AN1122
APPLICATION NOTE

Applying Protection and Unprotection to M29 Series Flash Memories

CONTENTS

INTRODUCTION

This application note describes the techniques used to protect and unprotect the blocks in M29 series Flash Memories. The application note applies to the Revision A, Revision B and Revision D M29 Series Flash memories, where the details of the Protect and Unprotect operations are not presented in the datasheet. The first version of the memories, with no revision letter, use algorithms presented in their respective datasheets.

When a block is protected the Flash's Program/Erase Controller will not allow changes to the block. It will not be possible, using any unwanted electrical signals or software commands, to change the data, either by accident or on purpose. This gives added confidence that, if the Flash is used for code storage then the product will always be able run after a reset, even if the software has erroneously tried to reprogram the Flash.

HISTORY OF BLOCK PROTECTION

In the early days of Flash the security of data was a major concern. Designers used to using EPROMs and ROMs were very concerned that the contents of Flash could become corrupt. The design of ROMs and EPROM does not allow them to become corrupt once in the system. To overcome this Flash manufacturers used several techniques to protect the Flash. The coded cycles required to execute the program or erase operations make it almost impossible for random write sequences (either from erroneous software or power-up/power-down electrical noise) to damage the contents of the Flash. All Flash Memories have a Lockout Voltage, V_{LKO}, to increase data security during power-up/power-down. For those who remained unconvinced about the security of Flash, Block Protection was also provided.

Recently designers have tended to rely on the coded cycles and Lockout Voltage to protect the Flash. Leaving the blocks unprotected has allowed greater flexibility because application code can be updated in-system without removing the Flash from the board. The security provided by the coded cycles and Lockout Voltage is rarely, if ever, breached, and the advantage of allowing in-system upgrades can be a greater benefit than the risk of losing or corrupting the code.

WHICH BLOCKS SHOULD BE PROTECTED?

If a designer is not happy with the protection provided by the coded cycles and the Lockout Voltage then a decision has to be made as to which blocks need to be protected and which ones do not need protection. A simple way to design the system is to use a small boot program that runs and performs a system check before the main application runs. The boot program is clearly very important because, without it, nothing will run. The boot program can also alert the user to an error in running the application by giving an indication that is more positive than merely not executing at all.

By incorporating a Download feature into the boot code the application code can be updated in-system. (A Download feature is one that allows the application code to be changed by the microprocessor.) Furthermore, if something goes wrong while the application code is downloading, the system will still boot and it will remain possible to attempt another download of the application.

If any blocks are going to be protected, the one with the boot code should be the first choice. This allows systems that have become corrupt to inform the user of the problem. The system can be fixed without the Flash being removed from the circuit board (not easy since most Flash is surface mount).

The designer may still not be happy with the protection provided for the application code. If the designer chooses to protect the application code then field updates are not really feasible. Depending on the Flash it may be necessary to remove the Flash from the circuit board to perform application upgrades. However, there are products where the code is fixed and no field upgrades are ever expected (e.g. hard disk drives). Flash may be a preferred choice over EPROM for these applications because it allows other blocks to store data, a facility that EPROMs cannot provide.

STMicroelectronics does not recommend protecting blocks that have data in them, unless the data is only updated as often as the application code. The number of times that protection can be applied and removed is not guaranteed. Once a design is able to remove protection without external intervention then it is no longer any more secure than a design that does not use protection at all. The main mode of failure for unprotected blocks is through software failure and, once software has control of the unprotect operation, it can corrupt the application code as easily as changing the data.

TECHNIQUES FOR CONTROLLING BLOCK PROTECTION

Unlike the Command Interface of the Program/Erase Controller, the technique for protecting and unprotecting blocks changes between different Flash memory suppliers. For example, following the techniques for AMD parts will not work on STMicroelectronics parts. Care should be taken when changing drivers for one part to work on another.

There are three techniques for controlling whether a block is protected or not. Not all techniques are available on all types of Flash Memory. The techniques available are:

1. Programmer Technique (using special bus operations, not available on a standard microprocessor bus).
2. In-System Technique (for Flash Memories already surface mounted onto the circuit board).
3. Temporary Unprotection (again, for in-system Flash Memories).

Technique 1 (Programmer Technique) is available on all types of Flash Memory. Technique 2 (In-System Technique) is available on ‘B’ and ‘D’ revision memories with a Reset/Block Temporary Unprotect pin; these include memories such as the M29F400B, M29W400B, M29F160B, etc. Technique 3 (Temporary Unprotection) is available on all Flash Memories with a Reset/Block Temporary Unprotect pin.

Note that the revision letter appears after the part number, but this can be confused with the Array Matrix selection for Bottom Boot Block devices. For example the M29F400B device is an old revision memory, the new revision is M29F400BB. The family name is M29F400 for the old family and M29F400B for the new family. Take care when deciding which part and revision is being used. If in doubt use the full part name and refer to the correct version of the Data Sheet.

Not all Flash Memories allow the protection of each block to be set individually. Devices such as the M29F080A, M29F016B, M29F080D, M29F016D and M29F032D require the blocks to be protected in groups. On these Flash Memories it is only necessary to follow the protection flowchart for one of the blocks in the group and all of the blocks in the group will become protected. The Protection Groups for these parts are listed in the Block Addresses Table of their Data Sheets.

Programmer Technique

Programmer Equipment Manufacturers should follow Figure 1, Programmer Equipment Protection Flowchart, to protect blocks in the Flash. Figure 2, Programmer Equipment Unprotection Flowchart, should be used to unprotect all blocks at once. Due to the internal design it is necessary to protect all blocks before starting the unprotect operation. Furthermore, all blocks must be unprotected at once. It is not possible to unprotect a single block in the same way as it is not possible to erase a single byte in the Flash Memory array.

The Flowcharts should be followed exactly and should not be aborted before reaching the end. The unprotect operation can take several seconds, so it might be useful to inform the users that the operation is progressing, otherwise they may believe that nothing is happening.

In-System Technique

Once the Flash has been fitted onto the circuit board the in-system technique for controlling protection should be used. This has been included in the new Flash Memories because it is difficult to remove surface mount parts; a five minute update task can take several days while the part is sent away for removal, returned, updated and then sent away again for fitting. As already described, the In-System technique is only available on 'B' and 'D' revision Flash Memories which have a Reset/Block Temporary Unprotect pin.

To use the In-System technique follow the Flowchart in Figure 3, In-System Protection Flowchart, to protect blocks in the Flash. Use the Flowchart in Figure 4, In-System Unprotection Flowchart, to unprotect all the blocks at once. Again, it is necessary to protect every block before unprotecting all the blocks.

The Flowcharts should be followed exactly and not aborted before reaching the end. The unprotect operation can take several seconds, so it might be useful to inform the users that the operation is progressing, otherwise they may believe that nothing is happening.

Temporary Unprotection

Flash Memories that have a Reset/Block Temporary Unprotect, $\overline{RP}$, pin can have the best of all worlds. Blocks can be protected during the manufacture of the product after the code is programmed into the Flash. Then, if the product is returned later for upgrade, the Reset/Block Temporary Unprotect pin can be raised to V_{ID}, temporarily cancelling the protection on the protected blocks. When the Reset/Block Temporary Unprotect pin drops outside the V_{ID} region the blocks that were previously protected become protected again.

Notes on the Reset/Temporary Unprotect Pin and V_{ID}

By putting the source of V_{ID} external to the product and not within the control of the microprocessor it will be impossible for the Flash to become corrupt during normal operation. The Reset/Block Temporary Unprotect pin can be put on a header on the board to allow for simple connection during upgrades. It is important to include circuitry that will prevent the Reset line to other parts in the system being dragged to V_{ID} and also prevent the Reset/Block Temporary Unprotect pin from making the transition from V_{IH} to V_{ID} faster than V_{PHPHH}. If field updates are required then a jumper could be used to raise the Reset/Block Temporary Unprotect pin to V_{ID}, the system is still secure because the microprocessor cannot put the jumper on itself.

THE THREE OPERATIONS INVOLVED IN PROTECTING AND UNPROTECTING BLOCKS

The bits that hold the protection status of a block are standard Flash memory cells. They can be programmed individually (to protect the block) and erased in bulk to unprotect all blocks. Unlike the cells in the memory array, the Program/Erase Controller does not take care of the specific signals required to program and erase the protection cells. These have to be performed carefully by following the Protection and Unprotection Flowcharts.

There are three distinct operations that involve the protection bits in a Flash Memory. These are:

1. Reading the protection status of a block
2. Programming a protection bit of a block
3. Verifying a protection bit of a block

The protection bits are read as normal Flash memory cells, the output of the cell is fed to a sensing amplifier that decodes the value stored in the cell to a ‘1’ or a ‘0’. The protection bits can be read using the Autoselect command, they are also output as part of the protection verify operation.

Programming a protection bit involves removing charge from the Floating Gate of the flash cell. Charge is removed a bit at a time to avoid removing too much charge; removing too much charge from the Floating Gate can damage the cell. Each time the Protect part of the flowcharts is performed more charge is removed from the Floating Gate.

The verify stage of protecting a bit reads the block protection status while applying a bias to the sense amplifier. This ensures the cell is programmed strongly enough to always read correctly under any condition (for example when very hot or very cold). When protecting it is very important to ensure that the cell outputs the protected state using the verify command, otherwise the cell may output the wrong value under other conditions, allowing Program and Erase operation to change the contents of a cell that the user thought was protected.

The same three stages exist when unprotecting a block too. When unprotecting however, the unprotect operation works on all cells at once, whereas the unprotection verify operation needs to check each protection bit individually. During the unprotect verify, the opposite bias is applied to the sense amplifier to ensure that the cell is erased strongly enough to always read correctly under any condition. It is vital that the unprotection verify is carried out on all blocks and that none are missed.

It is not recommended that the user read the status of the protection bits (operation 1. above) in order know if the sector is protected or not since there is no margin in this operation. The best method is using the unprotect verify and protect verify bits. If the user does however read the status of the protection bits (eg. in Auto Select) then a way to improve margin is to read over all ranges of operation of the device for example at both 4.5V and 5.5V for a 5V device (2.7V and 3.6V for a 3V device).

BLOCK ADDRESSES IN THE FLOWCHARTS

To protect and unprotect the blocks the addresses need to be specified correctly for the block(s) of interest. When protecting the Flash, many of the address pins do not need to be specified (they are “Don’t Care”), however some do need to be correct.

In all Flash Memory devices A0, A1, A6 need to be correct during Block Protection, as well as the bits that specify the BLOCK ADDRESS. In addition to these, when using the Programmer Technique A12 and A15 need to be correct during Blocks Unprotection (the M29F080A and M29W008A parts are special and require A12, A13, A15 and A16 to be correct). The flowcharts specify the values that these addresses should take. When a BLOCK ADDRESS is required in the flowchart any address in the block that keeps A0, A1, and A6 correct may be chosen. A simple way to choose the address is to choose the first address in the block from the Block Addresses Table of the Data Sheet and OR it with the values of A0, A1 and A6, leaving the other bits at ’0’.

When using Flash Memories with a $\overline{\text{BYTE}}$ pin take care that the addresses are specified in the Flash Memory address space, not the address space of the microprocessor. Usually A1 of the microprocessor ends up connected to A0 of the Flash, so the addresses must be shifted accordingly.

CONCLUSIONS

Block protection can be a useful feature for people who require utmost security for the contents of the memory. Usually only the block containing the boot code will be protected.

Flash Memories from STMicroelectronics provide several techniques for the user to choose between for controlling the protection of blocks. Programmer Equipment manufacturers can use the techniques that are compatible with old parts, Product Manufacturers can use in-system techniques to change the block protection after the Flash has been put on the circuit board. Temporary Unprotection can be used to update application code without compromising the strictest of security specifications.

Whatever your block protection requirements, STMicroelectronics Flash Memories have a suitable solution for you.

REVISION HISTORY

Date	Version	Revision Details
April 1999	-01	First Issue
August 1999	-02	In-system protection figures revised: added additional “write 60h” box at the top of the flowchart. This is required by the internal algorithms.
November 1999	-03	Clearer specification of when the three techniques can be used. Modified to account for A12, A13, A15 and A16 being required at V_{IH} during the Programmer Technique Blocks Unprotection.
June 2000	-04	Added the section on the Three Operations Involved in Protecting and Unprotecting Blocks. Added a paragraph warning about the differences between the protect/ unprotect operations in AMD and STMicroelectronics Flash Memories. Added notes to the In-System Unprotection Flowchart.
March 2001	-05	Added extra paragraph in the “Three Operations...” section to clarify that the sector protection status bit has no margin. Added references to M29 D version

Figure 1. Programmer Equipment Protection Flowchart

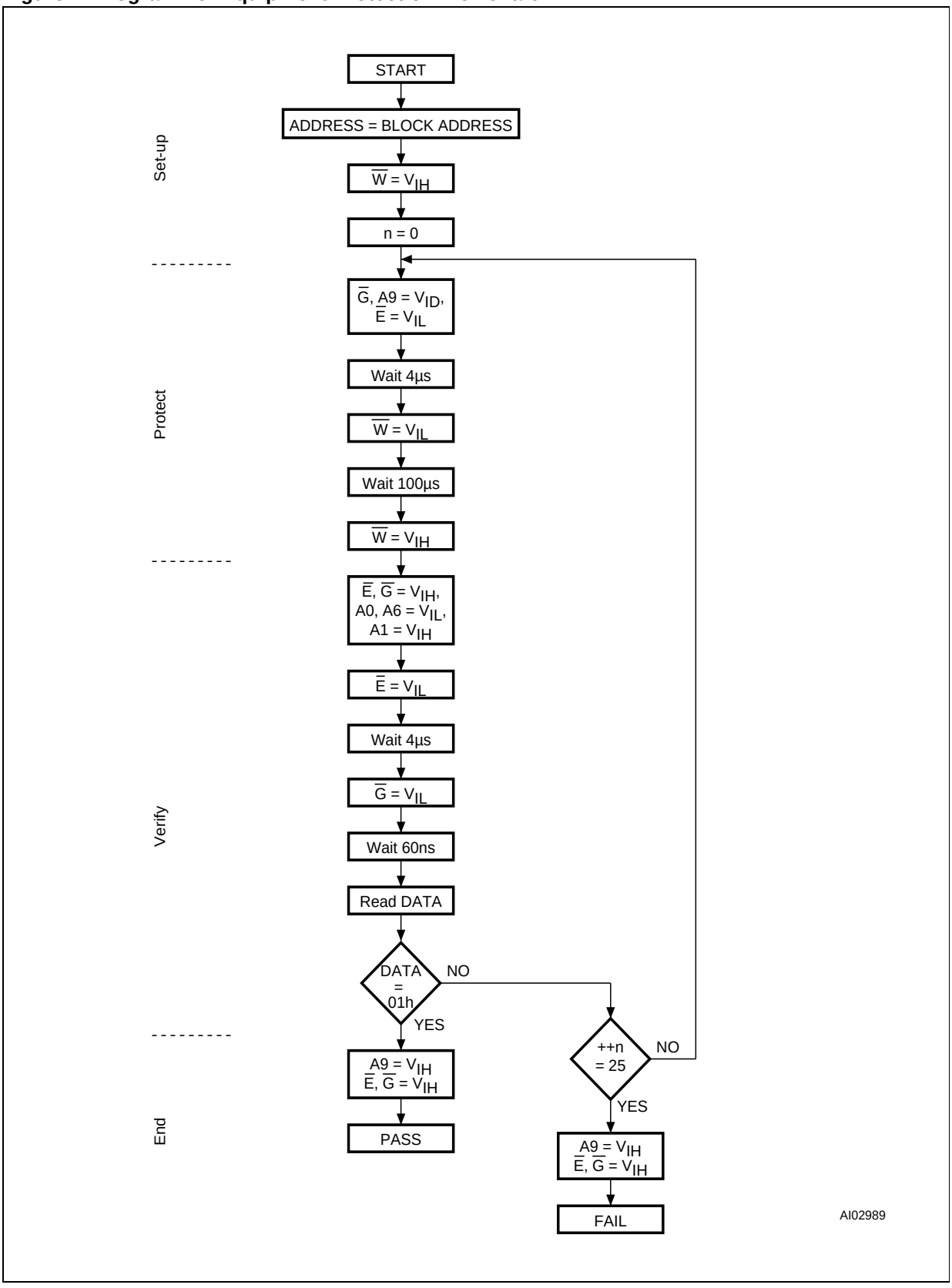
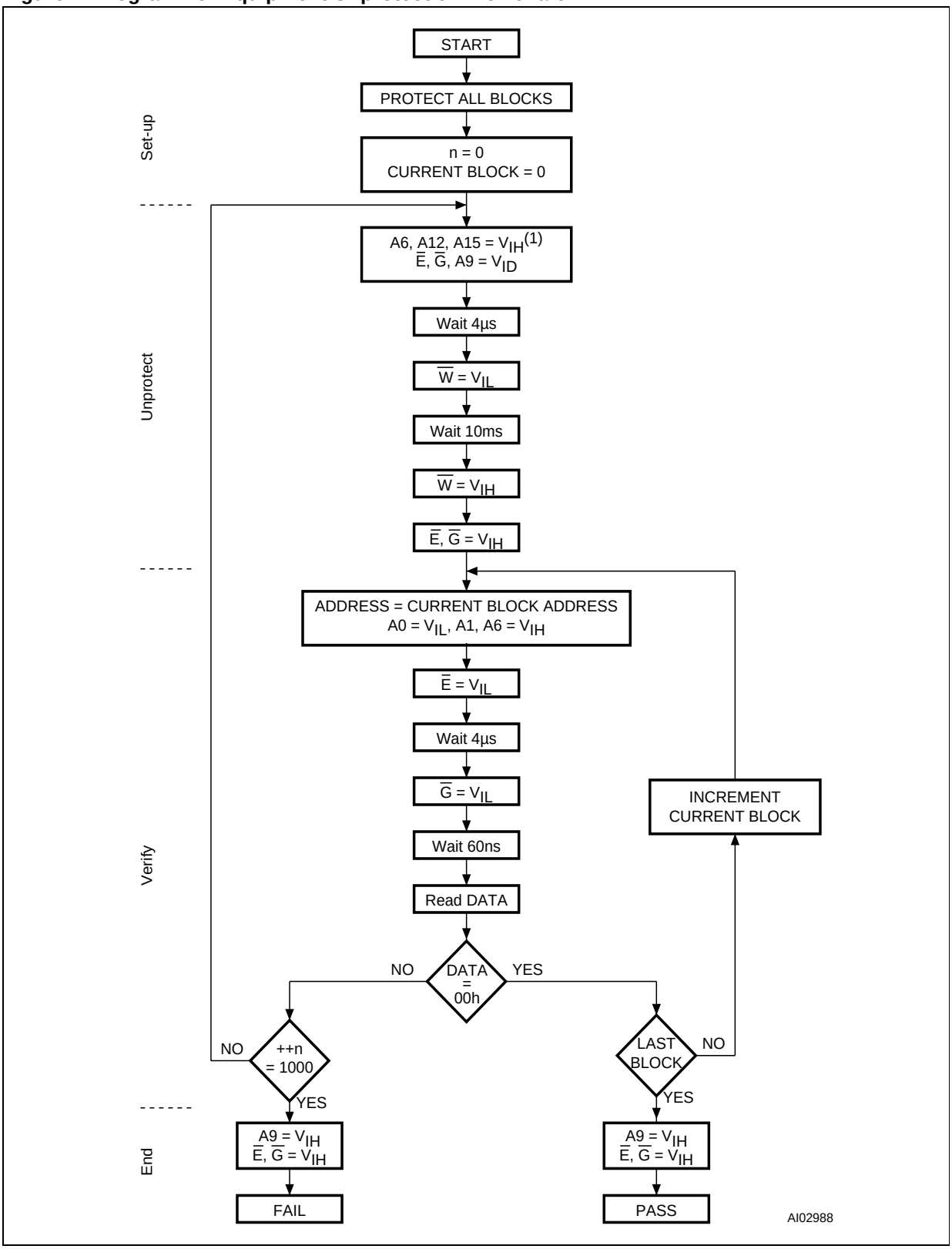


Figure 2. Programmer Equipment Unprotection Flowchart



Note: 1. For the M29F080A and the M29W008A set A6, A12, A13, A15 and A16 to V_{IH}.

Figure 3. In-System Protection Flowchart

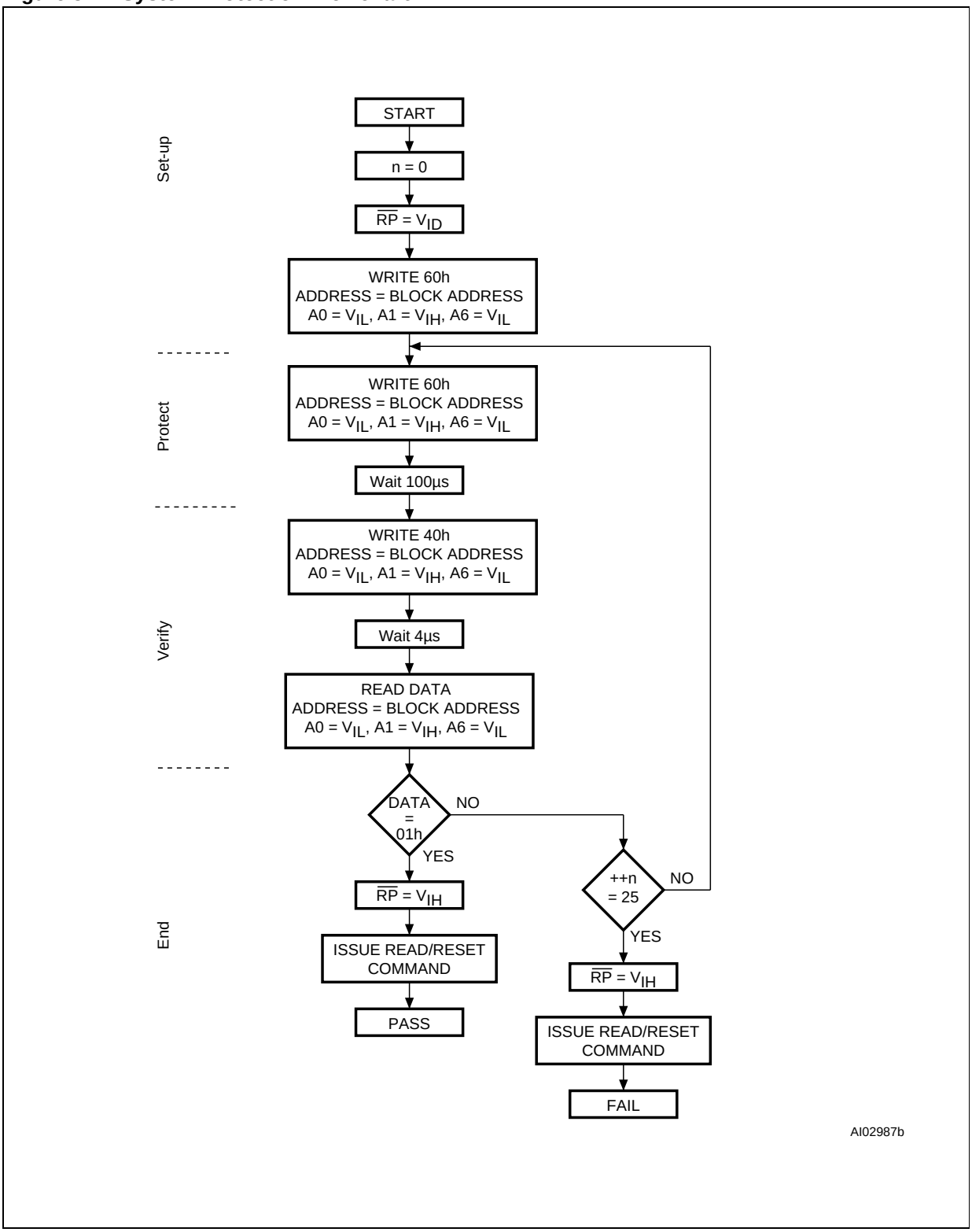
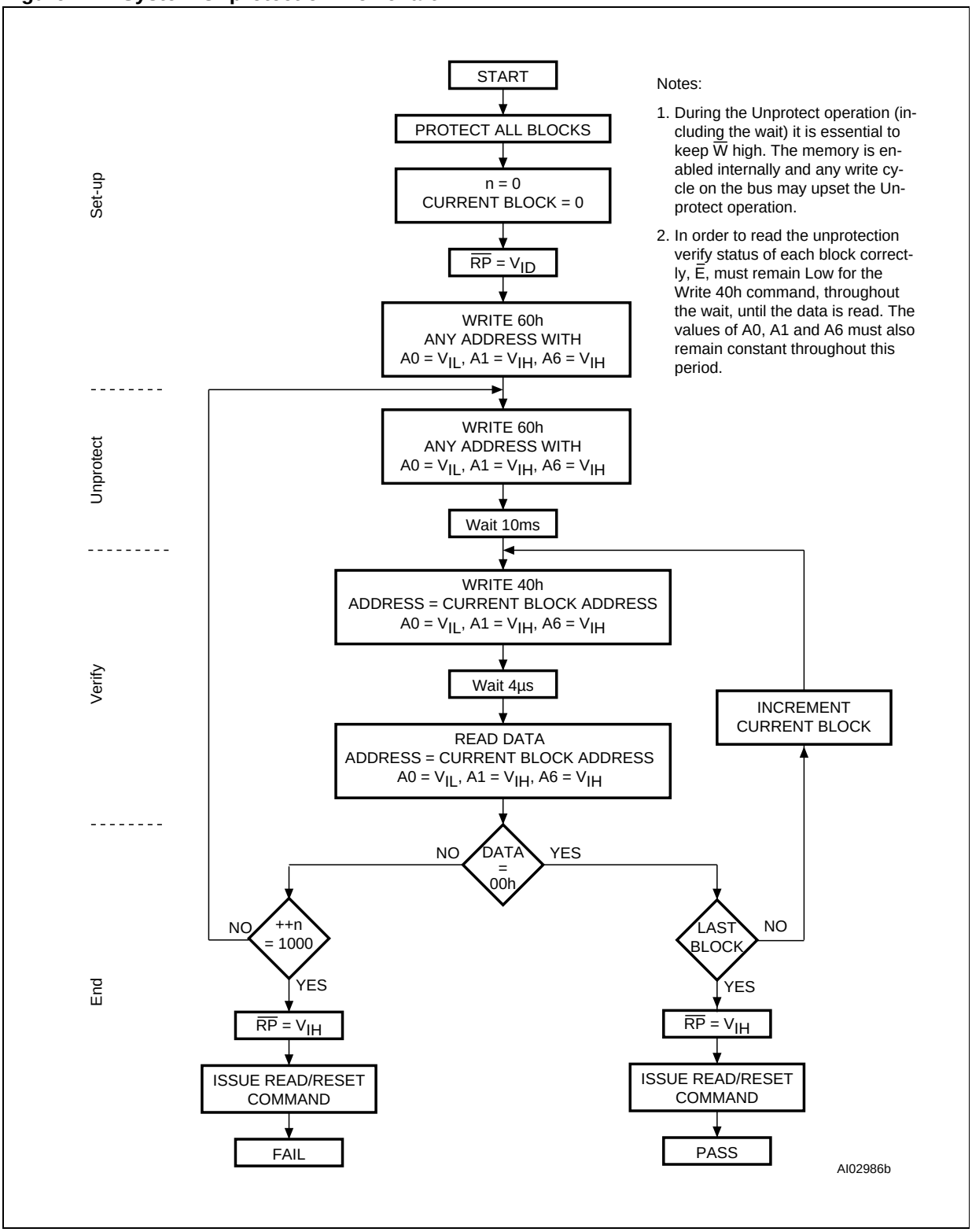


Figure 4. In-System Unprotection Flowchart



Notes:

1. During the Unprotect operation (including the wait) it is essential to keep $\overline{W}$ high. The memory is enabled internally and any write cycle on the bus may upset the Unprotect operation.
2. In order to read the unprotection verify status of each block correctly, $\overline{E}$, must remain Low for the Write 40h command, throughout the wait, until the data is read. The values of A0, A1 and A6 must also remain constant throughout this period.

If you have any questions or suggestion concerning the matters raised in this document please send them to the following electronic mail address:

ask.memory@st.com (for general enquiries)

Please remember to include your name, company, location, telephone number and fax number.

littlediode
COMPONENTS FOR ALL

Littlediode supplies new, hard to find or obsolete electronic components and semiconductors all over the world.

With over two million different components listed you are sure to find the part you need.

Feel free to visit us today at our online store:

LittleDiode.com

Looking forward to providing you with the best possible service.