Z80 INTERFACING TECHNIQUES FOR DYNAMIC RAM

Application Note

INTRODUCTION

Since the introduction of second generation microprocessors, there has been a steady increase in the need for larger RAM memory for microcomputer systems. This need for larger RAM memory is due in part to the availability of higher level languages such as PL/M, PL/Z, FORTRAN, BASIC and COBOL. Until now, when faced with the need to add memory to a microcomputer system, most designers have chosen static memories such as the 2102 1Kx1 or possibly one of the new 4Kx1 static memories. However, as most mini or mainframe memory designers have learned, 16-pin dynamic memories are often the best overall choice for reliability, low power, performance, and board density. This same philosophy is true for a microcomputer system. Why then have microcomputer designers been reluctant to use dynamic memory in their system? The most important reason is that second generation microprocessors such as the 8080 and 6800 do not provide the necessary signals to easily interface dynamic memories into a microcomputer system.

Today, with the introduction of the Z80, a true third generation microprocessor, not only can a microcomputer designer increase system throughput by the use of more powerful instructions, but he can also easily interface either static or dynamic memories into the microcomputer system. This application note provides specific examples of how to interface 16-pin dynamic memories to the Z80.

OPERATION OF 16-PIN DYNAMIC MEMORIES

The 16-pin dynamic memory concept, pioneered by MOSTEK, uses a unique address multiplexing technique which allows memories as large as 16, 384 bits x 1 to be packaged in a 16-pin package. For example the MK4027 (4,096x1 dynamic MOS RAM) and the MK4116 (16,384x1 dynamic MOS RAM) both use address multiplexing to load the address bits into memory. The MK4027 needs 12 address bits to select 1 out of 4,096 locations, while the MK4116 requires 14 bits to select 1 out of 16,384. The internal memories of the MK4027 and MK4116 can be thought of as a matrix. The MK4027 matrix can be thought of as 64x64, and the MK4116 as 128x128. To select a particular location, a row and column address is supplied to the memory. For the MK4027, address bits A_0-A_5 are the row address, and bits A_6-A_{11} are the column addresses. For the MK4116, address bits A_0-A_6 are the row address, and A_7-A_{13} are the column address. The row and column addresses are strobed into the memory by two negative going clocks called Row Address Strobe ($\overline{RAS}$) and Column Address Strobe ($\overline{CAS}$). By the use of $\overline{RAS}$ and $\overline{CAS}$, the address bits are latched into the memory for access to the desired memory location.

Dynamic memories store their data in the form of a charge on a small capacitor. In order for the dynamic memory to retain valid data, this charge must be periodically restored. The process by which data is restored in a dynamic memory is known as refreshing. A refresh cycle is performed on a row of data each time a read or write cycle is performed on any bit within the given row. A row consists of 64 locations for the MK4027 and 128 locations for the MK4116. The refresh period for the MK4027 and the MK4116 is 2ms which means that the memory will retain a row of data for 2ms without a refresh. Therefore, to refresh all rows within 2ms, a refresh cycle must be executed every 32μs (2ms÷64) for the MK4027 and 16μs (2ms÷128) for the MK4116.

To ensure that every row within a given memory is refreshed within the specified time, a refresh row address counter must be implemented either in external hardware or as an internal CPU function as in the Z80. (Discussed in more detail under Z80 Refresh Control and Timing.) The refresh row address counter should be incremented each time that a refresh cycle is executed. When a refresh is performed, all RAMs in the system should be loaded with the refresh row address. For the MK4027 and the MK4116, a refresh cycle consists of loading the refresh row address on the address lines and then generating a $\overline{RAS}$ for all RAMs in the system. This is known as a $\overline{RAS}$ only refresh. The row that was addressed will be refreshed in each memory. The $\overline{RAS}$ only refresh prevents a conflict between the outputs of all the RAMs by disabling the output on the MK4116, and maintaining the output state from the previous memory cycle on the MK4027.

Z80 TIMING AND MEMORY CONTROL SIGNALS

The Z80 was designed to make the job of interfacing

to dynamic memories easier. One of the reasons the Z80 makes dynamic memory interfacing easier is because of the number of memory control signals that are available to the designer. The Z80 control signals associated with memory operations are:

MEMORY REQUEST ($\overline{MREQ}$) - Memory request signal indicating that the address bus holds a valid memory address for a memory read, memory write, or memory refresh cycle.

READ ($\overline{RD}$) - Read signal indicating that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

WRITE ($\overline{WR}$) - Write signal indicating that the CPU data bus hold valid data to be stored in the addressed memory or I/O device.

REFRESH ($\overline{RFSH}$) - Refresh signal indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current MREQ signal should be used to generate a refresh cycle for all dynamic memories in the system.

Figures 1a, 1b, and 1c show the timing relationships of the control signals, address bus, data bus and system clock Φ. By using these timing diagrams, a set of equations can be derived to show the worst case access times needed for dynamic memories with the Z80 operating at 2.5MHz.

The access time needed for the op code fetch cycle and the memory read cycle can be computed by equations 1 and 2.

(1) $t_{ACCESS\ OP\ CODE} = 3(t_c/2) - t_{DL\overline{\Phi}(MR)} - t_{S\Phi(D)}$

where: t_c = Clock period

$t_{DL\overline{\Phi}(MR)}$ = $\overline{MREQ}$ delay from falling edge of clock.

$t_{s\Phi(D)}$ = Data setup time to rising edge of clock during op code fetch cycle.

let: t_C = 400ns; $t_{DL\overline{\Phi}(MR)}$ = 100ns; $t_{s\Phi}$ = 50ns

then: $t_{ACCESS\ OP\ CODE}$ = 450ns

(2) $t_{ACCESS\ MEMORY\ READ} = 4(t_c/2) - t_{DL\overline{\Phi}(MR)} - t_{S\overline{\Phi}(D)}$

where: t_C = Clock period

$t_{DL\overline{\Phi}(MR)}$ = $\overline{MREQ}$ delay from falling edge of clock

$t_{S\overline{\Phi}(D)}$ = Data Setup time to falling edge of clock

let: t_C = 400ns; $t_{DL\ (MR)}$ = 100ns; $t_{S\ (D)\ \overline{\Phi}}$ = 60ns

then: $t_{ACCESS\ MEMORY\ READ}$ = 640ns

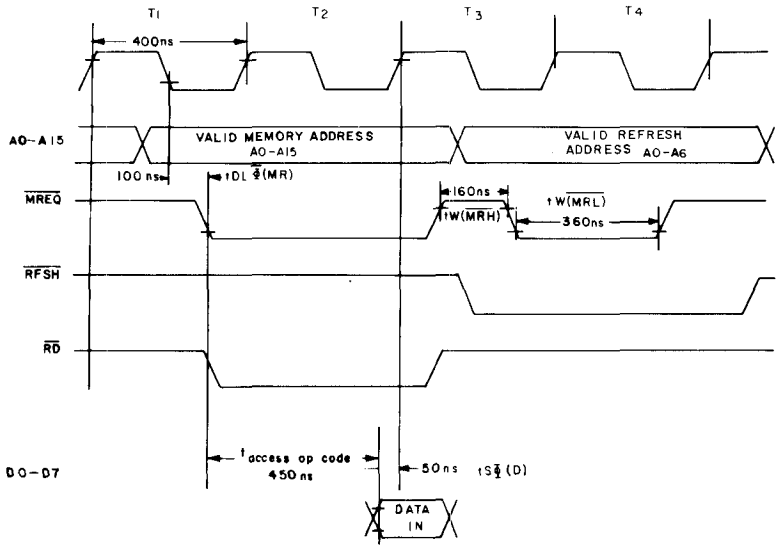
The access times computed in equations 1 and 2 are overall worst case access times required by the CPU. The overall access times must include all TTL buffer delays and the access time for the memory device. For example, a typical dynamic memory design would have the following characteristics, (see Figure 2).

The example in Figure 2 shows an overall access time of 336ns. This would more than satisfy the 450ns required for the op code fetch and the 640ns required for a memory read.

CPU $\overline{MREQ}$ buffer delay	12ns (8T97)
Memory gating and timing delays	40ns
Memory device access time	250ns (MK4027/4116-4)
Memory data bus buffer delay	17ns (8T28)
CPU data bus buffer delay	17ns (8T28)
	336ns

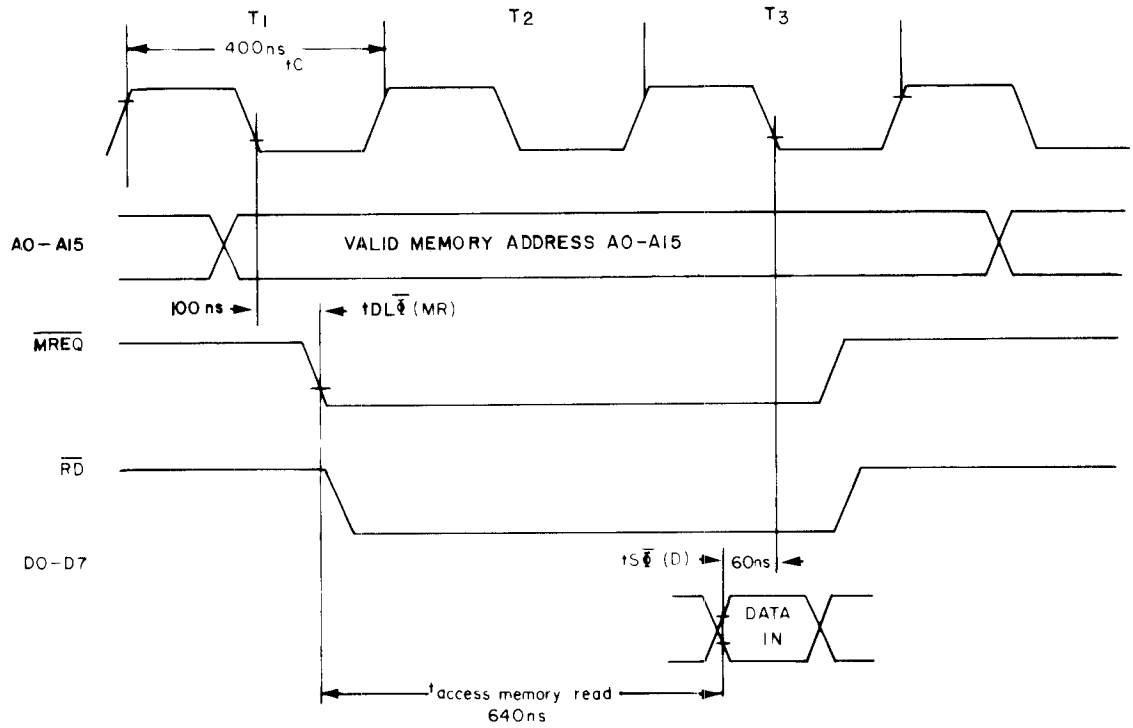
OP CODE FETCH TIMING

Figure 1a.



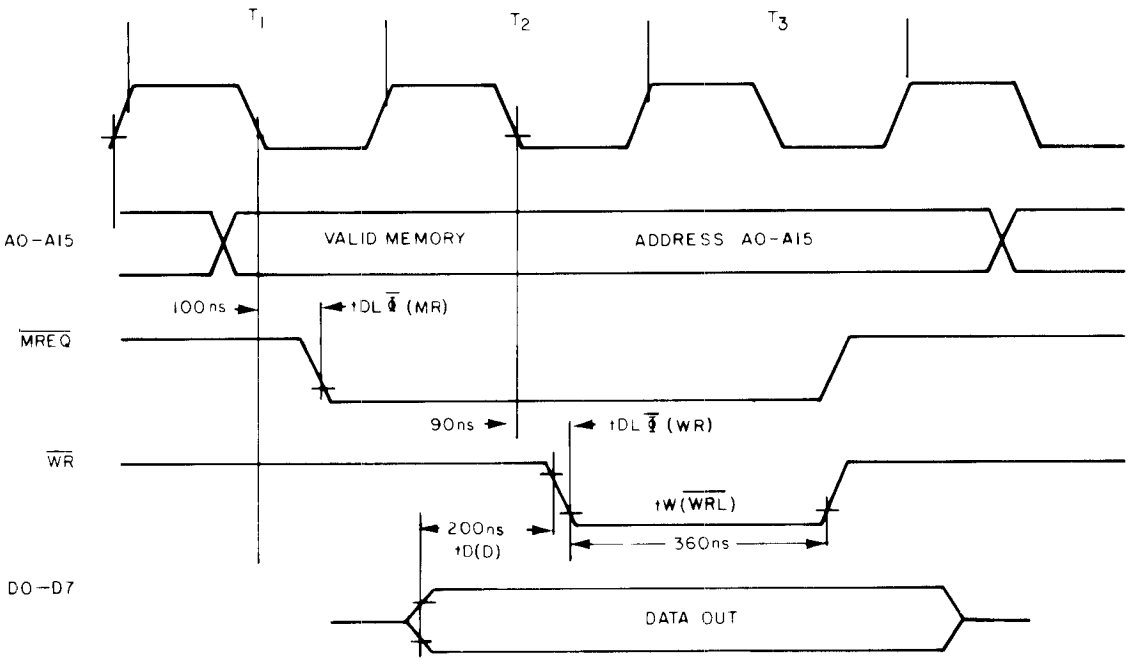
MEMORY READ TIMING

Figure 1b.



MEMORY WRITE TIMING

Figure 1c.



Z80 REFRESH CONTROL AND TIMING

One of the most important features provided by the Z80 for interfacing to dynamic memories is the execution of a refresh cycle every time an op code fetch cycle is performed. By placing the refresh cycle in the op code fetch, the Z80 does not have to allocate time in the form of "wait states" or by "stretching" the clock to perform the refresh cycle. In other words, the refresh cycle is "totally transparent" to the CPU and does not decrease the system throughput (see Figure 1a). The refresh cycle is transparent to the CPU because, once the op code has been fetched from memory during states T_1 and T_2, the memory would normally be idle during states T_3 and T_4.

Therefore, by placing the refresh in the T_3 and T_4 states of the op code fetch, no time is lost for refreshing dynamic memory. The critical timing parameters involving the Z80 and dynamic memories during the refresh cycle are: $t_{W(MRH)}$ and $t_{W(MRL)}$. The parameter known as $t_{W(MRH)}$ refers to the time that $\overline{MREQ}$ is high during the op code fetch between the fetch of the op code and the refresh cycle. This time is known as "precharge" for dynamic memories and is necessary to allow certain internal nodes of the RAM to be charged-up for another memory cycle. The equation for the minimum $t_{W(MRH)}$ time period is:

(3) $t_{W(MRH)} = t_{W(\Phi H)} + t_f - 30$

where: $t_{W(\Phi H)}$ is clock pulse width high
t_f is clock fall time

let: $t_{W(\Phi H)} = 180ns$; $t_f = 10ns$

then: $t_{W(MRH)} = 160ns$ (min)

A $t_{W(MRH)}$ of 160ns is more than adequate to meet the worst case precharge times for most dynamic RAMs. For example, the MK4027-4 and the MK4116-4 require a 120ns precharge.The other refresh cycle parameter of importance to dynamic RAMs is $t_{W(MRL)}$, (the time that $\overline{MREQ}$ is low during the refresh cycle). This time is important because $\overline{MREQ}$ is used to directly generate $\overline{RAS}$. The equation for the minimum time period is:

(4) $t_{W(MRL)} = t_c - 40$

where: t_c is the clock period

let: $t_c = 400ns$

then: $t_{W(MRL)} = 360ns$

A 360ns $t_{W(MRL)}$ exceeds the 250ns min $\overline{RAS}$ time required for the MK4027-4 and the MK4116-4.

By controlling the refresh internally with the Z80, the designer must be aware of one limitation. The limitation is that to refresh memory properly, the Z80 CPU must be able to execute op codes since the refresh cycle occurs during the op code fetch. The following conditions cause the execution of op codes to be inhibited, and will destroy the contents of dynamic memory.

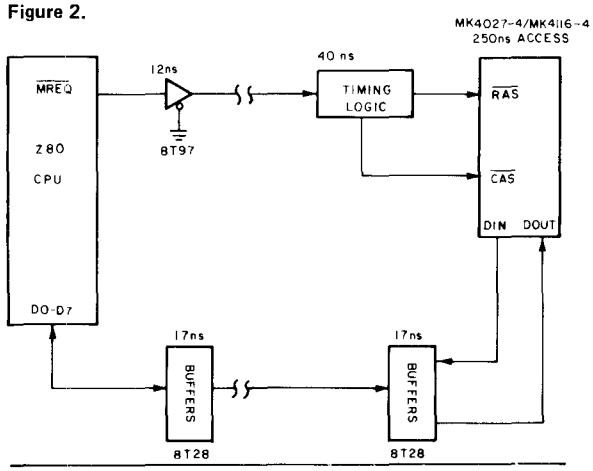
(1) Prolonged reset > 1ms
(2) Prolonged wait state operation > 1ms
(3) Prolonged bus acknowledge (DMA) > 1ms
(4) Φ clock of < 1.216 MHz for 16K RAMs
< .608 MHz for 4K RAMs

The clocks rate in number 4 are based on the Z80 continually executing the worst case instruction which is an EX (SP), HL that executes in 19 T states. Therefore, by operating the Z80 at or above these clocks frequencies, the user is ensured that the dynamic memories in the system will be refreshed properly.

Remember to refresh memory properly, the Z80 must be able to execute op codes!

DELAY FOR A TYPICAL MEMORY SYSTEM

Figure 2.



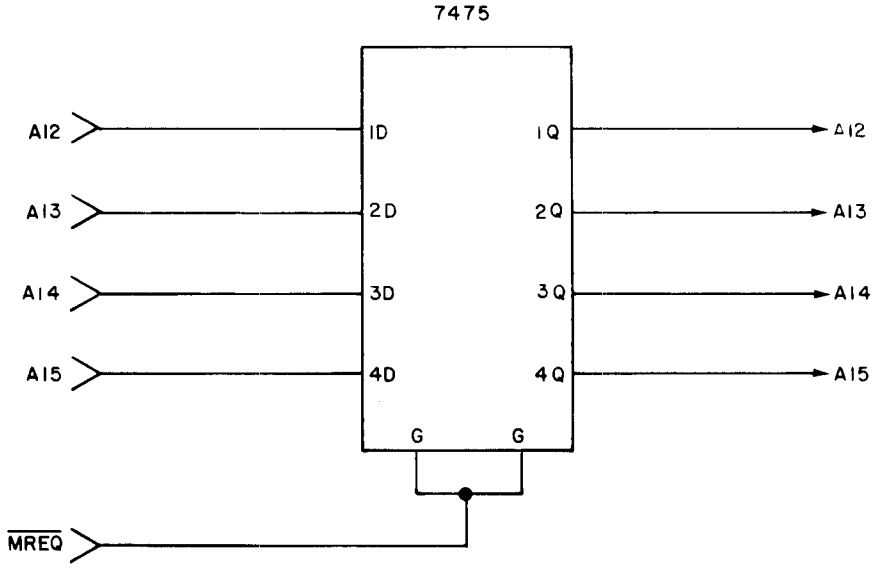
SUPPORT CIRCUITS FOR DYNAMIC MEMORY INTERFACE

Two support circuits are necessary to ensure reliable operation of dynamic memory with the Z80.

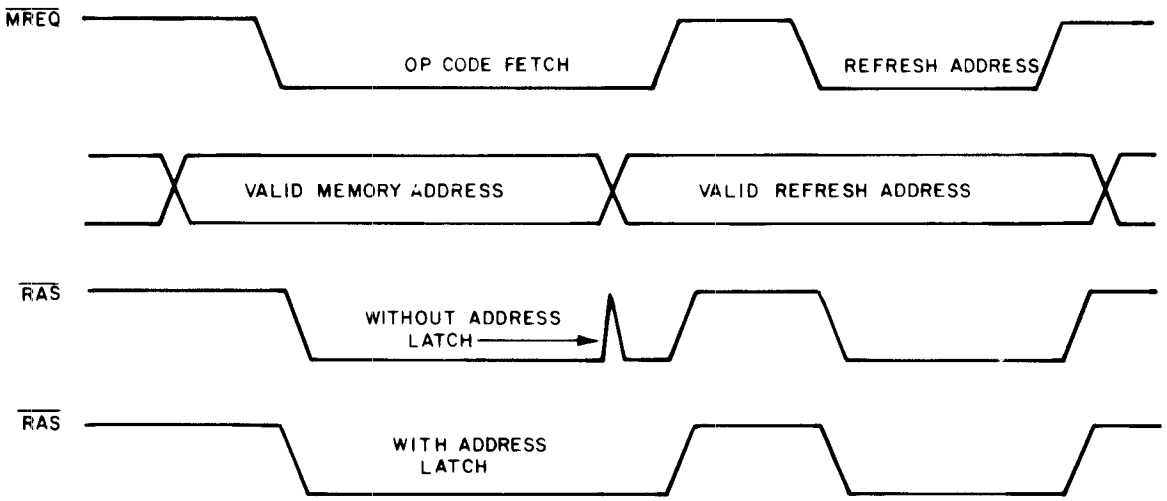
The first of these circuits is an address latch shown in Figure 3. The latch is used to hold addresses A_{12}-A_{15} while $\overline{MREQ}$ is active. This action is necessary because the Z80 does not ensure the validity of the address bus at the end of the op code fetch (see Figure 4). This action does not directly affect dynamic memories because they latch addresses internally. The problem comes from the address decoder which generates $\overline{RAS}$. If the address lines which drive the decoder are allowed to change while $\overline{MREQ}$ is low, then a "glitch" can occur on the $\overline{RAS}$ line or lines (if more than one row of RAMs are used) which may have the effect of destroying one row of data.

The second support circuit is used to generate a power on and short manual reset pulse. Recall from the discussion under Z80 Timing and Memory Con-

ADDRESS LATCH
Figure 3.



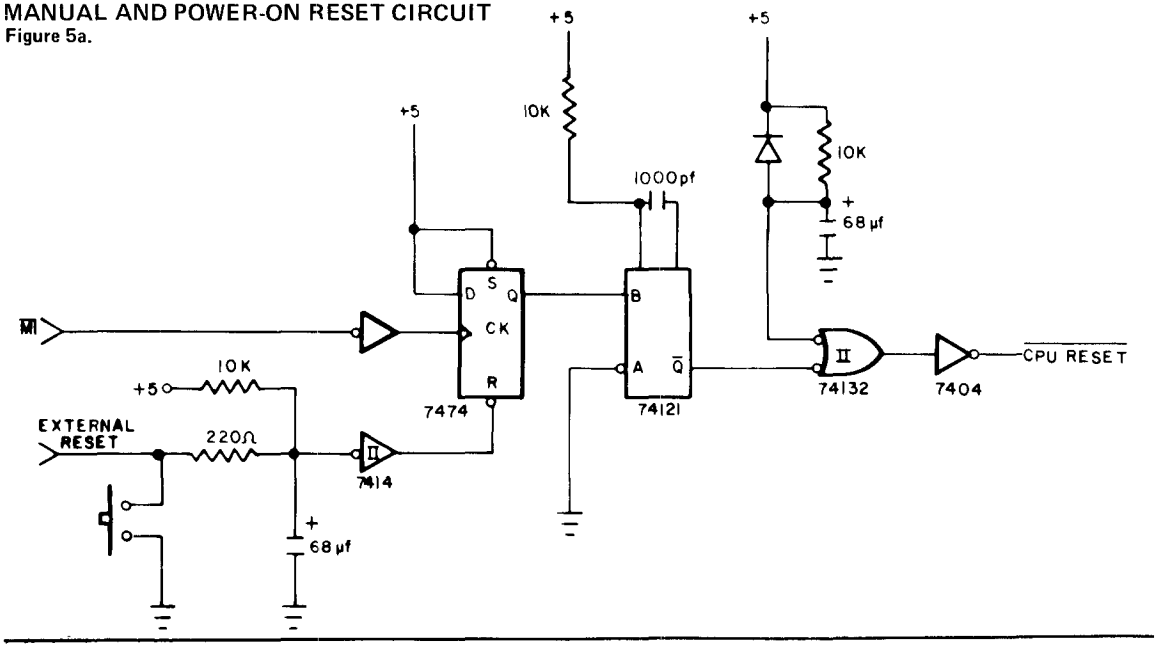
$\overline{RAS}$ TIMING WITH AND WITHOUT ADDRESS LATCH
Figure 4.



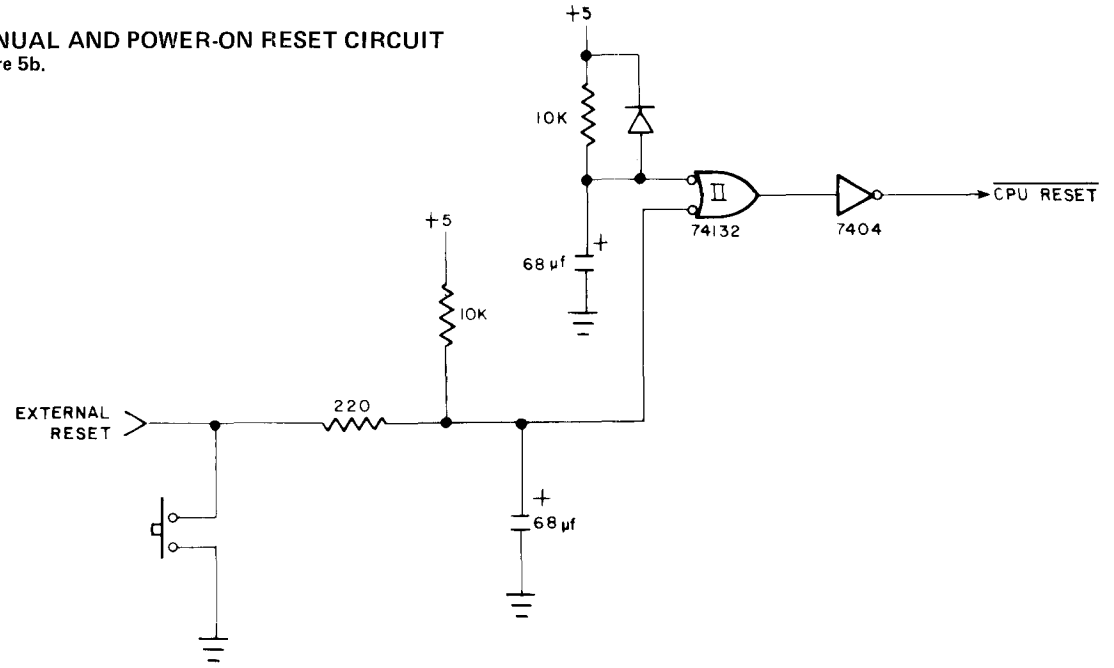
trol Signals that one of the conditions that will cause dynamic memory to be destroyed is a reset pulse of duration greater than 1ms. The circuit shown in Figure 5a can be used to generate a short reset pulse from either a push button or an external source. Additionally the manual reset is synchronized to the start of an M1 cycle so that the reset will not fall during the middle of a memory cycle. Along with the manual reset, the circuit will also generate a power on reset.

If it is not necessary that the contents of the dynamic memory be preserved, then the reset circuit shown in Figure 5b may be used to generate a manual or power on reset.

MANUAL AND POWER-ON RESET CIRCUIT
Figure 5a.

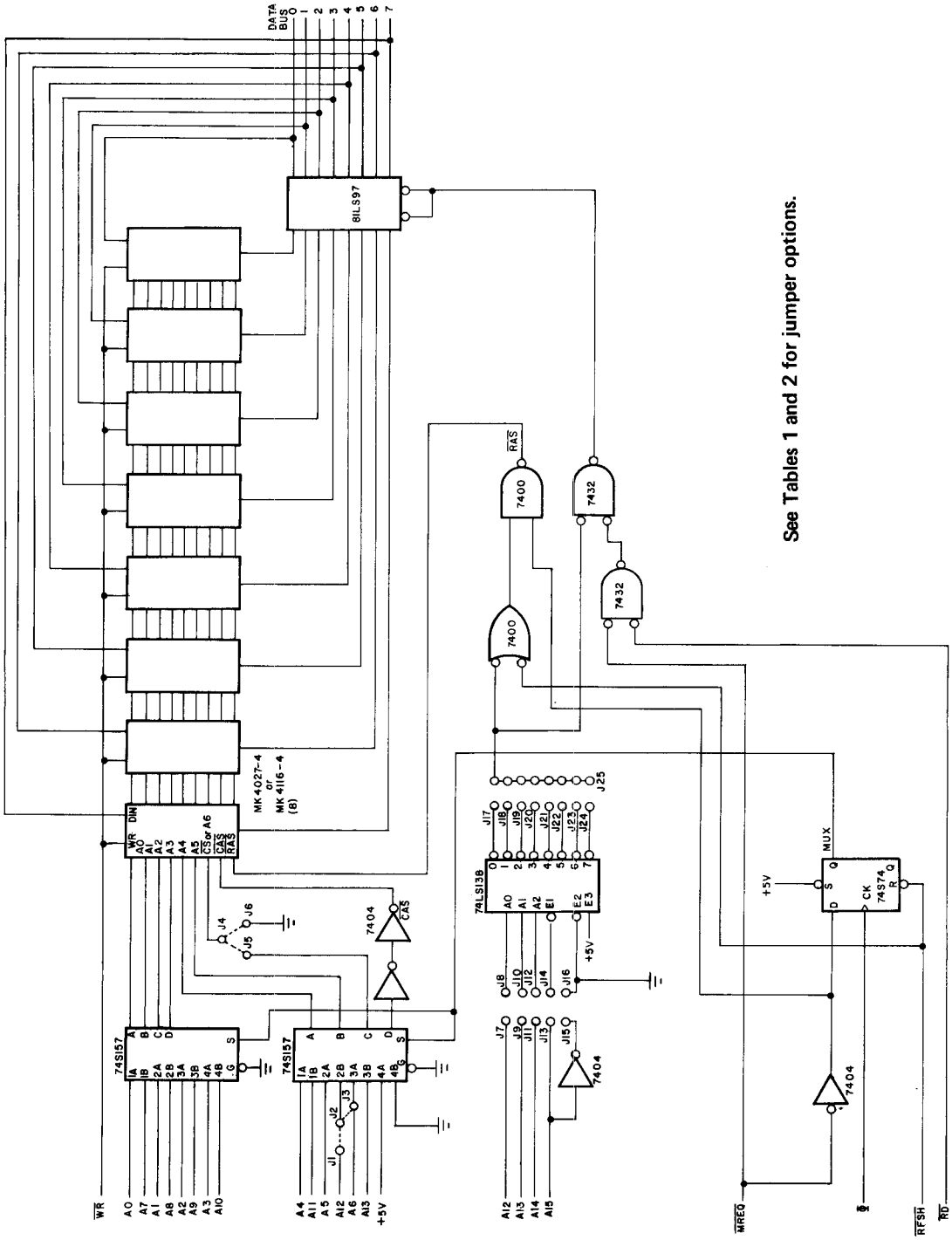


MANUAL AND POWER-ON RESET CIRCUIT
Figure 5b.



DESIGN EXAMPLE NO. 1 SCHEMATIC DIAGRAM

Figure 6.



DESIGN EXAMPLES FOR INTERFACING THE Z80 TO DYNAMIC MEMORY

To illustrate the interface between the Z80 and dynamic memory, two design examples are presented. Example number 1 is for a 4K/16Kx8 memory and the example number 2 is a 16K/64Kx8 memory.

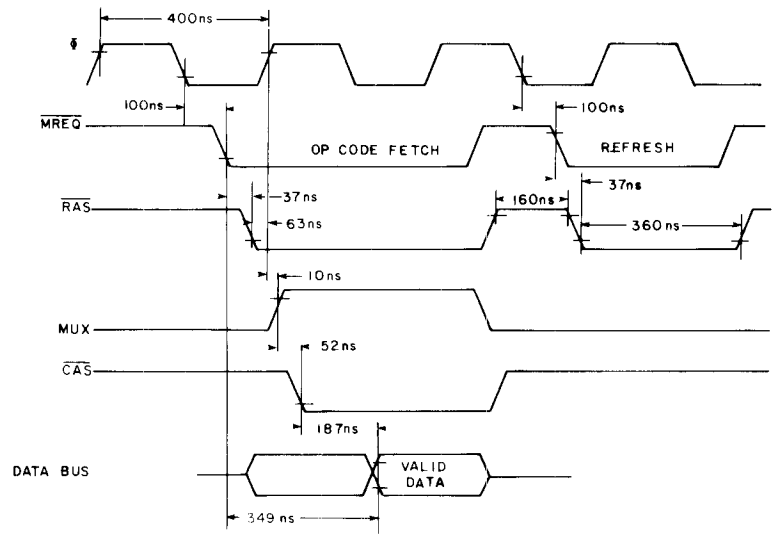
Design Example Number 1: 4K/16Kx8 Memory

This design example describes a 4K/16Kx8 memory that is best suited for a small single board Z80 based microcomputer system. The memory devices used in the example are the MK4027 (4,096x1 MOS Dynamic RAM) and the MK4116 (16,384x1 MOS Dynamic RAM). A very important feature of this design is the ease in which the memory can be expanded from a 4Kx8 to a 16Kx8 memory. This is made possible by the use of jumper options which configure the memory for either the MK4027 or the MK4116. See Table 1 and 2 for jumper options.

Figure 6 shows the schematic diagram for the 4K/16Kx8 memory. A timing diagram for the Z80 control signals and memory control signals is shown in Figure 7. The operation of the circuit may be described as follows: $\overline{RAS}$ is generated by NANDing MREQ with RFSH + ADDRESS DECODE. $\overline{RFSH}$ is generated directly from the Z80 while address decode comes from the 74LS138 decoder. Address decode indicates that the address on the bus falls within the memory boundaries of the memory. If an op code fetch or memory read is being executed the 81LS97 output buffer will be enabled at approximately the same time as $\overline{RAS}$ is generated for the memory array. The output buffer is enabled only during an op code fetch or memory read when $\overline{ADDRESS\ DECODE}$, $\overline{MREQ}$, and $\overline{RD}$ are all low. The switch multiplexer signal (MUX) is generated on the rising edge of Φ after $\overline{MREQ}$ has gone low during an op code fetch, memory read or memory write. After MUX is generated and the address multiplexers switch from the row address to column address, $\overline{CAS}$ will be generated. $\overline{CAS}$ comes from one of the outputs of the multiplexer and is delayed by two gate delays to ensure that the proper column address set-up time will be achieved. Once $\overline{RAS}$ and $\overline{CAS}$ have been generated for the memory array, the memory will then access the desired location for a read or write operation.

7404	22ns	Generate $\overline{RAS}$ from $\overline{MREQ}$
7400	15ns	
	63ns	$\overline{RAS}$ to rising edge of Φ
74S74	10ns	Φ to MUX
74S157	15ns	Generate $\overline{CAS}$ from MUX
7404	22ns	
7404	15ns	
t_{CAC}	165ns	$\overline{CAS}$ access time
81LS97	22ns	Output buffer delay
	349ns	Worst case access

DESIGN EXAMPLE NO. 1 MEMORY TIMING
Figure 7.



The worst case access time required by the CPU for the op code fetch is 450ns (from equation 1); therefore, the circuit exceeds the required access time by 101ns (worst case).

The circuit shown in Figure 6 provides excellent performance when used as a small on board memory. The memory size should be held at eight devices because there is not sufficient timing margin to allow the interface circuit to drive a larger memory array.

Design Example Number 2: 16Kx8 Memory

This design example describes a 16K/64Kx8 memory which is best suited for a Z80 based microcomputer system where a large amount of RAM is desired. The memory devices used in this example are the same as for the first example, the MK4027 and the MK4116. Again as with the first example, the memory may be expanded from a 16Kx8 to a 64Kx8 by reconfiguring jumpers. See Table 3 and 4 for jumper options.

Figure 8 shows the schematic diagram for the 16K/64K memory. A timing diagram is shown in Figure 9. The operation of the circuit can be described as follows: $\overline{RAS}$ is generated by NANDing MREQ with ADDRESS DECODE (from the two 74LSI38s) + RFSH. Only one row of RAMs will receive a $\overline{RAS}$ during an op code fetch, memory read or memory write. However, a $\overline{RAS}$ will be generated for all rows within the array during a refresh cycle. $\overline{MREQ}$ is inverted and fed into a TTL compatible delay line to generate MUX and $\overline{CAS}$. (This particular approach differs from the method used in example number 1 in that all memory timing is referenced to MREQ, whereas the circuit in example number 1 bases its memory timing from both $\overline{MREQ}$ and the clock. Both methods offer good results, however, the TTL delay line approach offers the best control over the memory timing.) MUX is generated 65ns later and is used to switch the 74157 multiplexers from the row to the column address. The 65ns delay was chosen to allow adequate margin for the row address hold time t_{RAH}. At 110ns, $\overline{CAS}$ is generated from the delay line and NANDed with RFSH, which inhibits a $\overline{CAS}$ during refresh cycle. After $\overline{CAS}$ is applied to the memory, the desired location is then accessed. A worst case access timing analysis for the circuit shown in Figure 8 can be computed as follows:

74LS14	22ns	Generate $\overline{RAS}$ from $\overline{MREQ}$
74LS00	15ns	
delay line	50ns	MUX from $\overline{RAS}$
delay line	45ns	$\overline{CAS}$ delay from MUX
7400	20ns	
t_{CAC}	165ns	Access time from $\overline{CAS}$
8833	30ns	Output buffer delay
	347ns	

The required access time from the CPU is 450ns (from equation 1). This leaves 103ns of margin for additional CPU buffers on the control and address lines. This particular circuit offers excellent results for an application which requires a large amount of RAM memory. As mentioned earlier, the memory timing used in this example offers the best control over the memory timing and would be ideally suited for an application which required direct memory access (DMA).

4K x 8 CONFIGURATION(MK4027) JUMPER

Table 1

CONNECT: J13 to J14	
ADDRESS	CONNECT
0000-0FFF	J17 to J25
1000-1FFF	J18 to J25
2000-2FFF	J19 to J25
3000-3FFF	J20 to J25
4000-4FFF	J21 to J25
5000-5FFF	J22 to J25
6000-6FFF	J23 to J25
7000-7FFF	J24 to J25

Connect: J2 to J3, J4 to J6, J7 to J8, J9 to J10, J11 to J12

CONNECT: J14 to J15	
ADDRESS	CONNECT
8000-8FFF	J17 to J25
9000-9FFF	J18 to J25
A000-AFFF	J19 to J25
B000-BFFF	J20 to J25
C000-CFFF	J21 to J25
D000-DFFF	J22 to J25
E000-EFFF	J23 to J25
F000-FFFF	J24 to J25

16K x 8 CONFIGURATION (MK4116) JUMPER CONNECTIONS

Table 2

CONNECT:	ADDRESS	CONNECT
J1 to J2, J4 to J5	0-3FFF	J17 to J25
J8 to J11	4000-7FFF	J18 to J25
J10 to J13	8000-BFFF	J19 to J25
J12 to J16	C000-FFFF	J20 to J25
J14 to J16		

16K x 8 CONFIGURATION (MK4027)

Table 3

CONNECT:	J1 to J3
	J5 to J6
	J7 to J8
	J9 to J10
	J11 to J12
	J13 to J14

ADDRESS:	0-3FFF	ADDRESS:	4000-7FFF	ADDRESS:	8000-BFFF	ADDRESS:	C000-FFFF
CONNECT:	J24 to J25	CONNECT:	J16 to J17	CONNECT:	J40 to J41	CONNECT:	J32 to J33
	J26 to J27		J18 to J19		J42 to J43		J34 to J35
	J28 to J29		J20 to J21		J44 to J43		J36 to J37
	J30 to J31		J22 to J23		J46 to J47		J38 to J39

64K x 8 CONFIGURATION(MK4116)

Table 4

CONNECT:	J1 to J2
	J4 to J5
	J8 to J11
	J10 to J13
	J12 to J15
	J14 to J15

ADDRESS: 0-FFFF	
CONNECT:	J32 to J33
	J34 to J35
	J36 to J37
	J38 to J39

SYSTEM PERFORMANCE CHARACTERISTICS

Table 5

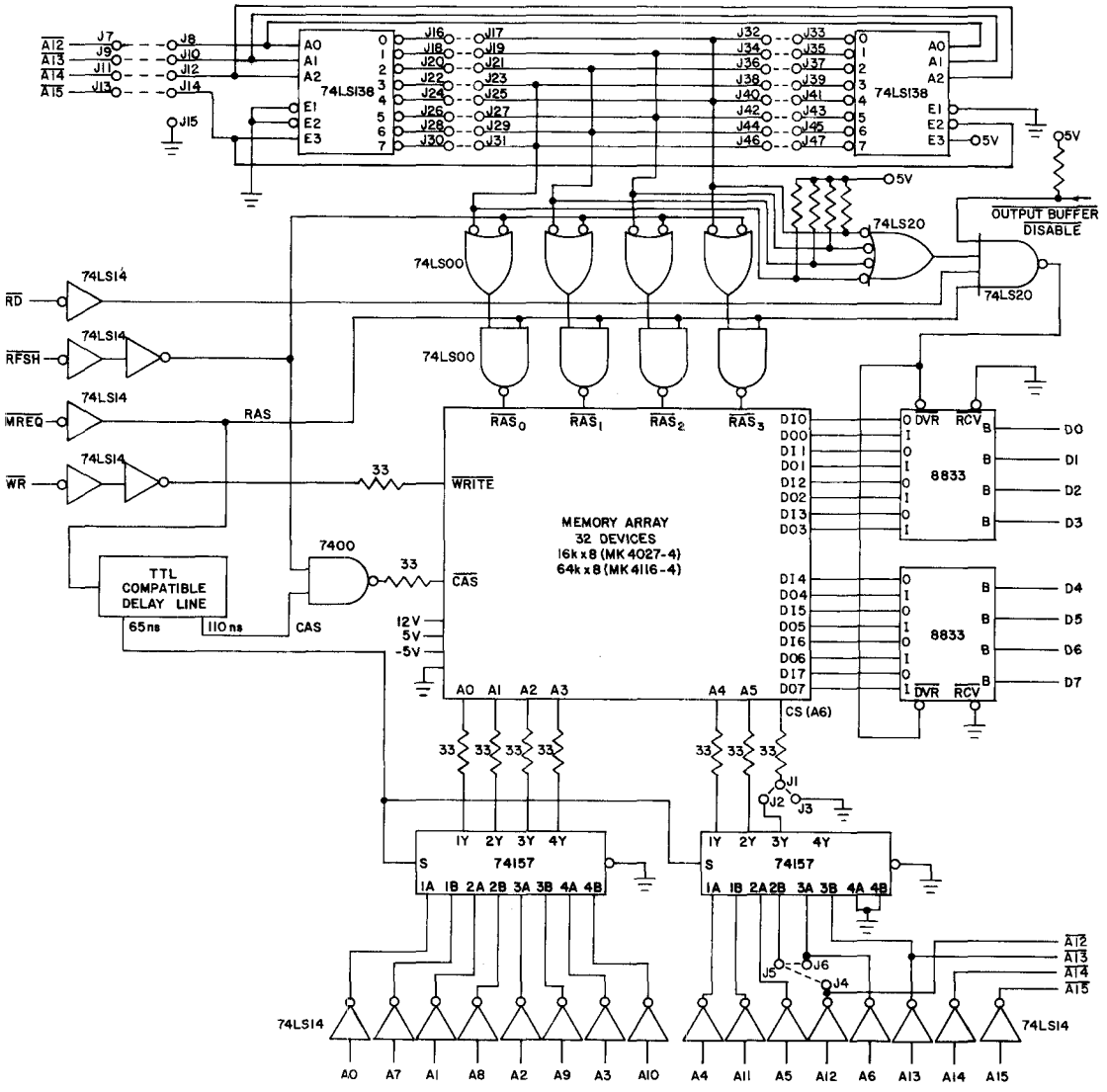
The system characteristics for the preceeding design examples are shown in Table 5.

EXAMPLE #	MEMORY CAPACITY	MEMORY ACCESS	POWER REQUIREMENTS
1	4K/16Kx8	349ns max.	+12V @ 0.0250 A max. +5V @ 0.422 A max.* -5V @ 0.030 A max.
2	16K/64Kx8	347ns max.	+12V @ 0.600 A max. +5V @ 0.550 A max. * -5V @ 0.030 A max.

*All power requirements are max.; operating temperature 0°C to 70°C ambient, max +12V current computed with Z80 executing continuous op code fetch cycles from RAM at 1.6 μs intervals.

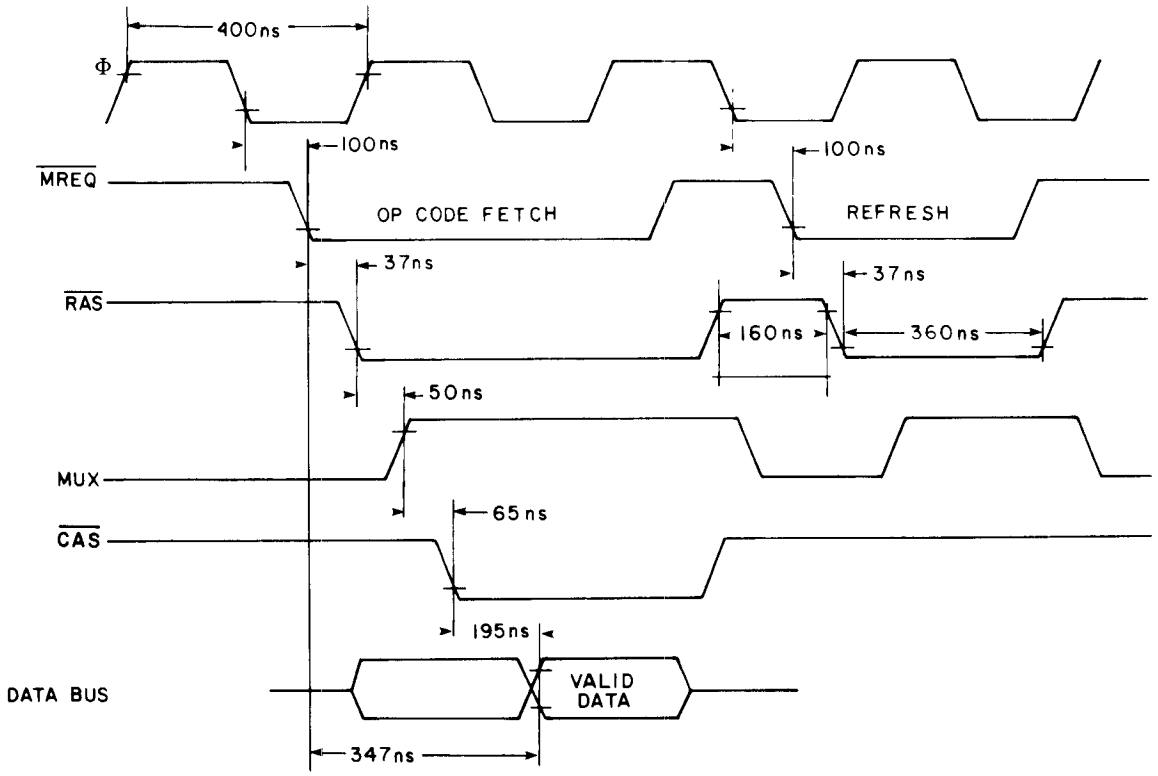
DESIGN EXAMPLE NO. 2 SCHEMATIC DIAGRAM

Figure 8.



FOR JUMPER OPTIONS SEE TABLES 3 AND 4

DESIGN EXAMPLE NO. 2 MEMORY TIMING
Figure 9.



PRINTED CIRCUIT LAYOUT

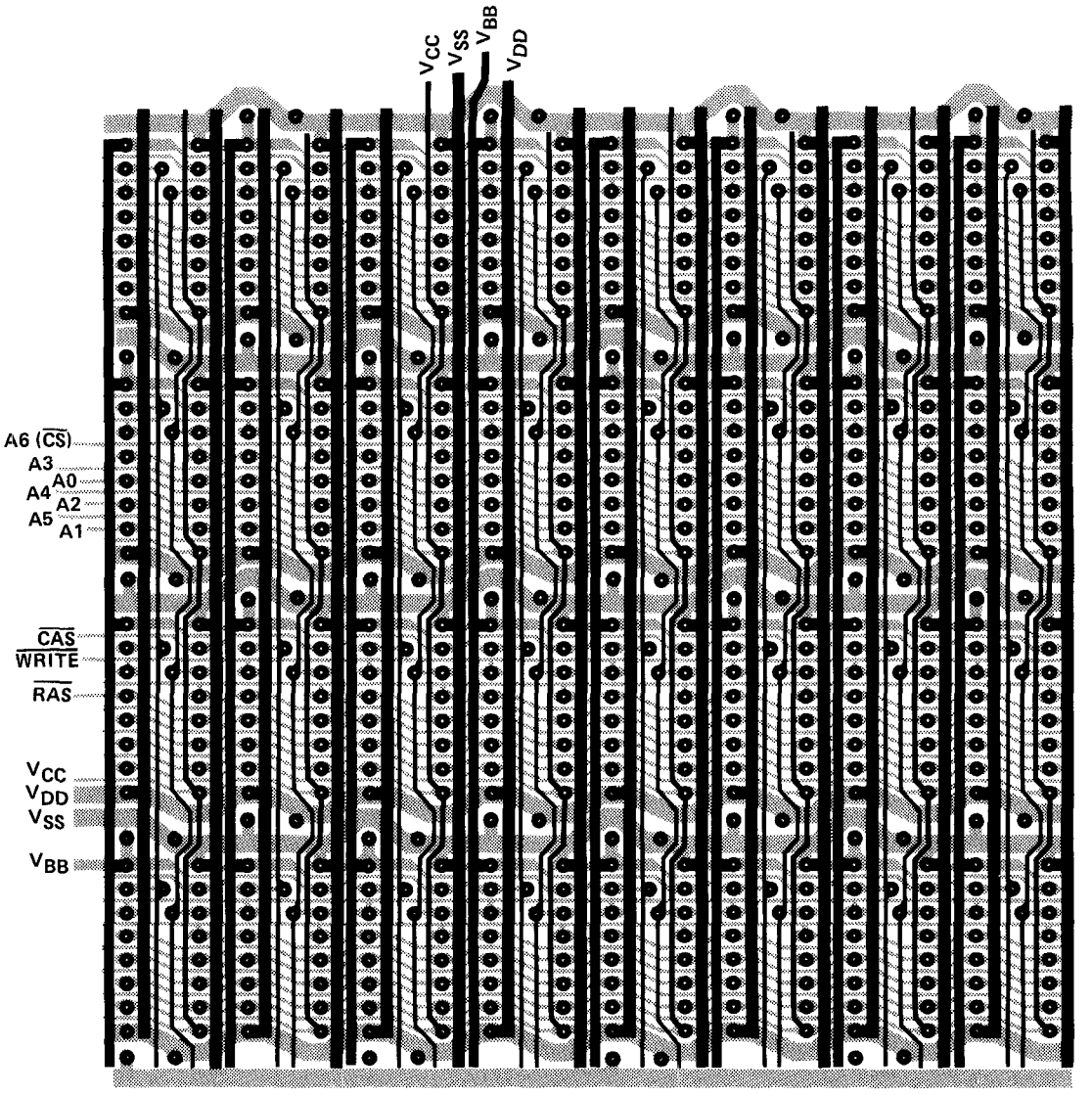
One of the most important parts of a dynamic memory design is the printed circuit layout. Figure 10 illustrates a recommended layout for 32 devices. A very important factor in the P.C. layout is the power distribution. Proper power distribution on the V_{DD} and V_{BB} supply lines is necessary because of the transient current characteristics which dynamic memories exhibit. To achieve proper power distribution, V_{DD}, V_{BB}, V_{CC} and ground should be laid out in a grid to help minimize the power distribution impedance. Along with good power distribution, adequate capacitive bypassing for each device in the memory array is necessary. In addition to the individual by-passing capacitors, it is recommended that each supply (V_{BB}, V_{CC} and V_{DD}) be bypassed with an electrolytic capacitor 20μF.

By using good power distribution techniques and using the recommended number of bypassing capacitors, the designer can minimize the amount of noise in the memory array. Other layout considerations are the placement of signal lines. Lines such as address, chip select, column address strobe, and write should be bussed together as rows; then, bus all rows together at one end of the array. Interconnection between rows should be avoided. Row address strobe lines should be bussed together as a row, then connected to the appropriate $\overline{RAS}$ driver. TTL drivers for the memory array signals should be located as close as possible to the array to help minimize signal noise.

For a large memory array such as the one shown in design example number 2, series terminating resistors should be used to minimize the amount of negative undershoot. These resistors should be used on the address lines, $\overline{CAS}$ and $\overline{WRITE}$, and have values between 20 Ω to a 33 Ω .

The layout for a 32 device array can be put in a 5" x 5" area on a two sided printed circuit board.

SUGGESTED P. C. LAYOUT FOR MK4027 or MK4116
Figure 10.



4MHz Z80 DYNAMIC MEMORY INTERFACE CONSIDERATIONS

A 4MHz Z80 is available for the microcomputer designer who needs higher system throughput. Considerations which must be faced by the designer when interfacing the 4MHz Z80 to dynamic memory are the need for memories with faster access times and for providing minimum RAM precharge time. The access times required for dynamic memory interfaced to a 4MHz Z80 can be computed from equations 1 and 2 under Z80 Timing and Memory Control Signals.

Access time for op code fetch for 4MHz Z80,
let: $t_C = 250ns$; $t_{D\overline{L\Phi}(MR)} = 75ns$; $t_{s\overline{\Phi}(D)} = 35ns$
then: $t_{ACCESS\ OP\ CODE} = 265ns$

Access time for memory read for 4MHz Z80,
let: $t_C = 250ns$; $t_{D\overline{L\Phi}(MR)} = 75ns$; $t_{s\overline{\Phi}(D)} = 50ns$
then: $t_{ACCESS\ MEMORY\ READ} = 375ns$

The problem of faster access times can be solved by using 200ns memories such as the MK4027-3 or MK4116-3. Depending on the number of buffer delays in the system, the designer may have to use 150ns memories such as the MK4027-2 or MK4116-2. The most critical problem that exists when interfacing dynamic memory to the 4MHz Z80 is the RAM precharge time (trp). This parameter is called $t_{W(MRH)}$ on the Z80 and can be computed by the following equation.

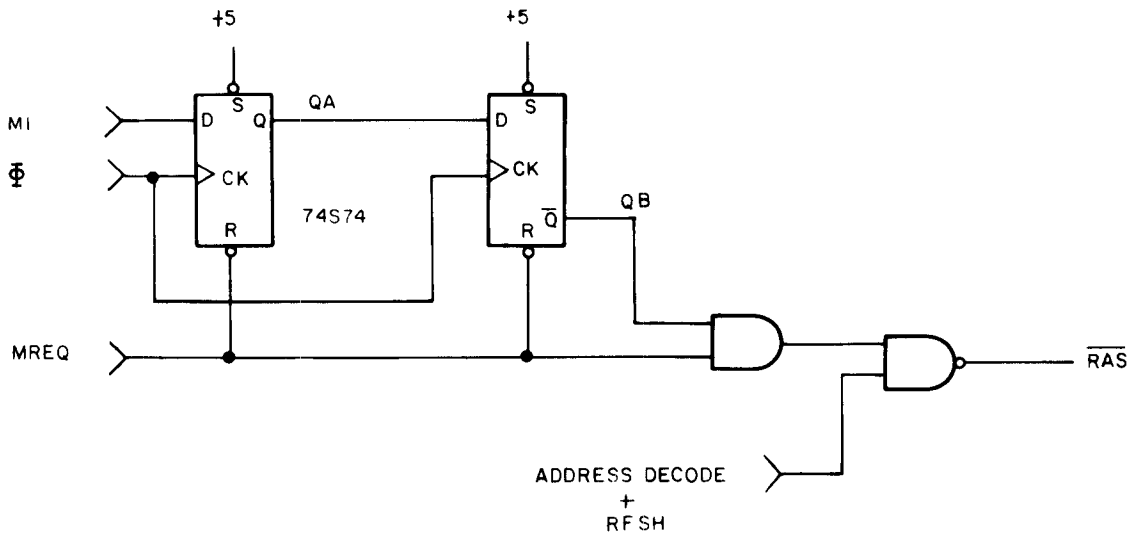
(4) $t_{W(RH)} = t_{W(\Phi H)} + t_f - 20ns$
let: $t_{W(\Phi H)} = 110ns$; $t_f = 5ns$
then: $t_{W(MRH)} = 95ns$

A $t_{W(MRH)}$ of 95ns will not meet the minimum precharge time of the MK4027-2 or MK4116-2 which is 100ns. The MK4027-3 and MK4116-3 require a 120ns precharge. Figure 11 shows a circuit that will lengthen the $t_{W(MRH)}$ pulse from 95ns to a minimum of 126ns while only inserting one gate delay into the access timing chain. Figure 12 shows the timing for the circuit of Figure 11. The operation of the circuit in Figure 11 can be explained as follows: The D flip flops are held in a reset condition until MREQ goes to its active state. After MREQ goes active, on the next positive clock edge, the D input of U1 and U2 will be transferred to the outputs of the flip flops. Output QA will go high if M1 was high when Φ clocked U1. Output QB will go low on the next positive going clock edge, which will cause the output of U3 to go low and force the output of U4, which is $\overline{RAS}$, high. The flip flops will be reset when MREQ goes inactive.

The circuit shown in Figure 11 will give a minimum of 126ns precharge for dynamic memories, with the Z80 operating at 4MHz. The 126ns $t_{W(MRH)}$ is computed as follows.

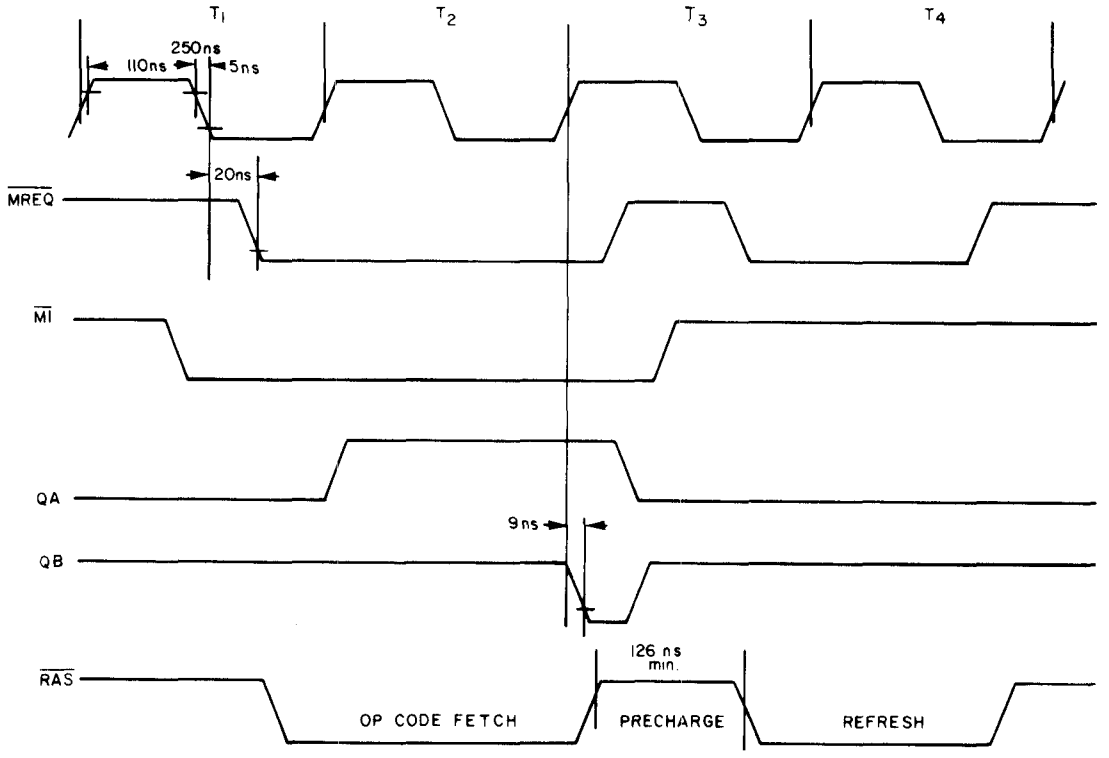
110ns	$t_{W(\Phi H)}$ - clock pulse width high (min)
5ns	t_F - clock full time (min)
20ns	$t_{D\overline{L\Phi}(MR)}$ - MREQ delay (min)
-9ns	74S74 delay (min)
126ns	$t_{W(MRH)}$ modified (min)

4MHz Z80 PRECHARGE EXTENDER FOR DYNAMIC MEMORIES
Figure 11



TIMING DIAGRAM FOR 4MHz Z80 PRECHARGE EXTENDER

Figure 12



APPENDIX

MEMORY TEST ROUTINE

This section is intended to give the microcomputer designer a memory diagnostic suitable for testing memory systems such as the ones shown in Section VI.

The routine is a modified address storage test with an incrementing pattern. A complete test requires 256_{10} passes, which will execute in less than 4 minutes for a 16Kx8 memory. If an error occurs, the program will store the pattern in location '2C'H and the address of the error at locations '2D'H and '2E'H.

The program is set up to test memory starting at location '2F'H up to the end of the block of memory defined by the bytes located at 'OC'H and 'OD'H. The test may be set up to start at any location by modifying locations '03'H - '04'H and '11'H - '12'H with the starting address that is desired.

```
                              MXRTS  LISTING                    PAGE   0001
LOC   OBJ CODE   STMT SOURCE STATEMENT

                 0001 ;TRANSLATED FROM DEC 1976 INTERFACE MAGAZINE
                 0002 ;
                 0003 ;THIS IS A MODIFIED ADDRESS STORAGE TEST WITH AN
                 0004 ;INCREMENTING PATTERN
                 0005 ;
                 0006 ;256 PASSES MUST BE EXECUTED BEFORE THE MEMORY IS
                 0007 ;COMPLETELY TESTED.
                 0008 ;
                 0009 ;IF AN ERROR OCCURS, THE PATTERN WILL BE STORED
                 0010 ;AT LOCATION '002C'H AND THE ADDRESS OF THE
                 0011 ;ERROR LOCATION WILL BE STORED AT '002D'H AND
                 0012 ;'002E'H.
                 0013 ;
```

MEMORY TEST ROUTINE (Cont'd.)

```
                        0014 ;THE CONTENTS OF LOCATIONS '000C'H AND '001D'H
                        0015 ;SHOULD BE SELECTED ACCORDING TO THE FOLLOWING
                        0016 ;MEMORY SIZE TO BE TESTED
                        0017 ;
                        0018 ;TOP OF MEMORY TO
                        0019 ;BE TESTED                          VALUE OF EPAGE
                        0020 ;
                        0021 ;       4K                                  '10'H
                        0022 ;       8K                                  '20'H
                        0023 ;      16K                                  '40'H
                        0024 ;      32K                                  '80'H
                        0025 ;      48K                                  'C0'H
                        0026 ;      64K                                  'FF'H
                        0027 ;
                        0028 ;THE PROGRAM IS SET UP TO START TESTING AT
                        0029 ;LOCATION '002F'H. THE STARTING ADDRESS FOR THE
                        0030 ;TEST CAN BE MODIFIED BY CHANGING LOCATIONS
                        0031 ;'0003-0004'H AND '0011-0012'H.
                        0032 ;
                        0033 ;TEST TIME FOR A 16K X 8 MEMORY IS APPROX. 4 MIN
                        0034 ;
0000                    0035          ORG   0000H
0000   0600             0036          LD    B,0          ;CLEAR B PATRN MODIFIER
                        0037 ;LOAD UP MEMORY
0002   212F00           0038 LOOP:    LD    HL,START     ;GET STARTING ADDR
0005   7D               0039 FILL:    LD    A,L          ;LOW BYTE TO ACCM
0006   AC               0040          XOR   H            ;XOR WITH HIGH BYTE
0007   A8               0041          XOR   B            ;XOR WITH PATTERN
0008   77               0042          LD    (HL),A       ;STORE IN ADDR
0009   23               0043          INC   HL           ;INCREMENT ADDR
000A   7C               0044          LD    A,H          ;LOAD HIGH BYTE OF ADDR
000B   FE10             0045          CP    EPAGE        ;COMPARE WITH STOP ADDR
000D   C20500           0046          JP    NZ,FILL      ;NOT DONE,GO BACK
                        0047 ;READ AND CHECK TEST DATE
0010   212F00           0048          LD    HL,START     ;GET STARTING ADDR
0013   7D               0049 TEST:    LD    A,L          ;LOAD LOW BYTE
0014   AC               0050          XOR   H            ;XOR WITH HIGH BYTE
0015   A8               0051          XOR   B            ;XOR WITH MODIFIER
0016   BE               0052          CP    (HL)         ;COMPARE WITH MEMORY LOC
0017   C22500           0053          JP    NZ,FXIT      ;ERROR EXIT
001A   23               0054          INC   HL           ;UPDATE MEMORY ADDRESS
001B   7C               0055          LD    A,H          ;LOAD HIGH BYTE
001C   FE10             0056          CP    EPAGE        ;COMPARE WITH STOP ADDR
001E   C21300           0057          JP    NZ,TEST      ;LOOP BACK
0021   04               0058          INC   B            ;UPDATE MODIFIER
```

```
                                MXRTS  LISTING                    PAGE   0002
 LOC   OBJ CODE   STMT SOURCE STATEMENT

0022   C30200           0059          JP    LOOP         ;RST WITH NEW MODIFIER
                        0060 ;ERROR EXIT
0025   222D00           0061 FXIT:    LD    (BYTE),HL    ;SAVE ERROR ADDRESS
0028   322C00           0062          LD    (PATRN),A    ;SAVE BAD PATTERN
002B   76               0063          HALT               ;FLAG OPERATOR
002C                    0064 PATRN:   DEFS  1
002D                    0065 BYTE:    DEFS  2
002F   2F00             0066 START:   DEFW  $
                        0068 EPAGE:   EQU   10H          ;SET UP FOR 4K TEST
                        0069          END
```