True Confessions:

How I Relate to KIM

Yogesh M Gupta
118 E Main St
New Concord OH 43762

I recently purchased a KIM-1 microcomputer card from MOS Technology (See "KIM-O-Sabee?" in the April BYTE, page 14 and "A Date With KIM" in the May issue, page 8). In my opinion, KIM-1 offers one of the best bargains to a computer experimenter for the price ($245 for the card + $4.50 for shipping and handling). However, the hobbyist may be faced with a few problems, as I was. The intent of this article is to solve some of these problems.

Clock Stretch and Random Access Memories

The cheapest random access memory in experimenters' markets today is the standard 2102 static memory which averages approximately 0.15¢ per bit. During a write cycle, the inexpensive slow versions of this device require the data to be stable for 800 ns before the trailing edge and data hold time of 100 ns after the trailing edge of the write pulse. Even if the MOS Technology 6502 processor is slowed down to 250 kHz to obtain the data stability, there is still not enough data hold time for the slow chips.

I solved this problem by implementing the circuit shown in figure 1. This circuit allows the 6502 processor to use a mixture of fast and slow 2102 memory devices in the same system. The processor cycle is main-

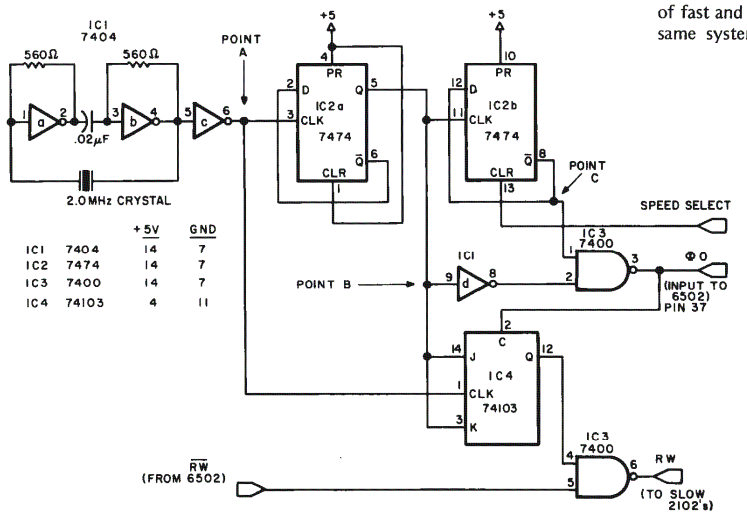

Figure 1: A circuit which creates an alternative slow clock cycle for the 6502 processor on the KIM-1 board under control of a "SPEED SELECT" line generated by slow memories. SPEED SELECT = 0 for fast cycles, SPEED SELECT = 1 for slow cycles. This circuit requires a 2.0 MHz crystal.

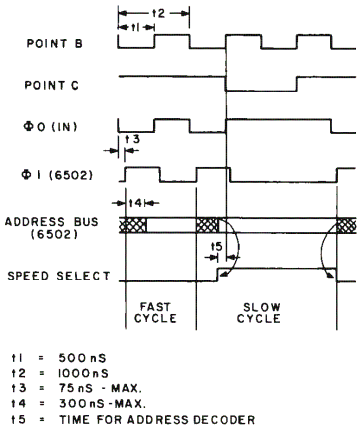

Figure 2: Method 1 SPEED SELECT Discipline. In this method, fast cycles are the rule, slow cycles are the exception. Refer to figure 1 for points B and C. Invalid data on the address bus is indicated by the crosshatched areas.

tained at 1.0 μs for the fast memory access, while for the slower 2102s, the cycle is automatically stretched to 2.0 μs.

Sometimes integrated circuits behave in ways that are not predicted by, or are overlooked by, their manufacturers. This modification of KIM-1 to enable the clock stretching function is accomplished by removing the usual KIM-1 6502 clock generation circuitry, and simply driving the ϕ_0 pin of the 6502 directly from a TTL clock source which is external to the chip. This mode of operation is not documented in the *6502 Hardware Manual* of MOS Technology, but it worked quite satisfactorily in my system. The intention of the designers of the 6502 was that the clock generation logic on the chip would be used with external components setting the frequency of the oscillator.

The SPEED SELECT signal to stretch the

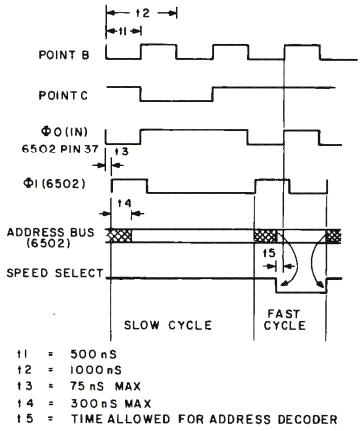

Figure 3: Method 2 SPEED SELECT Discipline. In this method, slow cycles are the rule, fast cycles are the exception. Refer to figure 1 for points B and C of the timing diagram. Invalid data on the address bus is indicated by the crosshatched areas.

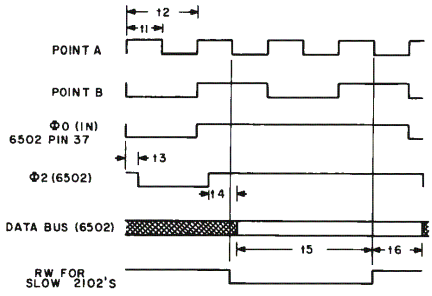

Figure 4: Write Cycle for Slow 2102 Memories. The timing requirement is that valid data be present on the bus when the RW signal to the memory changes from 0 (write state) to 1 (read state). The crosshatched areas indicate when data is invalid on the data bus.

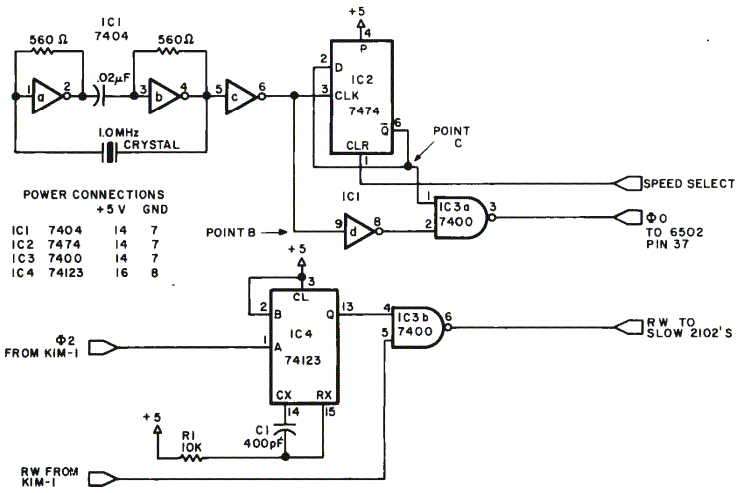

Figure 5: Alternate Slow Clock Generation Circuit. In this circuit, the original KIM-1 crystal can be used, since a digitally controlled timing cycle is replaced by the 74123 oneshot.

cycle is generated by address bus decoding logic using one of the following two methods.

Method 1: Normally the SPEED SELECT signal is kept low so the processor cycle is 1.0 μs. However, this signal goes high when the processor addresses the slow memory region causing the cycle to stretch to 2.0 μs. See figure 2 for the timing relationships.

Method 2: Normally the SPEED SELECT signal is kept high so that the processor cycle time is 2.0 μs to access slow memory. However, this signal goes low when the processor addresses the fast memory devices causing the cycle time to be only 1.0 μs. See figure 3 for the timing relationships.

The circuit shown in figure 1 will allow a data stability of 950 ns before the trailing edge and data hold time of 350 ns after the trailing edge of the Write Pulse for the slow 2102s. See figure 4 for the timing relationships.

However, the KIM-1 board comes with a 1.0 MHz crystal. Figure 5 shows an alternative circuit using a 1.0 MHz crystal. The timing relationships to control SPEED SELECT signal are the same as shown in figures 2 and 3. The RW signal for the slow memory is generated in this case by using a 74123 oneshot. The value of the RC constant for the 74123 is chosen to provide a nominal output pulse width of 1.2 μs. This allows a data stability of 1.0 μs before the trailing edge and data hold time of 300 ns after the trailing edge of the write pulse for the slow memories. Figure 6 shows the resultant timing relationships. It should be noted that the output pulse width of the 74123 can only tolerate a ± 16.66% variation, and still permit successful operation of the 2102 memory devices. This tolerance may require selection of precision parts for the external resistor and capacitor of the oneshot.

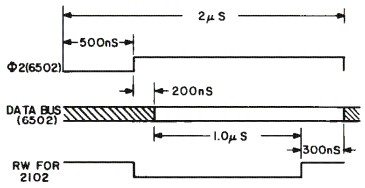

Figure 6: Write Cycle for Slow 2102 Memories using the circuit of figure 5. The output pulse width of RW is adjusted to 1.2 μs nominally. (Check the results on your scope even if you use other than precision parts of the values shown for R1 and C1 in figure 5.)

Bus Expansion

The 6502 bus is only capable of driving one standard TTL load. If more drive capability is needed, the tristate drivers such as the 8T97 or DM8833 parts may be used.

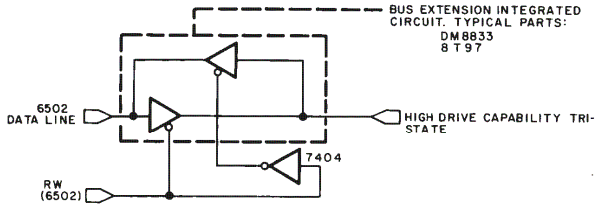

Figure 7: Use of a Bus Extension Integrated Circuit. In order to tie in extra memory or peripherals, a bus extension is required. The typical logic diagram of a simple attempt which will not always work is shown here. (Conflicts can arise.)

However, you must be very careful when using an extended data bus. If you enable the drivers by RW signal as shown in figure 7, then during read mode, the drivers for the existing KIM-1 memory (eg: 74125s) can be turned on simultaneously. The low level output current of 74125s is only 16 mA and is not sufficient to pull down a turned on 8T97 type driver to logic 0 level. Therefore, during read mode the bus extension tristate drivers should be turned off when the existing on board KIM-1 memory (RAM, 6530-002 and 6530-003) is being accessed as shown in figure 8. In actual implementation the DECODE ENABLE signal may be the same as the one needed on the application connector of the KIM board (when more than 8 K memory is needed).

Interrupt Prioritizing Logic

The *KIM-1 Hardware Manual* (Section 2.3.3) describes a few approaches to implement interrupt priority logic; but I found them either inefficient (software time) or expensive (use of ROM). The circuit shown in figure 9 provides a cost effective compromise. The interrupts from the peripheral devices are latched in by the $\phi 2$ signal. This inhibits the priority encoder from generating a false vector (if the interrupts from the peripherals are changing while the 6502 is fetching the vector). In response to IRQ, the 6502 fetches the vector from hexadecimal locations FFFE and FFFF. During these fetch cycles, the 6530-002 is disabled by letting the decode enable signal go high on the application connector. Therefore, the vector generated by this circuit is fetched by the 6502 instead, and the program goes to one of the locations from 0200 to 021C. This segment of memory serves as a vector table with pointers to the individual interrupt service routines as follows:

```
0200        JMP VEC0
0204        JMP VEC1
0208        JMP VEC2
020C        JMP VEC3
0210        JMP VEC4
0214        JMP VEC5
0218        JMP VEC6
021C        JMP VEC7
```

The actual service routines will reside in locations VEC0 through VEC7 for the respective interrupts. It should be noted that each vector in the table requires 4 locations. (Only 3 locations are needed for a jump but

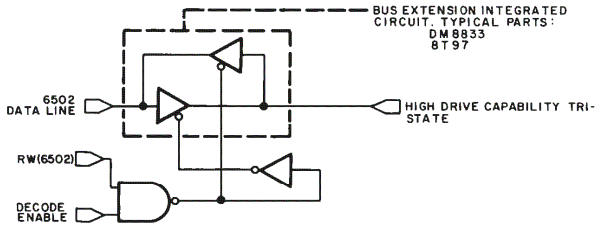

Figure 8: Adding a gate to the bus extension control resolves a potential conflict through the use of a decode enable signal which is high if external memory is referenced, low if memory on the KIM-1 board is referenced.

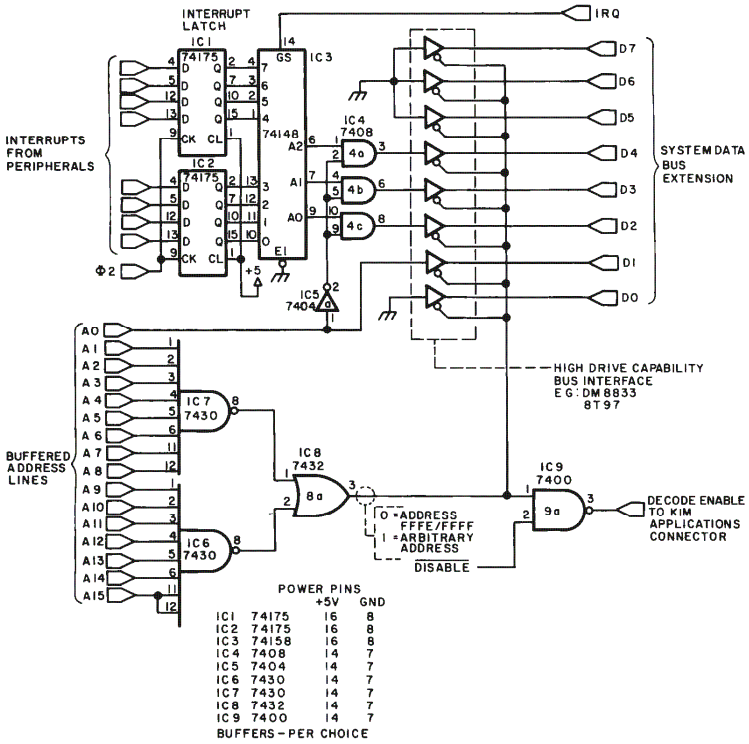

Figure 9: By disabling normal address decode through the DECODE ENABLE pin for the KIM-1 applications connector, an alternate source of the interrupt vector at locations FFFF and FFFE can be created which accomplishes interrupt prioritizing functions.

the extra location is a requirement of a simple hardware design.) JMP instruction takes only three locations, so your software might use the fourth location to save the accumulator, eg:

0200	PHA
0201	JMP
0202	VEC0 (LOW)
0203	VEC0 (HIGH)

This architecture will re-map the 1 K resident RAM on the KIM board as follows:

0000 through 00FF	Page 0
0100 through 01FF	Stack
0200 through 021F	Vector Table
0220 through 03FF	Applications

The disable signal in figure 9 will deselect existing KIM-1 memory when low. This is implemented for memory expansion as described earlier. However, if memory expansion is not desired the signal may be fixed to a logic 1 level.

Halt?

Another problem one faces is how to debug the software when the processor does not have a HLT instruction. You can single step the program instructions on KIM-1, but this feature does not help the programs which involve multiple levels of loops or critical peripheral timing controls. The obvious solution is to use the BRK (software interrupt) instruction. However, this would require software overhead in every interrupt service routine to determine whether it was a hardware or a software interrupt. On the KIM-1 system, I found the sequence JSR 05 1C (Jump to subroutine at location 1C05) more useful for this purpose instead. The execution of JSR causes the program to jump to an input monitor loop and display of the address (PC + 2) on the KIM board. PC is the location where the JSR was executed.■