How to Design with Am29Fxxx Embedded Algorithm

Application Note

by Kumar Prabhat

Advanced Micro Devices

This design note provides a general overview of the Embedded Algorithm and write operation status bits (DQ7–DQ3) that are incorporated in AMD's Flash memory devices and discusses any system level implementation issues associated with them. The Am29F010, a 5.0 Volt-only 1 Mbit Flash device is used as an example. It is highly recommended that this design note be used with the Am29F010 data sheet. Please note that the details on write operation status bits (DQ7–DQ5) provided in this design note may also be used with the Am28F256A, Am29F512A, Am28F010A and Am28F020A flash devices.

EMBEDDED PROGRAMMING OPERATION

Overview

The Am29F010 device is programmed on a byte per byte basis using a four bus cycle command sequence. Addresses are latched on the falling edge of $\overline{WE}$ or $\overline{CE}$, whichever happens later. Data is latched on the rising edge of $\overline{WE}$ or $\overline{CE}$, whichever happens first. The rising edge of $\overline{WE}$ (or $\overline{CE}$) begins the programming operation. The Am29F010 device supports both $\overline{WE}$ or $\overline{CE}$ controlled write operations. Upon executing the Embedded Programming command sequence, the system is not required to provide further controls or timings. The device will automatically provide internally generated program pulses and verify the programmed cell margin. An Embedded Programming operation is completed when the data on DQ7 is equivalent to the data written, at which time the device returns to the read mode. The flowchart for Embedded Programming is shown below.

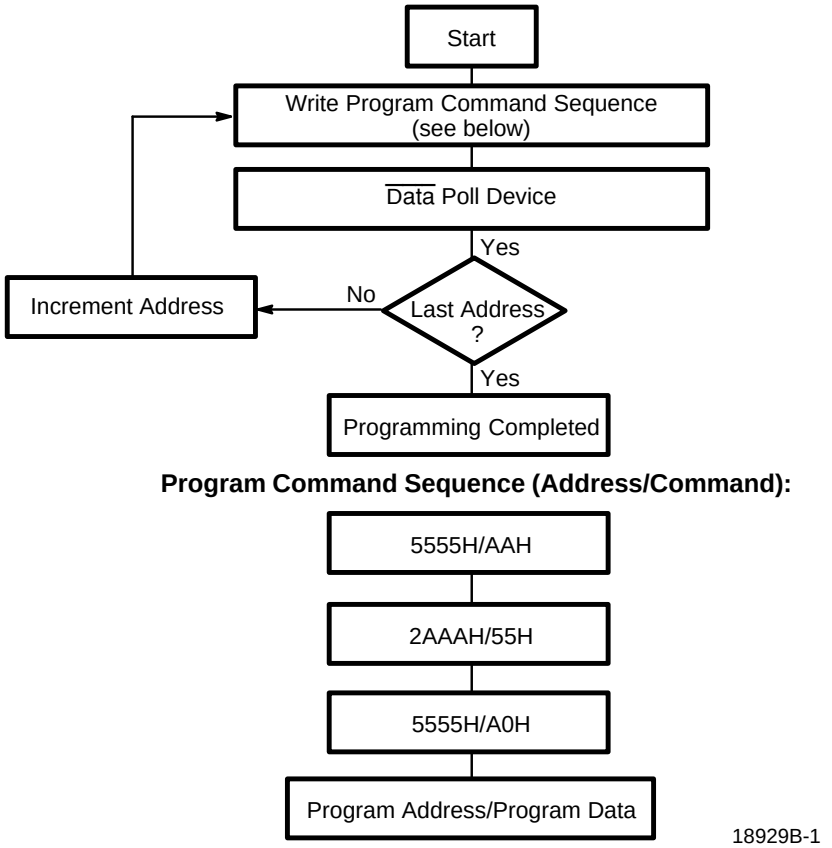


Figure 1. Embedded Programming Flowchart

Publication# **18929** Rev. **B** Amendment **/0**
Issue Date: **November 1995**

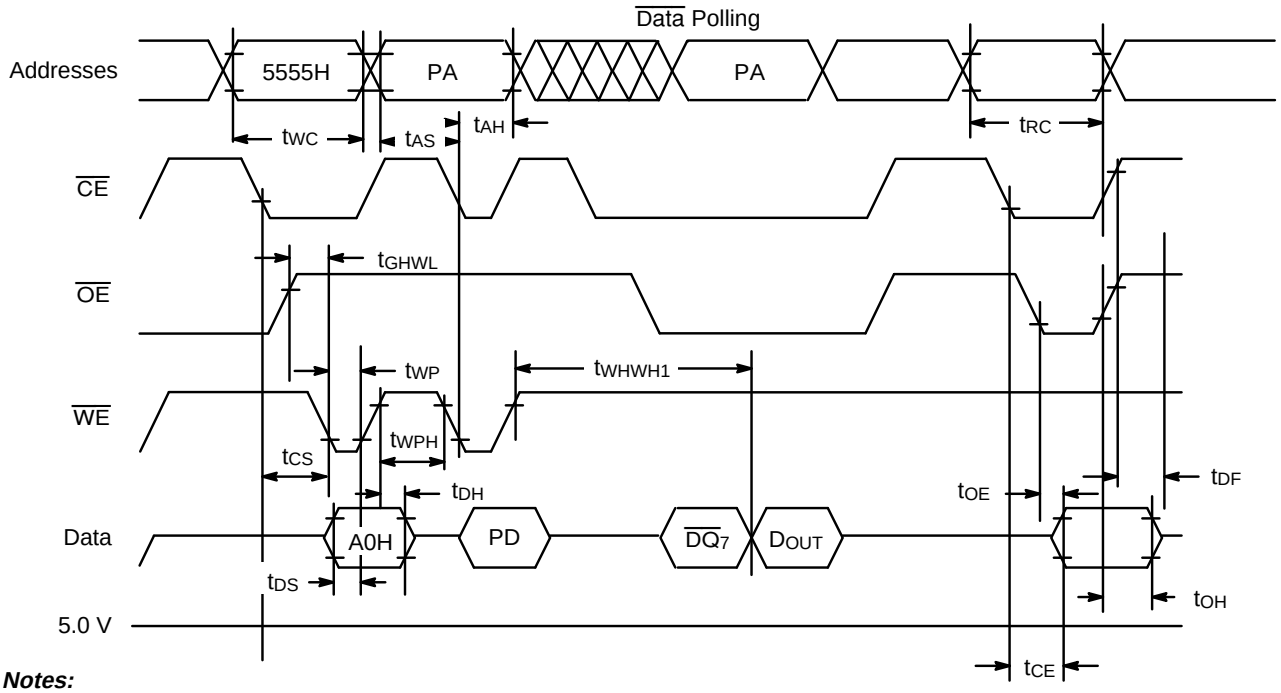
Implementation

Addresses are latched on the falling edge of $\overline{WE}$ during the Embedded Program command execution. Hence the system is not required to keep the address stable during the entire Programming operation. However, once the device completes the Embedded Programming operation, it returns to the read mode and address is no longer latched. Therefore, the device requires that valid address to the device be supplied by the system at this particular instant of time. Otherwise, the system will never read valid data on DQ7.

A system designer has two design alternatives to implement the Embedded Programming Algorithm:

- The system may initiate the $\overline{Data}$ Polling operation immediately after the the Embedded Programming command sequence is written.
- Once the system executes the Embedded Programming command sequence, the system microprocessor may take away the address from the device and thus is free to perform other tasks. In this case, the system microprocessor is required to keep track of the valid address which can be done by loading the address into a temporary register or any memory location. When the system microprocessor comes back to perform the $\overline{Data}$ Polling operation, it should reassert the same address.

However, since the Embedded Programming operation takes only 14–28 µs, it may be easier for the system microprocessor to start the $\overline{Data}$ Polling operation immediately after it has written an Embedded Programming Command instead of coming back and reasserting the valid address during the $\overline{Data}$ Polling operation. The option of either method is left to the system designer's choice. Figure 2 illustrates the timing diagram for the Embedded Programming operation.



Notes:

1. *PA is address of the memory location to be programmed.*
2. *PD is data to be programmed at byte address.*
3. $\overline{DQ}7$ *is the output of the complement of the data written to the device.*
4. D_{OUT} *is the output of the data written to the device.*
5. *Figure indicates last two bus cycles of four bus cycle sequence.*

Figure 2. Embedded Programming Operation

EMBEDDED ERASE

Overview

When executing the Embedded Erase Algorithm command sequence the device automatically will preprogram and verify the entire memory array for an all ‘zero’ data pattern prior to electrical erase. The system is not required to provide any controls or timings during this operation. The automatic erase begins on the rising edge of the last $\overline{WE}$ pulse in the command sequence and terminates when the data on DQ7 is ‘1’. The flowchart for the Embedded Erase operation is shown below.

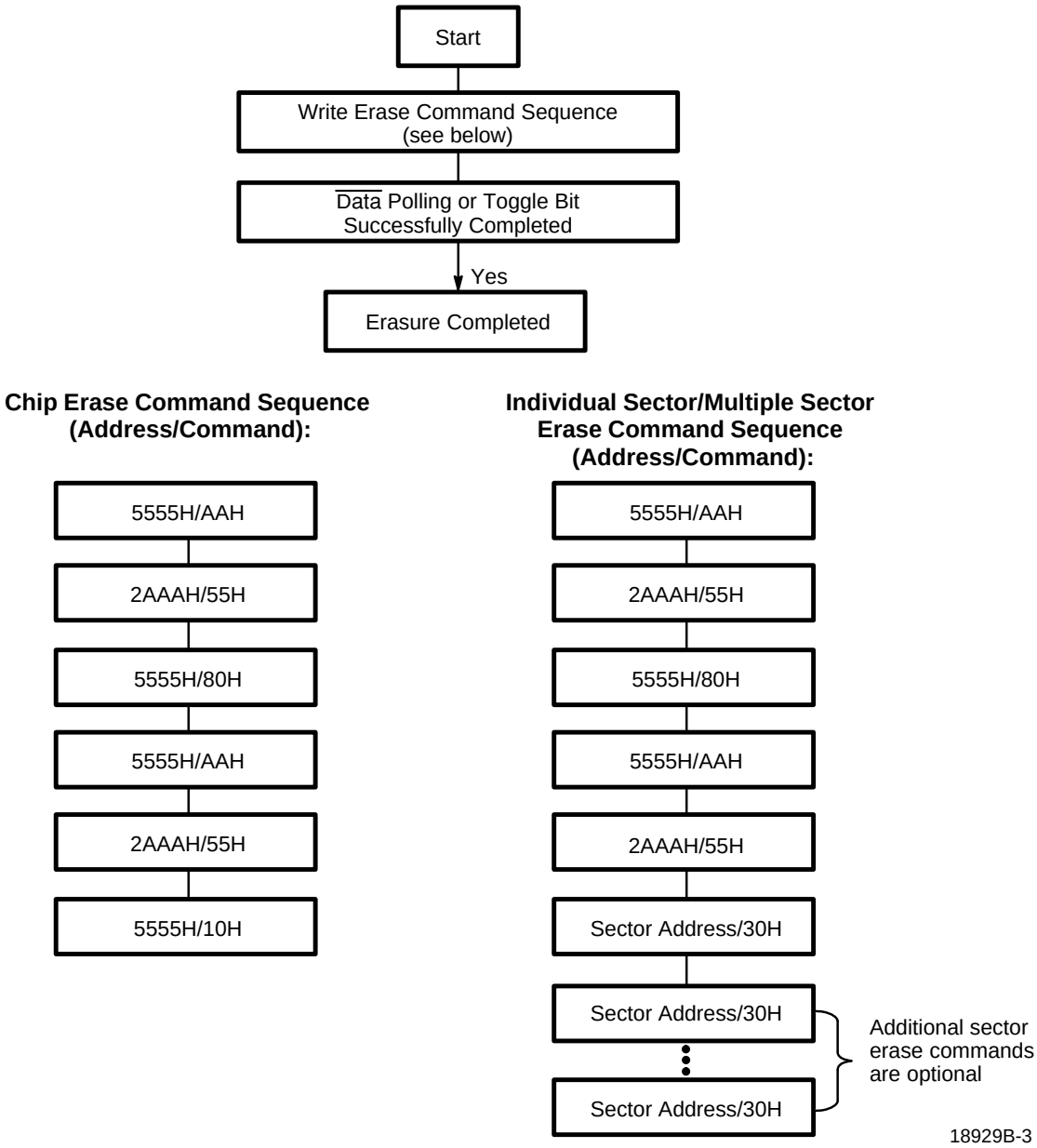


Figure 3. Embedded Erase Flowchart

Implementation

Similar to the Embedded Programming operation, once the device completes the Embedded Erase operation it returns to the read mode and addresses are no longer latched. Therefore, the device requires that a valid address input (sector address within any of the sectors being erased) to the device be supplied by the system at this particular instant of time. Otherwise, the system will never read a “1” on the DQ7 bit.

A system designer has two design alternatives to implement the Embedded Erase Algorithm:

- The system may initiate the $\overline{\text{Data}}$ Polling operation immediately after the Embedded Erase command sequence is written
- Once the system executes the Embedded Erase command sequence, the system microprocessor takes away the address from the device and thus is free to perform other tasks. In this case, the system microprocessor is required to keep track of one of the valid sector addresses (sectors being erased) and when it comes back for performing the $\overline{\text{Data}}$ Polling operation, it should reassert the same address.

Since the Embedded Erase operation takes a significant amount of time (typically 1 second), the second method would provide better system performance by freeing up the CPU for other system level tasks. The system can generate an interrupt on a regular interval to initiate the $\overline{\text{Data}}$ Polling operation to determine the status of the Embedded Erase operation. However, the choice of either option has been left to the system designer.

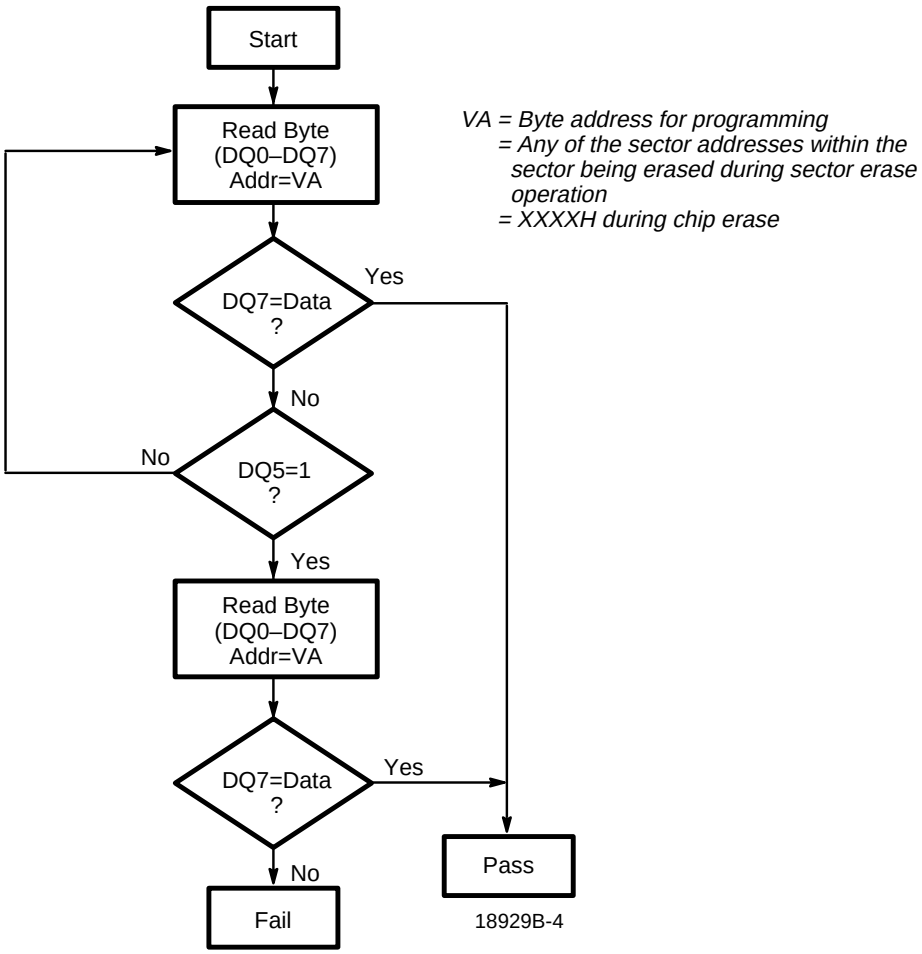
For the chip erase operation, if the device does not include any protected sectors, $\overline{\text{Data}}$ Polling may be performed at any address. When sectors are protected, $\overline{\text{Data}}$ Polling should be performed at any of the sector addresses which represent an unprotected sector.

WRITE OPERATION STATUS BITS

This section describes the operation of the Am29F010 write operation status bits (DQ3–DQ7). This section also describes the timing diagrams for the $\overline{\text{Data}}$ Polling (DQ7) and Toggle Bit (DQ6) operation.

DQ7–$\overline{\text{Data}}$ Polling

The Am29F010 device features the $\overline{\text{Data}}$ Polling operation as a method to indicate to the host system whether the Embedded Algorithms are in progress or completed. During the Embedded Program Algorithm, any attempt to read the device at address VA (Valid Address) will produce the compliment of the data last written to DQ7. Upon completion of the Embedded Program Algorithm, an attempt to read the device will produce the true data last written to DQ7. During the Embedded Erase Algorithm, an attempt to read the device will produce a "0" at the DQ7 output. Upon completion of the Embedded Erase Algorithm, an attempt to read the device will produce a "1" at the DQ7 output. The flowchart for the $\overline{\text{Data}}$ Polling operation (DQ7) is shown below.

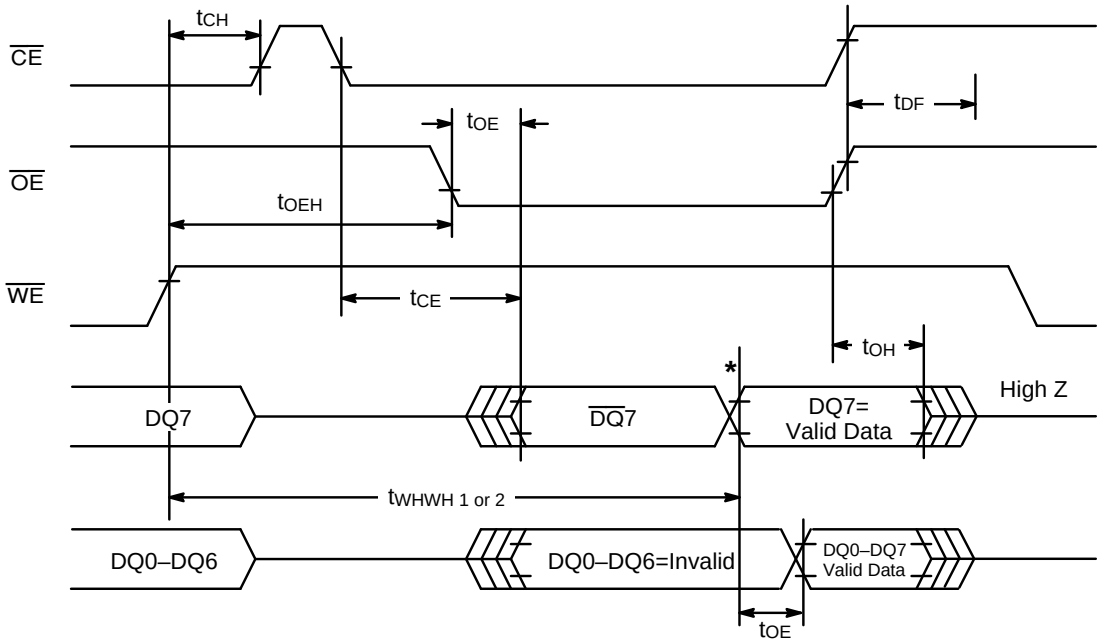


Note:

1. DQ7 is rechecked even if DQ5 = "1" because DQ7 may change simultaneously with DQ5.

Figure 4. DQ7—$\overline{\text{Data}}$ Polling Flowchart

Once the Embedded Algorithm operation is close to being completed, the Am29F010 data pins (DQ0–DQ7) may change asynchronously while the output enable ($\overline{OE}$) is asserted low. This means that the device is driving status information on DQ7 at one instant of time and then changing to the byte's valid data at the next instant of time. Depending on when the system samples the DQ7 output, it may read the status or the valid data. Even if the device has completed the Embedded operation and DQ7 has a valid data, the data outputs on DQ0–DQ6 may still be invalid since the switching time for the individual data bits (DQ0–DQ7) may not be the same. This is due to the fact that the internal delay paths for the individual data bits (DQ0–DQ7) are different. The valid data will be provided only after a certain time delay ($<t_{OE}$). This has been explained in the timing diagram shown below.



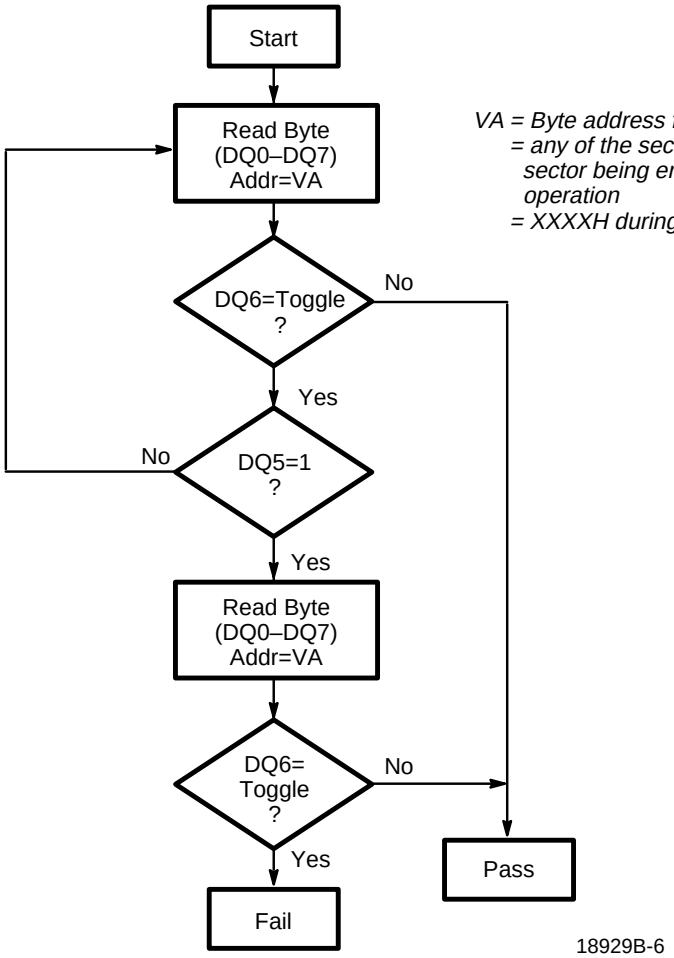
**DQ7=Valid Data (The device has completed the Embedded operation).*

18929B-5

Figure 5. DQ7—$\overline{Data}$ Polling Timing Diagram

DQ6—Toggle Bit

The device also features a “Toggle Bit” operation as another method that indicates the status of the Embedded Algorithm operations to the host system. During an Embedded Algorithm Program or Erase cycle, successive attempts to read (toggling $\overline{OE}$ or $\overline{CE}$) data from the device will result in DQ6 toggling between one and zero. Once the Embedded Algorithm Program or Erase cycle is completed, DQ6 will stop toggling and valid data on DQ0–DQ7 will be read on the next successive read attempt ($\overline{OE}$ going low). The flowchart for the Toggle Bit operation (DQ6) is shown below.

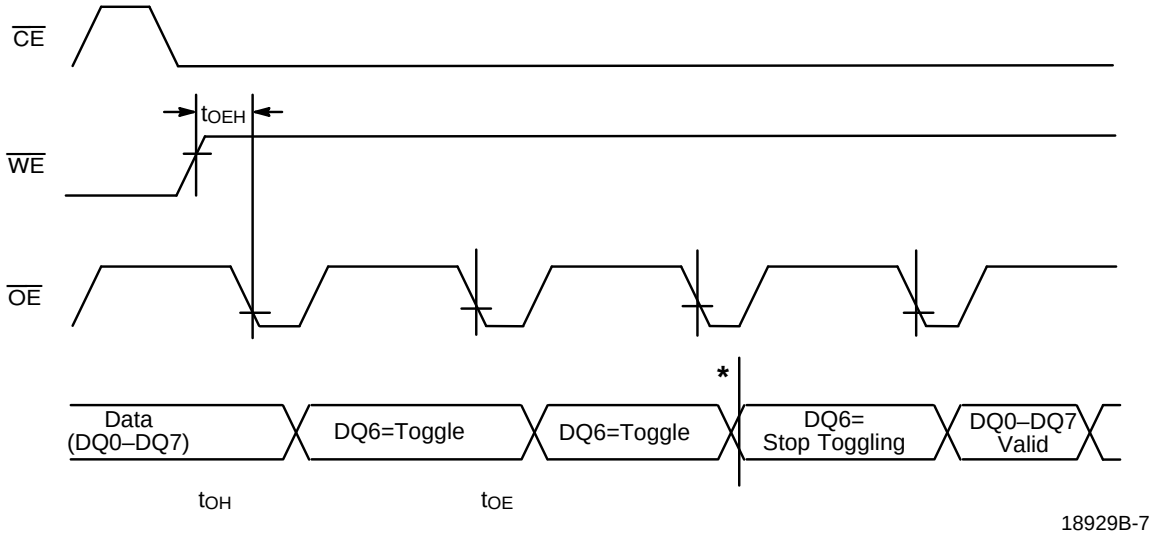


Note:

1. *DQ6 is rechecked even if DQ5 = “1” because DQ6 may stop toggling at the same time as DQ5 changing to “1”.*

Figure 6. DQ6—Toggle Bit Flowchart

Please note that even if the device completes the Embedded Algorithm operation and DQ6 stops toggling, data bits DQ0–DQ7 may not be valid during the current bus cycle. This happens since the internal circuitry may be switching from a status mode to the read mode. Since this time delay is always less than tOE ($\overline{OE}$ access time), the next successive read attempt ($\overline{OE}$ going low) will provide the valid data on DQ0–DQ7. This has been explained in the timing diagram shown below.



Note:

**DQ6 stops toggling (The device has completed the Embedded operation).*

Figure 7. DQ6—Toggle Bit Timing Diagram

The Am29F010 provides the $\overline{Data}$ Polling (DQ7) and Toggle Bit (DQ6) operations as two alternatives to determine the write operation status. However, a system designer is free to perform the complete byte verification instead of implementing either of these two methods.

DQ5—EXCEEDED TIMING LIMITS

The Am29F010 will also be able to indicate through DQ5 if the program or erase time has exceeded the specified limits **(internal pulse count)**. Under these conditions DQ5 will produce a "1". This is a failure condition which indicates that either the program or erase cycle was not successfully completed.

- If this failure condition occurs during sector erase operation, it specifies that a particular sector is bad and it may not be reused; however, other sectors are still functional and may be used for the program or erase operation. To use other sectors, reset the device by writing the Reset command sequence and then executing the program or erase command sequence. This allows the system to continue to use other active sectors in the device.
- If this failure condition occurs during the chip erase operation, it indicates one of the following:
  - The entire chip is bad and should not be reused
  - One or more sectors are bad. The system should be able to determine bad sectors by reading the DQ5 bit for individual sectors.
- If this failure condition occurs during the Byte Programming operation, it specifies that the entire sector containing that byte is bad and may not be reused.

The DQ5 failure condition may also appear if a user tries to program a non-blank location without first erasing it. In this case, the device locks out and never completes the Embedded Algorithm operation. Hence the system never reads a valid data on DQ7 bit and DQ6 never stops toggling. Once the device exceeds timing limits (internal pulse counts), the DQ5 bit will indicate a "1". Please note that this is *not* a device failure condition since the device was incorrectly used. Under this illegal condition, the system is required to reset the device by writing the Reset command sequence before the device can be used again.

DQ3—SECTOR ERASE COMMAND TIME-OUT FLAG

Overview

Sector erase is a six bus cycle operation similar to that used by standard EEPROMs. There are two unlock write cycles followed by writing the "set-up" command. Two more unlock write cycles are then followed by writing the sector erase command. On this sixth bus cycle, the sector address is latched on the falling edge of $\overline{WE}$ while the sector erase command (30h) is latched on the rising edge of $\overline{WE}$. Multiple sectors may be erased by writing the above six bus cycle operations followed by subsequent writes of sector erase commands to all other addresses in the sectors that need to be erased concurrently. The following is an example:

7th Bus Cycle	8th Bus Cycle	9th Bus Cycle
SA1/30H	SA2/30H	SA3/30H

After the completion of the initial sector erase command sequence, the sector erase time-out of 100 µs will begin. Every time the system writes an additional sector erase command, the time-out window is reset. The device will indicate this time-out through the DQ3 bit. If the DQ3 bit is high, the internally controlled erase cycle has begun. Any attempts to write additional commands to the device will be ignored until the erase operation is completed as indicated by $\overline{Data}$ Polling or Toggle Bit. If DQ3 is low, the sector erase timer window is still open and the device will accept additional sector erase commands provided that these additional sector erase commands are written within the 100 µs time-out window.

Implementation

Once the first sector erase command sequence is written and the sector erase time out has begun, the system software should read the status of DQ3 (at any address) prior to writing any sector erase command to determine whether the 100 µs time-out window is still open. The system software should also read the status of DQ3 following each sector erase command to verify that the command has been accepted.

Chip Erase

The chip erase command should not be used on devices that use sector erase commands. Likewise, sector erase commands should not be used on devices that use the chip erase command.

Sector Erase

If the multiple sector erase command is used, multiple sectors should be erased in groups to ensure that a group of sectors is exposed to the same number of program/erase cycles. In addition, the chip erase command should not be used on a device that uses sector erase or multiple sector erase commands.

DQ2–DQ0

Reserved for future use.