Running the 2 MHz S.D. Systems' ExpandoRAM at 4 MHz

by W. Howard Adams

The S.D. Systems' ExpandoRAM (the original 2 MHz version) is a quality product; the design examplifies that dynamic RAM can be of real benefit to a computer system. This design draws less current than any conventional static RAM design. The ExpandoRAM is 64K dense, eliminating the redundant, power-hungry TTL buffering and decoding circuitry necessary in multi-board memory systems. The board runs at 2 MHz with no wait states. It is clear that a great deal of forethought was put into the design since 8080, 8085 and Z80 processor timing differences have all been accounted for. S.D. Systems did a fine job in producing a super reliable 2 MHz card!

However, S.D. Systems could have gone further; the design has the potential to run at 4 MHz, with an added wait state. The purpose of this article is to save replacement costs of new memory by allowing 4 MHz operation through a simple modification.

The first and obvious requirement is to add a wait state to memory references. Refer to Figure 1 for a suggested implementation of a single wait state circuit. With very little time and effort it can be wire-wrapped in a kluge area, if a wait state circuit does not already exist in the system.

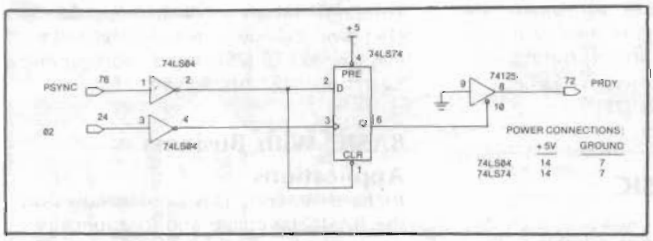


Figure 1
Adding One Wait State
(Every Memory Reference).

The second requirement is that the system's Z80 processor card must generate RFSH (refresh, active low) onto bus pin 66 which is used by the RAM board for refresh. Assure that on the ExpandoRAM card, E11 to E12 is jumpered and E10 to E12 is open. The ExpandoRAM's refreshing is now under the complete control of the Z80's timing.

W. Howard Adams, Eagles Computer Works, P.O. Box 22664, Denver, Co 80224

Before presenting the last requirement, some discussion of the board's only shortcoming is in order so that the modification will be better understood. Refer to the timing diagram in Figure 2 of this article, and to the schematic in the ExpandoRAM manual.

The problem is the generation of memory REFRESH CYCLEs which last much too long and run into the next M-cycle timing, completely obliterating it! As a fix was being sought, it was interesting to note how close to 4 MHz the board can work. While running a memory test there really were very few errors, and the addresses at which errors did occur were randomly located. This is a classic symptom of refresh interference!

The timing diagram of Figure 2 has one wait state in it, and is of an M1 cycle. Its inherent transparent REFRESH CYCLE occurs during the T4 state. Notice that the M1 and RFSH bus signals are for all intents and purposes the same signal, only the definitions of their active states are complemented. On the RAM board schematic observe that the RFSH signal eventually triggers one-shot U5-10, and a REFRESH CYCLE begins. U5-5 goes high, and U5-12 goes low throughout the REFRESH CYCLE, resulting in four occurances necessary to successfully refresh the RAMs.

First, U5-12 going low blocks the TIME-OF-CAS signal at U13-1 from being seen at U13-3 during a refresh. This permits the so-called RAS only REFRESH CYCLE, which is desirable.

Secondly, the active low at U5-12 also causes U8-8, 6, 11 and 3 to all go high enabling RAS drivers outputs at U3-8, 6, 11 and 3 to go low at TIME-OF-RAS. This produces the refresh RAS at all banks simultaneously, giving the RAM array its breath of fresh air.

Thirdly, U5-5 enables the refresh addresses of U11 and U15 to the RAM array by raising Select (pin 1) of the U21 and U16 multiplexers high.

Finally, after a time delay through R1 and C6 at U9-1, and onto U10-5, the active low leading edge at U10-6, defined as MEMORY CYCLE, triggers U5-1. Firing this one shot produces the TIME-OF-RAS signal, and a REFRESH CYCLE starts.

The timing diagram clearly differentiates between the M1 opcode fetch and the refresh portion. Note how long the REFRESH CYCLE waveform at U5-5 lasts past

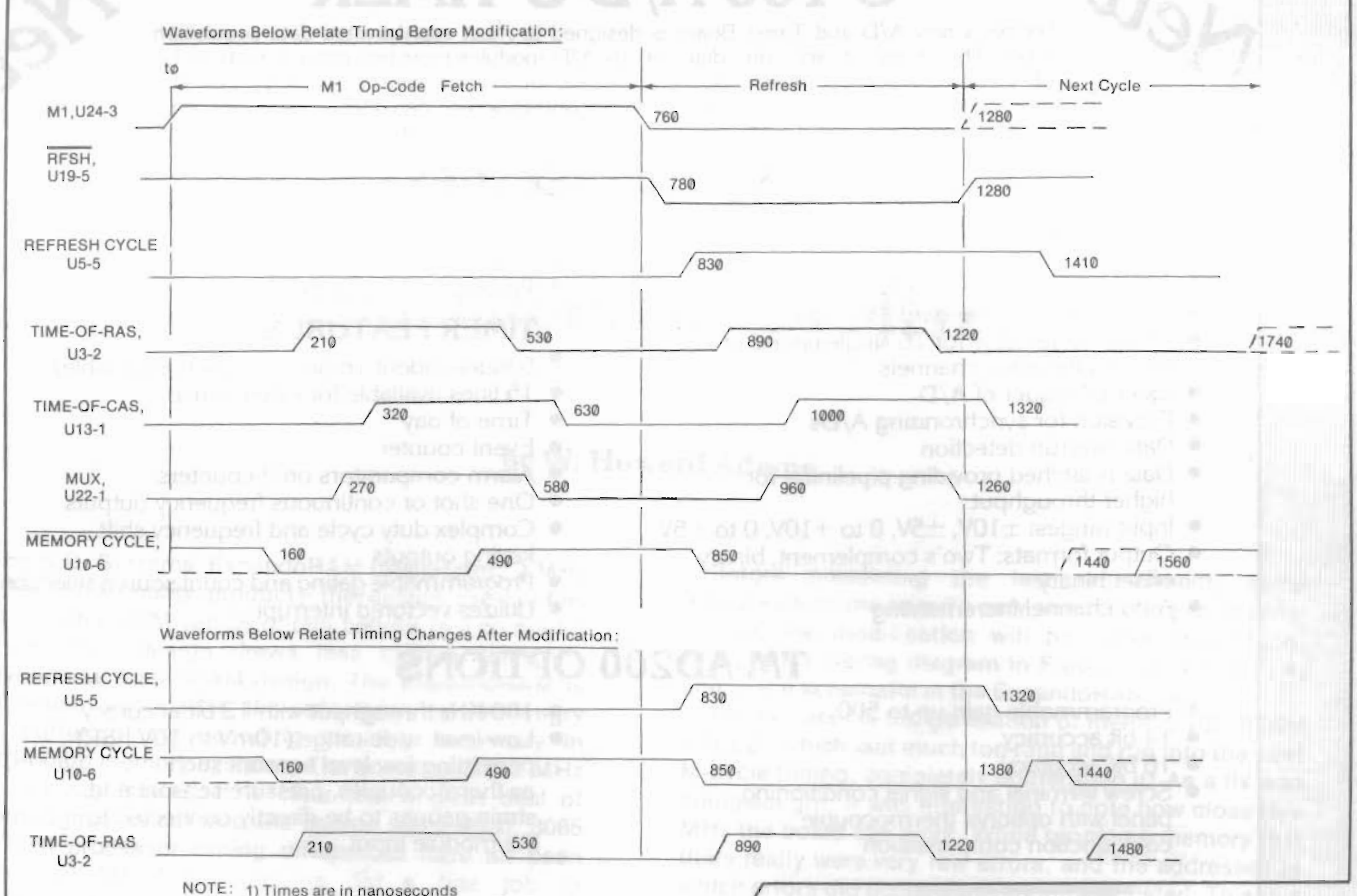


Figure 2
Timing Diagram—SDS ExpandoRAM
(Driven by 4 MHz Intersystems
Z-80 CPU Processor Board)

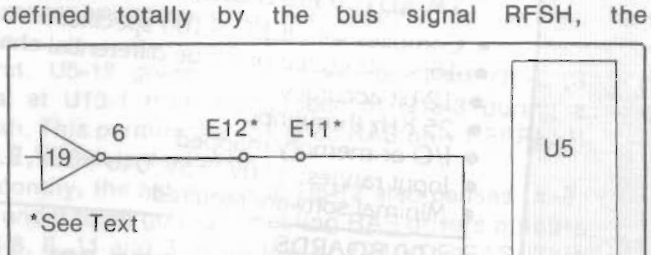
the alloted time for refresh! The culprit is: this signal lasts too long! As long as this signal is high, U10-6 is forced low, and no new cycle can begin and is therefore totally lost due to the synchronous nature of the S-100 Bus.

One quick fix could be shortening the REFRESH CYCLE timeout by reducing the value of R3 from 6.8K ohms. This is not a good method, since a one-shot does not provide consistent precise timing. There is an easier way which guarantees proper results without any component changing or trace cutting. Only one jumper is required. The proposed modification will also work should the board be returned to 2 MHz operation, so long as RFSH at pin 66 is provided.

The modification is simple. Pull U5 from its socket. Bend U5 pin 11 outward, and replace U5 back into its socket, assuring that all but pin 11 are in the socket. Run a jumper from U5-11 (which is hanging out) to U19-6. This jumper should be installed on the component side of the board, and dressed by running it under capacitors so it will not snag on neighboring pc boards during installation and removal.

A look at the timing diagram relates how the fix works. The new REFRESH CYCLE is cleared 100 nanoseconds sooner, and observe that the next M-cycle's trigger had been previously masked. The fix produces the necessary TIME-OF-RAS which had been lost. Refer to Figure 3 for the schematic changes necessary to correct the original.

The modification now keeps the REFRESH CYCLE one-shot cleared during non-refresh time. When it is time to do a refresh, the clear is removed, allowing the one-shot to fire. At the end of the REFRESH CYCLE, as defined totally by the bus signal RFSH, the



Schematic Changes Caused by Modification

1) U5-11 Removed from Logic 1 (i.e., +5V)
2) U5-11 Connected to U19-6 (or E11, or E12)
3) Only pertinent I.C.'s and pins are related that exist on the ExpandoRAM Card.

Figure 3
Modification
Schematic—SDS ExpandoRAM

modification clears the one shot efficiently and asynchronously.

The 2 MHz ExpandoRAM should now function at 4 MHz with its required wait state without sacrificing system throughput too drastically. (The particular board used to find this fix is populated with 250 nanosecond RAMs, and the only wait state needed is during the shorter accessing of the M1 cycle. So this system is hardly being held back at all!)

So go ahead and spend that memory replacement money on the new peripheral you've wanted for so long. There's still plenty of life in the 2 MHz ExpandoRAM with this new 4 MHz system! □