The Benefits of Atmel's RAPID™ Programming Algorithm

UV Erasable EPROM

Application Note

Introduction

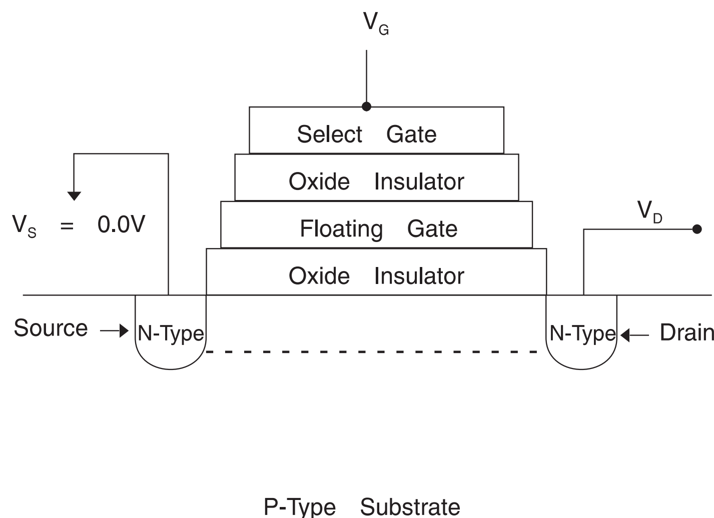
In designing and manufacturing certain modern-day products, the methods used to build these products are often as important to the design engineer as the components themselves. This is true about programmable memory devices as well, especially EPROMs. Most EPROM vendors use their own unique programming algorithm, which is based on the process used to make EPROMs, the design engineer needs to know about the algorithm during the system design cycle to insure that the EPROMs can ultimately be programmed.

This application note details the Atmel RAPID programming algorithm and briefly explains why this algorithm is superior to others. In addition, it will give an introduction to EPROM technology and the mechanics of programming. These should provide a basic understanding in the growing field of EPROMs.

Programming EPROMs the RAPID™ Way

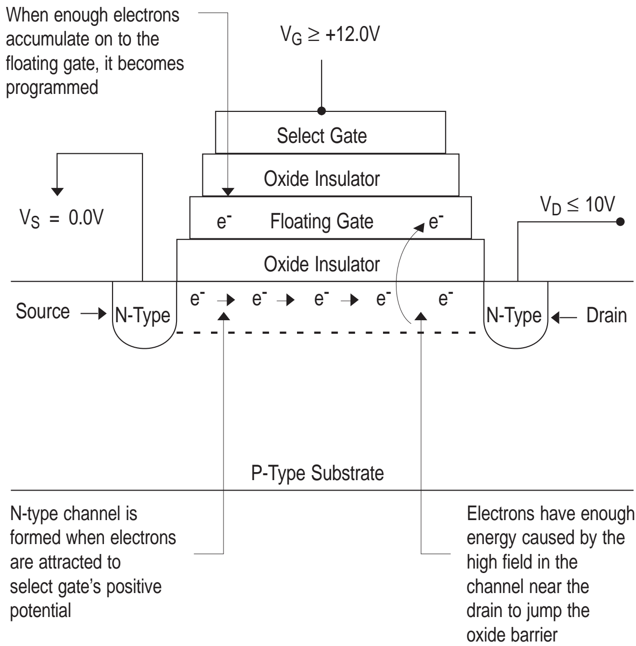
Several years ago, when Atmel reduced the geometry of its EPROM products from 1.5µ to 1.2µ linewidth, the Company adopted an entirely new programming algorithm for these devices. A reason for this algorithm change was to improve programming yields and lengthen long-term data retention. This was accomplished by using a shorter programming-pulse length during programming. The new RAPID algorithm reduces the 1 ms programming pulse width of the original FAST algorithm to only 100 µs, and it completely eliminates extra overprogramming pulses. The advantages of the RAPID programming algorithm are production proven even with today's advanced 0.5 micron EPROM technology.

Figure 1. Cross Section of a Typical EPROM Cell



Rev. 0578A–10/98

Figure 2. Process of Hot Electron Injection



There's more to the RAPID algorithm than shorter programming times and cost savings. It has a special way of checking that each cell is correctly programmed, and that cells are programmed with the required amount of charge. In fact, the RAPID algorithm even guarantees that the EPROM is correctly programmed. Programming algorithms of the FAST type, or their relatives, the QUICK-PULSE types, check each memory location for the programmed data immediately after programming that location. This check, which takes place before the final verify at the end of the programming cycle, is basically an "insurance" check, because it is performed at an elevated voltage, which is a worst-case condition. There is a flaw, however, in this type of programming algorithm: memory locations that have been previously programmed can be partially erased by programming subsequent locations (due to the elevated voltage on the same row or column in the memory array) and marginally programmed cells will go virtually undetected. The question is, doesn't the programmer check each device during verify after programming? Wouldn't those failures be caught then? Not necessarily, because when parts are checked during the program verify mode, the voltage is not elevated as high as it was during programming.

But higher yields and increased reliability aren't the only benefits the RAPID algorithm provides, it also takes less time to program these devices. The RAPID algorithm can reduce the programming overhead costs by a factor of 40! Here's how it works:

If you program an AT27C512R, 512K EPROM in a single-device programmer, using the FAST or any other type of 1 ms algorithm (1 ms initial pulse, plus 3 ms overprogramming pulse) the time spent programming will be:

$524288 \text{ bits} \div 8 \text{ bits/byte} = 65536 \text{ bytes}$

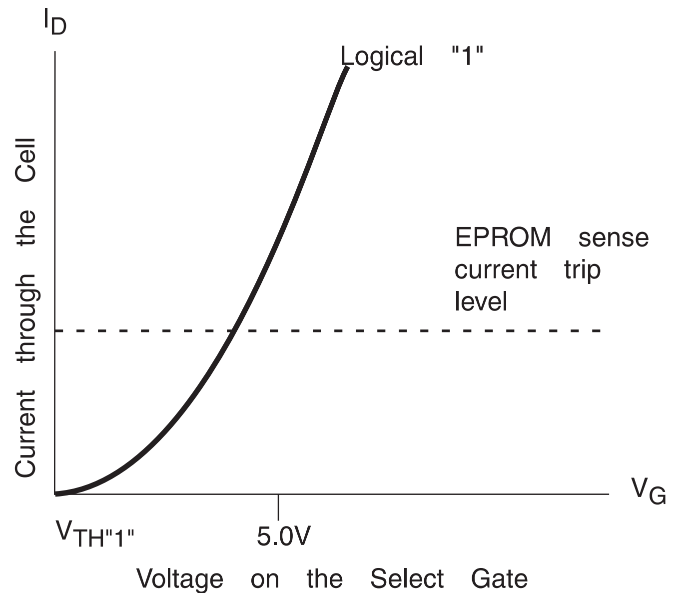
$65536 \text{ bytes} \times 0.004 \text{ seconds/byte} = 262 \text{ seconds}$

That's 262 seconds, or 4 minutes and 22 seconds. This works out to about a 75 cents programming cost, assuming an operator's rate of $10 per hour. Here's where the cost savings start: since we cannot reduce the number of bits to program, we reduce the total programming time by shortening the programming pulse width. Using 100 µs per byte, this is what happens:

$65536 \text{ bytes} \times 0.0001 \text{ seconds/byte} = 6.5 \text{ seconds}$

This amount of programming-time savings is what can be expected when using the RAPID algorithm. The big improvement is from reducing the total byte-programming time from 4 ms to 100 µs. With this example, total programming cost is about three cents. The RAPID algorithm can actually save up to 72 cents per device. Imagine how much can be saved with 10,000 EPROMs!

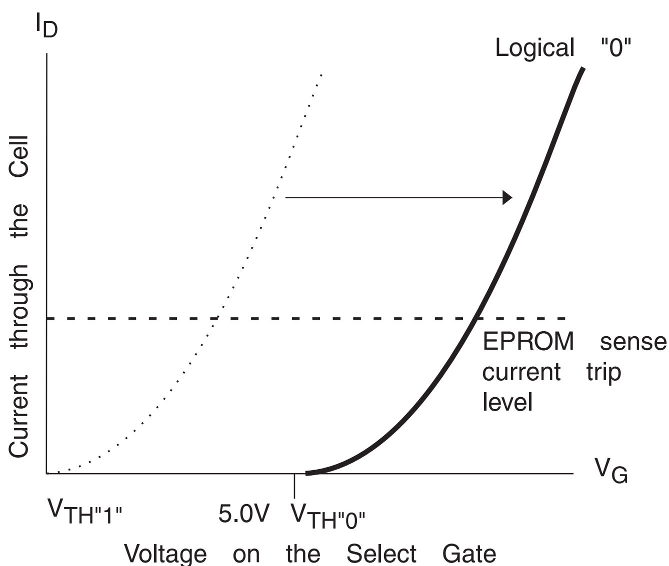
Figure 3. Unprogrammed Cell



The RAPID programming algorithm was designed to fix this oversight. First, it goes through the entire device and programs every cell without checking. Then it goes back to the beginning of the memory array and verifies the data in each cell at the elevated voltage. Once the device passes, another final verification is done at 5V. The RAPID algorithm will do a better job at preventing any marginally programmed parts from passing the programmer than other algorithms.

An important fringe benefit of the RAPID algorithm, because of the way it guarantees successful programmability, is long-term data retention. Basically, long-term data retention is how long the EPROM stays programmed, which is typically greater than ten years. Although long-term data retention is not the same as device programmability, they are related in this way: programmability tells how well the electrons have accumulated on the EPROM's floating gate, long-term data retention tells how long the electrons will stay there. The programming algorithm has an overwhelming influence on programmability, making it an overwhelming influence on long-term data retention as well. Therefore, a poor programming algorithm, one that doesn't guarantee programmability, can be responsible for poor long-term data retention. The RAPID algorithm can add years of data retention to your parts, because of the way it checks for programmability. Marginally programmed parts just don't stand a chance of getting past the programmer.

Figure 4. Programmed Cell. Note how V_{TH} raises after electrons are accumulated on the EPROM floating gate from programming.



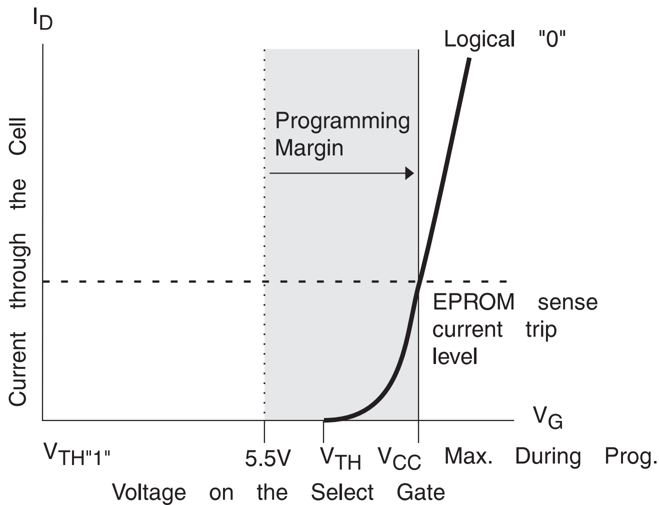
EPROM Programming, How it Works

Contemporary EPROM programming algorithms can be divided into two main sections, programming and verifying (or reading). Programming begins by selecting the desired voltage levels and byte address. It continues with a programming pulse applied to that byte, followed by a verify at the elevated V_{CC} used for programming. Verifying checks the data in two passes with the original data, with V_{CC} set to 5.5V on the first pass, and 4.5V on the second.

Basically, EPROMs are programmed through the accumulation of electrons on the floating gate of an N-Channel EPROM cell (see Figure 1) by the process of hot-electron injection. Hot-electron injection is where electrons, flowing as a current between the drain and source of a saturated EPROM cell, gain enough energy from the high electric field to jump the oxide barrier between the channel and the floating gate (see Figure 2). Before programming, the MOS threshold voltage, V_{TH} (otherwise known as the gate threshold voltage) of the erased floating-gate EPROM cell is about 1.0V to 2.0V (see Figure 3). After programming, its threshold voltage is about 6.5V to 9.0V, due to the accumulated electrons on the floating gate. In read mode, the address decoding circuitry in the chip selects the desired cell by pulling the gate voltage of the cell to V_{CC}. Since V_{CC} is typically 4.5V to 5.5V, an erased cell with a V_{TH} = 1.5V would be turned on (Figure 3), while a programmed cell with a V_{TH} = 7.5V would remain off (see Figure 4). This floating-gate process is how a single MOSFET-like transistor can provide for the two logic levels used in digital circuitry.

If V_{CC} is gradually raised in voltage to a point near the threshold voltage of a programmed EPROM cell, the cell would just begin to conduct, and would no longer appear to be programmed. This point, where the programmed EPROM cell begins to look unprogrammed, is defined as the programming margin (see Figure 5). The value of the programming margin can, in some cases, be simply equal to the value of the V_{CC} voltage present during programming. This is why the RAPID algorithm holds the value of V_{CC} constant at 6.5V during programming; to insure that each EPROM cell has a programming margin of at least that voltage. This margin is verified by reading each byte twice, once during the initial programming operation and again during the final read (or verify) operation, where the data from the EPROM is compared to the desired data. The difference between the value of V_{CC} during programming (the guaranteed programming margin) and the 5.5V V_{CC} maximum supply rating (from the data sheet) serves as a reliability guardband for long-term data retention and, more importantly, for system noise immunity. Poor programming margin can reduce system noise immunity and lead to EPROM chip instability due to power-supply noise on the V_{CC} pin. This instability can cause oscillations and read-mode data glitching that can be a problem in even in

Figure 5. Programming Margin. To find programming margin, increase gate voltage (V_{CC}) until the first "0" turns into a "1".

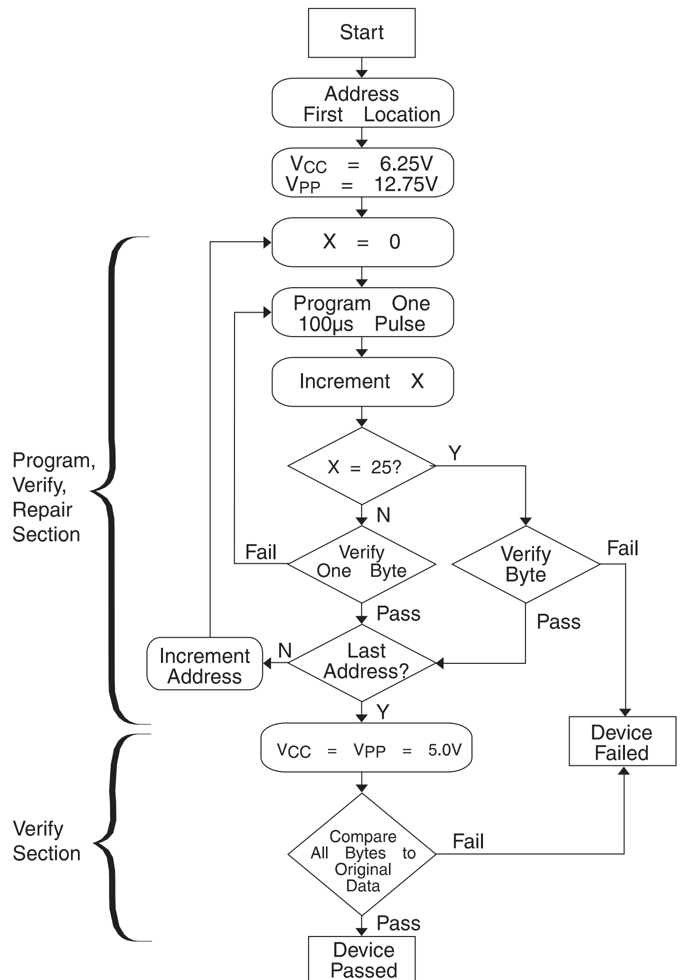


Guaranteeing Programmability

Most people might ask, "What's in a programming algorithm? Aren't they all the same?" That question would have been answered with a resounding "YES" 10 years ago when, quite frankly, they were the same. But it's not true today. There are over 20 manufacturers making EPROMs, and few of them use the same programming algorithm. Today, the programming algorithm is as important to EPROM testing as the actual device testing procedure. In fact, the device test procedures are often (if not always) based upon the programming algorithm. The programming algorithm has a direct effect on EPROM test yield, and manufacturers select their programming algorithms so they can obtain the highest yield possible. Additionally, the programming algorithm is directly responsible for the number of devices that pass the customers programmer, which is called programming yield. This is of vital importance to an EPROM manufacturer like Atmel, since the worst place for the slowest and most noiseless of systems. Since power-supply noise is a somewhat random occurrence, data errors can happen intermittently, which can undermine the reliability and integrity of the host system. These problems can be avoided by using the programming algorithm recommended by the EPROM chip vendor. The higher the guaranteed programming margin, the less likely any problems will occur.

Another important benefit of high-programming margin is that it extends the long-term data retention of the device. If the 6.0V programming margin (FAST algorithm) on the EPROM gradually diminishes to 5.5V over a 10-year time span, the randomly occurring noise spikes on the V_{CC} line can cause the EPROM to yield faulty data. On the other hand, given the same discharge rate (as a function of the silicon processing), an EPROM with a programming margin of 6.5V (RAPID algorithm) would take over 20 years to reach the 5.5V threshold that would lead to faulty data yield. All things being equal, better programming margin leads to longer data retention.

Figure 6. QUICK-PULSE Type



an EPROM to fail programming is in the customer's programmer. With this in mind, let's look at how the RAPID algorithm can guarantee better programmability than a common type of quick-pulsing algorithm.

We'll begin by comparing a common type of quick-pulsing algorithm with the Atmel RAPID algorithm. Examine Figure 6, which is the flowchart for the QUICK-PULSE type of algorithm. If you look very closely you will see that the algorithm is broken up in to two major sections. The main part is the program/verify section, the other part is the final verify section. Basically, the first section starts at byte address 0000H, programs the eight EPROM cells at that address, and verifies that those cells contain the correct data with a verify at 6.25V on V_{CC}. If the byte passes, it goes on to the next byte. If it fails, it repeats everything up to 25 times before it fails the device. The second section lowers the V_{CC} voltage to 5.0V and checks if all address locations read with the correct data. Although the flowchart specifies a one-pass final verify at 5.0V, many programmers verify in two passes, one with V_{CC} at 4.75V and the other with V_{CC} at 5.25V.

Now examine Figure 7, the Atmel RAPID algorithm. It looks similar to the quick-pulsing type of algorithm, but with a slight difference. If you look closely you'll see that it consists of three sections instead of just two. The first section is the programming section, where the programmer programs every location in the EPROM without verifying. Next there is the verify/repair section, where the programmer starts at the beginning of the EPROM and verifies every location for the correct data at 6.5V. Any cells that don't pass are reprogrammed up to ten times before the device is failed. The last section lowers V_{CC} to 5.0V and does a final verify of the data (here again, most programmers verify in two passes, one with V_{CC} at 4.75V and the other with V_{CC} at 5.25V). This type of programming algorithm is called a two-pass algorithm, because it goes through the memory array twice during programming.

Well, this all sounds fine, but what difference can the programming algorithm possibly make? We can find the answer to that question in a particularly sneaky deprogramming mode that EPROMs can exhibit. We all know that EPROMs are erased by exposing them to short-wave ultraviolet light, right? Nothing more than applying Einstein's discovery of the photoelectric effect. But there is another erasure mode that can occur, one that people in the E^2PROM business know about. If you were to examine some EPROM cells in an electron microscope, you might find a few that have small, tooth-like projections (called asperities) on the top of the floating gate polysilicon. These projections won't affect the normal operation of the EPROM, but they could give you problems during programming. When you program a row of cells on an EPROM, cells that have been previously programmed still feel the full brunt of the high V_{PP} voltage on their gates when subsequent cells on the same row are programmed, because all of the cells on a row have their gates connected together. The combination of high voltage on the gate and ground on the drain and source causes an intense electric field in each previously programmed cell. If any one of the cells on that row have these tooth-like projections on their floating gate polysilicon, the resulting electric field in the oxide above the projections will be much more intense than normal. This intensified electric field can give some of the electrons on the floating gate enough energy to jump the oxide barrier, thereby partially erasing the EPROM cell. This unwanted effect, called programming erase, can be responsible for poor programming margins unless the programming algorithm takes this problem into account.

Figure 7. RAPID Programming Algorithm

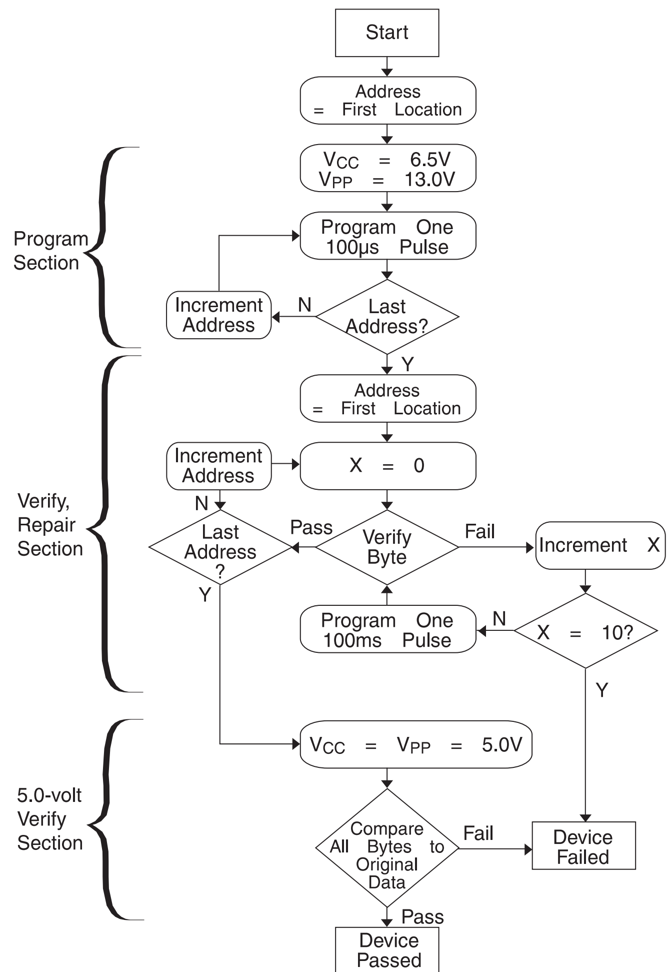


Figure 8. Row of EPROM cells from AT27C010. Note that the programming margin of each cell is 0, which allows each bit to read a “1”.

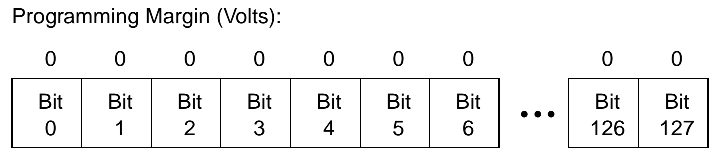


Figure 9. Bit 0 has been programmed, (QUICK-PULSE algorithm).

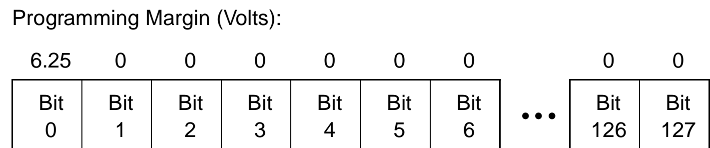
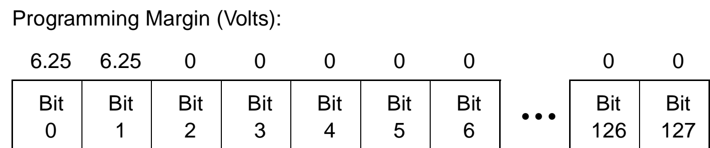


Figure 10. Bit 0 and bit 1 have been programmed.



Before we continue, it’s important to realize that this type of cell doesn’t have a **reliability** problem, it has a **programmability** problem. This cell will have the same long-term data retention as any other cell in the device, even if it loses part of its programming charge. Although it is an EPROM, it has the same charge retention characteristics as many manufacturers’ E^2PROM cells that use this type of erasure mode, and they all exhibit excellent long-term data retention. The challenge is to find these low-margin cells in the device with our programming algorithm, and to repair them so that the device functions normally.

Let’s see what kind of impact a cell like this can have on programming margin by programming a row of EPROM cells from our AT27C010 one-megabit EPROM with both algorithms. The array geometry on the one-megabit is 128 columns by 1024 rows, by 8 outputs. This means that a single row from a single output has 128 EPROM cells. Let’s say that the second cell on this row, bit 1 (we’ll call them bits and start with bit 0), has an asperity, just like the one mentioned above. When we go to program bits 2, 3, 4, etc., the voltage present on the gate of bit 1 causes the E^2PROM-like erasure mode. Given enough subsequent bits to program, bit 1 may lose enough charge to appear unprogrammed. Let’s take a look at how the QUICK-PULSE type of algorithm will fail the device, or even worse, pass it with poor programming margin. Then we’ll see how the RAPID algorithm will program it such that it works perfectly!

If we examine Figure 8, we see the row of EPROM cells taken from our AT27C010 one-megabit device. Recall that bit 1 is the cell that’s having the programmability problem, while the rest of the bits function normally. For the sake of example, let’s say that for each subsequent bit after bit 1 that’s programmed, bit 1 will lose eight millivolts (mV) of programming margin. Let’s also assume that the nominal programming margin for each cell is at least the value of V_{CC} present during programming, which is 6.25V for QUICK-PULSE type algorithms and 6.5V for the RAPID algorithm. Starting with the QUICK-PULSE type of algorithm at bit 0, we program it, verify it, and find that it passes (remember our flowchart from Figure 6?) with the correct margin (see Figure 9). We move to cell 1, program it, verify it, and it passes (see Figure 10). Remember, bit 1 only loses voltage margin when subsequent cells are programmed. Now we move on to bit 2, program it, verify it, and in the process reduce bit 1’s programming margin down to 6.242V (see Figure 11). Next we go to bit 3, program it, verify it, and in turn reduce bit 1’s programming margin down to 6.234V (see Figure 12). This process continues until we get to bit 127. By this time bit 1 has experienced 126 subsequent cell programming cycles, and its programming margin will be reduced to 5.242V (see Figure 13). Since the QUICK-PULSE type of algorithm does its high-voltage verify immediately after programming, the algorithm has no way of knowing what has happened to bit 1, once it finishes programming it. Only when the algorithm does its final verify with V_{CC} set at 5.25V could it detect that bit 1 is not fully programmed.

In this example we were able to detect bit 1 as being bad, and we would fail the device. But what if bit 1’s erasure rate was slightly less than 8 mV per subsequent cell, say 7.7 mV? Bit 1’s margin might be somewhere around 5.3V, which would probably pass the 5.25V verify check on our programmer. But remember the problem that we discussed earlier, about the power supply noise glitches messing up the operation of devices with low programming margin? A device with only 5.3V of margin is a prime candidate for this type of problem. A small noise glitch occurring during data access on the V_{CC} line of this EPROM could easily change the output from a “0” to a “1”. And, to make matters worse, this problem would probably occur randomly; the eventual diagnosis being that the device was intermittent. The unfortunate truth is that there is nothing wrong with the EPROM, it’s the programming algorithm that’s at fault.

So let’s go back to our row of 128 EPROM cells, erase them, and reprogram them with the RAPID algorithm. Remember that with the RAPID algorithm the initial pro-

Figure 11. Bits 0, 1, and 2 are programmed. Notice that bit 1 has been slightly erased.

Programming Margin (Volts):

6.25	6.242	6.25	0	0	0	0		0	0
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	•••	Bit 126	Bit 127

Figure 12. Bit 3 has just been programmed. Notice that bit 1 has been further erased.

Programming Margin (Volts):

6.25	6.234	6.25	6.25	0	0	0		0	0
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	•••	Bit 126	Bit 127

Figure 13. The entire row has been programmed. Notice how much bit 1 has been erased.

Programming Margin (Volts):

6.25	5.242	6.25	6.25	6.25	6.25	6.25		6.25	6.25
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	•••	Bit 126	Bit 127

Figure 14. Bit 0 has just been reprogrammed (RAPID algorithm).

Programming Margin (Volts):

6.50	0	0	0	0	0	0		0	0
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	•••	Bit 126	Bit 127

Figure 15. Bit 1 has just been programmed.

Programming Margin (Volts):

6.50	6.50	0	0	0	0	0		0	0
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	•••	Bit 126	Bit 127

Figure 16. Bit 2 has just been programmed. Notice how bit 1 has been slightly erased again.

Programming Margin (Volts):

6.50	6.492	6.50	0	0	0	0		0	0
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	•••	Bit 126	Bit 127

gram and verify routines are located in different sections of the algorithm, they are not contained within the same loop. Starting at bit 0, we program it (to 6.5V this time, see Figure 14). Then move to bit 1, and program it (see Figure 15). Next to bit 2, program it, and in turn reduce bit 1's programming margin to 6.492V (see Figure 16). Then on to bit 3, program it, and further reduce bit 1's programming margin to 6.484V. We continue programming until we get to bit 127, and you'll find that the programming margin for all the cells looks similar to figure 8 (see Figure 17). But wait, we're not finished yet. We move back to the beginning of the EPROM array, which is bit 0, and verify that it has 6.5V of programming margin. Since we are verifying at 6.5V, we pass it. We now move to bit 1 and notice that its programming margin is 5.492V. This fails our 6.5V verify, so we program it one more time and raise its margin back to 6.5V, then pass it (see Figure 18). Then we move to bit 2, and pass it, since its programming margin is also 6.5V. Notice that we didn't deprogram bit 1 in the process of verifying bit 2. We only deprogram bit 1 when we program subsequent cells; reading or verifying (which is reading) doesn't generate the intense electric fields needed to deprogram EPROM cells. After verifying (and repairing) this row of cells, we return V_{CC} to 5.25V, do a final data verify, then pass the row (see Figure 18). Now compare Figure 18 with Figure 13. That's how the RAPID algorithm can guarantee programmability!

Well, you may ask, what if we had five problem cells on the same row? Wouldn't the additional programming pulses during the verify function deprogram previously programmed and verified cells? They probably would, but the maximum amount of deprogramming on the first bit (using this model) would be only 32 mV (4 x 8 mV). This gives us a programming margin of 6.468V, which is still an excellent programming margin.

When you compare the QUICK-PULSE type algorithms to the RAPID algorithm, there really is no comparison. The RAPID algorithm simply guarantees programmability, and we demonstrated this with the deprogramming bit example,

which is one of the trickiest programming problems you can have. But the RAPID algorithm caught the problem, and repaired the bit so that the EPROM will function normally.

Figure 17. The entire row has just been programmed. Notice how much bit 1 has been erased.

Programming Margin (Volts):

6.50	5.492	6.50	6.50	6.50	6.50	6.50		6.50	6.50
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	•••	Bit 126	Bit 127

Figure 18. The entire row has just be verified at 6.50 volts. Notice how bit 1 has been repaired, its margin being returned to 6.50 volts using the RAPID algorithm.

Programming Margin (Volts):

6.50	6.50	6.50	6.50	6.50	6.50	6.50		6.50	6.50
Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	•••	Bit 126	Bit 127

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel U.K., Ltd.
Coliseum Business Centre
Riverside Way
Camberley, Surrey GU15 3YL
England
TEL (44) 1276-686677
FAX (44) 1276-686697

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road
Tsimshatsui East
Kowloon, Hong Kong
TEL (852) 27219778
FAX (852) 27221369

Japan
Atmel Japan K.K.
Tonetsu Shinkawa Bldg., 9F
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4 42 53 60 00
FAX (33) 4 42 53 60 01

Fax-on-Demand
North America:
1-(800) 292-8635
International:
1-(408) 441-0732

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

BBS
1-(408) 436-4309

© Atmel Corporation 1998.
Atmel Corporation makes no warranty for the use of its products, other than those expressly contained in the Company's standard warranty which is detailed in Atmel's Terms and Conditions located on the Company's website. The Company assumes no responsibility for any errors which may appear in this document, reserves the right to change devices or specifications detailed herein at any time without notice, and does not make any commitment to update the information contained herein. No licenses to patents or other intellectual property of Atmel are granted by the Company in connection with the sale of Atmel products, expressly or by implication. Atmel's products are not authorized for use as critical components in life support devices or systems.

Marks bearing ® and/or ™ are registered trademarks and trademarks of Atmel Corporation.

Terms and product names in this document may be trademarks of others.

Printed on recycled paper.

0578A–10/98/xM