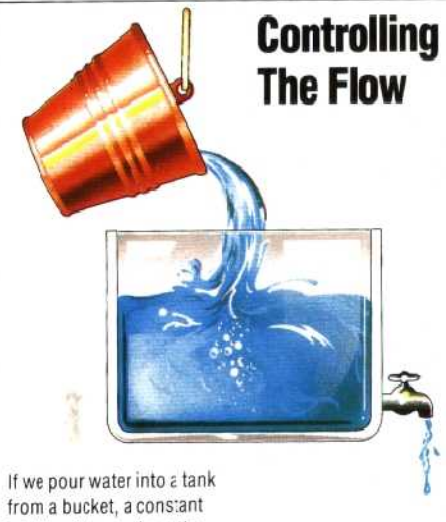
Controlling The Flow



If we pour water into a tank from a bucket, a constant trickle will come from the outlet pipe. In just the same way, the computer 'pours' data into the buffer, which releases it at the much slower speed at which the printer, for example, works

this happening, but with certain disk operating systems, or with certain types of applications software (involving a lot of processing of information) there may be a slight delay between a key being pressed and its appearance on the screen. A few operating systems allow the type ahead buffer to be turned on or off, or even allow the size of the buffer to be altered by the user.

The precise way in which the software is organised to handle buffers varies depending on what the buffer is supposed to do, but generally it will be necessary to set aside a few bytes to be used as counters and flags. It will be necessary to know how many bytes have been read out from a full buffer before more are written in, otherwise important data may be destroyed before it has been used.

Hardware Memory Buffers

Readers with printers may have noticed how slow they seem to be, especially when printing out a long program listing or document. Most computer operating systems are not able to do anything else while the printer is in use, so if it takes a long time to finish printing, the user will just have to sit at the screen waiting for the printout to end. Many manufacturers now offer add-on print buffers, usually in the form of a box that connects between the computer and the printer. These boxes effectively speed up the printer from the computer's point of view. Although the printer does not actually print any faster, the box contains extra, dedicated memory (sometimes as much as 16 Kbytes) with its own built-in software. To the computer it appears exactly like a faster printer. When the computer has to print a file, it sends bytes out to the printer until it receives a 'busy' signal, meaning that the printer cannot accept any more bytes. The computer then has to wait until the 'busy' line goes 'false', indicating that the printer is once again able to accept data. Even though printers usually have a small memory buffer built in, this is often not more than two Kbytes and it will not allow the computer to send more data until this buffer is empty. The add-on hardware memory buffers contain more memory, so they can accept much more data before sending a 'busy' signal to the computer. If the buffer is big enough, it may be able to hold all the data to be printed in one go, so the computer will be able to get on with other tasks while the buffer sends the data at a slower rate to the printer.

Memory is often used rather like a large trunk or pigeon-hole rack to store programs and data, but it may instead be organised into stacks or buffers. Stacks are 'Last In First Out' or LIFO structures, whereas buffers are 'First In First Out' of FIFO structures. The analogy often made for stacks is that of the pile of plates supported by a spring occasionally seen at self-service counters. Plates are piled on the stack and the last one placed there will be the first one removed. Like buffers, stacks are also temporary memory areas and differ from buffers only in the order in which the data is input and retrieved. Stacks are used 'internally' in high level languages (in BASIC interpreters, for example) when information needs to be stored temporarily and recalled later. Consider this BASIC program fragment:

```
FOR X=1 TO 10
PRINT "X=";X
FOR Y=1 TO 10
GOSUB SCAN
NEXT Y
PRINT "CS=";CS
NEXT X
```

This is an example of nested FOR. . .NEXT loops. When the BASIC interpreter gets to the second FOR statement, it needs to remember which variable is used by the previous FOR(X in this case), and so it 'pushes' the information about the first FOR onto a stack. When the inner loop has been completed, it 'pops' the information from the top of the stack and knows that the current FOR uses variable X. Since FOR. . .NEXT loops can be nested as deeply as required, it may need to push information for several FORs onto the stack. When it pops the information from the stack, it clearly needs to have the information in the inverse order from that in which it was pushed.

Buffers, on the other hand, organise memory so that the first information entered is the first information output. Buffers are often used in input/output routines and are used as 'interfaces' between routines or devices working in different units or at different speeds. For example, an input routine in BASIC might work in units of lines, terminated by a Carriage Return <CR>, but the interpreter may work on the lines in units of one character. Buffers usually need a 'pointer' to indicate where in the buffer the next character should be written. The pointer would be a byte or several bytes containing the address of that character. The address would be incremented after each character had been stored.

Strong Signals
The logic inside your computer works at TTL levels. Transistor-Transistor-Logic signifies a binary 1 with five volts, and 0 with zero volts. However, though devices such as the CPU are capable of producing these voltages, they can't generate enough current to drive all the other chips that might be connected to each pin. Signal buffers are therefore connected to the CPU's output lines to magnify the amount of current available. Signal buffers are themselves small chips, each typically capable of acting as a buffer for six signals