MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by AN1127/D

AN1127

High Speed DRAM Design for the 40 MHz MC68EC030

This design demonstrates a memory solution using inexpensive DRAM to provide the performance and density required for today's high performance embedded control systems.

DIFFERENCES IN DRAM AND SRAM

While SRAMs are generally easier to interface at higher speeds, they are limited in size and by higher cost. DRAMs provide larger capacities at a lower price, but have increased interfacing complexity and slower access time. The DRAM addresses must be multiplexed to divide them between the row and column addresses. Instead of the SRAM's simple chip select, DRAMs require two signals to latch in the row and column addresses. The DRAM output enable and write enable signals are similar to their SRAM equivalents.

Except for the newer bursting SRAM like the MCM62940, SRAMs generally require less access time, but the time is constant for each address. Because the DRAM separates the addresses between row and column, access times between different memory locations vary greatly. If both row and column addresses are needed, the access time is 2 to 3 times longer than changing only the column addresses. The MC68EC030 exploits the difference in DRAM access time by bursting, which stores the next three long words in the MC68EC030 on-chip caches. There are three main types of DRAM: page mode, static column and nibble mode. Each of these allows subsequent accesses to memory locations in different ways. Page mode requires both the column address and the column address strobe to be changed to access the next address with the same row address. Static column requires only the column address to be changed to access the next address with the same row address. Nibble mode requires only the column address strobe to be changed, but only four contiguous memory locations can be accessed.

A final difference between the two memory types gives them their names. SRAMs (Static Random Access Memory) are static memory devices i.e., the data will remain as long as the part is powered. The DRAMs (Dynamic Random Access Memory) are dynamic and require periodic refreshing to avoid losing data. Because the DRAM are unavailable during refresh, a DRAM design may have lower performance than an SRAM design of the same speed. The performance difference depends on the number of MPU accesses attempted during refresh.

ACCESS TIMING

The MC68EC030 has three types of accesses suitable for DRAM; read access with burst, read access without burst, and write access without burst. Figure 1 shows the timing for a read access with burst. Figure 2 shows a read access without burst. Figure 3 shows a write access. The DRAM interface logic further divides these three types of accesses based on whether the access immediately follows the previous access or there was one or more idle clock periods in between. Figures 1, 2 and 3 show accesses with one or more idle clocks before AS going low. Figure 4 shows a burst read being immediately followed by a byte write access. The MC68EC030 uses about 60% of the bus with 5 clock initial, 2 clock bursting. The remaining 40% is available to other bus masters. Thus, for a large portion of the time, the accesses are not back-to-back and will have one or more idle clock periods in between accesses.

The read access with burst is the most challenging for the DRAM interface, but allows read accesses to fill the on-chip caches efficiently. Studies show that typical programs spend most of their execution time in a few main routines or tight loops (known as locality of reference). Therefore, for any given instruction, the probability is high that you will need the next sequential instruction or will loop back to a nearby instruction. In a burst access, the MC68EC030 reads the first long word of the burst access, and then reads the next three sequential long words. The MC68EC030 provides only the first longword's address. The remaining three longwords are modulo four offsets from the starting longword's address. The MC68EC030 requests the burst access by the Cache Burst REQuest (CBREQ) signal. The burst is acknowledged to the MC68EC030 by Synchronous TERMination (STERM) signal and Cache Burst ACKnowledge (CBACK).

The three types of accesses start the same. The multiplexer passes the row addresses through to the DRAM. The control logic asserts the row address strobe (RAS) signal. RAS is based on the MC68EC030 address strobe (AS) being asserted, a decode of the address pins, and the state machine indicating the RAS recovery time is over. Next, after a RAS assertion to row address hold time, the ROW address/COLumn address (ROW/COL) change switches the addresses to the column address of the first long word being accessed. The control logic asserts the column strobe (CS). Then the control logic asserts the STERM before the rising edge of the fifth clock. In a read without burst access, the DRAMs present all four bytes of data to the MC68EC030 and the MC68EC030 selects which bytes it needs. In a write access, the CS signals the DRAM which bytes to store. In a read burst access, the MC68EC030 always requires all 32 bits for each of the four accesses.

The CS are divided between the four byte lanes. These byte lanes are selected by a decode of the SIZe (SIZ1, SIZ0), lower two addresses (A1, A0) and the Read/Write (R/W). Upper Upper byte (UU) is the byte enable for D31 – D24. Upper Middle byte (UM) is the byte enable for D23 – D16. Lower Middle byte (LM) is the byte enable for D15 – D8. Lower Lower byte (LL) is the byte enable for D7 – D0.

```
UU    =    AS & R/W & A1 & A0          ; write access
       #    AS & R/W                    ; read access
UM    =    AS & R/W & SIZ0 & A1        ; write access word or long word
       #    AS & R/W & SIZ1 & A1        ; write access word or 3 byte
       #    AS & R/W & A1 & A0          ; write access byte 2
       #    AS & R/W                    ; read access
LM    =    AS & R/W & SIZ1 & SIZ0 & A1 ; write access long word
       #    AS & R/W & SIZ1 & SIZ0 & A1 ; write access 3 byte
       #    AS & R/W & SIZ0 & A1 & A0   ; write access word or long word
       #    AS & R/W & A1 & A0          ; write access byte 3
       #    AS & R/W                    ; read access
LL    =    AS & R/W & SIZ1 & SIZ0      ; write access long word
       #    AS & R/W & SIZ1 & SIZ0 & A0 ; write access 3 byte
       #    AS & R/W & SIZ1 & A1        ; write access word or 3 byte
       #    AS & R/W & A1 & A0          ; write access byte 4
       #    AS & R/W                    ; read access
```

Now, the access differs based on the status of the $\overline{\text{CBREQ}}$ of the MC68EC030. The CBREQ signals the state machine to add the extra burst clocks required. The MC68EC030 reads the data on the falling edge of the fifth clock of the access. The control logic changes the column address with that falling clock edge. The data hold time required by the MC68EC030 is guaranteed by the minimum switching time of the multiplexing PAL and the DRAM data hold time. The access continues with the STERM being asserted at every other rising clock edge and the data being presented to the MC68EC030 on the following falling edge.

The read bursts, read without burst and write accesses again become the same with the last long word access. For the read without burst and write access, the last long word access is also the first long word access. RAS goes high with the falling edge of the final clock of the access. The ROW/COL and CS go high with the rising edge of the clock after the last access. This completes the access. The logic delays the next access by one clock if the AS is asserted before the RAS is recharged.

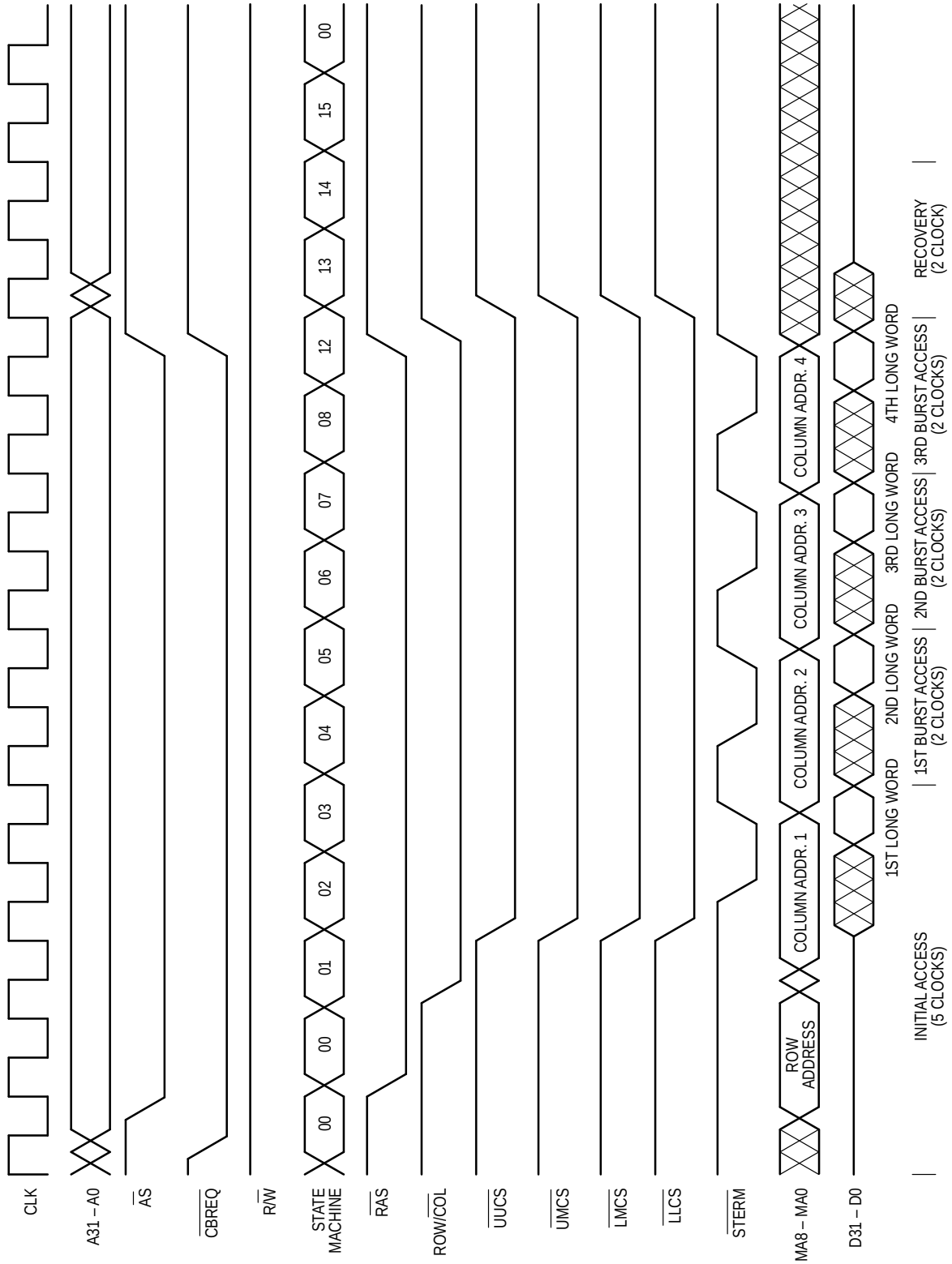

Figure 1. Burst Read Access Followed by Idle Clocks

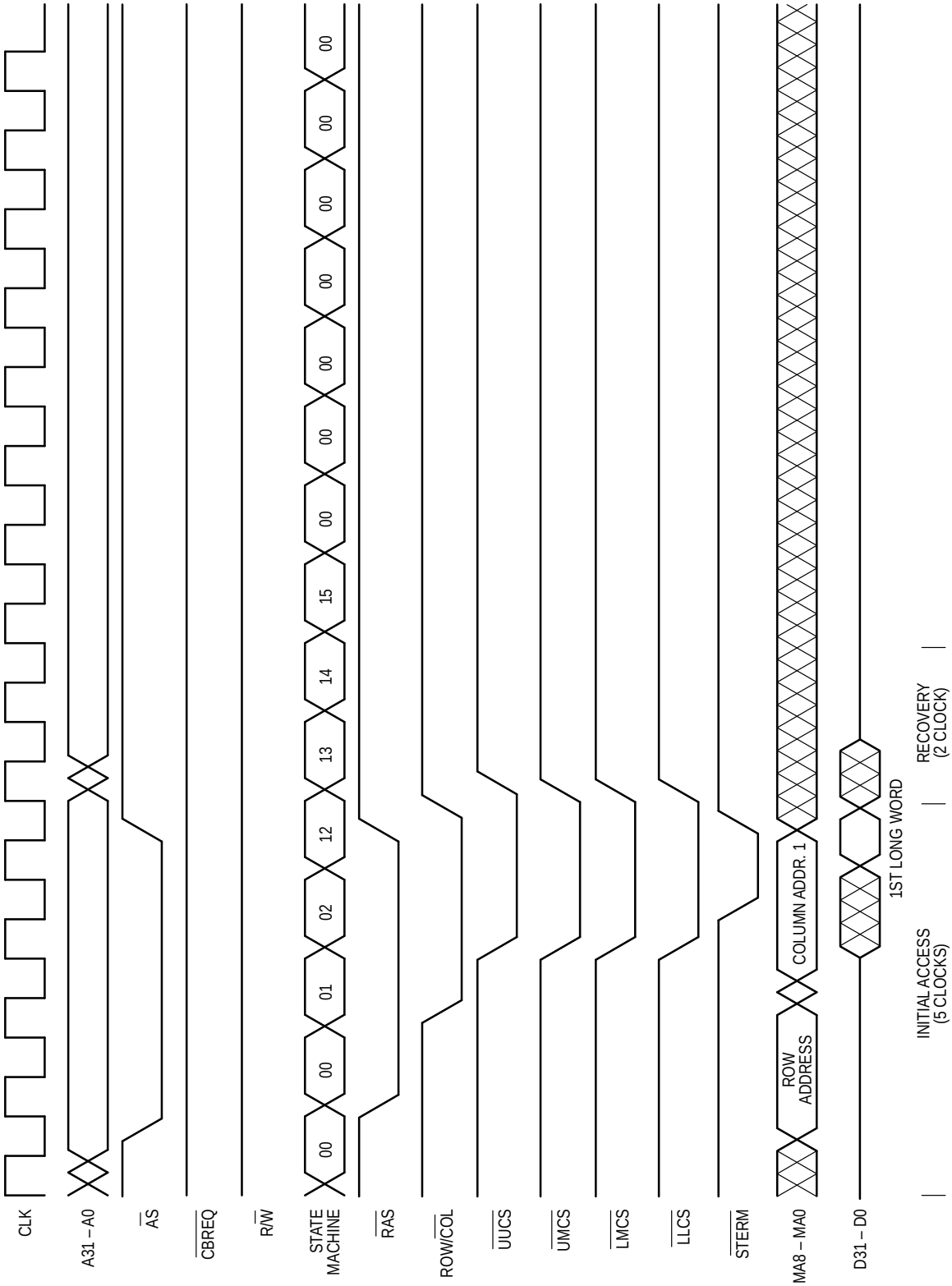

Figure 2. Non-Burst Read Access Followed by Idle Clocks

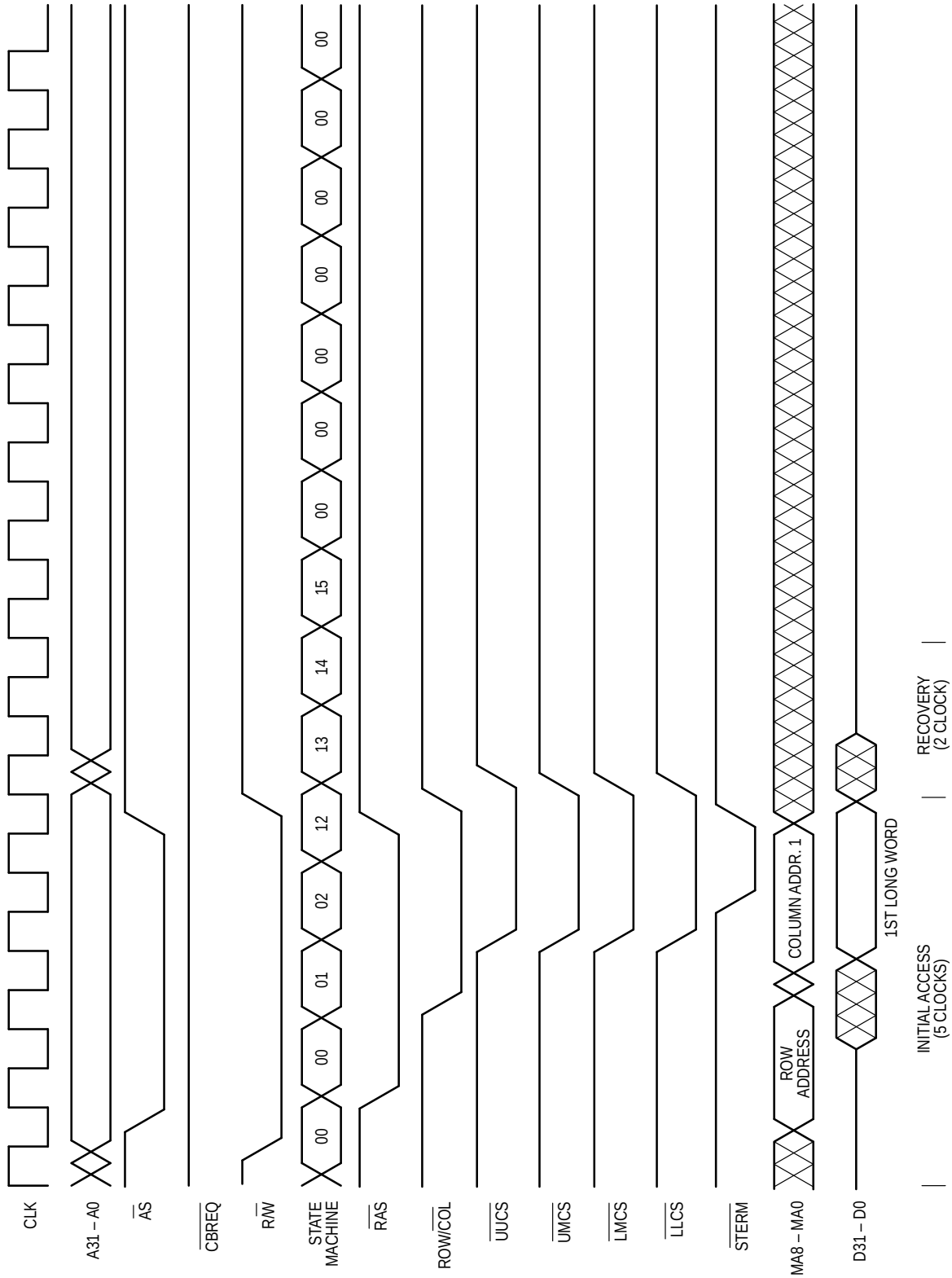

Figure 3. Non-Burst Write Access Followed by Idle Clocks

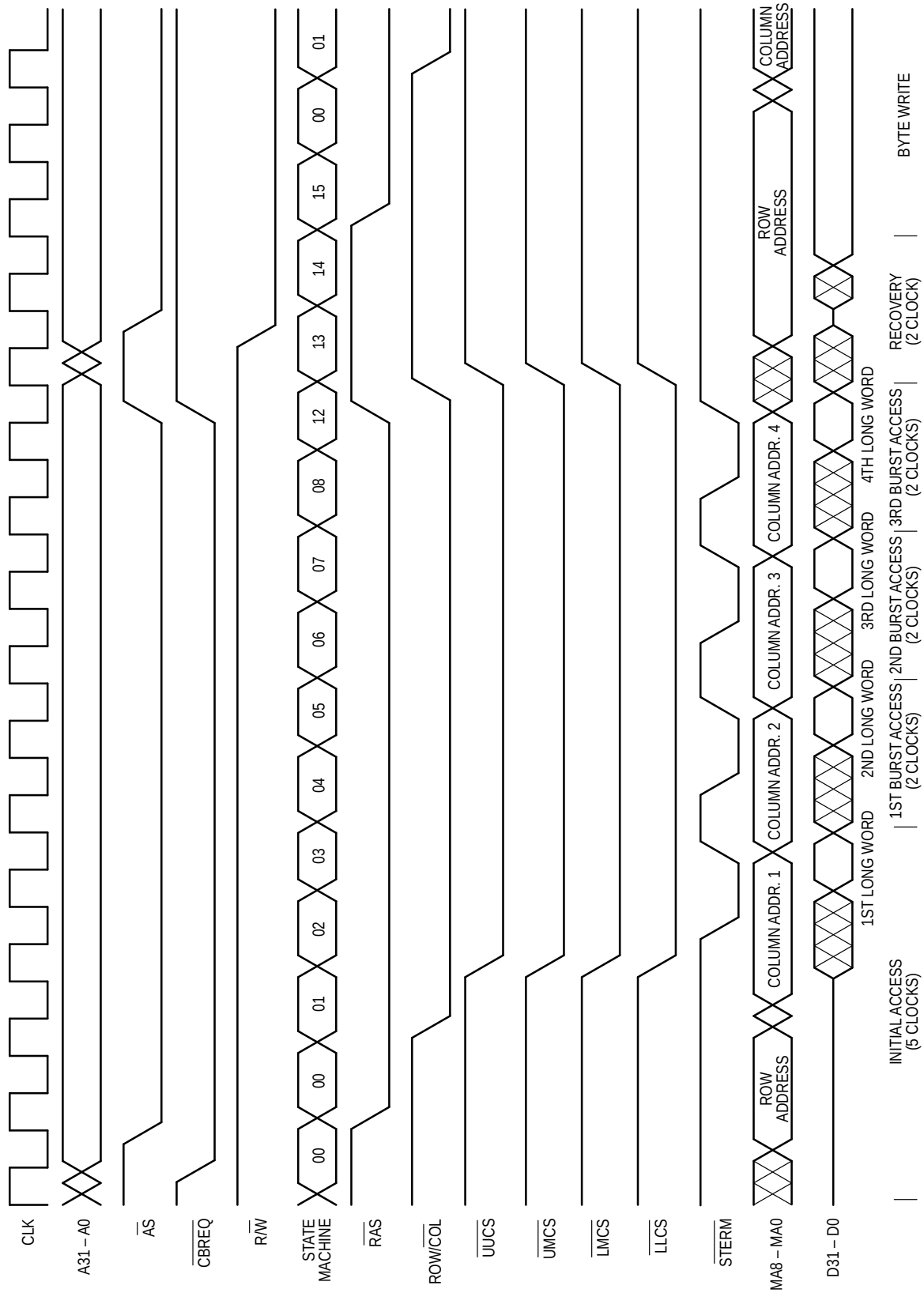

Figure 4. Burst Read Access Followed Immediately by a Byte Write Access

DESIGN OVERVIEW

This design (Figure 5) has five main parts; MC68EC030RP40, memory control logic, address multiplexing, one megabyte memory, and refresh control logic. These five parts interact to provide 5 clocks read initial access, 2 clocks read burst access, and 5 clocks write access at 40 MHz. The memory control logic is the most difficult part because of the tight timing requirements.

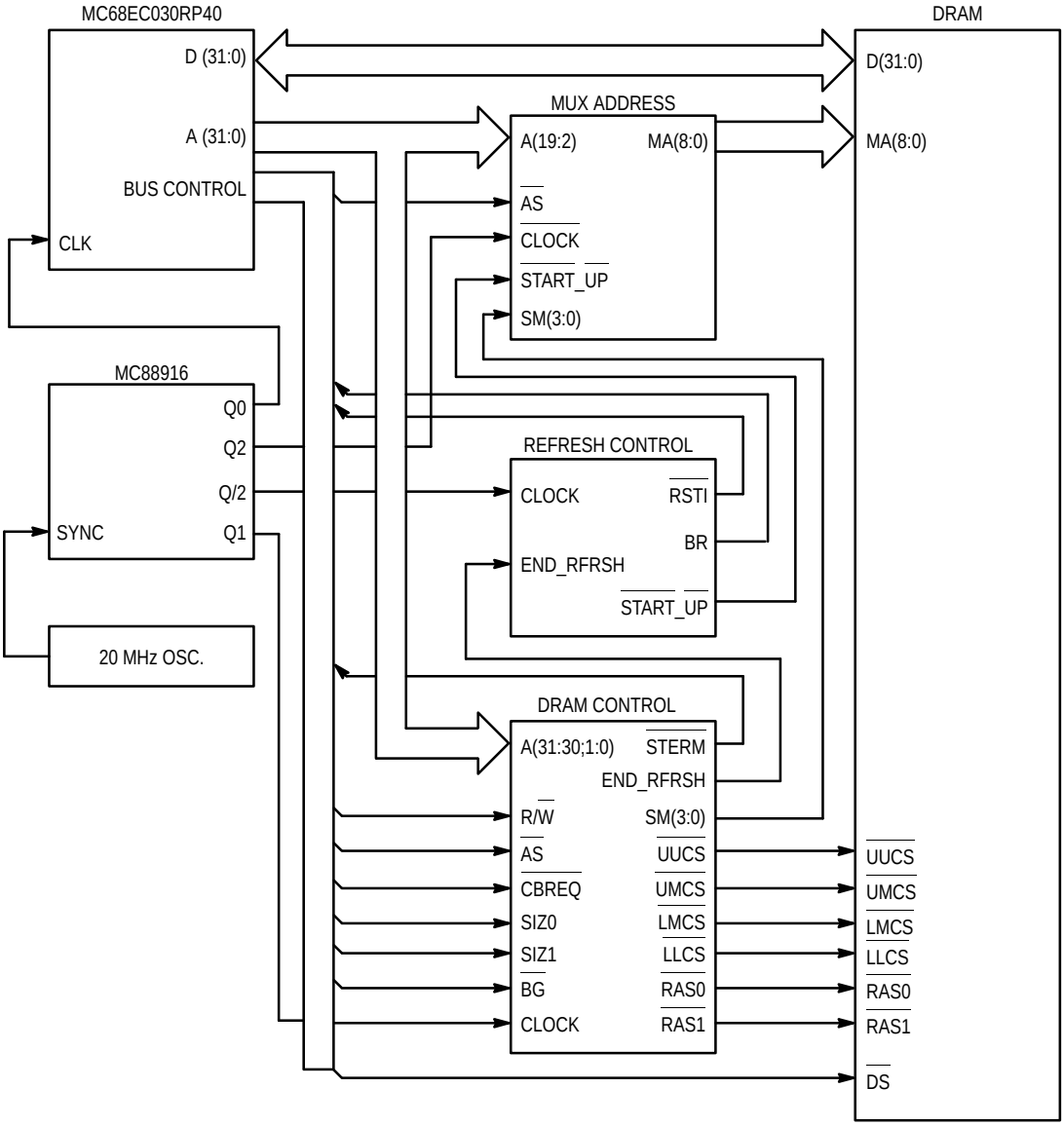

Figure 5. MC68EC030 System with DRAM Control

MEMORY CONTROL LOGIC

The memory control logic (Figure 6) has three subsections: DRAM address strobe generation, state machine generation, and access termination generation. The DRAM output enables connect to Data Strobe (DS) of the MC68EC030 The DRAMs drive the bus when RAS, CS, DS are asserted low and R/W is asserted high.

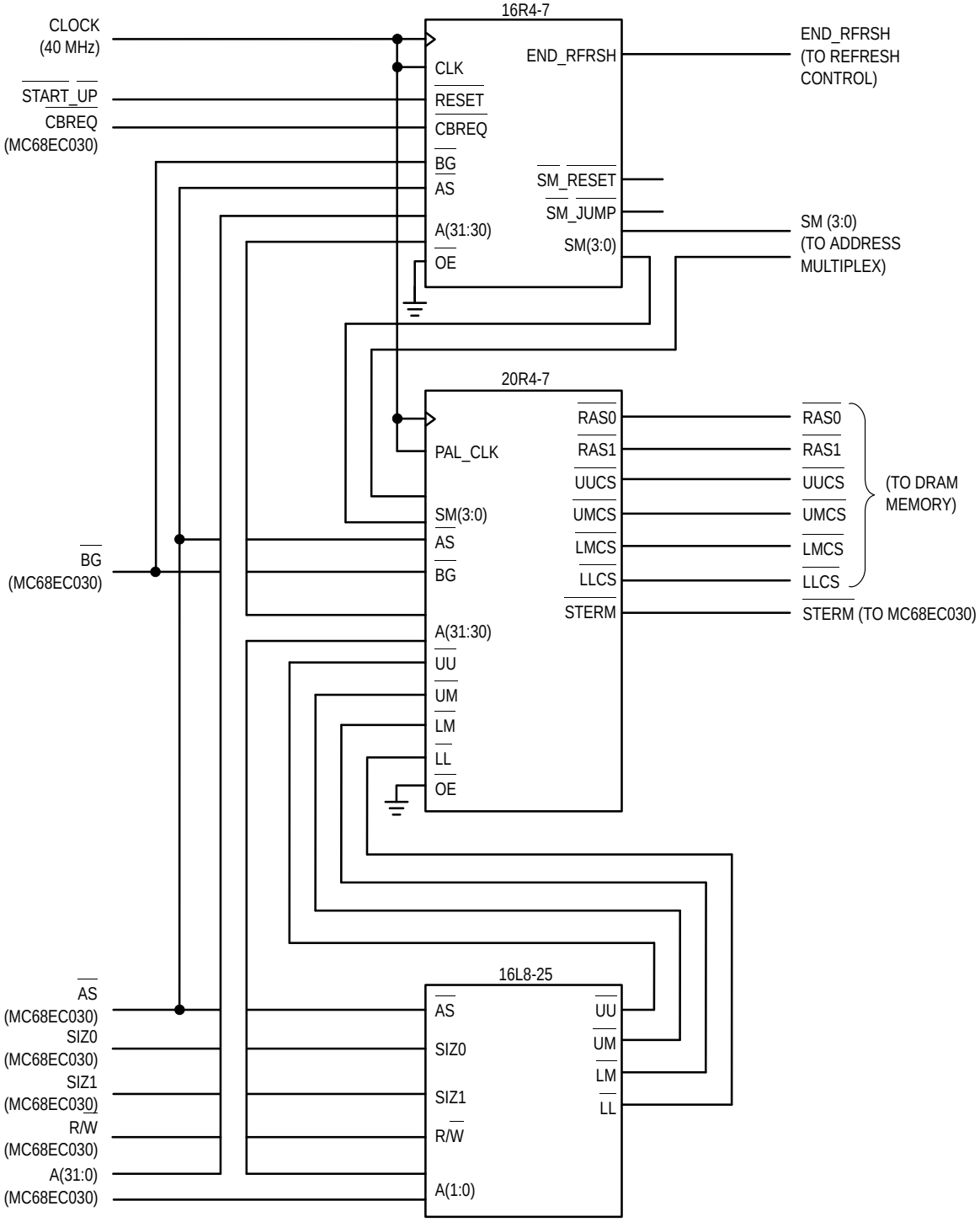

Figure 6. Memory Control Logic

The DRAMs address strobe generation provides the $\overline{RAS}$ and CS. These two signals tell the DRAMs when the row and column addresses are valid, respectively. These addresses must be valid concurrent or before their respective RAS or CS are asserted. In addition, the logic must hold the addresses for a certain amount of time after the respective strobes. The RAS is split into two signals, with identical signal generation for the RAS pair. The split decreases the capacitive loading the PALs must drive. A 7 ns PAL was used for the worst case timing of CS asserted to data valid on the first access. This provides 4 ns margin for clock skew and wire delay between DRAMs and processor. The CS uses the byte enables to determine which byte the DRAMs store on a write. The byte enables are generated in a separate PAL because of the lack of terms to decode the address and size signals.

The equation for $\overline{RAS}$ assertion consists of the $\overline{RAS}$ assertion in normal access and the $\overline{RAS}$ assertion for refreshes. For the purposes of this design, the memory is assumed to be on the second gigabyte of address space (A31 = 0, A30 = 1).

```
RASx    =   AS & A31 & A30 & SM3 & SM2                          ; RAS assert states 0 to 3
        #   AS & A31 & A30 & SM3 & SM2                          ; RAS assert states 4 to 7
        #   AS & A31 & A30 & SM3 & SM3 & SM1 & SM0              ; RAS assert state 8
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & CLK        : RAS assert state 12
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0              ; RAS assert state 15
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh RAS state 2, 3
        #   AS & BG & SM3 & SM2                                 ; refresh RAS state 4 to 7
```

The equation for $\overline{CS}$ assertion consists of the $\overline{CS}$ assertion in normal access and the $\overline{CS}$ assertion for refreshes. For the purposes of this design, the DRAM memory is assumed to be on the second gigabyte of address space (A31 = 0, A30 = 1).

```
UUCS    :=  AS & BG & SM3 & SM2 & SM0                           ; refresh CS state 1, 3
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh CS states 2, 3
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh CS state 4, 5
        #   AS & A31 & A30 & SM3 & SM2 & SM0 & UU               ; CS assert states 1, 3
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & UU               ; CS assert states 2, 3
        #   AS & A31 & A30 & SM3 & SM2 & UU                     ; CS assert states 4 to 7
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & UU         ; CS assert state 8
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & UU         ; CS assert state 12

UMCS    :=  AS & BG & SM3 & SM2 & SM0                           ; refresh CS state 1, 3
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh CS states 2, 3
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh CS state 4, 5
        #   AS & A31 & A30 & SM3 & SM2 & SM0 & UM               ; CS assert states 1, 3
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & UM               ; CS assert states 2, 3
        #   AS & A31 & A30 & SM3 & SM2 & UM                     ; CS assert states 4 to 7
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & UM         ; CS assert state 8
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & UM         ; CS assert state 12

LMCS    :=  AS & BG & SM3 & SM2 & SM0                           ; refresh CS state 1, 3
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh CS states 2, 3
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh CS state 4, 5
        #   AS & A31 & A30 & SM3 & SM2 & SM0 & LM               ; CS assert states 1, 3
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & LM               : CS assert states 2, 3
        #   AS & A31 & A30 & SM3 & SM2 & LM                     : CS assert states 4 to 7
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & LM         ; CS assert state 8
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & LM         ; CS assert state 12

LLCS    :=  AS & BG & SM3 & SM2 & SM0                           ; refresh CS state 1, 3
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh CS states 2, 3
        #   AS & BG & SM3 & SM2 & SM1                           ; refresh CS state 4, 5
        #   AS & A31 & A30 & SM3 & SM2 & SM0 & LL               ; CS assert states 1, 3
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & LL               ; CS assert states 2, 3
        #   AS & A31 & A30 & SM3 & SM2 & LL                     ; CS assert states 4 to 7
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & LL         ; CS assert state 8
        #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & LL         ; CS assert state 12
```

The state machine generation provides information on the state of the bus cycle to the rest of the control logic. The four bits of the state machine provide 16 states. The control logic does not use all the possible states. The state machine PAL includes two state machine control signals. State Machine JuMP (SMJMP) jumps the state to the closing sequence of states for a DRAM access. State Machine ReSeT (SMRST) resets the state to the initial state of a DRAM access. A 7 ns PAL meets the timing requirements of the state machine within a clock period. The state machine PAL recognizes and responds to the new state. The state machine control signals must meet the proper setup time to the next clock edge. This total sequence is 17.5 ns, 7.5 ns less than the 40 MHz clock period. A 10 ns PAL would use 27 ns for the three stages, 2 ns more than the 40 MHz clock period.

The $\overline{\text{SMRST}}$ equation holds the state machine in state 0 during reset and while the MC68EC030 is not accessing the DRAM. If the state machine is to be used for other peripherals (e.g., ROM timing), then the SMRST equation would have to be altered to enable the counting for other peripherals. For the purposes of this design, the memory is assumed to be on the second gigabyte of address space (A31 = 0, A30 = 1).

```
SMRST  =   AS & BG & SM3 & SM2 & SM1 & SM0          ; idle clock
       #   RESET                                    ; reset
       #   AS & A31                                 ; access to non DRAM
       #   AS & A31 & A30                           ; access to non DRAM
       #   SMRST & CLK                              ; hold for last half of clock
```

The $\overline{\text{SMJMP}}$ equation jumps the state at the end of a burst access, a non-burst access and a refresh. If the state machine is to be used for other peripherals (e.g., ROM timing), then the SMJMP equation would have to be altered to complete the counting for other peripherals. For the purposes of this design, the memory is assumed to be on the second gigabyte of address space (A31 = 0, A30 = 1).

```
SMJMP  =   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & CBREQ   ; end burst state 8
       #   AS & A31 & A30 & SM3 & SM2 & SM1 & SM0 & CBREQ   ; end non-burst state 2
       #   AS & BG & SM3 & SM2 & SM1 & SM0                  . end refresh state 5
```

The state machine equations are expressed both as their algorithm and their specific equatlons. If the state machine is to be used for other peripherals (e.g., ROM timing), then the SMJMP and SMRST equations would have to be altered to count for other peripherals. The state machine is a grey code counter, i.e., only one bit changes at a time. This prevents the difference in the high to low and low to high transition times from causing spikes in other decode logic. SM3 is the high order bit, SM0 is the low order bit.

```
                S00 = 0000    S01 = 0001    S02 = 0011    S03 = 0010
                S04 = 0110    S05 = 0111    S06 = 0101    S07 = 0100
                S08 = 1100    S09 = 1101    S10 = 1111    S11 = 1110
                S12 = 1010    S13 = 1011    S14 = 1001    S15 = 1000

STATE  Sn:
       IF       SMJMP        THEN          S12           ; jump to ending sequence
       ELSE IF  SMRST        THEN          S0            ; hold in reset
       ELSE     Sn + 1

SM3    :=   SMRST                                        ; state reset
       #    SM3 & SM2 & SMRST & SMJMP                    ; state 0 to 3
       #    SM3 & SM2 & SM1 & SMRST & SMJMP              ; state 4, 5
       #    SM3 & SM2 & SM1 & SM0 & SMRST & SMJMP        ; state 7

SM2    :=   SMRST                                        ; state reset
       #    SM3 & SM2 & SM1 & SM0 & SMRST & SMJMP        ; state 0
       #    SM3 & SM2 & SM0 & SMRST & SMJMP              ; state 1, 2
       #    SM3 & SM2 & SM1 & SM0 & SMRST & SMJMP        ; state 11
       #    SM3 & SM2 & SMRST & SMJMP                    ; state 12 to 15
       #    SMJMP                                        ; jump to state 12

SM1    :=   SMRST                                        ; state reset
       #    SM3 & SM2 & SM1 & SM0 & SMRST & SMJMP        ; state 5
       #    SM3 & SM2 & SM1 & SM0 & SMRST & SMJMP        ; state 6
       #    SM1 & SM0 & SMRST & SMJMP                    ; state 0, 7, 8, 15
       #    SM3 & SM2 & SM0 & SMRST & SMJMP              ; state 13, 14

SM0    :=   SMRST                                        ; state reset
       #    SM3 & SM2 & SM1 & SMRST & SMJMP              ; state 2, 3
       #    SM3 & SM2 & SM1 & SMRST & SMJMP              ; state 6, 7
       #    SM3 & SM2 & SM1 & SMRST & SMJMP              ; state 10, 11
       #    SM3 & SM2 & SM1 & SMRST & SMJMP              ; state 14, 15
       #    SMJMP                                        ; jump to state 12
```

The $\overline{\text{STERM}}$ signals the termination of a transfer to or from the processor based on the state machine state. The STERM is also connected to CBACK. This means that a burst request by the MC68ECC30 is acknowledged by the DRAM. If other 32-bit peripherals are not cached, then CBACK can be connected to RAS. The disadvantage to this is the increased capacitance load that RAS must drive. To convert this design to a non-bursting DRAM interface, pull the CBREQ inputs into the PALs high and do not connect STERM to CBACK. If the access was not a burst request, the MC68EC030 ignores the CBACK. The setup and hold times of the MC68EC030 on STERM requires a 7 ns PAL. The STERM has a 2 ns set up and 6 ns hold requirement for each rising edge. The control logic asserts STERM when the state

machine count is correct and the 40 MHz clock is low. The control logic continues to assert STERM while the clock is high. This means the STERM is asserted and negated off the falling edge of the clock.

```
STERM  =  AS & A31 & A30 & CLK & SM3 & SM2 & SM1 & SM0   ; last half of state 2
       #  AS & A31 & A30 & CLK & SM3 & SM2 & SM1 & SM0   ; last half of state 4
       #  AS & A31 & A30 & CLK & SM3 & SM2 & SM1 & SM0   ; last half of state 6
       #  AS & A31 & A30 & CLK & SM3 & SM2 & SM1 & SM0   ; last half of state 8
       #  STERM & CLK                                    ; keep first half of clock
```

ADDRESS MULTIPLEXING

The control logic handles the address multiplexing in two parts, row and column address multiplexing and burst address multiplexing. The DRAMs use multiplex address bits MA0 and MA1 for the bursting address. These pass through a PAL and multiplex A4 and A5 for the row address with A2 and A3 for the column address. The control logic increments MA0 and MA1 from A2 and A3 modulo 4 (00,01,10,11,00,...) for the four long words of the burst. The DRAM holds the data valid for 5 ns after the column address change for the burst. This allows the MC68EC030 data hold time to be easily met. Figure 7 shows the address multiplexing logic.

The MA0 and MA1 equations have three parts. A combinatorial equation that drives the DRAM, a registered equation that selects the next address, and a registered equation that says when to change the address.

```
MA0    =  ROW/COL & A4                          ; row address
       #  A2 & CLK & SM3 & SM2 & SM1 & SM0      ; state 1 col. addr.
       #  ROW/COL & MA0 & CLK                   ; column address hold
       #  ROW/COL & NXTA0 & CLK                 ; column address change

MA1    =  ROW/COL & A5                          ; row address
       #  A3 & CLK & SM3 & SM2 & SM1 & SM0      ; state 1 col. addr.
       #  ROW/COL & MA1 & CLK                   ; column address hold
       #  ROW/COL & NXTA1 & CLK                 ; column address hold

NXTA0  :=  CHANGE & MA0                         ; next lw col. addr.
       #  SM3 & SM2 & SM1 & A2                  ; 1st col. addr.., state 0, 1
       #  CHANGE & NXTA0                        ; hold column address

NXTA1  :=  CHANGE & MA0 & MA1                   ; next lw col. addr.
       #  CHANGE & MA0 & MA1                    ; next lw col. addr.
       #  SM3 & SM2 & SM1 & A3                  ; 1st col. addr.., state 0, 1
       #  CHANGE & NXTA1                        ; hold column address

CHANGE :=  SM3 & SM2 & SM1 & SM0                ; last half of state 1
       #  SM3 & SM2 & SM1 & SM0                 ; last half of state 3
       #  SM3 & SM2 & SM1 & SM0                 ; last half of state 5
       #  SM3 & SM2 & SM1 & SM0                 ; last half of state 7
```

The rest of the DRAM address lines are passed through MC74F258 multiplexers to switch between the row and column addresses. The row address must be valid concurrently or before the RAS generation and held for 10 ns. The column address must be valid concurrently or before the CS generation. There is a 5 ns data hold from the changing of the column address. The ROW/COL signals the change from row to column address to the multiplexers.

```
ROW/COL  =  SM3 & SM2 & SM1 & SM0 & AS          ; assert for col. addr. state 1
         #  ROW/COL & AS                        ; hold col. addr..
```

ONE MEGABYTE OF MEMORY

This is by far the easiest part of the design. It is eight MCM514258A 80 ns static column 256K x 4 DRAMs. The RAS, CS, and write enable signals are driven by the memory control logic. The address multiplexing logic drives the addresses. The output enable (G) is connected to the Data Strobe (DS) of the MC68EC030 to prevent bus contention on a read access followed immediately by a write access. The data bus must three-state within 25 ns of DS negation on a read access. The data must remain valid on a read for at least 0 ns after DS negating. The MCM514258A-80 three states its bus 0 to 20 ns after G going high. On a write, all byte lanes will be driven by the MC68EC030, but only certain byte lanes will have valid data. The valid bytes are selected by the CS that decode the address and size pins. Figure 8 shows the DRAM.

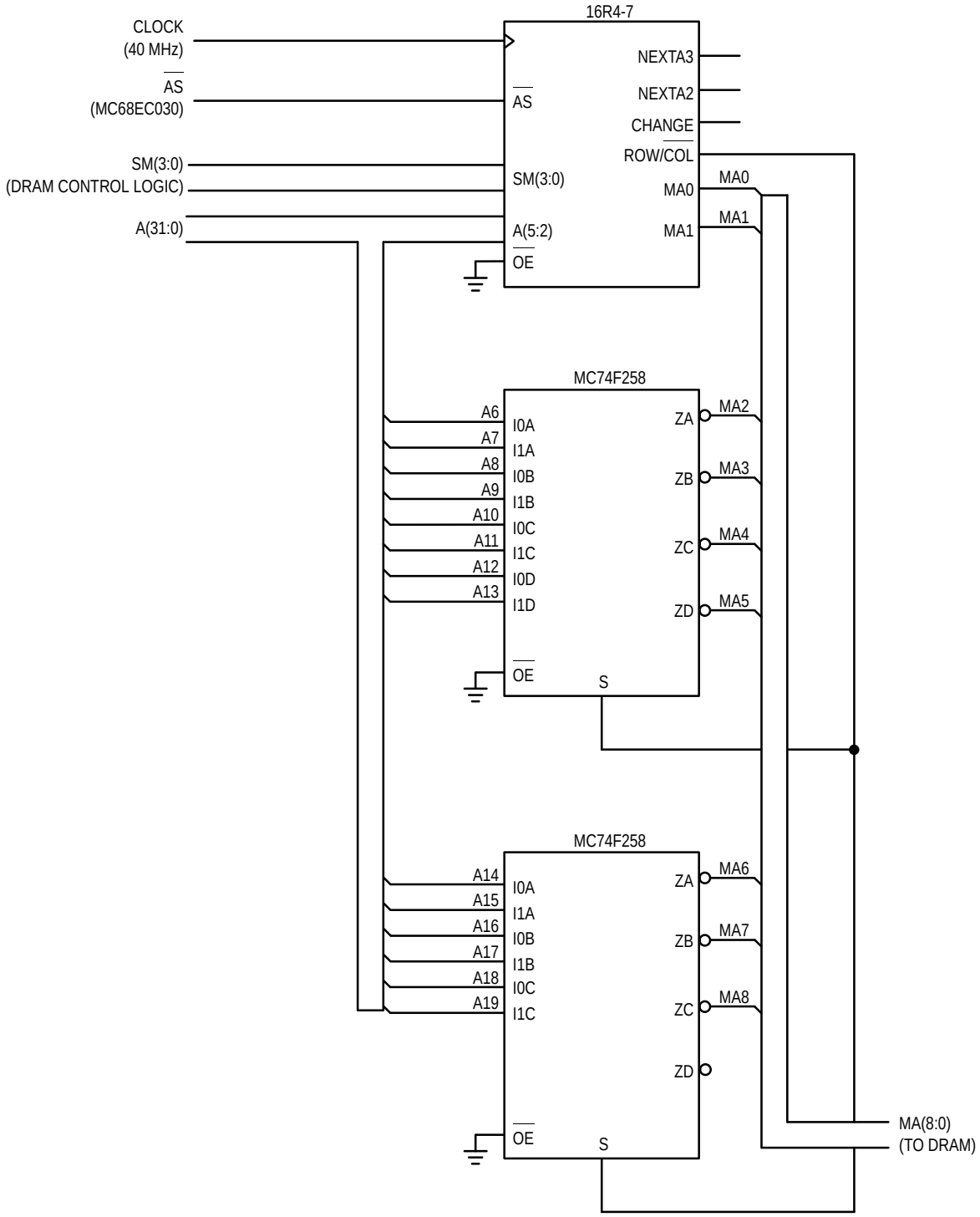

Figure 7. Address Multiplexing Logic

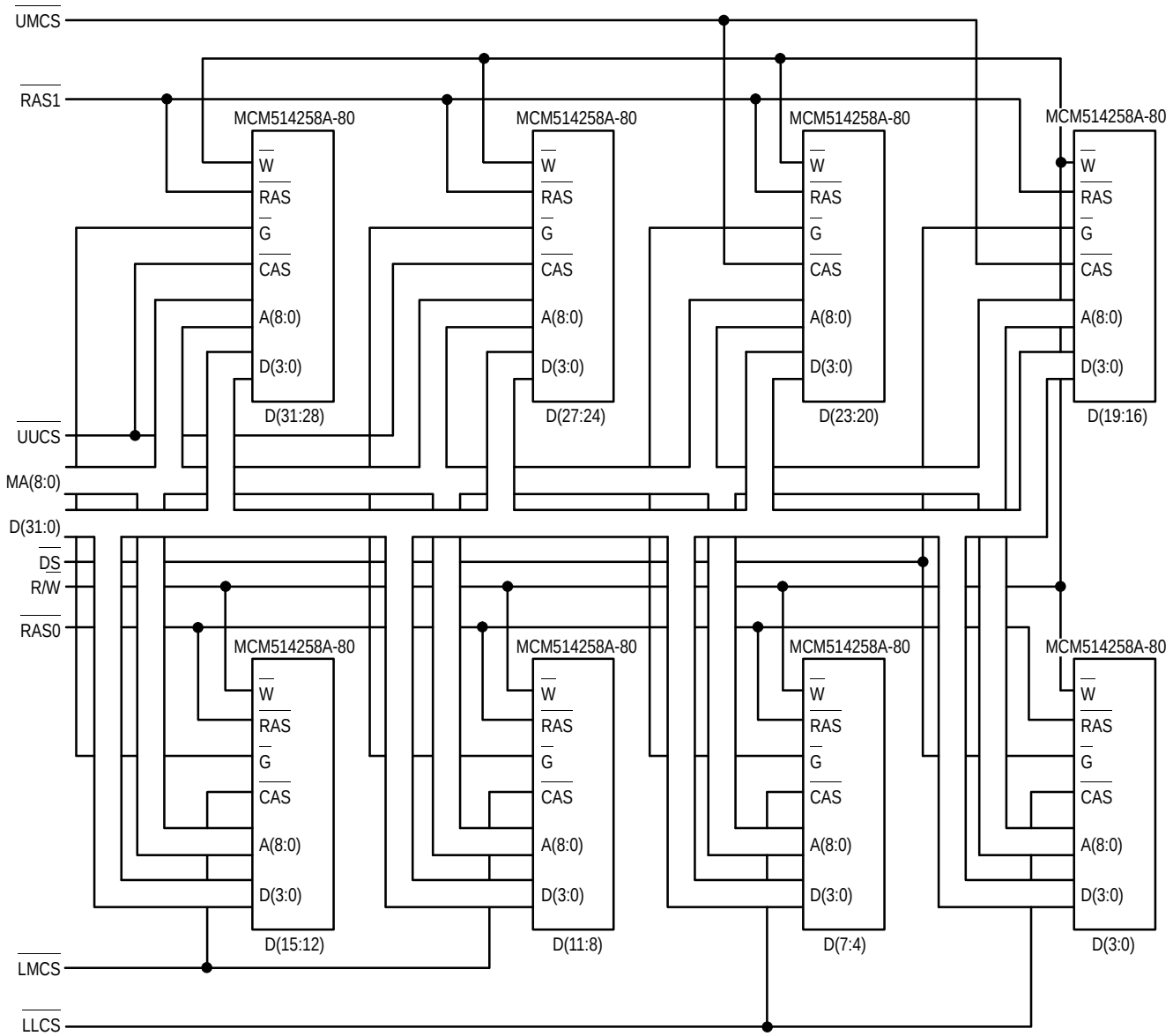

Figure 8. DRAM Control Logic

A larger memory is obtained by using eight 1M x 4 DRAMs (4 megabytes), thirty-two 1M x 1 DRAMs (4 megabytes) or thirty-two 4M x 1 DRAMs (16 megabytes). AS and DS signals are both used to minimize loading on either signal when using thirty-two DRAMs. Alternatively, the ROW/COL signal is attached to Bus Grant ACKnowledge (BGACK) of the MC68EC030 to provide a guaranteed bus inactive cycle between each DRAM cycle If ROW/COL is connected to BGACK, then the output enables can be grounded and the DRAM will three-state when CS negates.

REFRESH CONTROL LOGIC

The MCM514258A requires all 512 rows to be refreshed every 8 ms. This corresponds to one row every 15.625 µs. The refresh control logic signals every 12.8 µs that a refresh cycle is needed. The slightly higher refresh rate of the control logic provides margin to ensure the data is not lost and minimizes the amount of refresh logic required. In order for the refresh to work correctly, there should not be any access longer than 55 clocks. Since the longest read-modify-write for DRAM access would be 13 clocks, the access limit is dependent on other devices. The reason for the limit is the bus request, when signaling a required refresh, is asserted for up to 64 clocks. With bus arbitration, the refresh operation requires 9 clocks. The refresh, with its associated bus mastership, must occur within the remaining 55 clocks. A refresh rate of 12.8 µs means the DRAMs are not available to the MC68EC030 about 1.5% of the time. Since the MC68EC030 is not halted during this time and the internal caches are still available to the integer unit, this corresponds to less than 1.5% reduction in performance.

The refresh control logic consists of a MC74LS393 dual four bit counter. The first counter is clocked at 20 MHz, and in turn clocks the second counter. The four outputs of the second counter are NANDed through half of a MC74LS20 dual four input NAND gate to generate the Bus Request (BR). The MC68EC030 responds with a Bus Grant (BG). The state machine and DRAM control PALs use the BG signal to indicate that a refresh cycle should occur. At the end of the refresh cycle, the END_ReFReSH (END_RFRSH) clears the MC74LS393 counter to begin the count to the next refresh. Figure 9 shows the refresh control logic.

```
END_RFRSH     =     BG & AS & SM3 & SM2 & SM1 & SM0     ; end refresh state 4
              #     BG & AS & SM3 & SM2 & SM1 & SM0     ; end refresh state 5
              #     BG & AS & SM3 & SM2 & SM1 & SM0     ; end refresh state 12
```

The DRAM requires eight active cycles to establish the proper bias voltage, 200 μs after power-up. Two power-up reset signals provide this DRAM initialization. The first one negates 200 μs after power-up and the second one, to the processor, negates 350 μs after power-up. This ensures that the normal DRAM refresh logic provides at least the 8 active cycles required and guarantees the MC68EC030 40 MHz power-up reset timing requirement.

PERFORMANCE

The DRAM design shown here would provide approximately 2/3 to 3/4 of the performance of a no-wait-state-SRAM design at a significantly lower price. The DRAM design has a 5 clock initial access, 2 clock burst access and 2 clock recovery time. The recovery time affects the MC68EC030 only on back to back transfers (i.e., access followed immediately by another access, without any intervening bus idle clock periods). The MC68EC030 is not very likely to make back to back read accesses after a burst because of the on-board caches. Faster DRAMs would not increase the performance directly, just provide more setup margin. This increased margin could be used by multiplexing two banks of DRAM. This would double the amount of memory, provide 5 clock initial access, 1 clock burst access and 2 clock recharge time. One bank would be designated as even and the other odd. While the burst to the MC68EC030 would remain four long words, each bank would only provide two long words. The select between the two banks would correspond to the lower order column address (MA0) of the multiplexed address to the DRAM.

CONCLUSION

The DRAM interface to the MC68EC030 shown provides economical memory with relatively high performance. While slightly higher performance could be achieved through SRAM, the cost per byte is significantly higher.

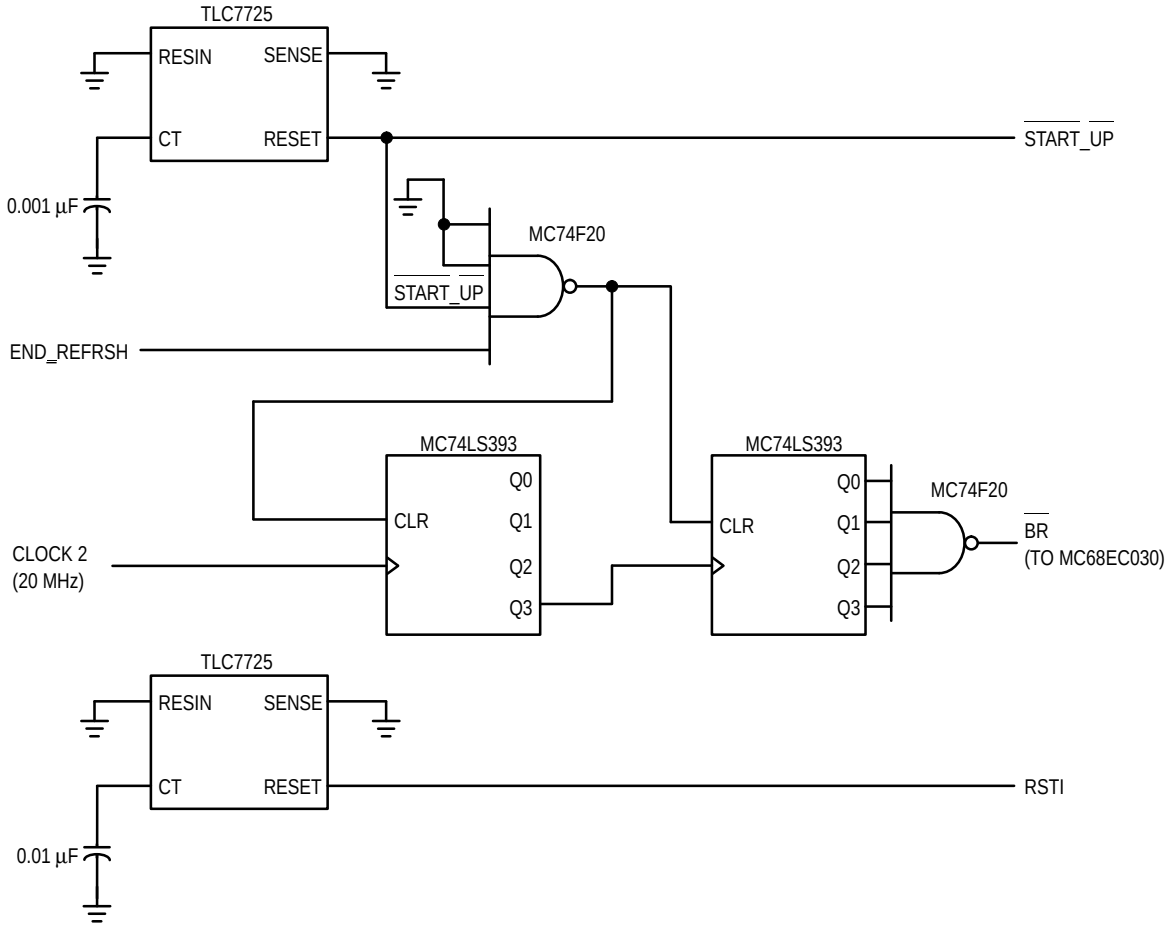

Figure 9. Refresh Control Logic

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.