intel®

AP-630

APPLICATION NOTE

# Designing for On-Board Programming Using the IEEE 1149.1 (JTAG) Access Port

November 1996

Order Number: 292186-002

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 7641
Mt. Prospect, IL 60056-7641

or call 1-800-879-4683



CG-041493

# CONTENTS

## REVISION HISTORY

| Number | Description |
|---|---|
| -001 | Original version |
| -002 | Minor changes throughout document. |

## 1.0 INTRODUCTION

Today's small hand-held electronic equipment is driving the miniaturization of printed-circuit boards (PCBs). The increased use of small form factor PCBs, with components installed on both sides, presents a problem for engineers who choose to perform On-Board Programming (OBP). Engineers typically perform OBP using Automatic-Test-Equipment (ATE) on larger PCBs. In space-constrained environments, or in high-speed designs, it is not always viable to incorporate the test land pads that link the PCB to an ATE bed-of-nails interface.

This document provides information on an alternative OBP method for small form factor PCB applications. You will learn about using the Joint Test Action Group Test Access Port (JTAG TAP) to perform OBP.

This application note provides details about the following topics:

- What is JTAG?
- Why use the JTAG TAP to program flash memory?
- JTAG programming performance
- Design considerations
- Programming strategies
- Vendor references

If you seek further technical information about programming Intel Flash memory, please contact Intel technical support. If you seek further information about JTAG programming solutions, refer to the vendor references in Appendix A.

## 2.0 WHAT IS JTAG?

The Joint Test Action Group (JTAG), formed in 1985 by key electronic manufacturers to create PCB and IC test standards. The JTAG proposal was approved in 1990 by IEEE as IEEE Standard 1149.1-1990 *Test Access Port and Boundary-Scan Architecture*. This standard specifies the hardware and software needed to enable boundary-scan testing. Since the 1990 approval, updates were published in 1993 as supplemental IEEE Std 1149.1a-1993 and in 1995 as supplement IEEE Std 1149.1b-1994.

The following topics briefly describe various components of a JTAG-compliant device:

**JTAG Pin Architecture**

A JTAG-compliant device includes the following pins in its architecture:

TCK – Test Clock Input. A 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.

Optional Pin:

TRST# – Test Reset Input (active low). Provides asynchronous initialization of the TAP controller.

A JTAG-compliant device can be a microprocessor, microcontroller, PLD, CPLD, FPGA, ASIC or any other discrete device that conforms to the 1149.1 specification.

**TAP Controller**

The TAP controller provides access to many of the test support functions built into the JTAG-compliant device. The TAP is a state machine. The state machine controls all operations for one JTAG-compliant device. Each JTAG-compliant device has its own TAP controller. You can sequence through the state machine functions via the TCK and TMS inputs.

**Boundary-Scan Register**

The 1149.1 specification stipulates that a single cell of a shift-register is designed into the IC logic and linked to every digital pin of the IC (see Figure 1). This single cell, known as the Boundary-Scan Cell (BSC), links the JTAG circuitry 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.

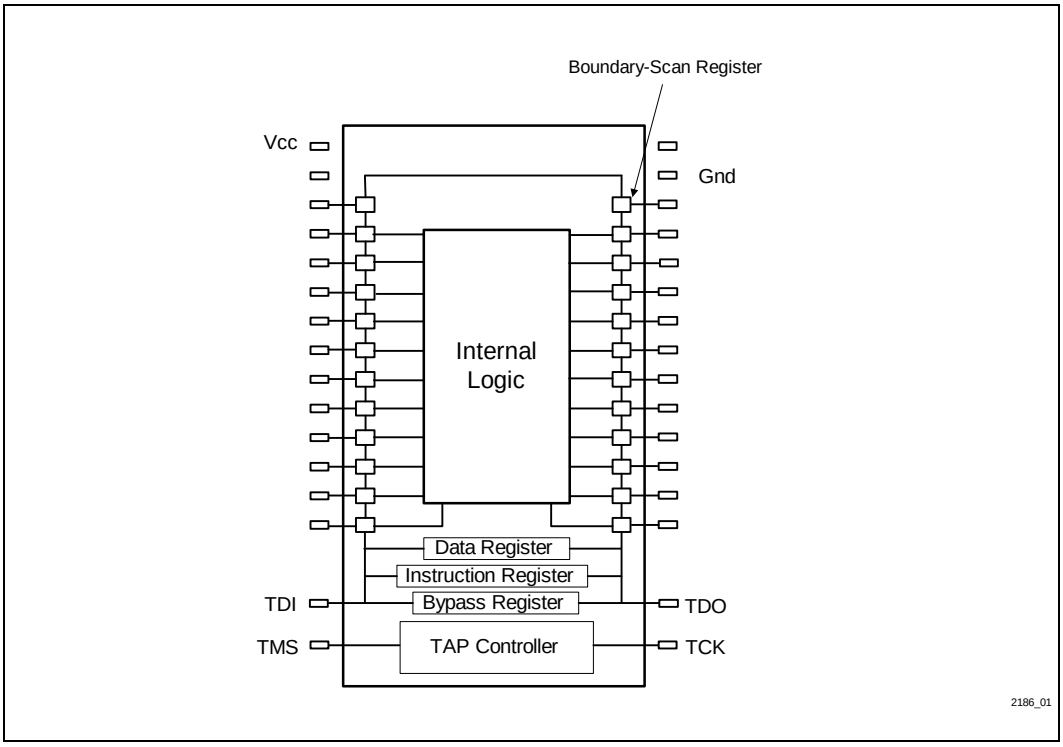


**Figure 1. The JTAG-Compliant IC Contains A Boundary-Scan Register Linked to Every Digital Input/Output Pin**

**JTAG Hardware Controller**

You communicate with the JTAG-compliant device by utilizing a hardware controller that either inserts into a PC add-in card slot or by using a stand-alone programmer. The controller connects to a JTAG-compliant PCB (see Figure 2). The JTAG-compliant device(s) must connect to all flash memory address, data and control signals. The flash memory does not need to be JTAG-compliant for this programming method to function. The JTAG hardware controller sends commands and data to the JTAG-compliant device, then propagates the data to the flash memory for programming.

JTAG hardware controllers provide a communication link with any JTAG-compliant device. You create the software to perform OBP programming functions. To perform JTAG programming operations at high speeds, you must optimize the code. In order to write optimized code, you must clearly understand flash memory programming algorithm operations and JTAG hardware controller requirements. Please review manufacturers' product specifications for this information.

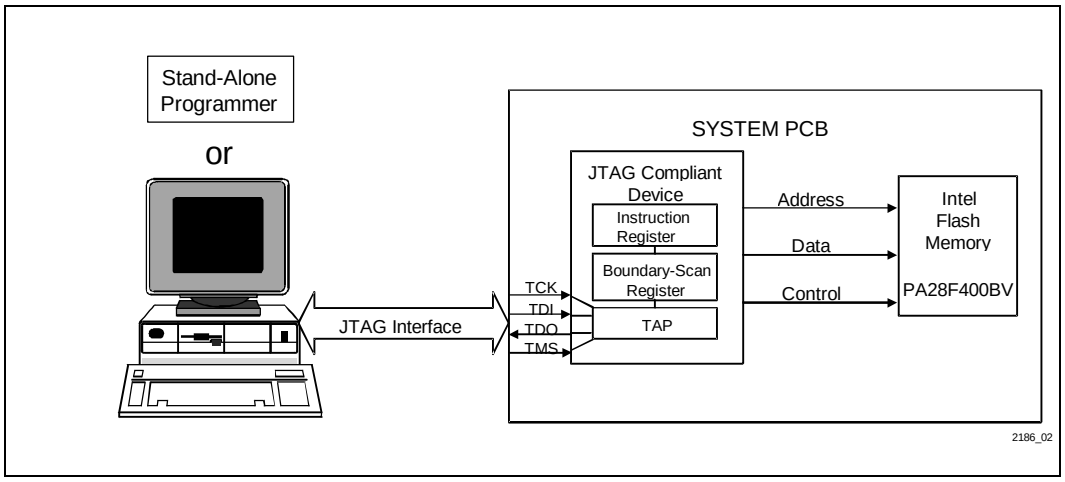


**Figure 2. You Can Control the Operation of JTAG-Compliant Devices via a PC That Contains a JTAG Controller or via a Stand-Alone Programmer**

**TAP Instructions**

The hardware controller communicates serially with the JTAG-compliant device through the TAP controller. You use the TCK and TMS inputs to clock in state-machine commands. Three TAP instructions manipulate the data:

**SAMPLE/PRELOAD**: Use this instruction to either SAMPLE the data currently contained in the BSCs, or to PRELOAD data into the BSCs.

**EXTEST**: When EXTEST is performed the BSCs attached to the JTAG-compliant device input pins act as sensors while the BSCs attached to output pins propagates data to interconnecting devices. The interconnecting devices may or may not be JTAG-compliant.

**BYPASS**: Reduces the BSC shift path through the device to a single bit register. For instance, if a device contains 401 BSCs and you execute the BYPASS instruction, the BSCs reduce to one for that device.

## 3.0 WHY USE THE JTAG TAP TO PROGRAM FLASH MEMORY?

As semiconductor technology advances, flash memory package dimensions continue to decrease. The latest flash memory package, Chip-Sized Package (CSP) is slightly larger than the actual DIE size, see Figure 3. In comparison, CSPs are approximately 50% smaller than TSOP devices. Designers often use CSPs in already space-constrained PCB applications. Increased use of advanced packaging heightens the need for new OBP capabilities. The JTAG TAP fits the OBP requirements of space-constrained PCB applications. Since the JTAG TAP requires only four test access points, manufacturers can use ATE or JTAG hardware controllers to serially program flash memory without utilizing a significant amount of PCB space.

Intel ships the majority of Small Outline Packages (SOP) in tape and reel shipping media. Tape and reel is economical and easier to use with surface-mount assembly equipment. Manufacturers save money by using this media. The increased use of SOPs and CSPs creates the need to reduce manual device handling. SOP and CSPs are ideal candidates for JTAG OBP alternatives.

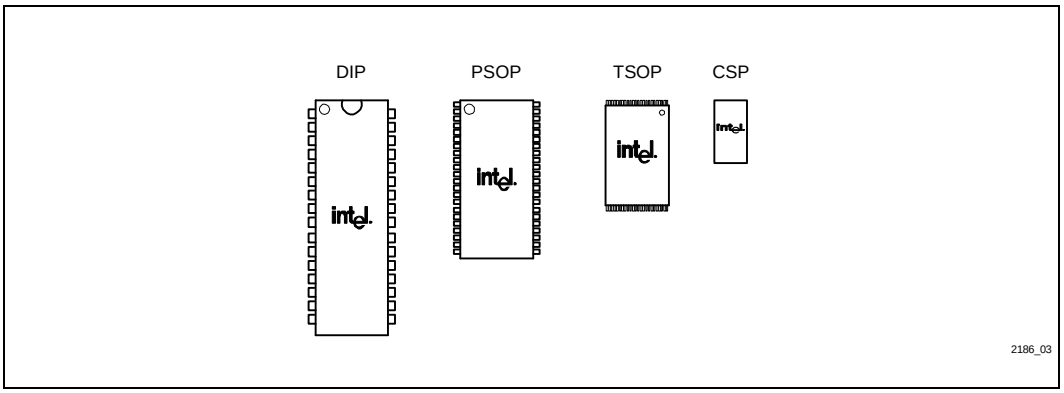


**Figure 3. Intel Flash Memory Package Dimensions Continually Decrease to Satisfy Requirements of Small Hand-Held Electronic Equipment**

## 4.0 JTAG PROGRAMMING PERFORMANCE

Engineers perform OBP by serially shifting data through the BSR and latching the data into the BSCs, see Figure 4. After loading the appropriate data in the BSCs you use the EXTEST instruction to propagate the BSC contents to the flash memory. For instance, if the JTAG-compliant device contains 224 BSCs then you use 224 TCKs to clock in each bit of data. This represents one BSR shift.

Each series of BSR shifts outputs one logical state, either high or low, to the flash memory. You clock each data bit in on the rising edge of TCK then latch the data bit into the BSCs. When you have completely loaded the BSC with all data bits, use the EXTEST instruction to perform a BSR shift. This outputs data to the flash memory. Each BSR shift increases programming time. Minimize the number of BSR shifts to obtain faster programming times.

Several factors can cause slower JTAG programming performance. When designing your JTAG OBP application consider the effect of the following variables:

- BSR length. A longer BSR means slower programming times. Use the BYPASS instruction on devices not directly connected to the flash memory or its controlling circuitry to reduce the BSR length when possible. (Make sure that the devices in BYPASS do have safe values on their outputs. An option here is to create a unique BSR chain, separate from the PCB test BSR chain, specifically for flash memory programming).
- Number of BSR shifts to perform flash memory writes. More BSR shifts takes more time.
- Flash memory density. Writing to more memory locations means longer programming time.
- Data bus width. Program flash memory 16 bits at a time when possible. (Some flash memories are configurable between 8 bit and 16 bit modes).
- TCK frequency. Slower TCK frequency means slower programming performance.

Intel Flash memories require four BSR shifts to perform byte/word programming for one address location, see Figure 5.

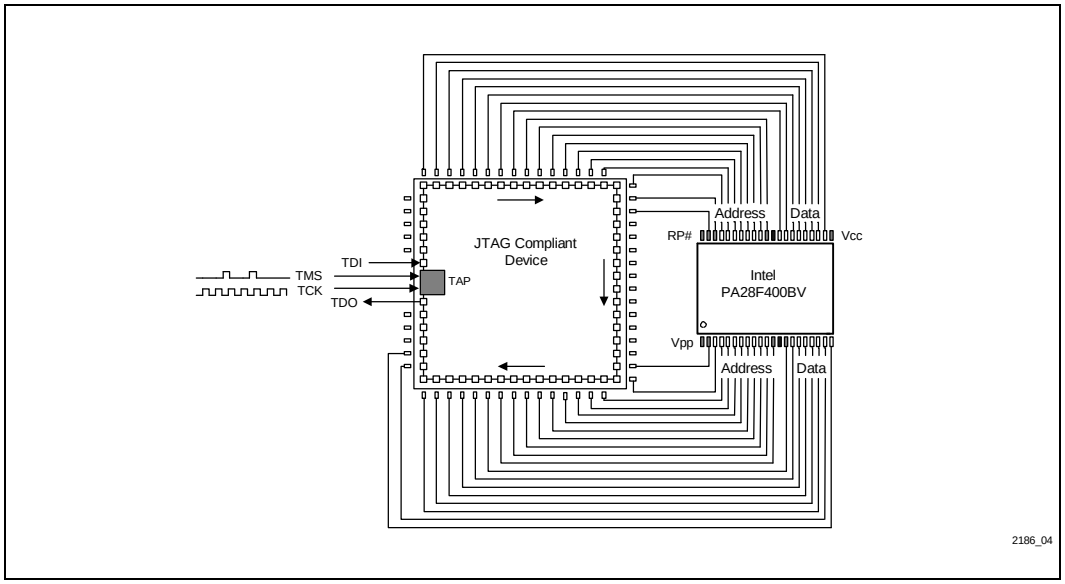


**Figure 4. Only Flash Memory Address, Data and Control Pins Connect to the BSR. $V_{CC}$, $V_{PP}$, RP#, and GND Do Not.**

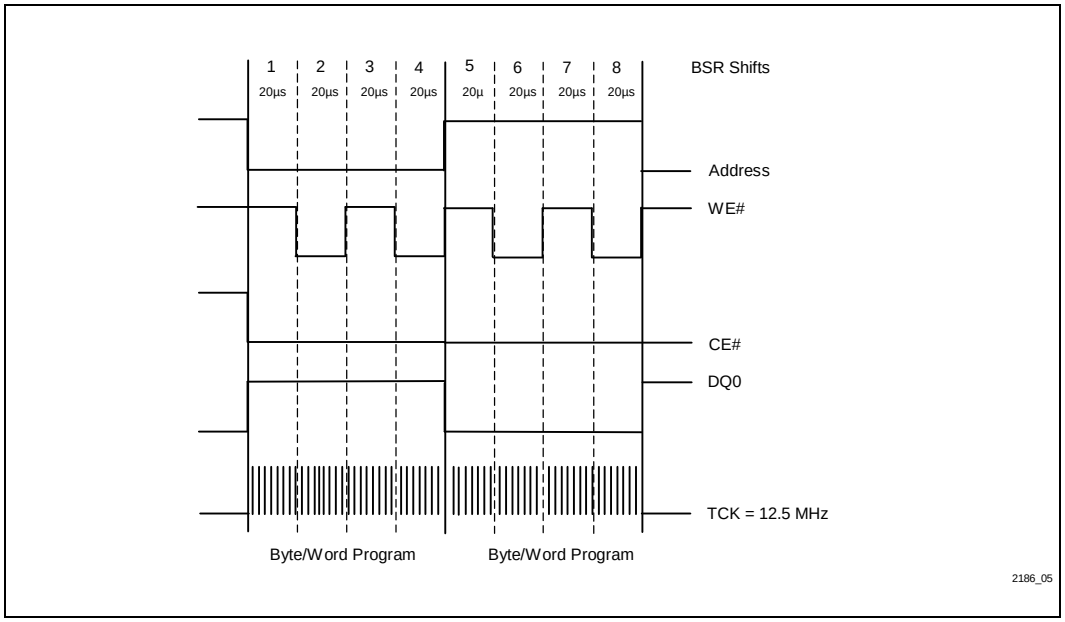


**Figure 5. Intel Flash Memories Require Four BSR Shifts per Address Location to Program One Byte/Word**

**Table 1. Emperical Lab Data for an Intel PA28F400BV Flash Memory**

| TCK frequency | 6 MHz | 12.5 MHz | 16 MHz |
|---|---|---|---|
| Boundary Scan Register length | 224 | 224 | 224 |
| BSR shifts per write cycle | 4 | 4 | 4 |
| Time to write one word | 168 µs | 80 µs | 63 µs |
| Time to program 4-Mbit flash device (x16) | 48.93 seconds | 23.49 seconds | 18.35 seconds |

Let's calculate the theoretical time required to program a 4-Mbit Intel Flash memory:

**A. Calculate time for one BSR shift:**

TCK = 80 ns Cycle Time (12.5 MHz TCK)

224 BSC bits × 80 ns (TCK) = 17.92 µs

**NOTE:**

Figure 5 indicates one BSR shift is equal to 20 µs. In example "A" we calculated one BSR shift as 17.92 µs. The additional 2.08 µs per BSR shift is a function of operating system overhead. This is the time it takes the data to transfer through the PC ISA bus to the JTAG-compliant device. For improved programming speeds you should seek out methods to reduce slow data transfers through the PC ISA bus.

**B. Calculate time to program one byte/word:**

4 signal transitions(1) × 17.92 µs = 71.68 µs

(1)4 signal transitions (BSR shifts) to drive "high" and "low" levels on address, control and data signals

**C. Calculate programming time for a 4-Mbit device:**

262,144 words (flash memory density) × 71.68 µs = 18.79 seconds

(Intel PA28F400BV Flash Memory)

**D. Calculate programming time for one 8-Kword boot block:**

8,192 words (boot block size) × 71.68 µs = 0.59 seconds

**E. The formula used to calculate JTAG programming performance:**

(TCK cycle time) × (BSC bits) × (BSR shifts) × (flash memory density) = programming time

See Table 1 for empirical data using different TCK frequencies. We collected this data using a stand-alone JTAG hardware controller connected to an Intel486™ DX2 computer. Please be aware that the theoretical calculations derived above do not reflect operating system overhead. Data for 12.5 MHz TCK indicates that 4-Mbit programming time is 23.49 seconds. Operating system overhead consumes 4.7 seconds of that time.

## 5.0 DESIGN CONSIDERATIONS

The following design considerations will help you successfully implement JTAG OBP:

- Each flash memory control pin, WE#, CE#, OE#, etc., must directly or indirectly connect to the BSR. For an example of an indirect connection see Figure 6. In the non-compliant design, the CE# pin does not connect to the BSR. You cannot use JTAG OBP to program flash memory with this design. To make this design function, connect the PLD "input from non-JTAG-compliant device" to a JTAG-compliant device. By doing this you gain control of the CE# pin via the BSR.

- Provide sufficient power to your PCB application. When calculating power requirements consider the need to supply power to the entire PCB application, not just the flash memories. (If you experience problems with transient voltage spikes in your application, please refer Appendix A for ordering information on the Intel technical paper *Designing for Successful Flash Memory Read and Verify Operations*. This technical paper explains how to reduce transient power spikes.)

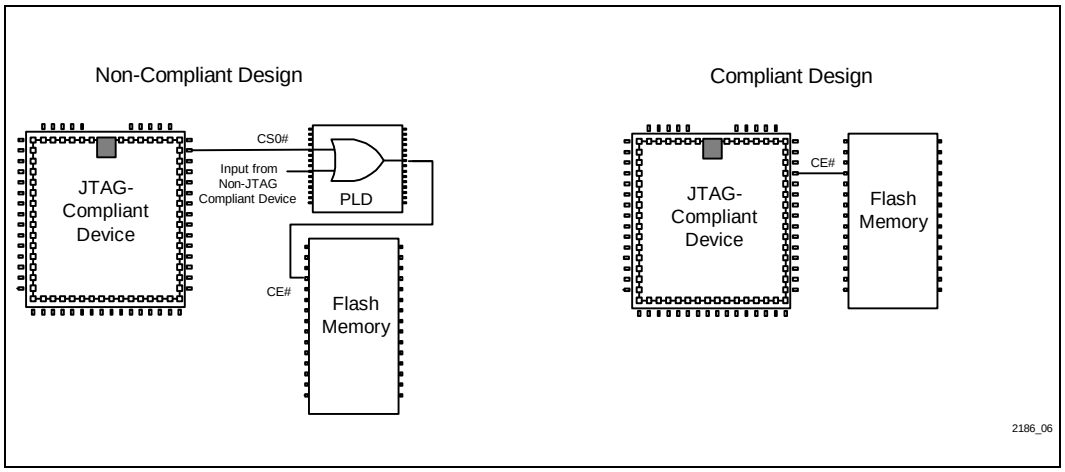


**Figure 6. To Perform JTAG OBP You Must Have Access to Every Flash Memory Pin from the BSR. In the Case of the Non-Compliant Design, You Cannot Control CE# because Its Input comes from a Non-JTAG-Compliant Source.**

- Keep the interface between the JTAG hardware controller and JTAG-compliant device as short as possible. This insures good signal integrity at high-frequencies.

- Provide a method to control the flash memory $V_{PP}$ and RP# inputs. These signals often connect to voltage pumps on the PCB. To enable programming you need control of these voltage pumps from the JTAG BSR.

- Be sure to account for all BSC bits that you clock into the BSR. One bit too many or one bit too few can have serious consequences on system hardware.

- Use ATE bed-of-nails test land pads, PCB trace vias, multiple pin header connector, an edge connector or some other method to access the JTAG TAP controller pins.

- For best programming results select TCK speeds of 6 MHz or greater. Slower clock speeds will function properly but programming time increases. Typically, TCK operates at one-half of the device clock speed. For example, a 25 MHz JTAG-compliant device has a TCK speed of 12.5 MHz. Be aware when selecting JTAG-compliant devices that undocumented limitations can cause programming problems. Ask device manufacturers specifically about errata or known limitations.

- Improve programming time by minimizing the number of BSR shifts necessary to execute byte/word programming. Select flash memory components that require the fewest programming bus cycles. Some flash memory components require four or more bus cycles to program one byte/word. Intel Flash memory components require only two bus cycles to program one byte/word.

## 6.0 PROGRAMMING STRATEGIES

You can choose from alternate JTAG OBP strategies to minimize programming times on manufacturing beat-rates. Some programming methods to consider:

- Use your JTAG compatible stand-alone programmer or JTAG hardware controllers to perform JTAG programming operations. Multiple programmers in parallel will reduce programming times.

- Program just the boot code using JTAG OBP. After programming the boot code use alternate off-line programming methods to program the remainder of the code in the flash memory without impacting the manufacturing line beat-rate.

- Use your ATE system, instead of the JTAG hardware controller, to perform JTAG programming operations. This method only works if you have space available on your PCB to place the four test land pads that interface with the ATE bed-of-nails test fixture.

- Field application engineers easily perform flash memory code updates in the field via JTAG-compliant devices. Some JTAG equipment vendors provide portable JTAG hardware controllers that plug into laptop computers. By connecting the JTAG hardware controller to the PCB application the engineer can update code from a laptop computer.

## 7.0 CONCLUSION

As demand for flash memory small outline packages increase, manufacturers seek alternative programming methods to eliminate device handling. Using the JTAG TAP is one OBP alternative suitable for space constrained PCB environments. You can use JTAG- compliant devices such as a microprocessor, microcontroller, buffers or ASICs to program flash memories. It does not matter what type of device it is as long as all flash memory address, data and control pins connect to the JTAG BSR.

Flash memory programming with the JTAG TAP is fast. We optimized the equipment and software used in our benchmark experiment for the best programming performance. If you follow the suggestions outlined in this appplication note you also can create an optimized programming setup to program flash memories quickly.

Each JTAG programming setup is different. Software and hardware engineers together should investigate the programming requirements of a particular application and implement ideas from this document as needed.

# APPENDIX A
# JTAG VENDOR REFERENCES

**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.

ASSET-InterTech
2201 N. Central Expressway
Suite 105
Richardson, TX 75080
telephone (214) 437-2800

Corelis
12607 Hidden Creek Way
Cerritos, CA 90703
telephone (310) 926-6727
fax (310) 404-6196

Data I/O Corp.
10525 Willows Rd. NE
Redmond, WA 98073
Customer Resource Center (800) 247-5700
fax (206) 882-1043
http://www.data-io.com

JTAG Technologies B.V.
5612 AN Eindhoven
The Netherlands
telephone ++31 40 295 08 70
fax ++31 40 246 84 71
internet: info@jtag.nl

Needhams Electronics
4630 Beloit Drive #20
Sacramento, CA 95838
telephone (916) 924-8037
fax (916) 924-8065

SMS Mikrocomputer Systeme GmbH
Im Grund 15
D-88239 Wangen
Germany
telephone 49-7522-97280
fax 49-7522-972850
http://www.sms-sprint.com

# APPENDIX B
# ADDITIONAL INFORMATION

## RELATED INTEL INFORMATION(1,2)

| Order Number | Document/Tool |
|---|---|
| 210830 | Intel *Flash Memory Databook* |
| 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* |
| 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. Visit Intel's World Wide Web home page at http://www.Intel.com for technical documentation and tools.

## OTHER RELATED INFORMATION

*IEEE Std 1149.1 Standard Test Access Port and Boundary-Scan Architecture* (available from IEEE, Inc,. 345 East 47th Street, New York, NY 100167, USA)

*The Boundary-Scan Handbook*, author: Kenneth P. Parker, publisher: Kluwer Academic Publishers

*Boundary-Scan Test*, authors: Harry Bleeker, Peter van den Eijnden, Frans de Jong, publisher: Kluwer Academic Publishers

AP-Note JTAG-102, *JTAG In-Circuit Programming of Flash Memory* available from Corelis (see Appendix A for vendor contact information)