intel®

AB-31

APPLICATION BRIEF

The 80C186XL/80C188XL Integrated Refresh Control Unit

GARRY MION
ECO SENIOR APPLICATIONS ENGINEER

November 1994

Order Number: 270520-003

Information in this document is provided in connection with Intel products. Intel assumes no liability whatsoever, including infringement of any patent or copyright, for sale and use of Intel products except as provided in Intel's Terms and Conditions of Sale for such products.

Intel retains the right to make changes to these specifications at any time, without notice. Microcomputer Products may have minor variations to this specification known as errata.

*Other brands and names are the property of their respective owners.

†Since publication of documents referenced in this document, registration of the Pentium, OverDrive and iCOMP trademarks has been issued to Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683

THE 80C186XL/80C188XL INTEGRATED REFRESH CONTROL UNIT

CONTENTS

The 80C186XL and 80C188XL incorporate a special control unit that integrates address and clock counters which, along with the **B**us **I**nterface **U**nit (BIU), facilitates dynamic memory refreshing. Refreshing is an operation required by dynamic memory to ensure data retention.

Dynamic memory refreshing can be controlled using anything from an exotic memory controller to a simple timer along with a DMA controller. In fact, the 80186 device accomplishes the task memory refreshing by using one of the internal timer/counters and a DMA channel. However, doing this meant that desirable internal functions were no longer available to do more useful work.

Dynamic memory, unlike static or non-volatile memory, always require some form of a memory controller to enable read and write operations. Therefore, even the most basic dynamic memory interface has a minimum set of support logic. The advent of programmable logic and highly integrated dynamic memory has made the job of designing a memory controller somewhat straightforward. However, directly supporting memory refresh can still complicate many controller designs.

The designer of a memory controller must take into account CPU-versus-refresh arbitration and must provide a mechanism to generate periodic refresh requests. Most dynamic memory devices now contain internal refresh address counters which eliminate the need for external refresh address generation. However, such devices tend to complicate a memory controller design. The 80C186XL simplifies dynamic memory controller design by integrating a refresh mechanism into the operation of the CPU.

This application brief is not intended to be a discussion of dynamic memory controller design. Instead, it will concentrate on the operation of the Refresh Control Unit of the 80C186XL, and how it can help simplify a memory controller.

The discussions on the following pages apply to BOTH the 80C186XL and 80C188XL except where noted.

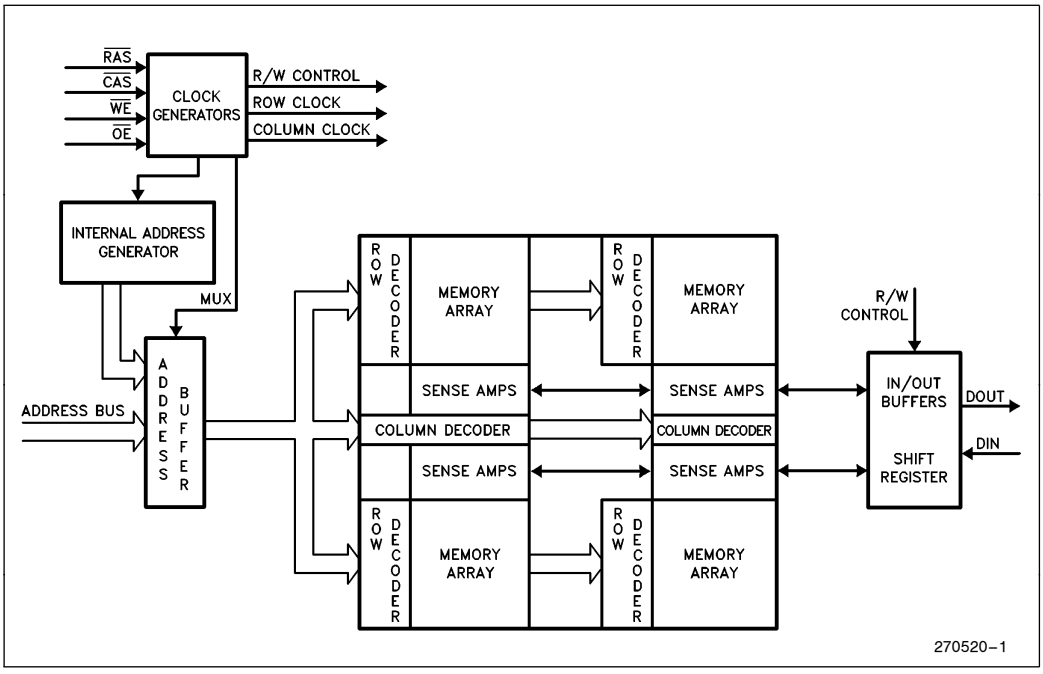


Figure 1. Random Access Memory Device

UNDERSTANDING DYNAMIC MEMORY

Before explaining how memory refreshing is accomplished, some understanding of a **D**ynamic **R**andom **A**ccess **M**emory (DRAM) device is needed. Figure 1 shows a simplified block diagram of a DRAM device, while a block diagram of a typical dynamic memory controller is shown in Figure 2.

The typical DRAM memory array is built as a matrix. Any bit or cell in the memory array is accessed by specifying a unique row and column address. As shown in Figure 1, the row and column addresses are multiplexed through one set of address inputs. Multiplexing the address inputs helps reduce the number of pins required to support large memory arrays. For instance, adding only one address bit will result in a memory array 4 times as large.

Two control lines, $\overline{RAS}$ and $\overline{CAS}$, are used to strobe an address into the memory chip. Figure 3 illustrates a timing diagram for a typical memory read access and the relationship between the $\overline{RAS}$ and $\overline{CAS}$ signals. The signal $\overline{MUX}$ controls which half of the address is presented to the memory devices. After generating the row address strobe ($\overline{RAS}$), the decoder selects a row of memory cells whose data value will be detected by a Sense Amplifier. The Sense Amplifier then presents the data to the column decoder. **Note that all cells associated to a row get accessed.** The fact that all cells within a row are accessed will be used later to explain why only the $\overline{RAS}$ portion of the memory address is required to refresh a device.

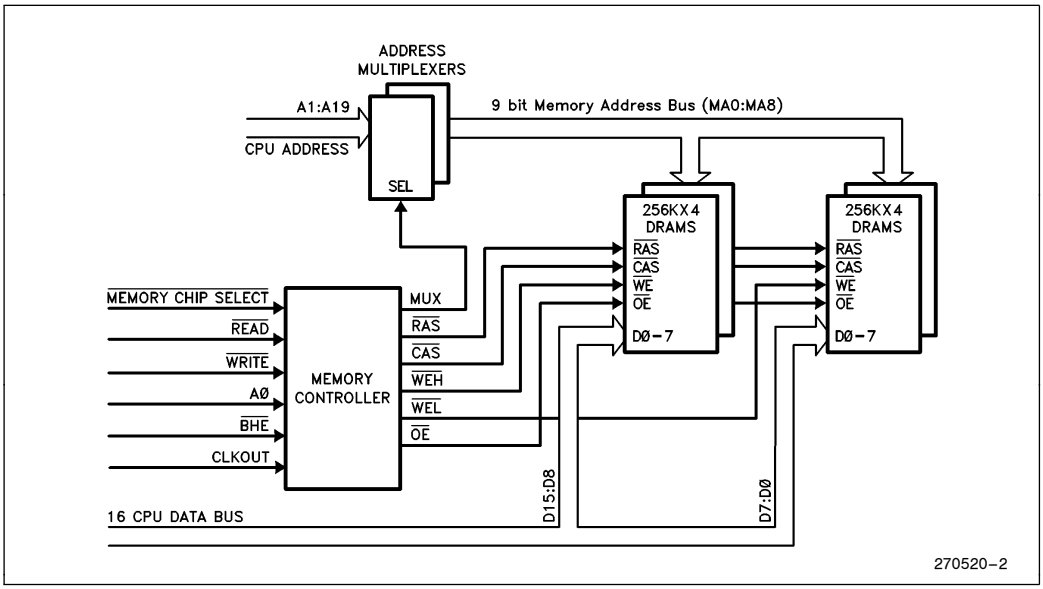


Figure 2. Minimum Configuration Memory Controller

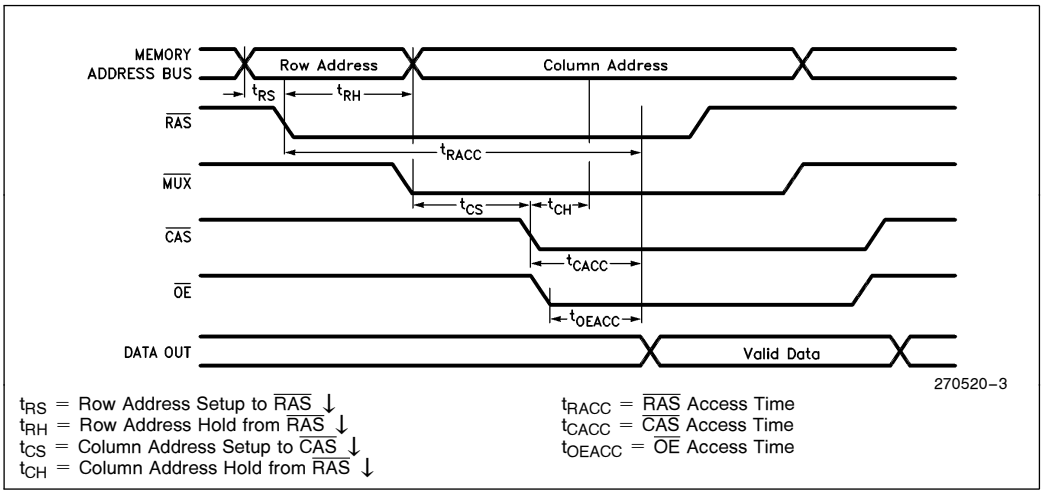


Figure 3. DRAM Signal Timings

When the column address is strobed, it is here that only one of the memory cells is selected. The memory cell will either be written to or read from depending on the the control signals $\overline{WE}$ and $\overline{OE}$ respectively. Since data from one entire row is presented to the column decoder, it is possible (on some devices) to simply cycle through column addresses to access additional data. The basic idea, however, is that two sets of addresses are required to access a memory cell within a memory array. Furthermore, specifying a single row address internally accesses all memory cells within that row.

The minimum memory controller interface consists of a sequencer and an address multiplexer. The sequencer is responsible for generating the correct control signals: $\overline{RAS}$; $\overline{CAS}$; $\overline{MUX}$; $\overline{WE}$; $\overline{OE}$. The address multiplexer logic is responsible for translating the processor address bus to the memory address bus. These two pieces of logic can exist in any form, from simple TTL gates to single chip solutions. However, what is missing from the simplified memory controller is a mechanism to perform memory refresh.

UNDERSTANDING MEMORY REFRESH

As indicated earlier, dynamic memory needs to be refreshed in order to maintain its data. Refreshing is accomplished whenever a memory cell is **accessed**. It is not necessary to read a memory location and then write the value back in order to refresh a memory cell. Simply cycling through a complete set of row addresses is all that is required. Remember, since a row accesses all memory cells associated to it, accessing all rows will access all the cells within the device.

Referring back to Figure 2, the 9 address bits presented to the memory devices are multiplexed from the 18 bits of address generated by the 80C186XL. In the design, address bits A1-A9 are presented during $\overline{RAS}$, while address bits A10-A18 are presented during $\overline{CAS}$. Note that address bit A0 is not used because the memory array is organized as word wide; A0 along with $\overline{BHE}$ are used to select one or both of the bytes within a word.

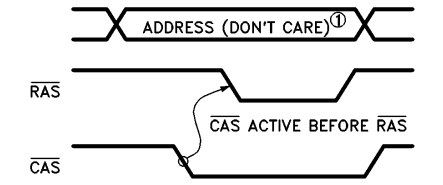
Cycling through row addresses is the only requirement needed to refresh a DRAM device. Using the example in Figure 2, 9 bits of address are needed. Nine bits represent 512 unique addresses, and the only requirement is that each unique address be regenerated every 8 ms (maximum refresh rate for most devices with 512 rows). An 8 ms refresh interval divided by 512 addresses results in an average refresh cycle rate of 15.625 microseconds. Therefore, every 15.625 microseconds a mechanism must exist that will access the DRAM device, each time presenting a new row address. Any rate faster than 15.625 microseconds is acceptable, but significantly faster times have the potential of decreasing memory performance.

WAYS TO REFRESH A MEMORY DEVICE

For most dynamic memory devices, there are several ways in which a refresh cycle can be run. The first and simplest way is to generate memory read cycles every 15.6 microseconds. Each new memory read cycle would generate a unique address. When refreshing is accomplished using memory read cycles, the memory controller is simplified. Only the basic control signals need to be generated, which are the minimum needed to access the memory anyway. Simplicity is, however, accompanied by one drawback; bus overhead. Using memory reads to perform DRAM refreshing means that one bus cycle every 15.6 microseconds is wasted. When operating at very slow speeds, a wasted bus cycle might appear to be significant. But if a bus cycle takes only, say, 320 nanoseconds to complete, running a refresh cycle every 15.6 microseconds represents a two percent hit in bus performance.

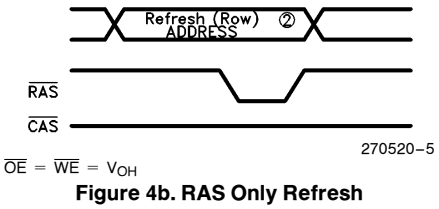
A second method relies on the fact that most dynamic memory devices now have built in refresh address mechanisms. DRAM refreshing can be accomplished by generating $\overline{CAS}$ before $\overline{RAS}$ signaling (see Figure 4a). This method requires that an external signal generate a periodic request to the DRAM controller to initiate the refresh cycle. A method similar to $\overline{CAS}$ before $\overline{RAS}$ refreshing is hidden refresh. Figure 4c illustrates the timing involved to perform hidden refresh. No request logic is needed, since the memory access itself is what initiates the refresh cycle. However, constant memory accessing is required in order to maintain refreshing. Once accessing stops, refreshing stops. Both of the methods described have the advantage of not consuming bus bandwidth, but require the memory controller to handle the somewhat different (from normal memory accessing) signaling requirements.

A final method is to implement a discrete design that supports refresh control and refresh address generation. The circuit details are shown in Figure 4b. A discrete design allows the most design flexibility and can be tailored to meet any system-to-memory interfacing requirements.



$\overline{OE} = \overline{WE} = V_{OH}$

Figure 4a. $\overline{CAS}$ before $\overline{RAS}$ Refresh



$\overline{OE} = \overline{WE} = V_{OH}$

Figure 4b. RAS Only Refresh

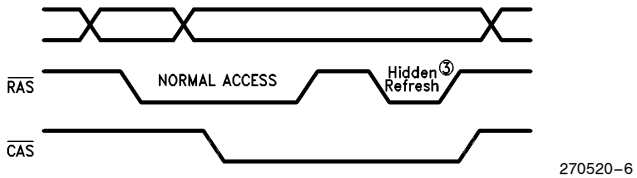


Figure 4c.

NOTES:
1. Refresh address provided internal to memory device.
2. Refresh address presented external to memory device.
3. Refresh address generated internally, and cycle does not effect memory access in progress (i.e. hidden).

Figure 4. Alternate Refreshing Methods

There are other methods available, most of which involve single-chip dedicated memory controllers. However, any memory controller design that performs the function of refreshing either directly or through external support circuitry has one major concern; arbitration between the refresh cycle and a normal memory access. The best way to make the operation of the DRAM memory controller a true slave to the operation of the CPU is to include refreshing as part of the functionality of the CPU. By offloading the task of memory refreshing onto the CPU, the memory controller can be simplified and dedicated to the duty of DRAM interfacing.

The idea that the 80C186XL refresh cycle is simply a memory read means that the dynamic memory control logic does not need to differentiate between refresh cycles and normal memory read cycles. This simplifies the design of the memory controller. There are no special signaling requirements needed, and $\overline{RAS}$ only refreshing (for low-power designs) can be easily accommodated. Further, since the request is generated internally and synchronous with the operation of the BIU, no special external logic needs to detect when a refresh cycle conflicts with a CPU access.

80C186XL REFRESH CONTROL FEATURES

The **R**efresh **C**ontrol **U**nit (RCU) of the 80C186XL consists of a 9-bit address counter, a 9-bit down counter, and support logic. The block diagram can be seen in Figure 5.

The 9-bit address counter is controlled by the BIU and used whenever a refresh bus cycle is executed. Thus, any dynamic memory device whose refresh address requirement does not exceed nine bits can be directly supported by the 80C186XL. A special register has been defined to allow the base (starting) address of the refresh memory region to be specified. This base address can be located on any 4 kilobyte boundary. Furthermore, if this refresh base address overlaps any of the defined chip select regions, the chip select defined for that region will go active.

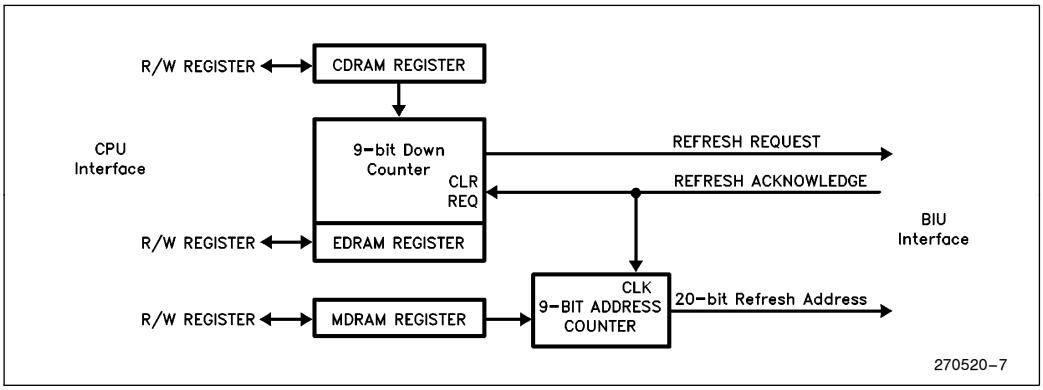


Figure 5. Refresh Control Unit Block Diagram

The 9-bit down counter initiates a refresh request. When the counter decrements to 1 (it decrements every clock cycle), a refresh request is presented to the BIU. When the bus is free, the BIU will run the refresh (memory) bus cycle. Note that since a refresh bus cycle is executed by the BIU, the faster refresh cycles are requested the greater the impact on bus performance. Referring back to the discussion of request rates, the maximum refresh period is typically 15.6 microseconds. With the 80C186XL operating at 12.5 MHz, this represents a refresh bus impact of only 2%. However, at 5 microseconds the bus impact is 15%. Therefore, the refresh request rate should be tailored to meet the needs of the dynamic memory and the system. The 80C186XL provides flexibility by allowing the request rate to be programmable in 80 ns steps (at 12.5 MHz).

To facilitate low power designs, the refresh bus cycle provides a mechanism whereby the dynamic memory devices can be turned off during refresh accesses. Low power control is accomplished by driving both address bit A0 and the control signal $\overline{BHE}$ to a high level. Essentially an invalid bus access condition exists, since A0 and $\overline{BHE}$ are used to indicate which half of the data bus is being accessed. When both are high during the access, the indication is that neither half of the bus is being used for the data transfer. This is acceptable for refresh bus cycles since no data is actually being transferred. If the memory controller takes advantage of this condition, the output enables of the dynamic memory devices (as well as the $\overline{CAS}$ strobe) can be disabled during refresh bus cycles, providing overall lower power consumption.

PROGRAMMING CHARACTERISTICS OF THE REFRESH CONTROL UNIT

A block of control registers are defined in the **P**eripheral **C**ontrol **B**lock (PCB) that define the operating characteristics of the refresh control unit (refer to Figure 5). These registers are only accessible when the 80C186XL is operating in enhanced mode. When in compatibility mode, the 80C186XL will ignore any reads or writes to the RCU registers.

The three registers associated with the refresh unit (MDRAM, CDRAM, EDRAM) provide the following features:

1) Enable/disable refresh unit
2) Establish a refresh request rate
3) Establish a refresh memory region
4) Examine the refresh down counter

It is not necessary to program any of these registers in a specific sequence, although the refresh request rate and refresh base address registers should be programmed before the refresh unit is enabled.

Programming the Memory Partition Register

The MDRAM register (Figure 6) is used to define address bits A13 through A19 of the 20 bit refresh address. This essentially establishes a memory region which will be accessed during refresh bus cycles. Typically, the refresh memory region will overlap a chip select that is used to access the dynamic memory. Overlapping the refresh memory region with a chip select memory region, means no additional external hardware is needed to support refresh bus cycles since it essentially operates the same as memory read cycles. When the 80C186XL is reset, the MDRAM register is initialized to zero.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MDRAM:	M6	M5	M4	M3	M2	M1	M0	0	0	0	0	0	0	0	0	0

bits 0–8: Reserved, should be written as a 0 to maintain future compatibility, will be read back as 0.
bits 9–15: M0–M6, are used to define address bits A13–A19 (respectively) of the 20-bit memory refresh address. These bits are set to 0 on RESET.

Figure 6. MDRAM Register Format

Figure 7 illustrates how the refresh address is generated. Address bits A10-A12 are not programmable and are always driven to a zero during a refresh bus cycle. Address bits A1 through A9 are derived by a 9-bit linear-feedback shift counter. The address counter is not ascending or contiguous, meaning that the counter does not start at 0 and increment to 511 before resetting back to 0. For refreshing purposes, it is not important that the address be contiguous and count up or down. Rather, the only requirement is that all combinations of the 512 addresses be cycled through before being repeated. Equation 1 provides the state definition of the 9-bit refresh address counter and can be used to determine the exact counting sequence. Figure 8 illustrates the gate logic used to create such a counter.

There are no limitations placed on the programming of the MDRAM register, but be aware that any chip select **memory** region that overlaps the address established by the MDRAM register will be activated during refresh bus cycles. Therefore, the register should be programmed to correspond to the chip select address that is activated for the dynamic memory partition.

Programming the Refresh Clock Register

The CDRAM register (Figure 9) is used to define the rate at which refresh requests will be internally generated. The CDRAM register is used to maintain the starting value of a down counter, which decrements each falling edge of CLKOUT. When the counter decrements to 1, a refresh request is generated and the counter is again loaded with the value contained in the CDRAM register. Initially, however, the contents of the CDRAM register is loaded into the down counter when the enable bit in the EDRAM register set. Thus, if the CDRAM register is changed, the new value will take effect when either the down counter reaches 1 and reloads itself, or whenever the E bit is written to a 1 (this is true whether the bit was previously set or not). When the 80C186XL is reset, the CDRAM register is initialized to zero. A value of zero in the CDRAM register is used to indicate the maximum count rate of 512 clocks.

Address Bit	**19**	**18**	**17**	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Physical Refresh Address	M6	M5	M4	M3	M2	M1	M0	0	0	0	C_{A8}	C_{A7}	C_{A6}	C_{A5}	C_{A4}	C_{A3}	C_{A2}	C_{A1}	C_{A0}	1

Bit 0: Always driven to a 1 (Logic High). This is true for both the 80C186XL/80C188XL.
Bits 1–9: C_{A0}–C_{A8}, are generated by the 9-bit Linear-Feedback shift counter.
Bits 10–12: Always driven to a 0 (Logic Low).
Bits 13–19: M0–M6, are defined by the MDRAM Register.

Figure 7. Physical Refresh Address Generation

$$
\begin{aligned}
C_{A0} &\leftarrow C_{A1}\\
C_{A1} &\leftarrow C_{A2}\\
C_{A2} &\leftarrow C_{A3}\\
C_{A3} &\leftarrow C_{A4}\\
C_{A4} &\leftarrow C_{A5}\\
C_{A5} &\leftarrow C_{A6}\\
C_{A6} &\leftarrow \text{If } (C_{A1}\text{–}C_{A6} = 111111B),\\
&\qquad \text{then } C_{A6} = \text{Inverted } C_{A0}\\
&\qquad \text{else } C_{A6} = ((C_{A0} \text{ .XOR. } C_{A1}) \text{ .XNOR. } (C_{A2} \text{ .XOR. } C_{A3}))\\
C_{A7}, C_{A8} &\leftarrow \text{If } (C_{A0} - C_{A6} = 0111111B)\\
&\qquad\qquad \text{then } C_{A7}, C_{A8} = C_{A7}, C_{A8} + 1\\
&\qquad\qquad \text{else } C_{A7}, C_{A8} = C_{A7}, C_{A8}
\end{aligned}
$$

Equation 1. Refresh Counter Operation

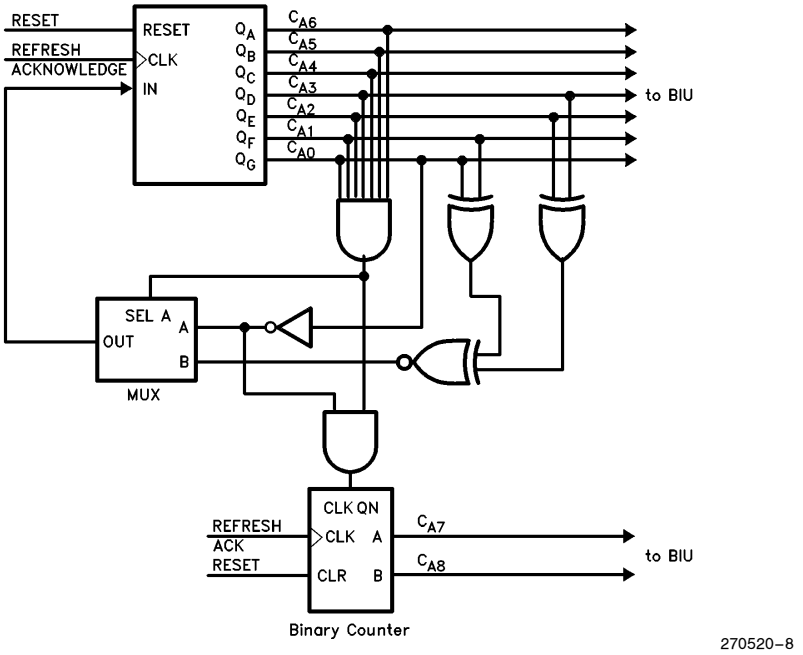


Figure 8. Logic Representation of Refresh Address Counter

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CDRAM:	0	0	0	0	0	0	0	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

Bits 0–8: C_0–C_8, define the number of CLKOUT cycles between each Refresh Request
0 = 512, 1 = 1, 2 = 2, . . . These bits are set to 0 on RESET.
Bits 9–15: Reserved, should be written as 0 to maintain future compatibility, will always be read back as 0.

Figure 9. CDRAM Register Format

The equation shown in Figure 10 can be used to determine the value of the CDRAM register needed to establish a desired refresh request rate. **Note that the equation is based on the internal operating frequency of the 80C186XL**. Therefore, the request rate is effected by any change in operating frequency. Modification of the operating frequency can occur in two ways: modifying the input clock or entering power-save mode. There is no upper limitation as to the frequency of refresh requests (other than programming), but there is a lower limit. This lower limit is based on the fact that the request rate can be no faster than the time it takes to service the request. Subsequently, **the minimum programming value of the CDRAM register should be 18 (12H)**. It is very doubtful that this will ever become a problem when operating at normal frequencies, since the refresh rate of most dynamic memories is well above this minimum programming value.

$$\left| \frac{R_{PERIOD}\ (\mu s) * FREQ\ (MHz)}{\#\text{Refresh Rows} + (\#\text{Refresh Rows} * \%\ \text{Overhead})} \right| = \text{CDRAM Register Valve}$$

R_{Period} = Maximum Refresh period specified by the DRAM manufacturer (time in microseconds).
FREQ = Operating Frequency at 80C186XL in megahertz.
#Refresh Rows = Total number of rows to be refreshed.
% Overhead = Derating factor that estimates the number of missed refresh requests (typically 1–5%).

Figure 10. Equation to Calculate Refresh Interval

However, when making use of the power-save feature of the 80C186XL, it is possible to lower the operating frequency such that it will prevent adequate refreshing rates. When operating at 12.5 MHz, dividing the clock by 16 results in a cycle time of 1.28 microseconds. Since the minimum value of the CDRAM is 18, the minimum refresh rate is 23.04 microseconds. 23 microseconds is not fast enough to service most dynamic memories. Therefore, caution must be exercised when using the power-save feature of the 80C186XL. When there is a need to keep dynamic memory alive, the clock should not be divided much below 2 MHz to avoid monopolizing the bus with refresh activity. If there is no desire to keep memory alive during power-save operation, then the refresh unit can simply be disabled during this time.

Programming the Refresh Enable Register

The EDRAM register (Figure 11) is used to enable and disable the refresh control unit. Furthermore, reading the register returns the current value of the down counter.

Setting the E bit enables the RCU and loads the value of the CDRAM register into the down counter. Whenever the E bit is cleared, the refresh control unit is disabled and the down counter is cleared. Disabling the refresh control unit does not change the contents of the refresh address counter (i.e. it is not cleared or initialized to any specific value). Thus, when the refresh unit is again enabled, the address generated will continue from where it left off. Resetting the 80C186XL automatically clears the E bit. There are no refresh bus cycles during a reset.

The current value of the down counter, as well as the present state of the E bit can be examined whenever the EDRAM register is read. Any unused bits will be returned as zero. Whenever the E bit is cleared, the T0 through T8 bits will be read as zero.

REFRESH CONTROL UNIT OPERATION

Figure 12 illustrates the two major operational functions of the refresh control unit that are responsible for initiating and controlling DRAM refresh bus cycles.

The down counter is loaded (with the contents of the CDRAM register) on the falling edge of CLKOUT, either when the E bit is set or whenever the counter decrements to 1. Once loaded, the down counter will decrement every falling edge of CLKOUT. It will continue to decrement as long as the E bit remains set.

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	E	0	0	0	0	0	0	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0

Bits 0–8: T_0–T_8, Refresh request down counter clock count. These bits are read only and represent the current value of the counter. Any write operation to these bits is ignored. These bits are set to 0 on $\overline{\text{RESET}}$ or when the E bit is cleared.
Bits 9–14: Reserved, should be written to as a 0 to maintain future compatibility, will always be read back as zero.
Bit 15: E, enables the operation of the refresh control unit. Setting the E bit will automatically load the request down counter. Clearing the E bit stops refresh operation and clears the Down Counter.

Figure 11. EDRAM Register Format

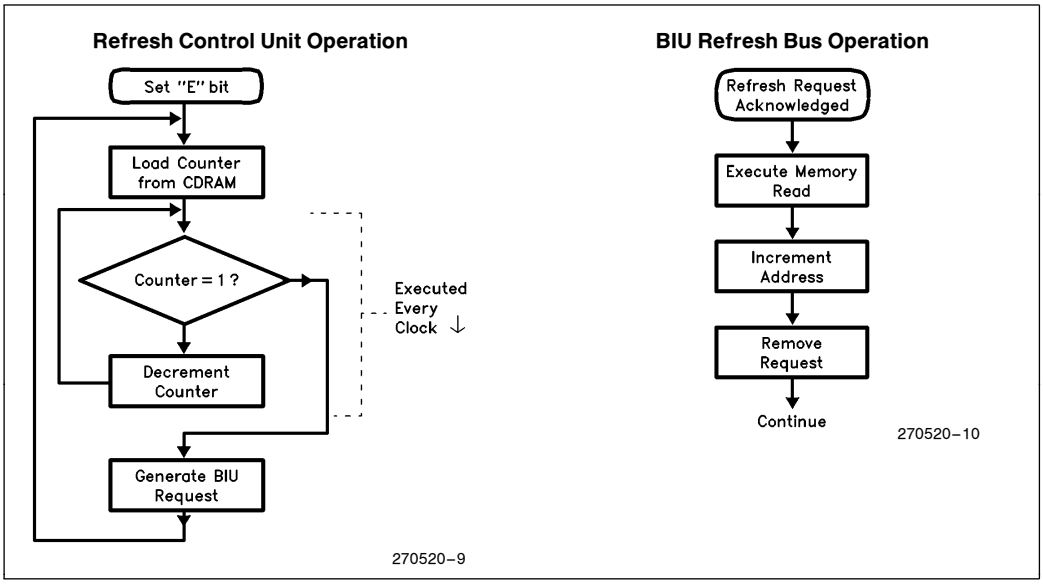


Figure 12. Flowchart of RCU Operation

When the down counter finally decrements to 1, two things will happen. First, a request is generated to the BIU to run a refresh bus cycle. The request remains active until the bus cycle is run. Second, the down counter is reloaded with the value contained in the CDRAM register. At this time, the down counter will again begin counting down every clock cycle, **it does not wait until the request has been serviced.** This is done to ensure that each refresh request occurs at the correct interval. Otherwise, if the down counter only started after the previous request were service, the time between refresh requests would also be a function of bus activity, which for the most part is unpredictable. When the BIU services the refresh request, it will clear the request and increment the refresh address.

80C188XL Address Considerations

The physical address that is generated during a refresh bus cycle is shown in Figure 7, and it applies to both the 80C186XL and 80C188XL. For the 80C188XL, this means that the lower address bit A0 will not toggle during refresh operation. Since the 80C188XL has an 8-bit external bus, A0 is used as part of memory address decoding. Whereas the 80C186XL, with its 16-bit external bus, uses A0 (along with $\overline{BHE}$) to select memory banks. Therefore, when designing 80C188XL memory subsystems it is important not to include A0 as part of the ROW address that is used as a refresh address. Appendix A illustrates Memory Address Multiplexing Techniques that can be applied to the 80C186XL and the 80C188XL.

MISSING REFRESH REQUESTS

Under most operating conditions, the frequency of refresh requests is a small percentage of the bus bandwidth. Still, there are several conditions that may prevent a refresh request from being serviced before another request is generated. These conditions include:

1) LOCKED Bus Cycles
2) Long Bus accesses (wait states)
3) Bus HOLD

LOCKED Bus Cycles

Whenever the bus is LOCKED, the CPU maintains control of the BIU and will not relinquish it until the locked operation is complete. Therefore, internal operations like refresh and DMA are not allowed to execute until the LOCKED instruction has completed. Where this presents the greatest problem is when an instruction such as a move string is executed, and is locked. The move string instruction can take from several clocks to hundreds of thousands of clocks to complete. Obviously anything that takes longer than 512 clocks to complete will always cause a refresh overflow.

Care should be taken not to generate long executing instructions that require bus accesses and are locked. The refresh request interval can be shortened to compensate for missing requests.

Long Bus Accesses

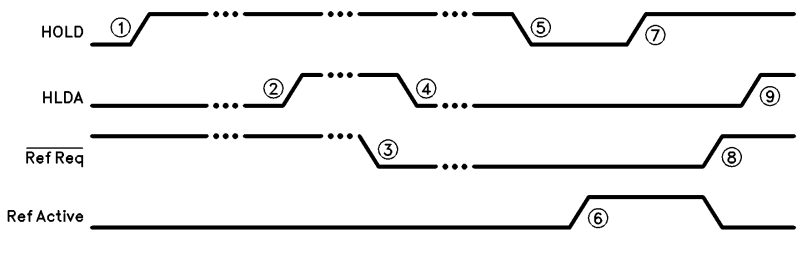
The 80C186XL does not provide any mechanism to abort or terminate a bus access in the event ready is not returned within a specified amount of time (the 80C186XL will infinitely wait for ready). Therefore, if a bus access is in progress when a refresh request is generated, the bus access must complete before the request will be serviced.

Bus HOLD

Special consideration is given when a refresh request is generated and the 80C186XL is currently being held off the bus due to a HOLD request.

When another bus master has control of the bus, the HLDA signal is kept active as long as the HOLD input remains active. If a refresh request is generated while HOLD is active, the 80C186XL will remove (drive inactive) the HLDA signal to indicate to the other bus master that the 80C186XL wishes to regain control of the bus (see Figure 13). If, **and only if**, the HOLD input is removed will the BIU begin to run the refresh bus cycle.

Therefore, it is the responsibility of the system designer to ensure that the 80C186XL can regain the bus if a refresh request is signaled. The sequence of HLDA going inactive while HOLD is active can be used to signal a pending refresh request. HOLD need only go inactive for one clock period to allow the refresh bus cycle to be run. If HOLD is again asserted, the 80C186XL will give up the bus after the refresh bus cycle has been run (provided there is not another refresh request generated during that time).



NOTES:
1. System generates HOLD request.
2. HLDA is returned and 80C186XL floats bus/control.
3. Refresh request is generated internal to 80C186XL.
4. 80C186XL lowers (removes) HLDA to signal that it wants the bus back.
5. 80C186XL waits until HOLD is lowered (removed) for at least 1 clock cycle (minimum HOLD setup and hold time) to execute the refresh bus cycle. If HOLD is never lowered, the 80C186XL will not take over the bus.
6. 80C186XL runs the refresh bus cycle.
7. HOLD can be again asserted after the 1 clock duration.
8. The refresh request is cleared after the bus cycle has been executed.
9. If HOLD was again asserted, the 80C186XL will immediately relinquish the bus back. If no HOLD occurred, normal CPU operation will resume.

Figure 13. HOLD/HLDA Timing and Refresh Request

EFFECTS OF MISSING REFRESH REQUESTS

If a refresh request has not been serviced before another request is generated, the new request is not recorded and is lost. For instance, if the interval between refresh request is 15 microseconds and one request is lost, then the time between two requests will be 30 microseconds when the next request is finally serviced. In this example, missing one request will add 15 μs to the total refresh time. If it is anticipated that refresh requests may be missed (due to programming or system operation), then the refresh request interval should be shortened to allow for missed requests.

Since the BIU is responsible for maintaining the refresh address counter, missing a refresh requests does **not** imply that refresh addresses are skipped. In fact, an address can never be skipped unless a reset occurs.

CONCLUSION

The Internal Refresh Control Unit of the 80C186XL and 80C188XL helps solve three issues concerning DRAM refreshing: a way to generate periodic refresh requests; a way to generate refresh addresses; a way to simplify DRAM memory controllers. Once a memory controller has been designed to handle the simple tasks of reading and writing the task of refreshing has already been built in.

APPENDIX A
TYPICAL DRAM ADDRESS GENERATION CONSIDERATIONS FOR 80C186XL/80C188XL

80C186XL DESIGNS

		Row Address (A0–AX)	Column Address (A0–AX)
64K x 1	(128K Bytes)	A1–A8	A9–A16
16K x 4	(32K Bytes)	A1–A8	A9–A14
256K x 1	(512K Bytes)	A1–A9	A10–A18
64K x 4	(128K Bytes)	A1–A8	A9–A16
1M x 1	(2M Bytes)	A1–A10	A11–A19 (+ Bank)
256K x 4	(512K Bytes)	A1–A9	A10–A18

80C188XL DESIGNS

NOTE:
Address bit A0 can be used in either RAS or CAS addresses, so long as it is not included in any refresh address bits.

		Row Address (A0–AX)	Column Address (A0–AX)
64K x 1	(64K Bytes)	A1–A7, A0	A8–A15
16K x 4	(16K Bytes)	A1–A7, A0	A8–A13
256K x 1	(256K Bytes)	A1–A8, A0	A9–A17
64K x 4	(64K Bytes)	A1–A8	A0, A9–A15
1M x 1	(1M Byte)	A1–A9, A0	A10–A19
256K x 4	(256K Bytes)	A1–A9	A0, A10–A17

RAM Type	RAS Add	CAS Add	Refresh Add
64K x 1	A0–A7	A0–A7	A0–A6
16K x 4	A0–A7	A0–A5	A0–A6
256K x 1	A0–A8	A0–A8	A0–A7
64K x 4	A0–A7	A0–A7	A0–A7
1M x 1	A0–A9	A0–A9	A0–A8
256K x 4	A0–A8	A0–A8	A0–A8