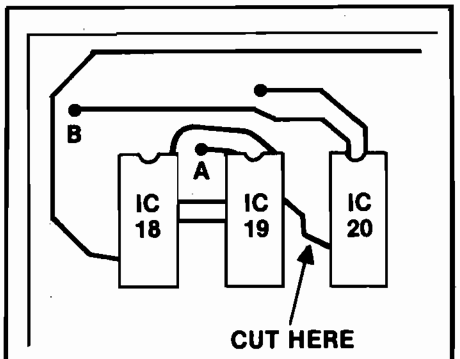
Speeding Up Your SWTP 6800

By Harry Fair

The process of attempting to decrease execution time can be approached from many different angles. Some timesavers are: streamlining programs, eliminating redundancies, avoiding goto functions, decreasing the arithmetic accuracy or even go to integer arithmetic, such as in TSC's Micro BASIC. You can use a compiler instead of an interpreter for higher level languages; however, you may find the time needed for compiling and saving the compiled code outweighs the advantages on shorter programs. You can also attack the root of the problem and increase the hardware speed.

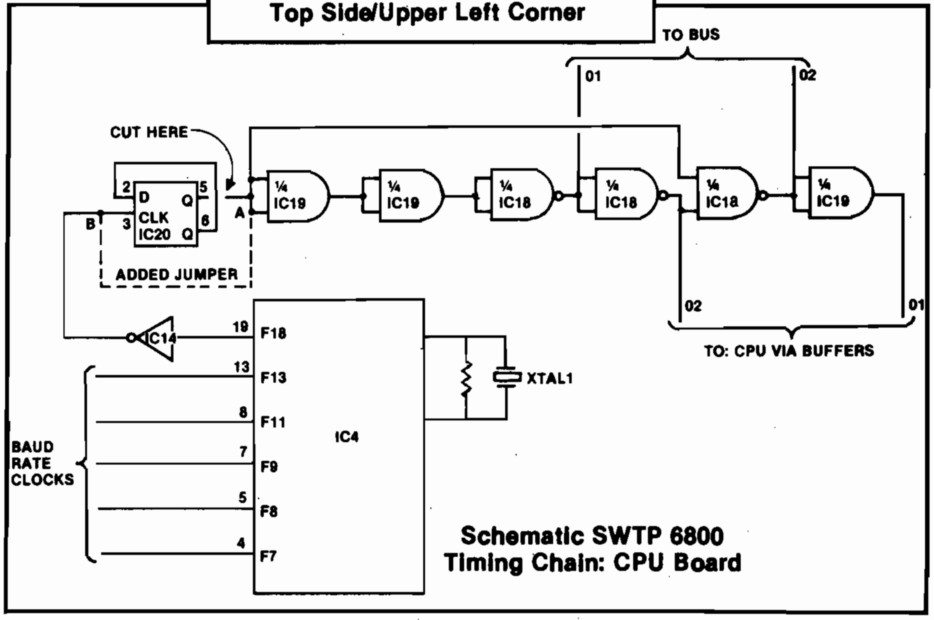
When I first began writing long statistical programs, I became painfully aware of the need for greater speed. Like all good programmers, I began to simplify and eliminate. I reduced the programs down to their very essence. I ran benchmark programs to find the fastest forms of program writing. I even declared all my variables in the start so they would be ordered in the table; but, alas, one program still took over 16 hours to run.



CPU Board Diagram
Top Side/Upper Left Corner

I began to look at other methods. A simpler form of BASIC with only integer arithmetic was out (I need the accuracy and decimals). A compiler seemed logical, but they were expensive and somewhat difficult to use. Finally there was the hardware with all those gates registers and counters.

I chose the latter. It wasn't long before I discovered some interesting things. Most of the 6800 chips sold since early 1977 are capable of running at 2.0 MHz and the Xtal on the CPU board runs at 1.8 MHz, not at .9 MHz. You only need to connect the faster clock to your CPU chip and everything the CPU does will only take half as long.



Schematic SWTP 6800
Timing Chain: CPU Board

The main clock circuits were not difficult either. IC4 (see schematic) is the main clock controlled by Xtal 1. It generates the baud rate frequencies for the PIAs and ACIAs. It also generates the 1.8 MHz clock at pin 19 which goes to a ÷ 2 counter in ½ of IC20. This in turn goes to IC18 and 19 where it is split into 01 and 02 clocks. These last two clocks drive the CPU, the PIAs and ACIAs. By bypassing the ÷ 2 counter in IC20, we can double the two main clocks, 01 and 02.

There are two ways of doing this depending on whether or not you used sockets for your ICs. If you did, pull IC20 out of its socket and bend pin 5 out a little. Reinstall it so pin 5 is *not* in the socket or touching anything. If you don't have IC20 in a socket, cut the trace between IC19 and 20 (see diagram). It's the only one running between the two ICs. Next, insert a jumper between hole A and hole B (see diagram).

Now that the CPU is capable of running twice as fast, examine some of the possibilities of what can be done with it. The most obvious is that programs will be executed in approximately one half the time. This is dependent on the extent of the I/O routines because the interfaces are still running at the same rate.

One of the more intriguing situations that has arisen out of this increase in clock rate to the CPU and interface chips (02) is that you can now record and play tapes through a standard cassette interface (Percom or SWTP) at higher baud rates. Since the CPU is executing MIKBUG routines twice as fast, it can generate and accept data in the MIKBUG format at rates up to 1200 baud. This higher rate works equally well for saving and loading BASIC programs and data.

Before making this modification, I suggest running one of the memory diagnostics such as CDAT-2 and timing how long it takes. After the modification is done and everything has been double checked, run the program again and retime it. The results should prove quite pleasing. If for some reason the modification was done incorrectly or you have one of the early chips that will not support the higher rate, attempting to rerun the program will prove futile.

The only problem that has cropped up so far has been with my BFD-68 disk system. It seems that every once in a while I get a disk error when I'm copying programs from one disk to another. The people at Smoke Signal Broadcasting tell me this is because of the software timing loops in the system. They are aware of the problem and are working on a solution. It happens so infrequently though it does not seem to be of much consequence.

The other thing to be aware of is, if program loops are used to interface with the real world, these loops will have to be lengthened to compensate for the faster execution time. Except for these two things, I have experienced no problems or lost bits in over a thousand hours of use. I have noticed, however, an increase in reliability with my tape interface even at the higher baud rates.☐