TECHNICAL
PAPER

Improving Programming Throughput of Automated Flash Memories

February 1997

Order Number: 297769-002

CONTENTS

PAGE

1.0 INTRODUCTION 5

2.0 PROGRAMMING IMPLEMENTATIONS 5

2.1 Key Considerations 5
2.2 Reducing Off-Line Programming Expenses . 5
2.3 Benefits of Integration into Manufacturing Flow 6

3.0 REVIEW OF ON-CHIP AUTOMATION..... 6

3.1 Command User Interface 6
3.2 Write State Machine 6
3.3 Data Comparator 7
3.4 Status Register 9

4.0 OBSOLETING EPROM LEGACY PRACTICES–CHANGING THE PARADIGM . 9

4.1 Internal Program and Program Verification 10

5.0 ARRAY BLOCKING PROVIDES FLEXIBILITY TO SPEED PRODUCTION THROUGHPUT 11

5.1 Intel Array Blocking Options 11

6.0 WAVEFORM, TIMING, VOLTAGE, BUS WIDTH OPTIMIZATIONS 12

6.1 Faster Programming at 12V V_{PP} and 5V V_{CC} 12
6.2 Hold CE# Low Throughout Programming Cycle 13
6.3 Command Sequence Length Affects Programming Times 14
6.4 Timings and Address Cycling 14
6.5 Programming Word Wide–2x Faster Than Byte Mode 17

PAGE

7.0 WSM READY/BUSY INDICATION MODES .. 17

7.1 Monitor RY/BY# Output vs. Status Bit SR.7 Polling 17
7.2 Check Full Status after Completing all Program/Erase Operations 17

8.0 IMPACT OF HARDWARE DESIGN ON PROGRAMMING SPEED 19

8.1 "Intelligence" Proximity to Chip 19
8.2 Pin Driver Proximity to Chip 19
8.3 Buffer Proximity to Chip 19
8.4 Communication Bandwidth 19

9.0 USING SUPERSET DEVICE WRITE-PERFORMANCE FEATURES 19

9.1 Command Queue 21
9.2 Page Buffers 21
9.3 Extended Status Registers 21
9.4 Erase Queue and Erase All Unlocked Blocks Command 21

10.0 CONCLUSION 21

APPENDIX A: Additional Information 22

REVISION HISTORY

Number	Description
-001	Original version
-002	Made corrections to Figure 1 Added Internal Program Verification Circuit Diagram (new Figure 2) Updated Table, V_{CC} = 5V ± 0.5V, 5V ± 0.25V, T_A = 0°C to +70°C Added Table, V_{CC} = 3.3V ± 0.3V, T_A = 0°C to +70°C

1.0 INTRODUCTION

This paper sheds light on concepts that can be employed to accelerate programming of write-automated flash memories in manual/automated device programmers, Automated Test Equipment (ATE; i.e., board testers), JTAG/HSS (High Speed Serial) implementations, and in-system write applications.

Faster programming improves device throughput, an especially important attribute in manufacturing environments where the main thrust is to do things "faster, better, cheaper." Improved throughput increments, by as small a factor as even 10%, are very admirable when accumulated across large volume. By achieving 20–40%, the production line manager/ programming supervisor is bestowed recognition as a champion by the team. Greater than 50%, and that person reigns as hero in the battle to cut costs.

In this report, we'll examine many areas into which we can inject improvement. The following topics are covered:

- The two basic programming implementations, the individual methods that can be employed, and key considerations.
- On-chip program/erase automation.
- Obsoleting "EPROM legacy" practices.
- Internal program/erase verification.
- Programming subset of device array.
- Waveform, timing, voltage level, command sequence length and bus width optimizations
- Hardware ready/busy monitoring vs. status register polling.
- Full status check after a string of programs or block erase operations.
- Impact of hardware design.
- Superset write enhancement features.

2.0 PROGRAMMING IMPLEMENTATIONS

Several methods exist to program flash memories. Two broad implementation categories cover all methods: off-line and in-line.

An off-line implementation is any programming method that's performed outside the main manufacturing flow. This can be accomplished via gang, concurrent, or board-level programmers, automated programming systems (handler), or JTAG/HSS solutions. Off-line programming can be performed in-house or off-site. It can even involve in-system writes via the local processor; this usually requires some pre-programming of boot code in a previous step via any of the above methods or ATE.

In-line programming occurs in the main manufacturing flow. This can involve ATE during board test, component/automated programming equipment just prior to or integrated into the pick-n-place system, or some JTAG/HSS approach or board-level programmer at some point past assembly.

2.1 Key Considerations

Important to link with any programming consideration is the fact that, as flash densities increase, tightly-leaded Small Outline Packages become standard chip housings industry wide. And with this change, tape-and-reel is becoming the choice shipping media because of its cost advantages and because it fits/integrates into pick-n-place systems better than tray.

When production line managers ponder the issues surrounding flash memory programming, packaging and shipping media, two obvious questions arise: Which is more efficient/economical for my environment—programming off-line or in-line? Can board-level programming resolve my concerns—eliminate bent leads, facilitate tape-and-reel, reduce costs?

A variety of paths are available to opt from; some point to large-ticket items, e.g., utilization of existing ATE or investment in a new programmer-handler, while others involve less expensive solutions. Possessing in-depth knowledge of the capabilities and flexibility of today's automated flash memories affords you the necessary insight to make the best implementation choice for your particular environment.

2.2 Reducing Off-Line Programming Expenses

With off-line programming, whether in-house or contracted out, expenses are directly proportional to programming times. External expenses equate to $$/hour for equipment usage and manpower spread over units/hour. If units/hour increase, then cost/device should decrease. Done in-house, there are direct

(equipment, manpower, etc.) and hidden (software updates, maintenance, etc.) costs, both being positively influenced by greater throughput.

2.3 Benefits of Integrating Programming into the Manufacturing Flow

Fast programming times can facilitate integration into the manufacturing flow, so long as the line's beat rate is not hindered.

Existing ATE can be used to program at board test. This has several advantages:

- Eliminates capital expenditure for separate programming systems and the costs associated with keeping that equiment up-to-date.
- Reduces manpower requirements to operate and maintain separate programming systems.
- Facilitates shipment of product in lower-cost tape-and-reel media.
- Facilitates acceptance of product in a fine-pitch package— reduces potential lead distortion.
- Facilitates just-in-time programming—eliminates inventory of programmed units.

JTAG and High Speed Serial Programming (HSS) share the same advantages as ATE. Implementing JTAG/HSS, on the other hand, requires much less real estate—typically only four or five contact points, compared to an average of 48 for ATE.

3.0 REVIEW OF ON-CHIP AUTOMATION

Today's standard flash memory has on-chip program and erase automation circuitry. See Figure 1 for an example of the 28F800BV block diagram. This automation performs all operations that required external control on non-automated first-generation flash technology via the following main functional units:

- Command User Interface (CUI)
- Write State Machine (CUI)
- Data Comparator
- Status Register

It is this automation, included in second- and later-generation flash devices, specifically steps performed by the WSM and its support circuits, that allows us to shift beyond the EPROM programming paradigm. This paradigm being algorithm practices which were standard in EPROM programming routines, but are unnecessary for modern automated flash memories.

Before we move on, let's explore the activities that occur with and between the CUI, WSM, data comparator and status register in greater detail.

3.1 Command User Interface

The CUI is the outside world's request interface, its basic job is to arbitrate between system processor and internal device functions. The CUI serves this role via a command register to hold the issued request, a command decoder to interpret/translate that request, and the control logic to initiate it. Activities include CUI-WSM communications, read-path selection, and status register checking and clearing.

The CUI resides on the internal data bus. Commands are input on data pins DQ_{0-7} with CE# and WE# driven low, then get latched and interpreted after WE# is returned to V_{IH}. Address information is captured in the address latch and program data secures in the data register when WE# is driven high in the second bus cycle.

When the issued command is a read operation, e.g., from the ID or status registers or the memory array, the CUI ensures the output multiplexer gates that data to the output buffers.

3.2 Write State Machine

The WSM consists of an integrated oscillator and control circuitry to carry out program and erase operations. CUI-to-WSM signaling forwards translated user requests for processing and control. The WSM then generates signals that:

1. Initiate a strobe to bits requiring program or the block to erase.
2. Supervise strobe pulsewidth and associated timings.
3. Control the data comparator.
4. Request feedback from the data comparator to determine pulse repetition control and provide update to the status register.
5. Initiate an address counter for erase preconditioning or erase verify.

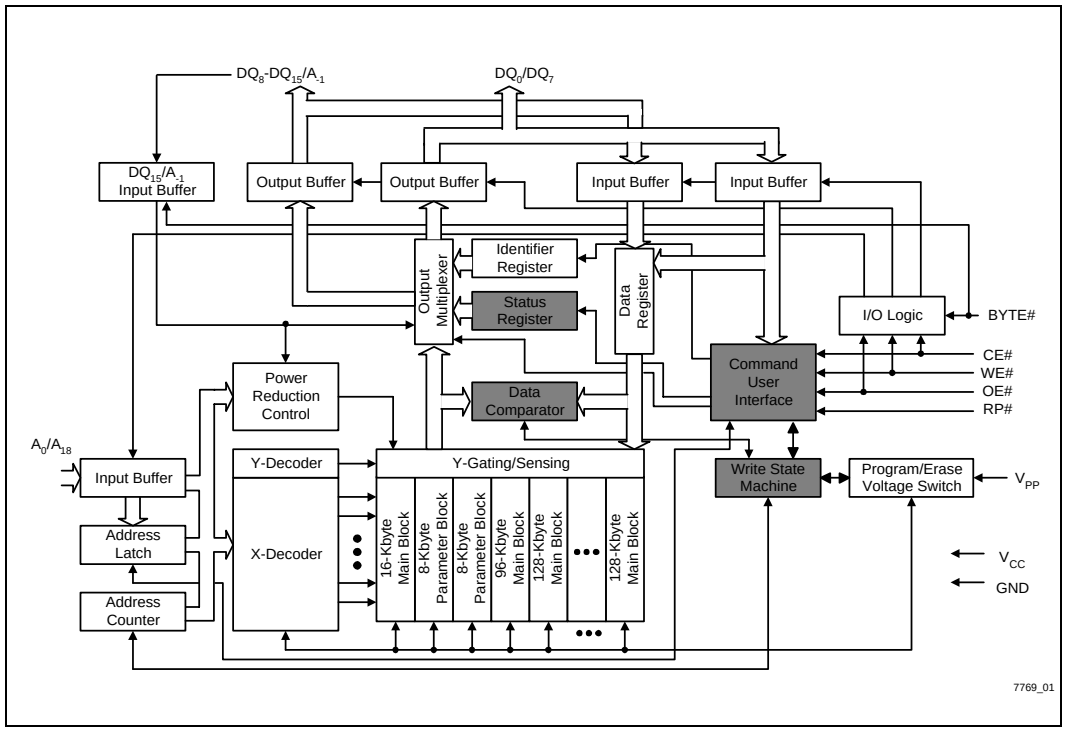


Figure 1. Automated Flash Memory Enhancements Enable Faster, Easier Programming

3.3 Data Comparator

The data comparator is controlled by the WSM. The WSM employs the data comparator during program and erase operations. The data comparator collates just-programmed cells against data stored in the device data register. See Figure 2 for expanded detail of 28F016SA program verify functionality. For erase, it compares erased locations against data value FF/FFFFh. Erased locations are cycled through the data comparator via an address counter circuit.

The data comparator reports the results of its collation to the WSM, which in turn determines if pulse repetition is required. If pulse repetition is not required, the WSM sends a signal to update the status register program or erase status bit.

Let's examine the margining function that occurs just ahead of the data comparator. This example is for a program operation:

> Current I_{PMRGN}, derived from a program margin bias read of the column containing the programmed cell, is fed into a sense amplifier with reference current I_{PREF} (current from a factory-set program reference circuit). If $I_{PMRGN} < I_{PREF}$ the sense amp outputs a zero. If $I_{PMRGN} > I_{PREF}$ the sense amp outputs a one. This operation occurs 8 or 16 in parallel depending on the bus width of the program operation. The 8 (16) output values are sent to the data comparator for collation against data stored in the data register.

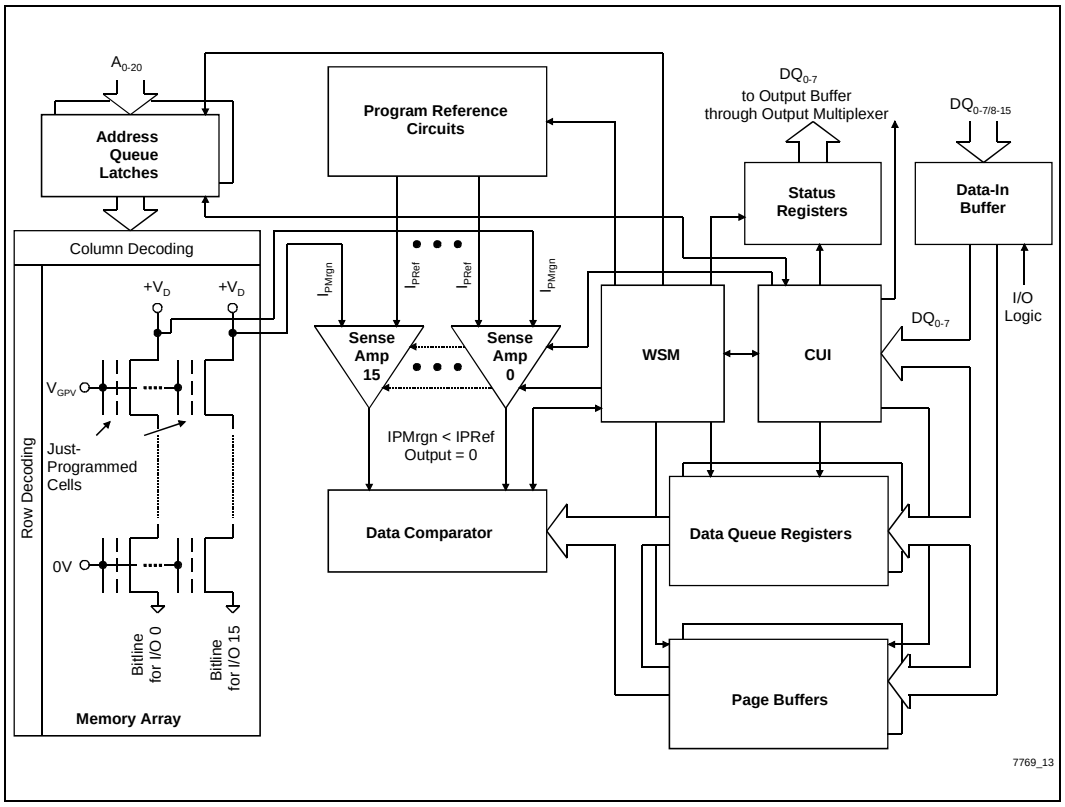


Figure 2. Internal Program Reference Circuits Provide a More Finely-Tuned Verify Than That Which Could Be Achieved Externally

Table 1. 28F800BV Status Register Bit Definition

WSMS	ESS	ES	PS	VPPS	R	R	R
7	6	5	4	3	2	1	0

	NOTES
SR.7 = WRITE STATE MACHINE STATUS (WSMS) 1 = Ready 0 = Busy	Check WSMS to determine Word/Byte Program or Block Erase completion before checking the Program or Erase Status bits.
SR.6 = ERASE-SUSPEND STATUS (ESS) 1 = Erase Suspended 0 = Erase In Progress/Completed	When Erase Suspend is issued the WSM halts execution and sets both WSMS and ESS to "1." The ESS bit remains set to "1" until an Erase Resume command is issued.
SR.5 = ERASE STATUS (ES) 1 = Error In Block Erasure 0 = Successful Block Erase	When ES is set to "1" the WSM has applied the max number of erase pulses to the block and is still unable to verify successful erasure.
SR.4 = PROGRAM STATUS (PS) 1 = Error in Word/Byte Program 0 = Successful Word/Byte Program	When PS is set to "1" the WSM has attempted but failed to program a word or byte.
SR.3 = V_{PP} STATUS (VPPS) 1 = V_{PP} Low Detect, Operation Abort 0 = V_{PP} OK	VPPS does not provide continuous indication of V_{PP} level. The WSM interrogates V_{PP} level only after the Program or Erase command sequences have been entered, and informs the system if V_{PP} has not been switched on. VPPS is not guaranteed to report accurate feedback between V_{PPLK} and V_{PPH}.
SR.2–SR.0 = RESERVED FOR FUTURE ENHANCEMENTS (R)	These bits are reserved for future use and should be masked out when polling the Status Register.

3.4 Status Register

The status register is the automation circuitry's other interface to the outside world. The WSM receives feedback from its support circuits, thereby allowing it to keep the status register current and CUI abreast. Table 1 shows that status bits provide ready/busy, operation success/error, and V_{PP}/suspend status indications.

Status register contents are driven out on DQ_{0-7} at the falling edge of CE# or OE#, whichever occurs last in the read cycle. Either pin must be driven high, then low again to send updated content to the output buffers.

4.0 OBSOLETING EPROM LEGACY PRACTICES–CHANGING THE PARADIGM

EPROM legacy practices, i.e., verification of each location as it's written and the final two-pass comparison of all words to original data, should be eliminated from the programming routines of automated flash memories.

The WSM and its supporting circuits perform internally-margined verification of all words as they're written. As such, it is redundant for a programming system to do the same. In fact, internal device margin settings are much more finely tuned for the sensing of program thresholds.

NOTE:

If any concerns exist over the programming hardware's buffer-to-flash data transfer integrity, it's suggested that a single-pass (nominal V_{CC}) post-program verify operation be performed.

Figure 3 compares an EPROM legacy algorithm to the 28F800BV automated flash memory flowchart. The individual word verify step, the $\pm V_{CC}$ post-program compare operation, and the pulsecount incrementing and checking functions have been removed. Section 4.1 provides the supporting text for these recommendations.

Blank check, an added step, is typically unnecessary in production environments. Intel Flash chips ship erased, so production programming systems need only erase devices in instances when test code is replaced or production code has recently been updated. Blank check should be made an option that can be turned off by the equipment operator.

4.1 Internal Program and Program Verification

After successful receipt and interpretation of the requested program operation, the CUI forwards a translated signal to the WSM. The WSM then supervises internal program and verify-path support circuits to perform the following tasks:

- program pulse control
- pulse-repetition control
- time-out control
- program verification
- status register update

Program verification occurs in two steps:

1. A margined-sensing scheme applies an elevated read voltage to just-programmed cells, the resulting bitline currents then feed individually into 16 sense amplifiers. The output of factory-set program reference circuits, adjusted to V_{tp} (program-threshold voltage), also feeds into the sense amplifiers.
2. The output of the sense amplifiers then route into a data comparator for collation. This collation compares sense amp output to data register content.

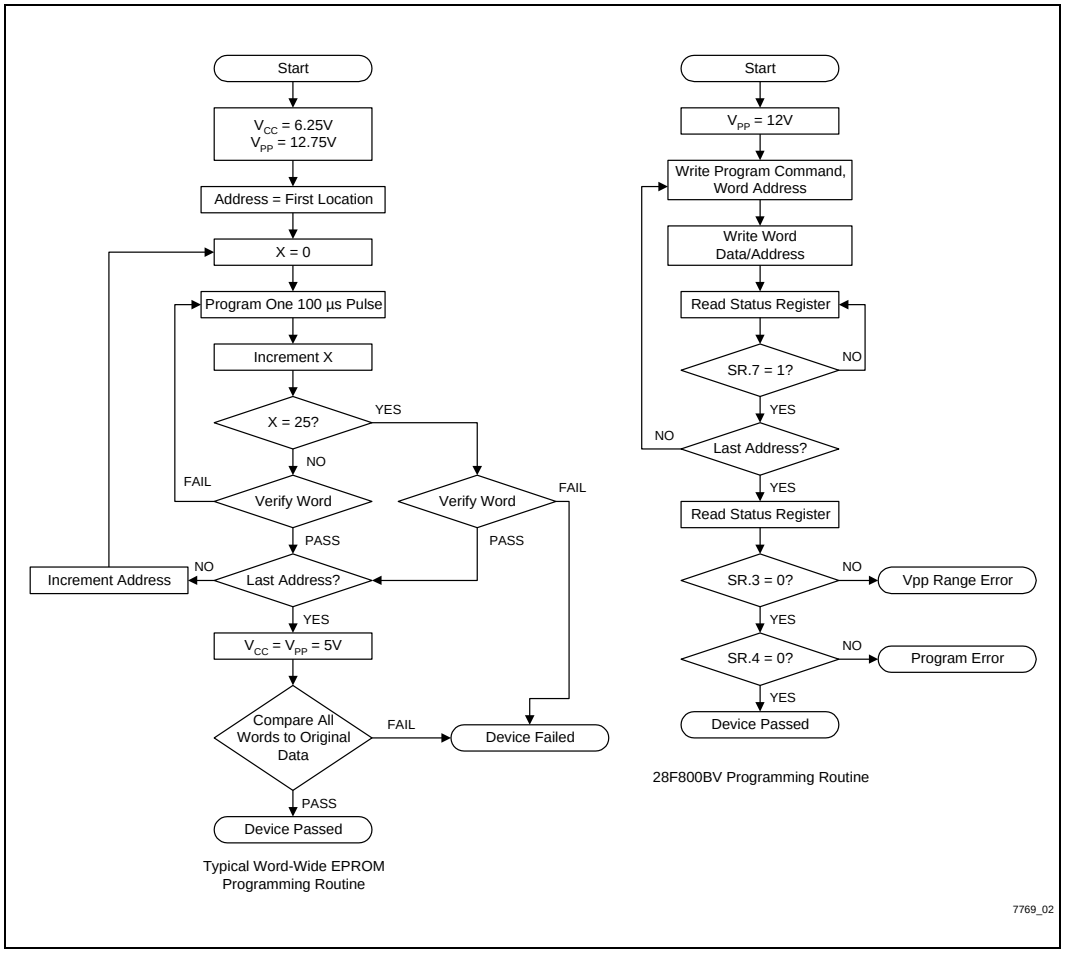


Figure 3. Major Time Savings: Word-by-Word Verify and Post-Program Compare Are Not Necessary for Automated Flash Memories

5.0 ARRAY BLOCKING PROVIDES FLEXIBILITY TO SPEED PRODUCTION THROUGHPUT

Memory blocking allows customers to segment code and data, and provides a means by which programming tool vendors can offer flexible solutions.

For example, if requirement is to program a subset range of the array, then it's wasted time/effort to write null data to remaining locations, worse yet if a post-program verify operation compares null locations to buffer. Additionally, full chip erase is wasted time/effort if only a portion, e.g., one block, requires update. The other blocks would then require re-write of the existing data. The programming tool can take advantage of this flexibility provided it has a user interface allowing selection of a range to program or blocks to erase.

There's a huge benefit to program the minimum amount of memory (enough to boot application), where afterwards in-system writes can take place. This may make feasible, programming on an ATE system that has a limited window of availability.

5.1 Intel Array Blocking Options

The high-integration boot block architecture has an asymmetrical segmentation with specialized block sizes. There's a dedicated hardware-locked block available at the top or bottom of the device memory map, two small parameter blocks for parameter storage or EEPROM emulation, and large main blocks, the count depending on device density. See Figure 4 for the 28F800BV memory map.

Intel high-density FlashFile™ and high-performance Fast Flash memories have symmetrically-sized blocks.

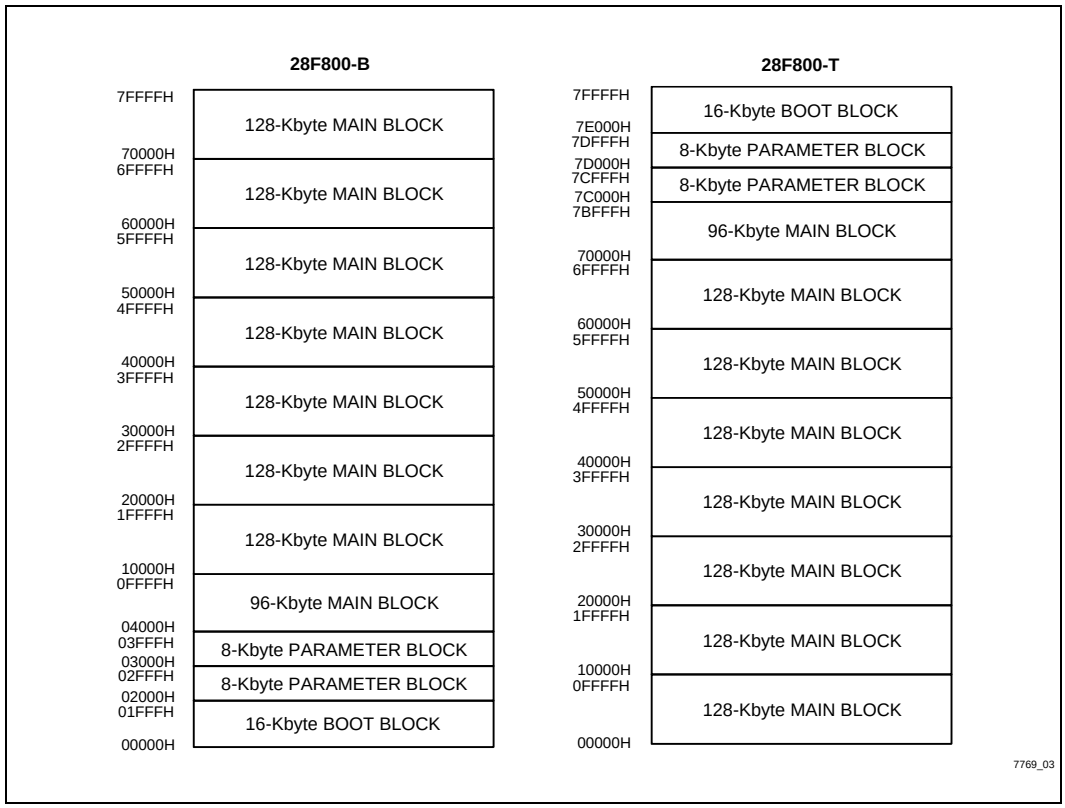


Figure 4. Flexibility to Program a Portion of the Device during Manufacturing, Such as the Boot Block, Improves Throughput

6.0 WAVEFORM, TIMING, VOLTAGE, BUS WIDTH OPTIMIZATIONS

In this section, we'll explore the advantages of higher V_{CC} and V_{PP} levels, x16 data writes and holding the CE# input low. Additionally, we'll look at the affects of command sequence length and system timing capabilities.

6.1 Faster Programming at 12V V_{PP} and 5V V_{CC}

Intel's SmartVoltage technology produces extremely versatile flash memories with a range of V_{CC} and V_{PP} options. The new 28F008SC SmartVoltage FlashFile memory, for example, includes 2.7V/3.3V/5V V_{CC} read and 3.3V/5V/12V V_{PP} programming. SmartVoltage devices are ideal for power-sensitive customer applications; they operate at lower voltage in-system, but are capable of programming at 12V V_{PP} for manufacturing.

As an example, the 28F008SC programs 64-Kbytes of data in 0.3 seconds minimum with 12V V_{PP} and 5V V_{CC}; block erase equals 0.3 seconds minimum at this setting. These times are significantly faster than those at lower voltages. Tables 2 and 3 show that 28F008SC program and erase performance is best with 12V V_{PP} and 5V V_{CC}.

Of course, the benefits of 12V V_{PP} are lost when software overhead hides them. This could be the case when WSM RY/BY# status is determined by polling status register bit SR.7. To take maximum advantage, the programming system should, after the second WE# rising edge, drive OE# low (to read SR.7) at the typical datasheet value for that device's program time (t_{WHQV1}).

Table 2. V_{CC} = 3.3V ± 0.3V, T_A = 0°C to +70°C

Sym	Parameter	Notes	3.3V V_{PP}			5V V_{PP}			12V V_{PP}			Unit
			Min	Typ(1)	Max	Min	Typ(1)	Max	Min	Typ(1)	Max	
t_{WHRH1}, t_{EHRH1}	Program Time	2	15	17	TBD	8.2	9.3	TBD	6.7	7.6	TBD	µs
	Block Write Time	2	1	1.1	TBD	0.5	0.5	TBD	0.4	0.5	TBD	sec
t_{WHRH2}, t_{EHRH2}	Block Erase Time	2	1.5	1.8	TBD	1	1.2	TBD	0.8	1.1	TBD	sec

Table 3. V_{CC} = 5V ± 0.5V, 5V ± 0.25V, T_A = 0°C to +70°C

Sym	Parameter	Notes	V_{PP} = 5V			V_{PP} = 12V			Unit
			Min	Typ(1)	Max	Min	Typ(1)	Max	
t_{WHRH1}, t_{EHRH1}	Program Time	2	6.5	8	TBD	4.8	6	TBD	µs
	Block Write Time	2	0.4	0.5	TBD	0.3	0.4	TBD	sec
t_{WHRH2}, t_{EHRH2}	Block Erase Time	2	0.9	1.1	TBD	0.3	1.0	TBD	sec

NOTES:

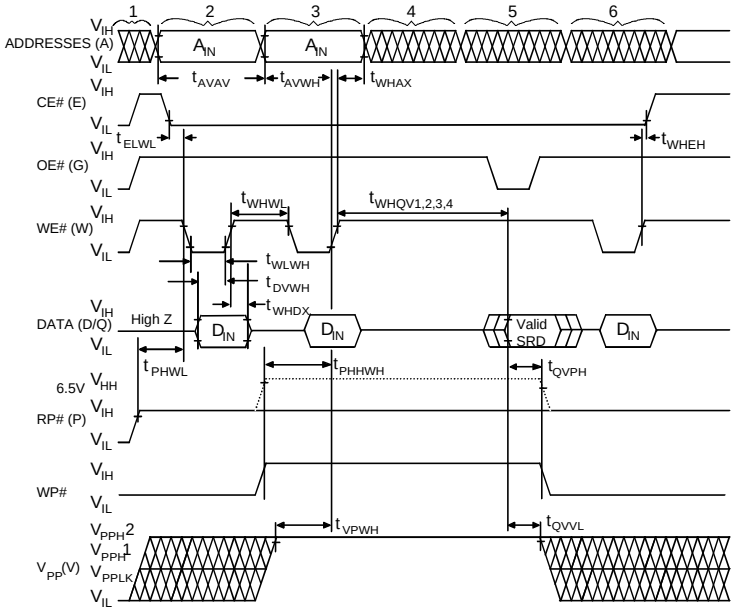
1. Typical values measured at T_A = +25°C and nominal voltages. Assumes corresponding lock-bits are not set. Subject to change based on device characterization.
2. Excludes system-level overhead.
3. These performance numbers are valid for all speed versions.
4. Sampled, but not 100% tested.
5. For more information, see the *Byte-Wide SmartVoltage FlashFile™ Memory Family 4, 8, 16 Mbit Datasheet,* literature number 290600.

6.2 Hold CE# Low Throughout Programming Cycle

CE# can be held at V_{IL} during all phases of the programming operation—command in, data in (WE# low) and status check (OE# low), throughout all address transitions. See Figure 5 for a modified WE#-controlled write waveform.
Eliminating needless CE#-to/from-WE#/OE# setup/hold times removes a significant amount of programming system overhead, the level of this factor determined by the device programmer or ATE system signal transition capability.

Note that 2–5 are the cycles that repeat for each location to be written. Cycle 6 occurs after the last location gets programmed, and could also be a power-down cycle.

Additionally, the address bus can be held constant throughout a location's programming cycle except in instances when a specific address is required for status register reads. This would be the case if a global or block status register (see Section 9.3 concerning Extended Status Registers) is requested or the status register is mapped at a unique location. In this situation, the waveforms would show either a read status register command cycle occurring during the automated program delay (cycle 4) and/or a specific A_{IN} for cycle 5.



NOTES:
The depicted cycles are:
1. V_{CC} power-up and standby
2. Write Program command
3. Issue data to program
4. Automated program delay
5. Read status register data
6. Write Read Array command

Figure 5. CE# Does Not Have to Toggle, with Associated Set-Up and Hold Times, around Every Write or Read Cycle

6.3 Command Sequence Length Affects Programming Times

Flash memory command sequence length (i.e., number of bus cycles) has direct repercussions on the performance capabilities of any programming system. Parts which require long, drawn-out command sequences negatively impact themselves in any programming method by adding to the degree of overhead that must be implemented to support that device. The effect of this is especially felt in JTAG/HSS serial-stream programming where each pin state must be shifted in serially.

For example, if each address requires a two-cycle sequence to unlock writes, in addition to the command and data-in bus cycles, overhead doubles in JTAG/HSS methodologies. Even systems that access device pins in parallel are slowed.

Additionally, Intel Flash memories do not require another command sequence to allow WSM ready/busy status bit indication out of the device. They go directly into status read mode after initiation of program or erase operations, saving overhead.

6.4 Timings and Address Cycling

Programming equipment and implementations that can achieve setup, hold, pulsewidth, delay and address cycle times at or near device datasheet minimums will attain device programming times close to theoretical.

Years back, when memory densities were small, lead pitch was in the 50–100 mil range, and competition not so fierce, device programmer support times were not an issue. Today, with larger arrays, smaller packaging, and the need cut costs, programming times become increasingly important and now shift into the forefront of awareness. Device programmer manufacturers lacked sufficient motivation to optimize support to its fullest potential. Besides, slower algorithms might require the purchase of additional systems. An informed consumer is much better prepared to make good choices, and to work with their equipment supplier to meet their needs.

Towards this goal of supplying cost-effective solutions to its customers, Intel is currently working with device programmer manufacturers and vendors of other solution types to optimize their algorithms.

In September 1996, benchmark experiments were performed using the 28F200BV on three industry-standard programmers. Erase, program, verify and total time were recorded utilizing a stop watch. Figures 6 through 9 are provided as examples of the recorded ranges.

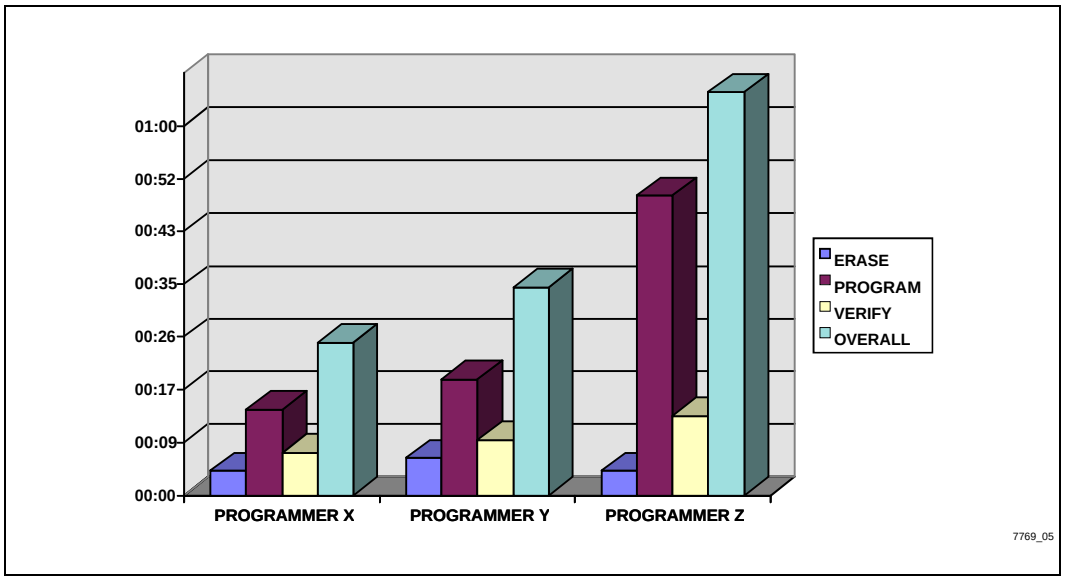
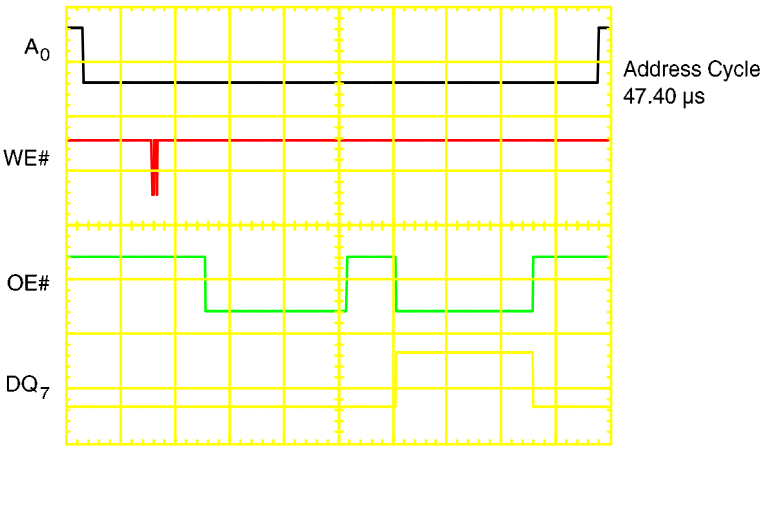


Figure 6. Erase, Program, Verify Times Vary across Programmers

	Erase Seconds	Program Seconds	Verify Seconds	Overall Minutes
Programmer X	4	14(1)	7	0:25
Programmer Y	6	19(1)	9	0:34
Programmer Z	4	49(1)	14	1:06

NOTE:

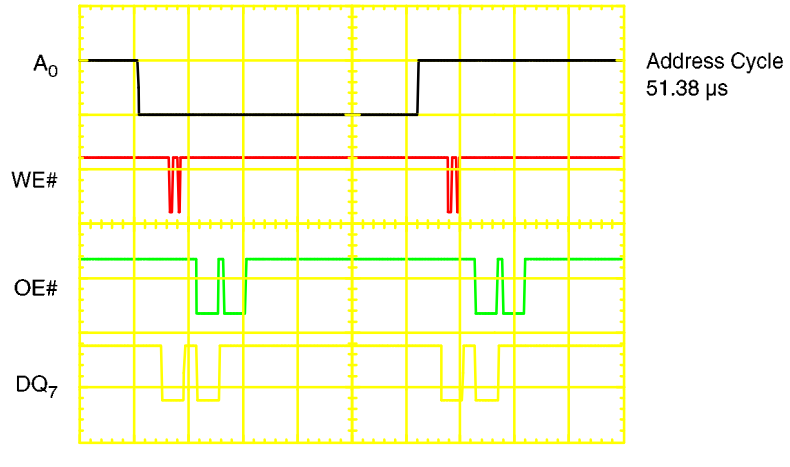
1. All memory cells were programmed to zero state.



NOTE:

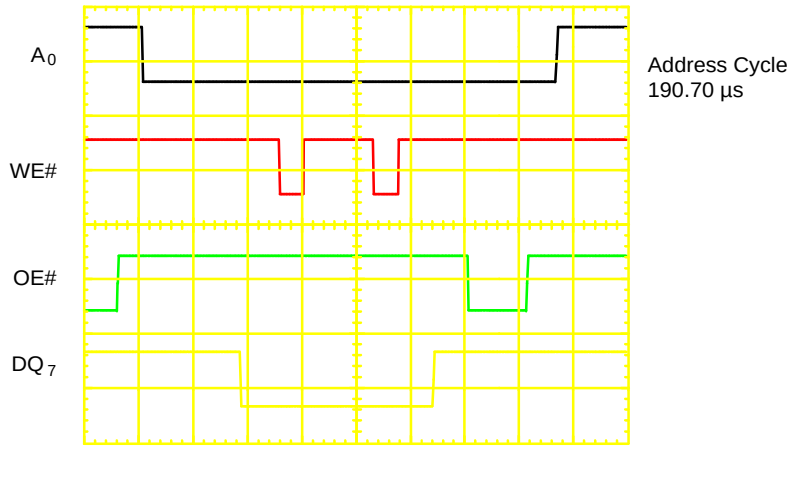
Horizontal time scale: 5 µs/Div.

Figure 7. Programmer X Has an Address Cycle Time of 47.40 µs



NOTE:
Horizontal time scale: 10 µs/Div.

Figure 8. Programmer Y Has an Address Cycle Time of 51.38 µs



NOTE:
Horizontal time scale: 25 µs/Div.

Figure 9. Programmer Z Has an Address Cycle Time of 190.70 µs

We can see from the waveform plot of Programmer X (Figure 7) that there's a great deal of room for improvement, 33–50% or more. If the first OE# low transition was delayed 3 µs, there would be no need to strobe OE# low a second time; this would shave 14 µs from the displayed address cycle time (refer to last sentence in Section 6.1). In addition, OE#'s pulsewidth should be shortened, cut in half, or more.

Programmer Y (Figure 8) has been positively affected by Intel's algorithm optimization work with the vendor. We see a long time from OE# high till the next address. Unfortunately, that's a function of system overhead that cannot be improved upon. The small savings to push OE# out a couple microseconds would be futile given the strobes are 5 µs in duration. Programmer Z (Figure 9) is currently undergoing Intel scrutiny, and should improve significantly.

6.5 Programming Word Wide–2× Faster Than Byte Mode

Several Intel Flash memory devices contain user-selectable bus width control. A dedicated input pin, BYTE#, driven low places device in x8 mode. BYTE# high puts the part in x16 operation.

This capability provides a means to program devices fast in x16 mode, regardless of the application's bus width.

7.0 WSM READY/BUSY INDICATION MODES

Intel FlashFile and Fast Flash memories indicate WSM status via a dedicated output pin, RY/BY#, and status register bit SR.7. In the next section, we'll discover why RY/BY# provides faster status indication than SR.7, and in Section 7.2 that the other status bits don't require checking each word/byte program or block erase operation.

7.1 Monitor RY/BY# Output vs. Status Bit SR.7 Polling

The RY/BY# pin was first introduced on Intel's 28F008SA FlashFile memory in 1992. It enhanced the performance capabilities of the first write-automated flash memory, the Intel 28F001BX introduced the year earlier, by providing fast hardware indication of internal WSM operation.

RY/BY# is constantly driven by the device and not tri-stated if CE# or OE# are brought to V_{IH}. RY/BY#'s default state after device power-up is V_{OH}. It transitions low to V_{OL} when a program or erase sequence is initiated, and RY/BY#'s rising edge (return to V_{OH}) alerts the system to operation completion.

RY/BY# is intended to interface the device to a system microprocessor rising-edge-triggered interrupt input. This type of indication, via hardware signaling of status, is faster than status bit polling. The reason, unlike SR.7, RY/BY# is not gated by the need to drive/toggle a control pin (OE#). To put this directly in the context of a programming system environment, RY/BY# signaling is not hidden by OE# inactive or the need/time involved to re-toggle OE# to get updated indication; as such, software overhead is reduced, which also improves performance.

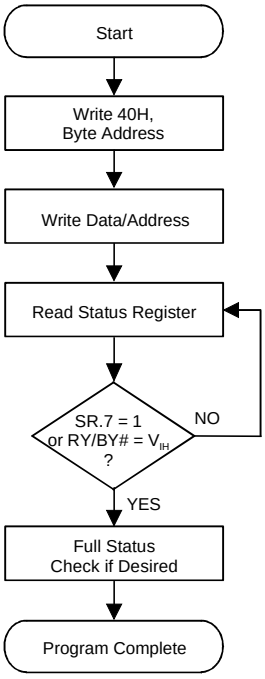
NOTE

Depending on the configuration of a device's RY/BY# pin, a pull-up resistor may be required. Check the datasheet for the particulars of the device you're working with.

7.2 Check Full Status after Completing All Program/Erase Operations

The programming system intelligence may program several bytes, or erase several blocks back-to-back, while monitoring RY/BY# or polling SR.7 to determine when the next Program or Block Erase command can be given. When all bytes are programmed, or all blocks erased, the system can then poll the other status flags to determine if all operations were successful. See Figure 10 for the 28F008SA automated byte programming flowchart.

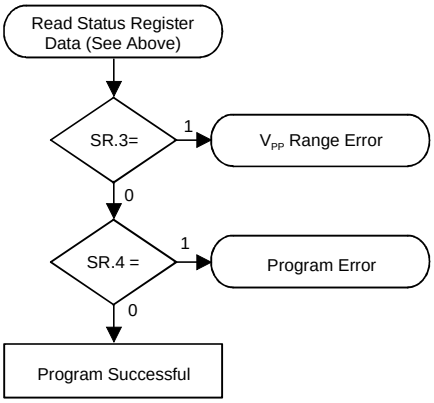
While other types of flash require the controlling microprocessor to watch for non-completion of program or erase within a specified time to indicate an error, Intel's implementation requires no external system timers or software timing loops. As such, the system can reduce its polling overhead while still identifying any potential error conditions.



Bus Operation	Command	Comments
Write	Setup Program	Data = 40H Addr = Byte to Program
Write	Program	Data = Data to Program Addr = Location to Program
Read or Monitor		Status Register Data Toggle CE# or OE# to Update SRD. RY/BY# Pin
Standby		Check SR.7 1 = WSM Ready 0 = WSM Busy

Repeat for subsequent program operations.
SR Full Status Check can be done after each program operation, or after a sequence of program operations.
Write FFH after the last write operation to reset device to read array mode.

FULL STATUS CHECK PROCEDURE



Bus Operation	Command	Comments
Standby		Check SR.3 1 = V_{PP} Low Detect
Standby		Check SR.4 1 = Byte Program Error

SR.3 MUST be cleared, if set during a program attempt, before further attempts are allowed by the Write State Machine.
SR.4 is only cleared by the Clear Status Register Command, in cases where multiple bytes are programmed before full status is checked.
If error is detected, clear the Status Register before attempting retry or other error recovery.

7769_09

Figure 10. A Full Status Check Procedure Does Not Have to Occur after Each Byte Programmed or Block Erased

8.0 IMPACT OF HARDWARE DESIGN ON PROGRAMMING SPEED

In the following sections, we'll touch on a few hardware design considerations that can provide a significant performance boost to a programming system. We'll cover impact relating to the proximity of the programming "intelligence," pin drivers and buffer memory to the DUT (Device Under Test) and review communication bandwidth issues.

8.1 "Intelligence" Proximity to Chip

Moving the programming intelligence closer to the DUT can improve programming performance of a system design. Taking an algorithm from the many stored on the PC, then dumping it into a reprogrammable logic device to create a device-specific state machine eliminates continual bus transfer of programming instructions, all but the initial download that generates the state machine.

8.2 Pin Driver Proximity to Chip

Having the pin driver electronics in close proximity to the DUT improves signal integrity. A better signal may negate the need to incorporate algorithm compensating techniques because problems are avoided.

Strong, fast pin driver circuits and a robust system power supply provide added benefit. It is important to locate the power supply close to the DUT. Doing so improves performance, especially when reading devices with fast output buffers.

8.3 Buffer Proximity to Chip

Making the buffer memory resident in the programming system increases the speed of writing to the DUT. Having the program data stored inside the programmer eliminates the need to transfer it from a host system.

Providing expansion capabilities for this buffer gives the unit a longer service lifetime, as well as offers lower price points for users that don't initially need a large capacity.

8.4 Communication Bandwidth

The bandwidth of the communication link and transfer rate of hard/floppy disk drives are important factors for the programming system designer to consider.

Movement of code/data traveling through slow ISA bus bottlenecks then down a parallel or serial cable are eliminated by stand-alone equipment that has its own built-in interface to system-level software and device support algorithms. Many gang programmers are designed this way.

For data transfer rates, memory cards (PCMCIA or custom) are far superior to hard or floppy disk drives. Another method which would be even faster is to store the code in a flash memory array directly on the processors data bus for fast execution. Of course, some interface would be required to get the code into that array from the vendor's distribution media.

9.0 USING SUPERSET DEVICE WRITE-PERFORMANCE FEATURES

New commands and hardware integration enhance write performance on Intel's 28F016SA/SV and 28F032SA. At the core of this third-generation flash design is a more sophisticated CUI with command queue, an enhanced WSM, queueable data registers and address latches, dual SRAM write buffers, global and block-specific status registers, additional status feedback and an erase tagging mechanism. See Figure 11 for the 28F016SA block diagram.

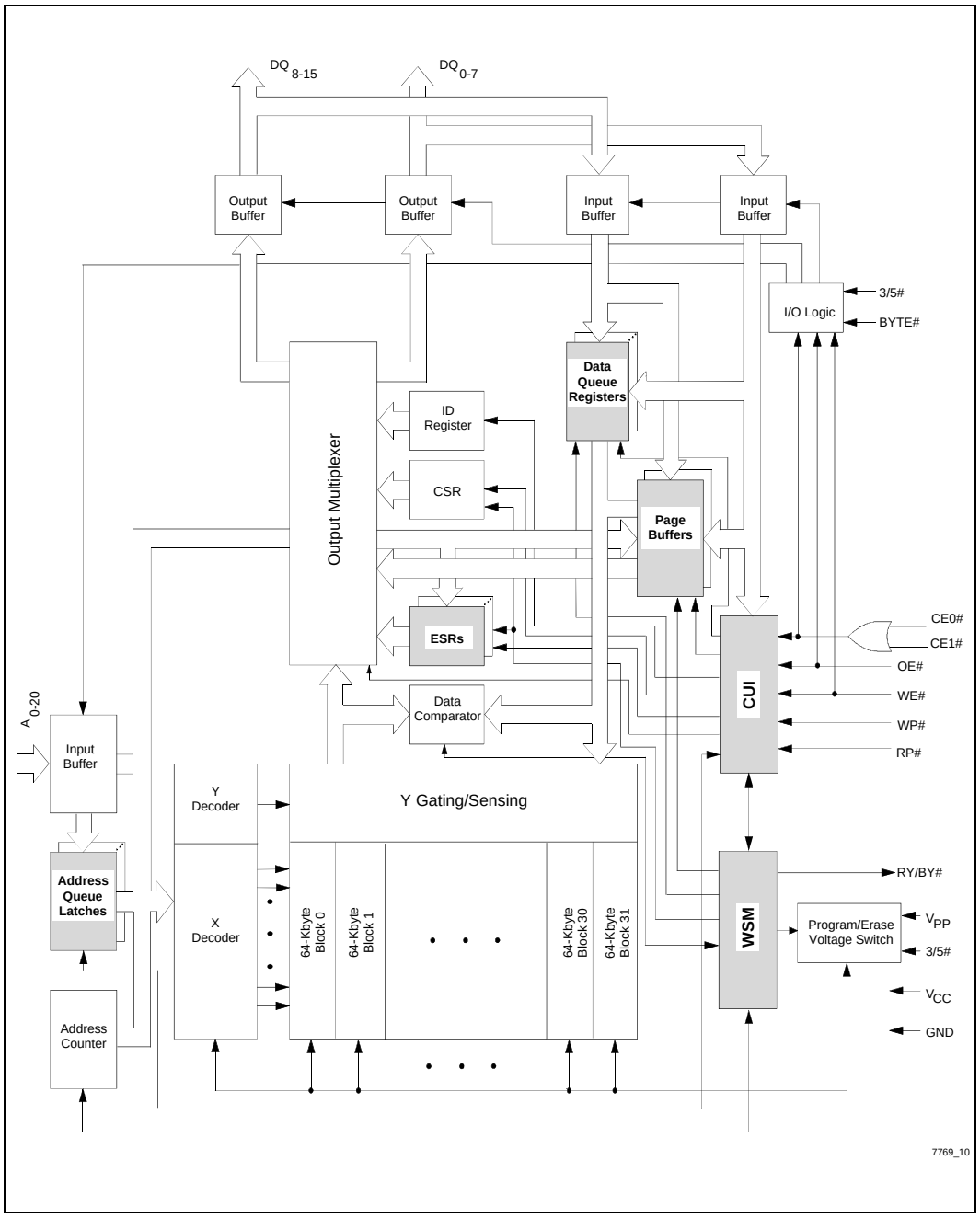


Figure 11. Architectural Evolution Includes Page Buffers, Queue Registers for Address, Data and Commands, and Extended Status Registers to Improve Write Performance

9.1 Command Queue

Advanced CUI sophistication involves the ability to queue multiple commands and enact new functions. For queueable commands, three CUI registers are available and the CUI control logic enhanced via a priority resolver to determine execution order.

The command queue serves as a “holding tank” for instructions requiring WSM activity. The 28F016SA queue is three deep, allowing two additional commands to be accepted while some other operation is current. Data and address latching mechanisms are also three deep to hold the information necessary for queued commands.

WSM feedback informs the CUI when new requests are allowed, and when previously latched ones can progress through the command queue.

Queueing hides system overhead when successively writing several words/bytes/pages to the array or erasing/locking many blocks in series.

9.2 Page Buffers

The 28F016SA contains two uniquely-selectable, 256-byte (or 128-word) SRAM page buffers to improve system program performance by as much as 4.8× over previous-generation flash devices. This gain is achieved via an optimized data-write caching scheme. Internal programming occurs x16, regardless of the data load width.

A sequential load command is used to fill the page buffer. The flash array is then written with the stored data. Page buffer write operations are queued in the same manner as other CUI commands. Therefore, the user can load a page, issue a page buffer write command, then load the second buffer, which automatically becomes available. The second page buffer write operation is then queued for execution. This pipelining of page buffer write operations results in very high write transfer rates to the flash array.

9.3 Extended Status Registers

To support this new architecture and the performance-enhancement commands, a Global Status Register and a set of 32 Block Status Registers are provided. These Extended Status Registers, which are addressable in each block memory space, convey status information at the chip and block level. Incremental indications such as queue status and page buffer availability are contained within the ESRs.

9.4 Erase Queue and Erase All Unlocked Blocks Command

The 28F016SA WSM contains its own queueing mechanism for block-erase operations, and its CUI, a new command to Erase All Unlocked Blocks. The erase queue is used when full-chip erase is not practical, i.e., slows performance of the programming system because it would be inefficient to reprogram data back into blocks not requiring content update or because there are locked blocks that do require a re-write. This queue is as deep as the number of erase blocks within the device.

If a block erase operation is currently running, and a second block erase is instructed, that new location is communicated to the WSM, which in turn marks that block in its tagging mechanism. This action is noted in conjunction with the initial erase command, and when that operation completes, the WSM looks at its tagged blocks to continue erase at the next logical location. In this manner, erase of many blocks does not tie up the command queue.

10.0 CONCLUSION

Time is money, and ideas that get applied which improve device throughput performance lower the costs associated with programming. This could result in significant reductions to off-line programming expenses, regardless of whether that function occurs in-house or at a distributor or independent programming center.

From the perspective of the in-line implementation, programming expeditiously facilitates integration into the manufacturing flow. This, in turn, accelerates acceptance of product in fine-pitch packaging and shipment in tape-and-reel media. Existing ATE can be used to program at board test, or an inexpensive JTAG/HSS solution can be deployed at the end of the line.

Bottom line, fast programming saves money.

APPENDIX A
ADDITIONAL INFORMATION(1,2)

Order Number	Document
290600	*Byte-Wide SmartVoltage FlashFile™ Memory Family 4, 8, 16 Mbit Datasheet*
290429	*28F008SA 8-Mbit FlashFile™ Memory Datasheet*
290539	*8-Mbit SmartVoltage Boot Block Flash Memory Family Datasheet*
297372	*16-Mbit Flash Product Family User's Manual*
292094	*AP-359 28F008SA Hardware Interfacing*
292099	*AP-364 28F008SA Automation and Algorithms*
292179	*AP-624 Introduction to On-Board Programming with Intel Flash Memory*
292185	*AP-629 Simplify Manufacturing by Using Automatic Test Equipment for On-Board Programming*
292186	*AP-630 Desigining for On-Board Progammming Using the IEEE 1149.1 (JTAG) Access Port*
294016	*ER-33 ETOX™ IV Flash Memory Technology: Insight to Intel's Fourth-Generation Process Innovation*

NOTES:

1. Please call the Intel Literature Center at (800) 548-4725 to request Intel documentation. International customers should contact their local Intel or distribution sales office.
2. Visit Intel's World Wide Web home page at http://www.intel.com for technical documentation and tools.