TECHNICAL NOTE

VARIOUS METHODS OF DRAM REFRESH

This article was originally published in 1994.

INTRODUCTION

DRAM refresh is the topic most misunderstood by designers due to the many ways refresh can be accomplished. This article addresses the most often asked questions about refresh. The two basic means of performing refresh, distributed and burst, are explained first, followed by the various ways to accomplish refresh: RAS#-ONLY REFRESH, CAS#-BEFORE-RAS# REFRESH and HIDDEN REFRESH.

STANDARD AND EXTENDED REFRESH

DRAMs are often referred to as either "standard refresh" or "extended refresh." Dividing the specified refresh time by the number of cycles required will determine if the DRAM is a standard refresh or an extended refresh device. If the result is 15.6µs, it is a standard refresh device, while a result of 125µs indicates an extended refresh device.

Table 1 lists some of the standard DRAMs and their refresh specifications.

Table 1
Standard DRAMs and Refresh Specifications

DRAM	REFRESH TIME	NUMBER OF CYCLES	REFRESH RATE
4 Meg x 1	16ms	1,024	15.6µs
256K x 16	8ms	512	15.6µs
256K x 16 (L version)	64ms	512	125µs
4 Meg x 4 (2K)	32ms	2,046	15.6µs
4 Meg x 4 (4K)	64ms	4,096	15.6µs

DISTRIBUTED REFRESH

Distributing the refresh cycles so that they are evenly spaced is known as distributed refresh. To perform distributed refresh on a standard DRAM, execute a refresh cycle every 15.6µs such that all rows are turned on before repeating the task. When not being refreshed, the DRAM can be read from or written to.

BURST REFRESH

Refresh may be achieved in a burst method by performing a series of refresh cycles, one right after the other until all rows have been accessed. During refresh other commands are not allowed. Below is a drawing representing burst and distributed refresh.

For example: a 4 Meg x 1 requires 1,024 consecutive refresh cycles, each of which will use 130ns (^{t}RC) for a 70ns device:

1,024 cycles ¥ 130ns = 133,120ns = 0.133ms
16ms - 0.133ms = 15.867ms

Approximately 0.13ms would be spent performing refresh, and the remaining 15.87ms could be spent reading and writing; then burst refresh would occur again, and so on.

Distributed refresh is the more common of the two refresh categories. The DRAM controller is set up to perform a refresh cycle every 15.6µs. Usually, this means the controller allows the current cycle to be completed and then holds off all instructions while a refresh is performed on the DRAM. The requested cycle is then allowed to resume.

REFRESH CYCLES

There are different cycles you can use to refresh DRAMs, all of which can be used in a distributed or burst method. There are three types listed in a standard data sheet:

- RAS#-ONLY REFRESH
- CAS#-BEFORE-RAS# REFRESH
- HIDDEN REFRESH

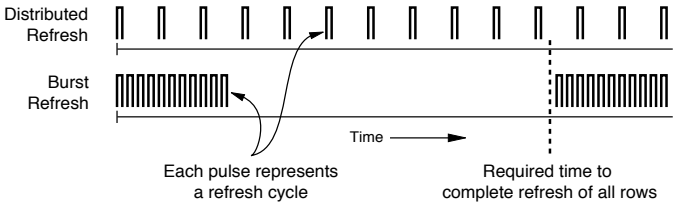

Figure 1
Burst and Destributed Refresh

RAS#-ONLY REFRESH

To perform a RAS#-ONLY REFRESH, a row address is put on the address lines and then RAS# is dropped. When RAS# falls, that row will be refreshed and as long as CAS# is held HIGH, the DQs will remain open. (See Figure 2.)

It is the DRAM controller's function to provide the addresses to be refreshed and make sure that all rows are being refreshed in the appropriate amount of time. The row order of refreshing does not matter; what is important is that each row be refreshed in the specified amount of time.

CAS#-BEFORE-RAS# REFRESH

CAS#-BEFORE-RAS# REFRESH, also known as CBR REFRESH, is a frequently used method of refresh because it is easy to use and offers the advantage of a power savings. A CBR REFRESH cycle is performed by dropping CAS# and then dropping RAS#. One refresh cycle will be performed each time RAS# falls. WE# must be held HIGH while RAS# falls. The DQs will remain open during the cycle.

Here's how CBR REFRESH works. The die contains an internal counter which is initialized to a random count when the device is powered up. Each time a CBR REFRESH is performed, the device refreshes a row based on the counter, and then the counter is incremented. When CBR REFRESH is performed again, the next row is refreshed and the counter is incremented. The counter will automatically wrap and continue when it reaches the end of its count. There is no way to reset the counter. The user does not have to supply or keep track of row addresses. A drawing of one CBR REFRESH cycle is shown in Figure 3. CAS# must be held LOW before and after RAS# falls to meet ^{t}CSR and ^{t}CHR. Figure 4 shows three CBR REFRESH cycles. In this drawing, CAS# stays LOW and only RAS# toggles. Every time RAS# falls a refresh cycle is performed. CAS# may be toggled each time, but it's not necessary.

CBR POWER SAVINGS

Since CBR REFRESH uses the internal counter and not an external address, the address buffers are powered-down. For power-sensitive applications, this can be a benefit because there is no additional current used in switching address lines on a bus, nor will the DRAMs pull extra power if the address voltage is at an intermediate state.

CBR REFRESH IS EASY TO USE

Since CBR REFRESH uses its own internal counter, there is not a concern about the controller having to supply the refresh addresses. Virtually all DRAMs support CBR REFRESH and the 15.6µs refresh rate, so you can design for CBR REFRESH at the distributed rate of 15.6µs and plug in many different DRAMs without having to worry about refresh. For example, the 4 Meg x 4 comes in two versions:

- 2,048 cycles in 32ms
- 4,096 cycles in 64ms

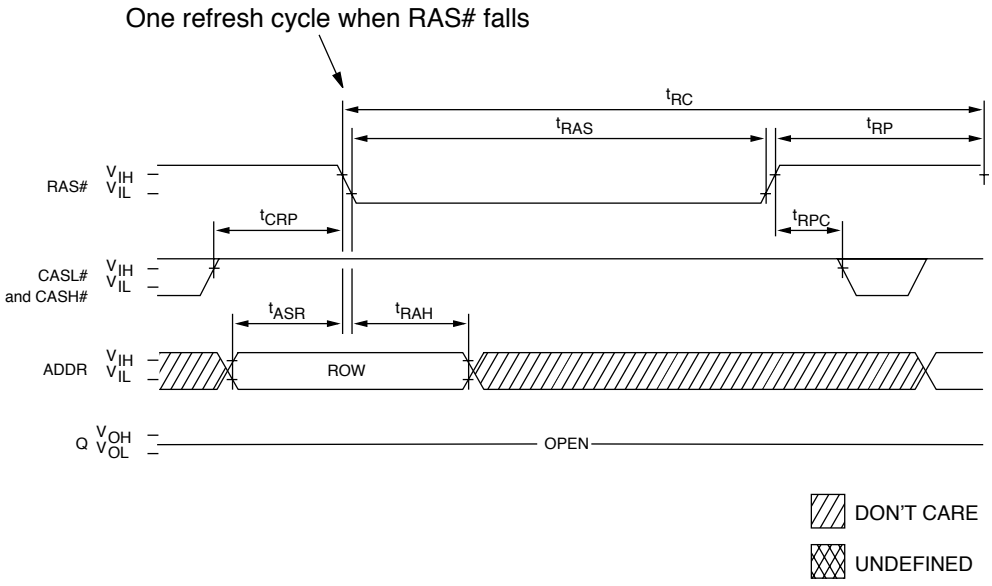

Figure 2
RAS#-Only Refresh

If CBR REFRESH is used, simply maintain the standard 15.6µs refresh rate. If RAS#-ONLY REFRESH is used, addresses must be supplied as follows:

- A0-A10 for the 2,048 cycle refresh
- A0-A11 for the 4,096 cycle refresh

HIDDEN REFRESH

In HIDDEN REFRESH, the user does a READ or WRITE cycle and then, leaving CAS# LOW, brings RAS# HIGH (for minimum of ^{t}RP) and then LOW. Since CAS# was LOW before RAS# went LOW, the part will execute a CBR REFRESH. In a READ cycle the output data will remain valid during the CBR REFRESH. The refresh is not "hidden" in the sense that you can hide the time it takes to refresh; instead, it is hidden in the sense that data-out will stay on the lines while performing the function. READ and HIDDEN REFRESH cycles will take the same amount of time: ^{t}RC. The two cycles together take 2 x ^{t}RC. If we were to do a READ and then follow it with a standard CBR REFRESH (instead of a HIDDEN

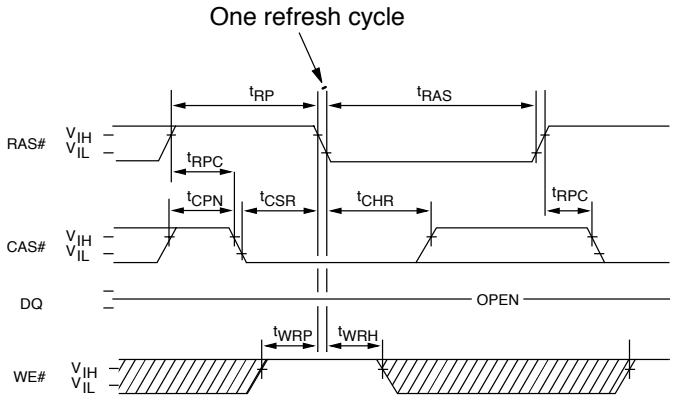

Figure 3
One CAS#-Before-RAS# Refresh Cycle

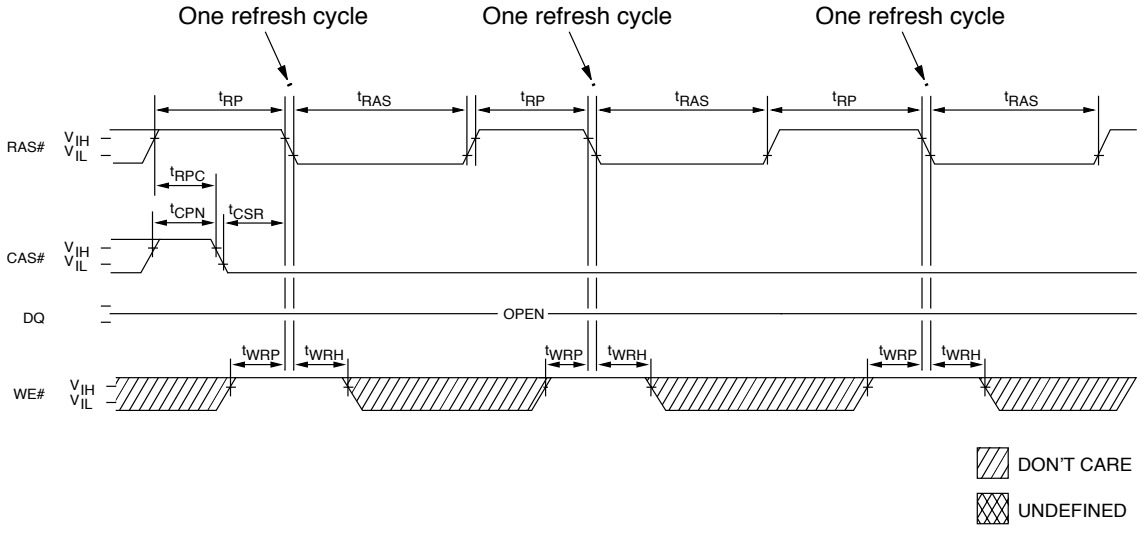

Figure 4
Three CAS#-Before-RAS# Refresh Cycles

REFRESH), this would take the same amount of time: 2 x ^{t}RC.

Figure 5 shows a READ followed by a HIDDEN REFRESH. Figure 6 shows a READ followed by a standard CBR REFRESH. The only difference between the two is that data-out is valid during the HIDDEN REFRESH.

SUMMARY

Three different cycles exist to perform refresh on a standard DRAM: RAS#-ONLY REFRESH, CBR REFRESH, and HIDDEN REFRESH. Each cycle can be used in a burst or distributed method, whichever best fits the designer's needs. It is strongly urged that CBR REFRESH be used to refresh the DRAM. Future DRAMs will most likely require CBR REFRESH only.

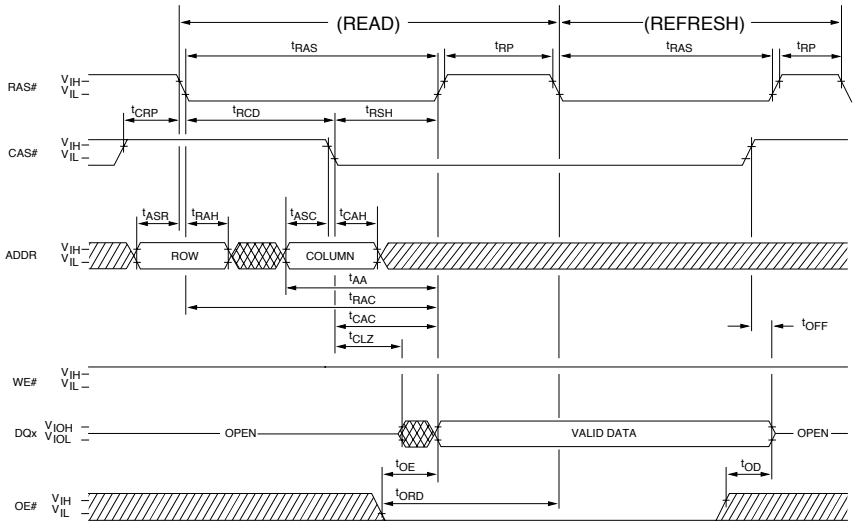

Figure 5
READ Cycle Followed by Hidden Refresh

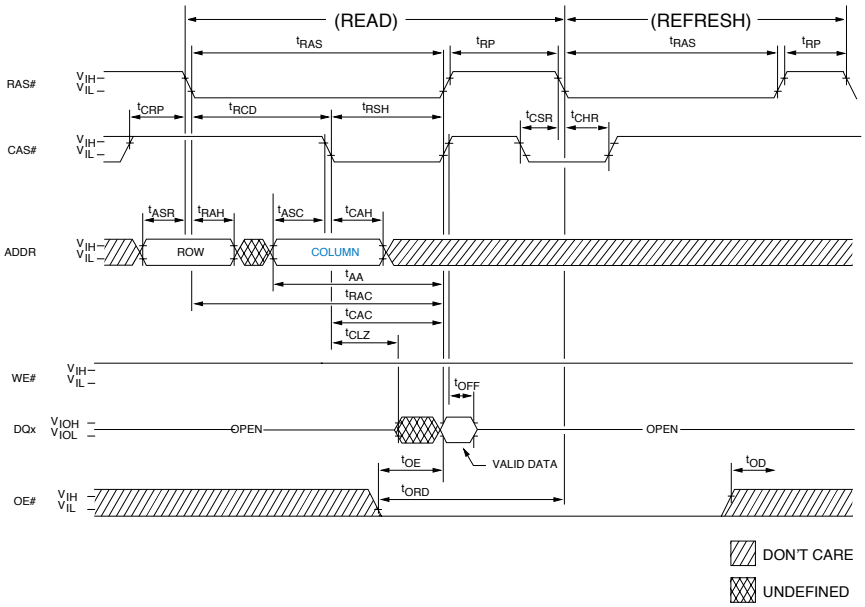

Figure 6
READ Cycle Followed by CBR REFRESH