AP-624

APPLICATION NOTE

Introduction to On-Board Programming with Intel Flash Memory

November 1996

Order Number: 292179-002

CONTENTS

REVISION HISTORY

Number	Item
-001	Original Version
-002	Updated boot block JTAG programming calculation to reflect word programming time.

1.0 INTRODUCTION

A variety of options exist to program Intel Flash memory components. These alternatives include:

- Engineering programmer—An operator loads and programs one component at a time.
- Gang programmer—The operator loads and programs multiple components simultaneously.
- Automated handling system—The component loads automatically into a programmer using a mechanical handler.
- On-Board Programming—Off-board hardware and software programs the flash memory while it is installed on the printed circuit board (PCB).
- In-System-Write—The system CPU programs the flash memory in the user application.

This application note explores the strengths, limitations, programming methods and design considerations for one common option: on-board programming (OBP).

Each OBP environment is unique and requires examination by hardware and software engineers to insure appropriate functionality. If you have OBP questions not covered in this application note, please contact your local OBP solution provider (see the Appendix) or Intel technical support for more information.

2.0 PROGRAMMER TERMINOLOGY DEFINITIONS

The terms “on-board programming” and “in-system write” are often inappropriately interchanged. To eliminate confusion, we will first review definitions of the three main programming techniques.

2.1 In-System-Write (ISW)

The system CPU executes flash memory program and erase algorithms (see Figure 1). New data comes from one of several sources: serial or parallel port, floppy or hard disk drive, modem, etc. You create the algorithms for flash memory component erase and program operations.

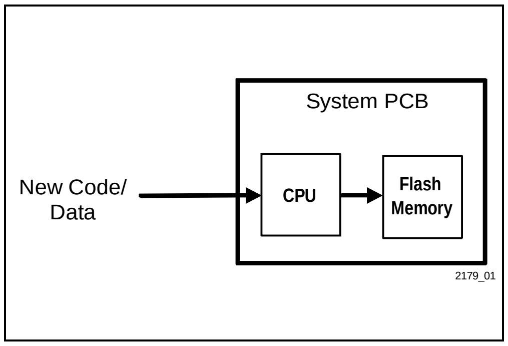

Figure 1. The System CPU Controls In-System Writes to the Flash Memory Component

2.2 On-Board Programmer (OBP)

In this approach, the OBP hardware controls update operations to the flash memory (see Figure 2), after first removing the PCB from the rest of the system. The OBP powers down the processor or holds it in a high-impedance state. The system PCB, with the flash memory installed, connects to an external “computer” such as a board tester or board level programmer. This off-board intelligence provides the necessary commands, voltages, control signals, addresses and data to erase and program the flash memory.

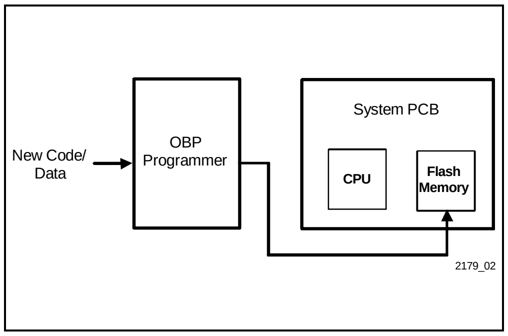

Figure 2. The OBP Programmer Controls Flash Memory Update Operations

2.3 Component Programmer

This approach was first made popular in the days of PROMs and EPROMs. In this approach, you remove the flash memory from the PCB and socket it in dedicated hardware called a component programmer (see Figure 3). Intel works closely with a wide range of component programmer vendors to ensure support for all of its flash memory products. The programmer manufacturer updates the flash memory algorithms for erase and program operations when necessary.

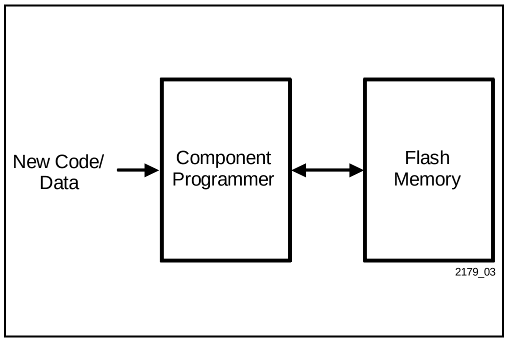

Figure 3. The Component Programmer Controls Update of the Flash Memory

3.0 ON-BOARD PROGRAMMING STRENGTHS AND LIMITATIONS

Each programming solution, including OBP, provides both strengths and limitations. You should analyze the production requirements of your project and consider all capabilities and shortcomings before determining if OBP is an appropriate programming solution for your project.

- **Strength: OBP Reduces Manual Handling of Flash Memory Components**

  As flash memory package dimensions decrease, pin-to-pin spacing narrows and susceptibility to mechanical problems increases, due to improper component handling.

  Manual programming is susceptible to mechanical problems when the operator handles each individual component. Misalignment of component pins occur when the component is inappropriately inserted or removed from programming sockets. You can repair misaligned pins, but the cost is high, the yield is not 100% and the rework creates delays in delivering the end product.

  OBP reduces susceptibility to misaligned component pins. With OBP, the flash memory is soldered onto the PCB using surface mount assembly equipment and operators do not handle the individual components. Without operator intervention, the risk of inadvertently damaging component pins is significantly reduced.

- **Strength: OBP Enables Just-In-Time Code Updates to Assembled Products**

  You can easily perform Just-In-Time (JIT) code updates to assembled products with OBP. For example, imagine the manufacturing process for cellular telephones, which are assembled, tested and stored until needed. When a customer orders the cellular telephone, you use OBP to update the flash memory with the most recent firmware, Just-In-Time. By performing JIT firmware updates, the customer receives the most recent cellular telephone features, including country- and region-specific language and other information, and the best value for the price.

  In contrast, let's investigate a case that does not use OBP. A preprogrammed flash memory component is soldered onto the cellular telephone PCB. After assembling the telephone, the software engineer discovers a firmware bug, fixes it and releases new firmware. A technician unsolders the flash memory component from the PCB, erases it and updates it in a component programmer with the latest code. The technician then resolders the flash memory to the PCB. The PCB rework required to update the telephone firmware significantly increases costs and creates delays in delivering the cellular telephone to the customer.

- **Limitation: OBP Can Increase System Cost and Development Time**

  Sometimes the design engineer must create additional hardware and/or software to perform OBP. For example, to link the OBP to the PCB the design engineer may need to add an interface connector. Additionally, to isolate certain PCB circuits from the programmer driver circuits, design engineers must incorporate jumpers or active devices, such as FET switches or tri-state buffers, into the design. The additional hardware costs increase the per-system bill of materials, while the additional software slows time-to-market.

- **Limitation: On-Board Programmers May Program Flash Memory Slower Than Other Programming Methods**

  The time needed to program flash memory with a board level programmer may be longer than the time required for component programmers. With board level programmers, programming time may increase due to operating system overhead. The board level programmer must contend with high capacitive loading on the PCB, with powering down peripheral circuits around the flash memory and with potential power and ground noise problems. Software precautions, such as increased settling time delays designed to avoid potential hardware problems, account for some increase in operating system overhead. This potentially results in a decreased manufacturing line beat-rate (the number of products that go through the manufacturing flow in a given amount of time).

4.0 OBP PLATFORMS

Be careful to select the appropriate programmer platform for OBP. This section identifies some technical items to consider when investigating board level programmers:

- The OBP power supply must be capable of supplying current to all components on the PCB
- Power supply tolerances must conform to Intel Flash memory requirements
- Address, data and control signal drive capability must meet Intel Flash memory specifications when applied to the flash memory component
- Minimize inductance and capacitance introduced to the circuit by the programmer and interface

Multiple board level programming platforms exist for flash memory programming. The following is a summary of each solution: Automatic-Test-Equipment, OBP Board Level Programmer, JTAG Test Access Port, and modified component programmers:

- **Automatic Test Equipment**

  Manufacturing engineers use Automatic Test Equipment (ATE) to perform functional testing on assembled PCBs. In addition, ATE can perform flash memory programming functions if desired.

  You can use a bed-of-nails interface to link your ATE with the PCB. Bed-of-nails refers to multiple spring loaded test points that connect to tester driver circuits. When you secure the PCB on top of these test points, some test points contact flash memory device pins and some contact PCB traces called test land pads which connect to flash memory device pins. The ATE software defines which test points are active. This software enables the ATE to drive signals on test points that contact appropriate flash memory device pins, while disabling test points that do not contact flash memory pins.

  If you intend to use ATE to program flash memory, you need both sufficient ATE hardware and software knowledge and sufficient flash memory product knowledge. You will create erase and program routines on your ATE. These routines are then integrated into the test flow and are used to program the flash memory. Your ATE vendor will most likely be able to assist you in developing necessary hardware and software (see the Appendix).

  ATE typically performs programming operations quicker than other OBP programming methods. To gain the best ATE programming performance, you must optimize the ATE code. Be sure to perform only necessary programming operations. A thorough understanding of flash memory programming requirements, as outlined in the device datasheet and other technical documentation, will help you reduce cycle times and minimize overall programming time. To check cycle time, connect an oscilloscope to the least significant address line. This will provide an accurate representation of programming cycle time on your ATE.

- **OBP Board Level Programmer**

  Component programmers designed specifically for board level programming are lower initial cost OBP solutions as compared to ATE solutions. Manufacturing engineers use board level programmers for either in-line programming (the programming process is part of the manufacturing assembly process) or off-line programming. With off-line programming, the OBP programs flash memory separate from the manufacturing assembly line.

  OBP programmer vendors provide erase and program algorithms with the purchase of a board level programmer. When semiconductor manu-facturers improve their product and/or launch new products, the programmer vendor updates the programming algorithm. New device support, or updated programming algorithms, are delivered to owners of board level programmers via mail or downloaded from manufacturers' bulletin board services.

The owner of a board level programmer needs to create a hardware interface linking the PCB application to the programmer. When the hardware interface is functional the user links commercially available erase and program algorithms to create a programming flow applicable to the project. The project design engineer will address issues such as device power up sequence, appropriate delay between different commands and power and GND transient voltage spikes. These issues contribute to increased programming times.

Device programming times with OBP programmers are typically longer than ATE programming times. Therefore, board programmers are commonly used for lower volume board programming.

- **Using the JTAG Test Access Port or Other Serial Channel to Program Flash Memory**

  The JTAG Test Access Port (TAP) is an emerging OBP method. JTAG communicates serially (one bit at a time) with the PCB application. JTAG is a viable programming alternative in a manufacturing environment if your PCB application contains a JTAG-compliant microprocessor and you can tolerate somewhat longer programming times than are capable with other OBP methods. Alternatively, JTAG can be used to program boot code into the flash memory, with the remainder of the device programmed via in-system write.

What Is JTAG?

The Joint Test Action Group (JTAG), formed in the 1980's by key electronic manufacturers, sought to create PCB and IC test standards. The JTAG team created the IEEE specification 1149.1-1990 *Standard Test Access Port and Boundary-Scan Architecture*. This specification defines precisely how to design JTAG logic into an IC to enable JTAG testing.

The JTAG specification stipulates that a single cell of a shift-register is designed into the IC logic and linked to every pin of the IC (see Figure 4). This single cell, known as the Boundary-Scan Cell (BSC), links the JTAG interface to the IC's internal core logic. All BSCs of a particular IC constitute the Boundary-Scan Register (BSR). BSR logic becomes active when performing JTAG testing, otherwise it remains passive under normal IC operation.

How Can I Use the JTAG Test Access Port to Program my Flash Memory?

By utilizing JTAG communication equipment that inserts into your PC add-in card slot and connects to your JTAG-compliant PCB application, you send commands and data through the JTAG Test Access Port to the system microprocessor. These commands and data instruct the microprocessor to program the flash memory (see Figure 5).

JTAG communication equipment (see Appendix) allows communication with any JTAG-compliant micro-processor. You create the software to program flash memory. Programming the flash memory at high speeds requires that you develop optimized code for the JTAG communication equipment, necessitating a clear understanding of the flash memory programming algorithm and JTAG communication equipment.

A JTAG compliant microprocessor has the following four pins included in its pin architecture:

TCK – Test Clock Input. A free running clock separate from the system clock.

TDI – Test Data In. Data is shifted into the JTAG compliant device via TDI.

TDO – Test Data Out. Data is shifted out of the JTAG compliant device via TDO.

TMS – Test Mode Select. TMS commands select test modes as defined in the JTAG specification.

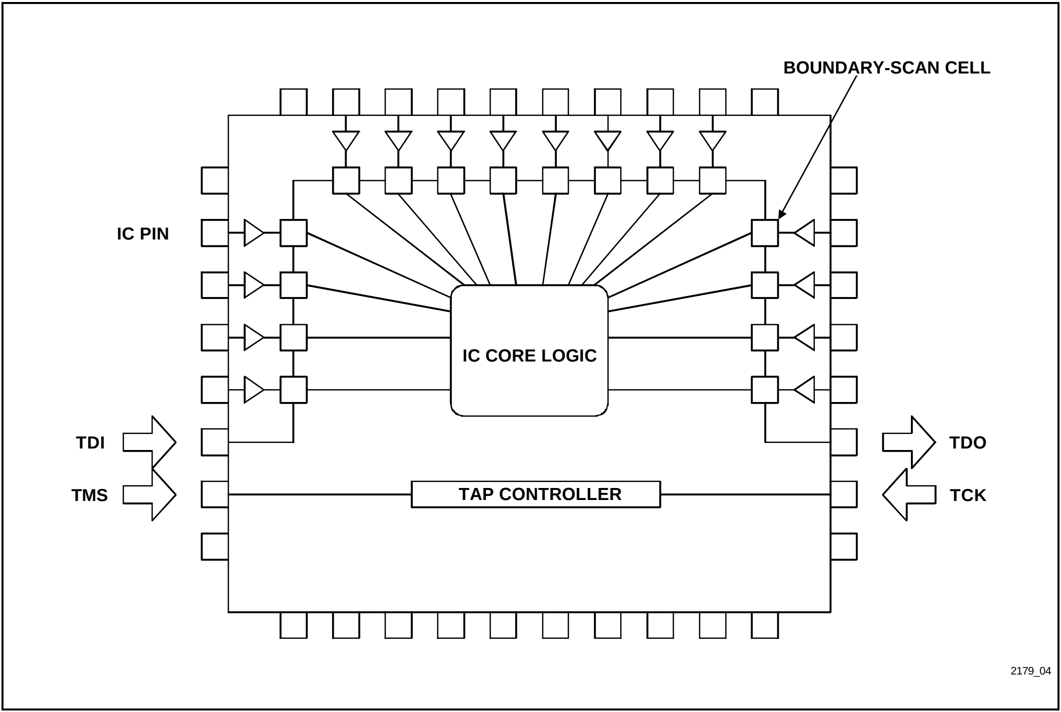

Figure 4. IC Containing JTAG Boundary-Scan Register

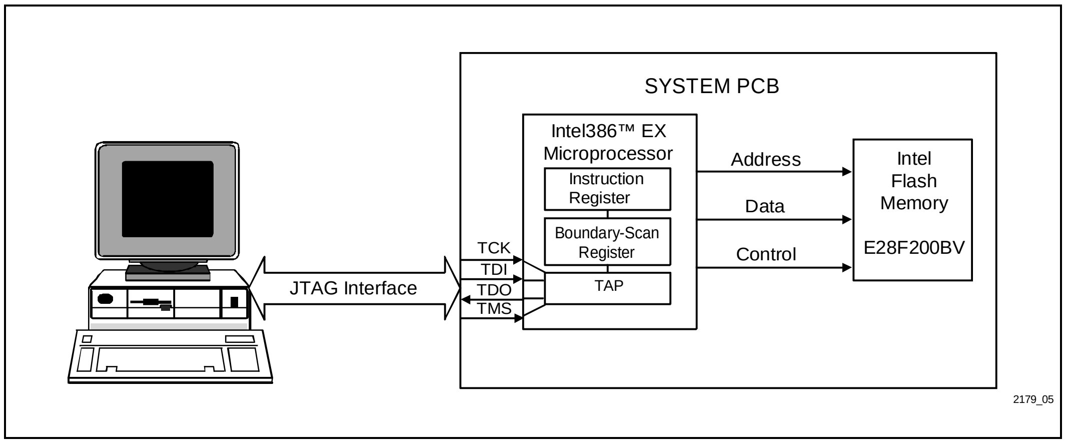

Figure 5. The System CPU's JTAG Test Access Port Provides an Interface to Program Flash Memory

How Fast Can I Program Flash Memory Using the JTAG Test Access Port?

Programming time depends on several variables: TCK frequency, JTAG-compliant microprocessor speed, number of BSC bits and flash memory density. This example will assume the following variables:

JTAG Programming Variables:

TCK = 40 ns Cycle Time (25 MHz TCK)

Number of BSC bits = 132
(Intel386™ EX Microprocessor in a 132-pin PQFP package)

Memory Density = 2 Mbit
(E28F200BV Intel Flash Memory)

Calculate JTAG Programming Time per 2-Mbit Data:

132 BSC bits × 40 ns (TCK) = 5.28 µs

*10 signal transitions × 5.28 µs = 52.8 µs / 1 word of data

128 Kwords (flash memory density) × 52.8 µs = 6.92 seconds to program 2 Mbit of data

* 10 signal transitions to drive "high" and "low" levels on address, control and data signals.

Calculate JTAG Programming Time for One 16-Kbyte Boot Block:

8 Kwords (Boot Block size) × 52.8 µs =
0.43 seconds to program 8 Kwords of data

Remember, each JTAG programming setup is different. Programming time calculations are dependent on variables associated with the individual setup and vary with different hardware configurations.

The techniques described in this section may also be applicable in some cases to proprietary serial communications methods offered with various CPU architectures. Examples include serial debug and high-speed input-output ports. Contact your CPU vendor for more information.

- **Use Your Component Programmer to Perform OBP**

In some cases, you may choose to use a component programmer to perform OBP. The main advantage of this approach is its low cost, especially if you already own a component programmer. However, this technique also has its limitations. Signal degradation from the programmer to the PCB can cause unreliable results. Also, the component programmer only supplies sufficient current to power the flash memory component, not the entire PCB.

To program flash memory using this method you need to build a cable to connect the component programmer to your OBP application. One end of the cable connects into the component programmer DIP socket, the other end connects to your OBP application.

If you select this method to perform OBP you should also adhere to the following guidelines:

— Use shielded heavy gauge wire for power, GND, address, control and data signals. The GND shield provides a solid connection from the component programmer to the PCB application and reduces susceptibility to signal crosstalk.

— Keep the cable length as short as possible.

— Place two V_{CC} bypass capacitors, 0.1 µF and 4.7 µF, at both ends of the cable.

To validate signal integrity at the PCB end of the cable use an oscilloscope to check address, control and data signals. Look for rise and fall signal transitions from 10% to 90%. These signal transitions must be free of oscillations. You only need to perform this check upon initial implementation of this programming method.

Since component programmers typically provide sufficient power for single components only, you should isolate the flash memory power, GND, addresses, data and control signals from the rest of the PCB during programming and erase operations. After updating the flash memory, reconnect these signals to the remainder of the system circuitry.

5.0 HARDWARE DESIGN CONSIDERATIONS FOR OBP INTERFACES

This section provides an overview of common design considerations to help reduce susceptibility to programming problems in the board level environment. The suggestions in this section apply to both interfaces between the OBP and PCB applications and to the PCB application.

The following high-speed design rules help reduce susceptibility to transient voltage spikes when incorporated in an OBP interface design:

Design Consideration 1: Minimize Power Supply Interconnections

Keep power supply interconnections to a minimum. Interconnections add inductance and capacitance to a circuit and decrease and distort the circuit's ability to operate at high frequencies. When you have fewer power supply interconnections to the interface board or PCB application you will experience less transient voltage spikes.

Keep V_{CC} and GND inductance minimal. Direct connections from the component socket to V_{CC} and GND planes on the system board is optimal to reduce inductance.

Minimize susceptibility to signal crosstalk. Crosstalk, the electrical influence of one signal on another, alters the high frequency operating characteristics in a circuit. To reduce signal crosstalk, avoid locating flash memory control signal PCB traces in close proximity to power supply traces or input/output traces.

Design Consideration 2: V_{CC} PCB Layout

To ensure proper flash memory operation, the V_{CC} voltage tolerance cannot exceed recommended device operating specifications. This design consideration helps to maintain a stabilized V_{CC}.

Transient voltage spikes on V_{CC} or GND are a common problem associated with OBP. Transient voltage spikes can permanently damage the flash memory. You should incorporate the following design precautions into the PCB design to minimize susceptibility to voltage spikes.

Your hardware design should incorporate a multiple layer PCB with dedicated V_{CC} and GND planes. The capacitive effect gained by using planes for power and GND help suppress high frequency transient voltage spikes. In addition, use bypass capacitors to reduce the amplitude of transient voltage spikes. One 0.1 µF V_{CC} bypass capacitor placed as close to the flash memory V_{CC} pin(s) as possible will reduce high frequency voltage spikes. For every eight flash memory components, a 4.7 µF bypass capacitor placed as close to the V_{CC} pin as possible will reduce low frequency voltage spikes. In the event that less than eight flash memory components are used a single 4.7 µF bypass capacitor is still necessary.

Design Consideration 3: V_{PP} PCB Layout

The V_{PP} power supply provides the high voltage necessary to perform flash memory erase and program operations. During these times, the component uses maximum V_{PP} current, I_{PP}. If you are simultaneously programming or erasing multiple flash memory components, the current requirement, I_{PP}, increases proportionately with the number of flash memory components used. Be sure that the board level programmer can supply sufficient I_{PP} current to program and erase all flash memory components. Maintain V_{PP} within specified tolerances at all times; over-voltage can damage the device and under-voltage can decrease device reliability. Check the board level programmer specifications to compare the maximum power supply current rating to the calculated current demand for your application. You must base the calculated current demand on the number of flash memory components in your application. Information gathered from this check will help determine if the board level programmer can perform erase and program operations on your application.

To avoid damage to other PCB components during programming and erase operations you must isolate V_{PP} to the flash memory component(s). Make V_{PP} PCB traces as wide as possible, .050 mils or greater in width, and as short as possible. Additionally, a 0.1µF bypass capacitor placed close to the flash memory V_{PP} pin will help reduce the amplitude of transient voltage spikes.

Design Consideration 4: Disable the Microprocessor before Programming

If your flash memory application contains a microprocessor and you are not performing JTAG programming, you must disable the microprocessor outputs before attempting to program the flash memory. You can accomplish this by using the microprocessor's RESET, HOLD or ONCE modes. Any of these modes place the microprocessor control signals and local bus in a high-impedance state. In this state you avoid bus contention between the microprocessor and flash memory signals.

In addition, be careful not to forward bias pins on peripheral components around the flash memory. You can avoid forward biasing by maintaining stringent voltage tolerances supplied by the OBP.

Design Consideration 5: Beware of V_{CC} and GND Transient Voltage Spikes

V_{CC} and GND transient voltage spikes are a common source of problems in embedded system applications. These transient spikes cause various programming and erase problems. False verify errors, a condition that occurs when the flash memory programs or erases correctly but fails to verify data correctly, is prevalent in some marginally designed system environments. Look for documentation of this problem, and possible solutions, in the Intel technical brief: *Designing for Successful Flash Memory Read and Verify Operations*. You can obtain this technical brief from the Intel literature center by requesting document number 297691.

Design Consideration 6: Do Not Combine Flash Memory Components from Different Manufacturers Unless Absolutely Necessary

Flash memory components from different manufacturers may share similar part numbers and device pin-out architectures but require vastly different programming algorithms. OBP problems occur if you mix flash memory components from different manufacturers into the same application. Avoid compatibility problems by using flash memory from a single manufacturer whenever possible.

Design Consideration 7: Isolate High-Voltage Flash Memory Signals from Other PCB Circuits

To avoid potential damage to circuits on the PCB application, you must isolate high-voltage flash memory signals from other PCB circuits. Vpp and RP# are set to 12V when programming or erasing some flash memory components. If these signals connect to other non-flash memory PCB circuits and are not isolated during programming or erase operations, the 12V will damage them. You must incorporate a method to disconnect high-voltage signals from other PCB circuits prior to programming or erasing the flash memory components.

6.0 SUMMARY

OBP is one of several programming alternatives available to flash memory users. OBP users can select programming vehicles that range from Automatic-Test-Equipment to dedicated board level programmers and JTAG Test Access Port programming schemes.

To insure that the OBP programming vehicle you select reliably programs flash memory, you should understand and implement the high-speed design considerations outlined in this document. Each design consideration applies to both the interface between the OBP programming vehicle and to the PCB application on which you perform OBP.

Each OBP programming technique is unique. An OBP method that applies to one manufacturing requirement may not apply to another. Software and hardware engineers should carefully analyze individual project requirements and implement suggestions from this document as required.

APPENDIX A
PROGRAMMER VENDORS

NOTE

OBP options were selected from products offered by a variety of vendors. Since this industry develops many new solutions each year, Intel recommends that designers contact vendors for their latest products. Intel will continue to work with the industry to develop optimum solutions for programming flash memories. The hardware vendor remains solely responsible for the design, sale, and functionality of its product, including liability arising from product infringement or product warranty.

Programmer Manufacturers:

Data I/O Corp.
10525 Willows Rd. NE
Redmond, WA 98073
(800) 247-5700 Customer Resource Center
http://www.data-io.com/

SMS Mikrocomputer Systeme GmbH
Im Grund 15
D-88239 Wangen
Germany
49-7522-97280
49-7522-972850 FAX

System General Corp.
1603-A S. Main St.
Milpitas, CA 95035
(800) 967-4776/408-263-6667
(408) 262-9220 FAX

BP Microsystems, Inc.
1000 N. Post Oak Rd. #225
Houston, TX 77055-7237
(800) 225-2102/713-688-4600
(713) 688-0920 FAX

Needham's Electronics
4630 Beloit Drive #20
Sacramento, CA 95838
(916) 924-8037
(916) 924-8065 FAX
http://www.quiknet.com/~needhams/

Elan Digital Systems Ltd.
Elan House, Little Park Farm Rd.
Segensworth West, Fareham
Hants PO15 5SJ
United Kingdom
44-1489-579799
44-1489-577516 FAX

Automatic-Test-Equipment Manufacturers:

Hewlett Packard
Contact your local HP Board
Test Representative

GenRad
2480 N. First Street, Suite 100
San Jose, CA 95131-1028
(408) 321-3512
(408) 432-0267 FAX

Teradyne
2625 Shadelands Drive
Walnut Creek, CA 94598
(510) 932-6900
(510) 932-7965 FAX

JTAG Equipment Manufacturer:

Corelis, Inc.
12607 Hidden Creek Way
Cerritos, CA 90703
(310) 926-6727
(310) 404-6196 FAX

APPENDIX B ADDITIONAL INFORMATION

Device datasheets provide in-depth information on device operating modes and specifications.

The *16-Mbit Flash Memory Product Family User's Manual* (order number 297372) gives detailed information on the enhanced automation of Intel's 16-/32-Mbit FlashFile™ and Fast Flash memories. Included flowcharts assist you in developing system software.

The following Intel documents deal specifically with software and hardware interfaces to Intel Flash memories:

RELATED INTEL INFORMATION(1, 2, 3)

Order Number	Document
292046	*AP-316 Using Flash Memory for In-System Reprogrammable Nonvolatile Storage*
292059	*AP-325 Guide to First Generation Flash Memory Reprogramming*
292077	*AP-341 Designing an Updateable BIOS Using Flash Memory*
292095	*AP-360 28F008SA Software Drivers*
292099	*AP-364 28F008SA Automation and Algorithms*
292126	*AP-377 16-Mbit Flash Product Family Software Drivers*
292148	*AP-604 Using Intel's Boot Block Flash Memory Parameter Blocks to Replace EEPROM*
292172	*AP-617 Additional Flash Data Protection Using* V_{PP}*, RP#, and WP#*
297691	Technical Paper: *Designing for Successful Flash Memory Read and Verify Operations*

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. Additional information can be requested from Intel's automated FaxBack* system at (800) 628-2283 or (916) 356-3105 (+44(0)793-496646 in Europe).
3. Visit Intel's World Wide Web home page at http://www.Intel.com for technical documentation and tools.