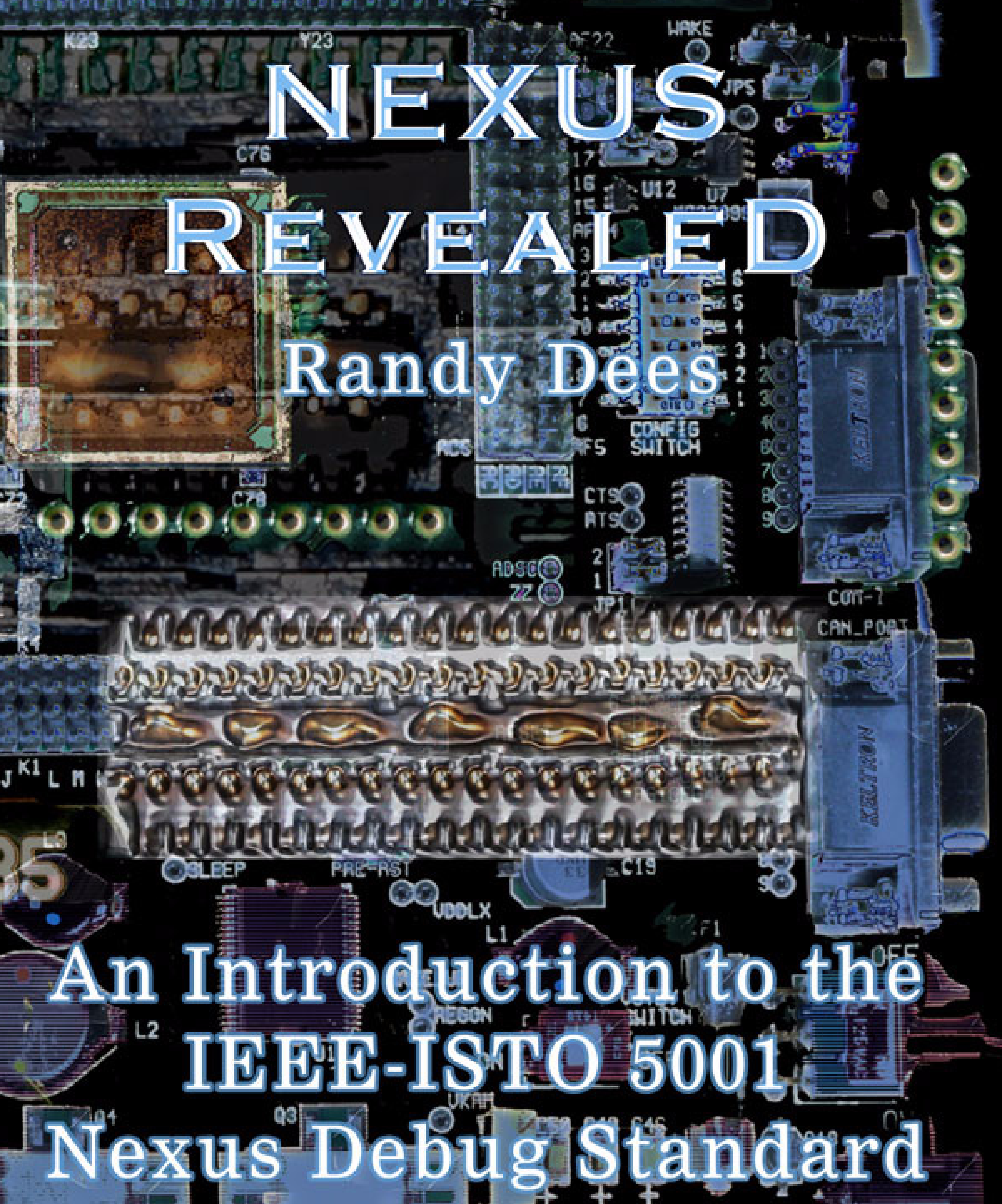

NEXUS REVEALED

Randy Dees

An Introduction to the IEEE-ISTO 5001 Nexus Debug Standard

This is a preview copy of the book, “Nexus Revealed” which was never completed.

Nexus Revealed

An Introduction to the IEEE-ISTO 5001 Nexus Debug Standard

By Randy Dees

Drippin’ Shine Designs

Nexus Revealed
An Introduction to the IEEE-ISTO 5001 Nexus Debug Standard

By Randy Dees

Published by
Drippin' Shine Designs

Preview Edition

Cover art and design by Mike Cipolla (mc3designs.com), based on photos by Randy Dees

Preface

In 1999 a new standard, the IEEE-ISTO 5001-1999 - informally known as Nexus, was published as a standard for debugging microcontrollers. The standard was updated in 2003. This book describes the standard in detail and its use and implementation on microcontrollers from several semiconductor manufacturers.

This first part of the book focuses on the standard itself. The second part of the book looks at the different implementations on several devices from multiple semiconductor vendors. The last part of the book looks at actually using Nexus in debugging microcontrollers.

The book is organized as follows:

Part 1 - In Theory

Chapter 1- An Introduction, History, and Overview: The first chapter provides an introduction to the needs for a debug standard, the history of getting to an implemented standard, and an overview of the standard itself.

Chapter 2 - Introduction to JTAG Debug: This chapter provides an overview of the IEEE 1149.1 standard for integrated circuits and its use as a debug interface.

Chapter 3 - Nexus Features: This chapter focuses on the features available via the Nexus Standard.

Chapter 4 - Nexus Auxiliary Port Messages and Auxiliary Port Message Protocol: This chapter delves into details of the Auxiliary Port messages that form the backbone of the Nexus debug standard.

Chapter 5 - Nexus Recommended Registers: The Nexus Standard provides a recommended set of registers to control the various features of the Nexus Standard. This chapter details the recommended register and bit descriptions.

Chapter 6 - Nexus Connectors: Due to the varied requirements of systems that use the Nexus Standard, multiple connector standards are recommended. This chapter describes all of the different connector options (styles and sizes) that are recommended for different types of environments.

Part II - In the Real World

Chapter 15 - Working with Nexus: This chapter includes pointers for using Nexus optimally for debugging microcontroller systems.

Chapter 16 - Poor Man's Trace: Even without advanced and sometimes expensive commercial debuggers that do trace reconstruction and completely decode Nexus messages, Nexus can be used with a simple logic analyzer to provide some debug capabilities above what can be done with a static debug interface. This chapter shows how a logic analyzer can be used to obtain additional information about the system being debugged.

Appendix A - References: Appendix A lists documentation that was either referenced in the writing of this book or is readily available for additional information on Nexus and devices that include the Nexus debug interface.

Appendix B - JEDEC Manufacturer Identifiers: The IEEE-ISTO 5001 Nexus Standard includes an option to use the IEEE 1149.1 (JTAG) interface for control of the Nexus debug interface, including using the JTAG Device Identification Register for device identification. This appendix contains an extract from the JEP106P standard of the manufacturer identification code used in the JTAG Device Identification Register of semiconductor manufacturers that are or have been members of the Nexus Consortium.

Appendix C - Debug Protocols: BDM, JTAG - What's the difference? A short summary of the different debug protocols from semiconductor manufacturers.

Appendix D - About the Author: This gives a look at the background and history of the author, Randy Dees, currently an employee of Freescale Semiconductor, Inc.

CHAPTER 1

An Introduction, History, and Overview

The Early Days of Debug

Debug on the very first microcontrollers consisted of a ROM monitor, a special set of software that ran on the target system that allowed the user to modify registers and memory, set breakpoints, and step though the software being developed. This allowed developers to monitor the target system. This was good when the code size was small, but as embedded systems became more complex, so did the software and the software development process. Emulators were the next evolution in the debug process. Emulators many times used Logic Analyzers to capture and analyze memory accesses. They could not, however, see internal operations without special versions of the devices that brought out visibility to internal resources. In the early 1980's, microprocessors started adding on-chip memory and on-chip peripherals creating the first microcontrollers. Emulators relied very heavily on bond-out "visibility" versions of a device and special devices called Port Replacement Units (PRU). PRUs allowed devices to substitute an address and data bus for some of the on-chip peripherals pins. The user could then see the program execution on the memory bus pins and still be able to use the peripherals whose functions were copied into the external PRU device. Motorola Semiconductor[1] introduced something they called Background Debug Mode (BDM) on the MC68300 family. This was expanded to many other families of Motorola microcontrollers, including the HC12, HCS08, HCS12, MPC500, and the MPC800 families. BDM allows the microcontroller itself to be connected to an external computer and used the microcontroller to debug itself. BDM gave access to read and write memory, execute instructions, start and stop programs, and even set breakpoints. BDM was supplemented on the MPC500 family with on-chip trace capabilities as well. Trace on devices running from external memory required only logic analyzers to determine program flow but as microcontroller systems became more complex and included internal memory and cache, trace became more complicated. In these cases, trace required some

1. Freescale Semiconductor became a publicly traded company in July 2004 after more than 50 years as a part of Motorola, Inc.

intrusion into the system, including affecting system performance and restricting use of some of the microcontroller pins. This required an ever increasing number of pins on a 32-bit MCU to support (32 data pins, 24-32 Address pins, Bus Control pins, and more). It became clear that as microcontrollers become more complicated, integrating more internal memory (both program memory and data memory), a new type of debug interface was needed.

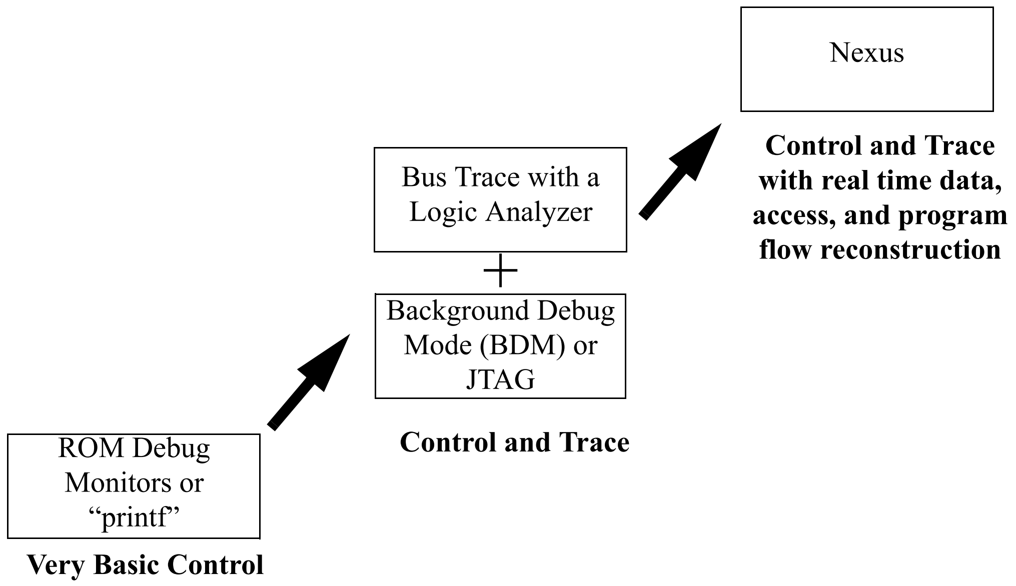
Nexus
Control and Trace with real time data, access, and program flow reconstruction
Bus Trace with a Logic Analyzer
+
Background Debug Mode (BDM) or JTAG
Control and Trace
ROM Debug Monitors or "printf"
Very Basic Control

FIGURE 1.1. **Evolution of Debug Solutions**

History Of Nexus

Growing from a white paper written jointly by Motorola Semiconductor (now Freescale Semiconductor, Inc.) and Hewlett Packard (now Agilent Technologies), the Nexus Consortium began back in 1998 as an idea to define an industry standard debugger interface for embedded micro-controllers. The original paper referred to this new standard as GEPDIS (Global Embedded Processor Debug Interface Systems). Soon other companies were enlisted, including other semiconductor companies (Hitachi Semiconductor - now Renasas, and Siemens - now Infineon Technologies) and tool vendors, and meetings were held in the United States, Europe, and Japan to work out the details of the new standard. Once the first revision of the standard had been hammered out by the initial group of companies, it was turned over to the IEEE-ISTO organization to manage the day to day operation of running a standards consortium where it was approved by the IEEE and became known as the IEEE-ISTO 5001-1999 Nexus Standard. In 2003, the Nexus Standard was updated with additional enhancements and corrections. The newest revision of the standard is known as the IEEE-ISTO 5001-2003. The current members of the Nexus Consortium include not just semiconductor companies and tool vendors, but also users of Nexus microcontrollers, and are shown in Table 1.1.

TABLE 1.1 Current Nexus Consortium Members (February 2005)

Semiconductor Manufacturers

Freescale Semiconductor, Inc.	ST Microelectronics
Infineon Technologies	

Tool Vendors

Ashling Microsystems	IAR Systems
dSPACE GmBH	iSYSTEM GmBH
ETAS, Inc.	Lauterbach
First Silicon Solutions (FS2)	Metrowerks
Greenhill's	Samtec
Hitex Development Tools	Wind River

User Companies

Ford Motor Company	Motorola, Inc.
Delphi Automotive Systems	Visteon
General Motors	

Over the last few years, some members of the Nexus Consortium have dropped out of the Consortium for varying reasons, but several made significant contributions. Some dropped out of the consortium when annual dues started being charged. Some of these are listed in Table 1.2.

TABLE 1.2 Past Nexus Consortium Members (January 2005)

Semiconductor Manufacturers

Alphamosaic Ltd.	Mitsubishi Semiconductor
Altera	National Semiconductor
Cygnal	Renases (formerly Hitachi)
Lucent Technologies	

Tool Vendors

ADL	Nohau
Agilent Technologies (formerly Hewlett Packard)	Noral
Applied Microsystems[a]	PLX
Hiware [b]	Tektronix
Macraigor Systems	Yokogawa

a. Now out of business. Some assets were bought by Metrowerks.

b. Now owned by Metrowerks.

The annual dues are low (in terms of comparative industry consortiums) with a three tier fee: semiconductor manufacturers, tool vendors and user companies, and the lowest, associate members (reserved for wholly owned subsidiaries of full members).

Why was Nexus Consortium Formed

The Nexus Consortium was formed to define a standard debug interface that could be used by multiple microcontroller manufacturers. The skyrocketing costs of developing new tools for every new microcontroller family, combined with the development time required for new tools drove a need for standardization. A secondary factor was the increase of System-On-a-Chip microcontrollers that included all of the system memory with the processor inside smaller packages that prevented access to the stem address and data bus. Two specific applications, automotive powertrain controllers and hard disk controllers, rely on access to parts of processor memory to tune the performance of the engine or hard disk “on the fly”. Without external memory, which SOCs eliminate, it becomes difficult to do this calibration work. Some advantages of Nexus:

- Reuse of MCU debug designs saves time and cost of both silicon creation and tool development.
- Reuse of tool debug hardware for multiple devices and architectures.
- Reduce development time for new processors by using known protocols.

First Public Demonstration

The first public showing of a Nexus microcontroller with tools was at the Embedded Systems show in Chicago (USA) February 28 - March 2, 2000 with a Hiware[2] debugger running on a Motorola M*Core based Nexus test microcontroller through an interface from Metrowerks with connections to a logic analyzer from Tektronics. A second showing occurred in London (England) on May 24-25, 2000.

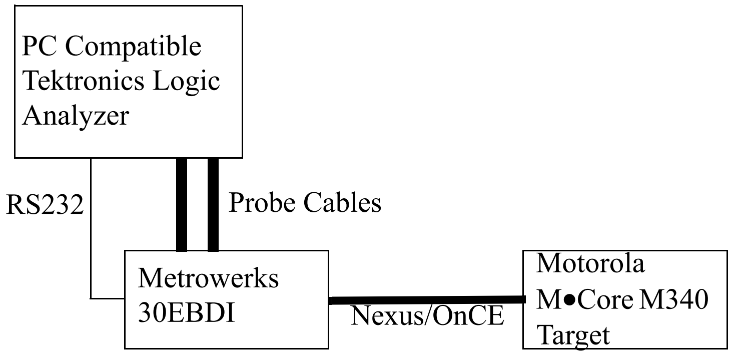
PC Compatible Tektronics Logic Analyzer
RS232
Probe Cables
Metrowerks 30EBDI
Nexus/OnCE
Motorola M•Core M340 Target

FIGURE 1.2. First Public Demonstration Showing of Nexus

This demonstration showed that not only was this proposal feasible, it was practical. The first production Nexus microcontroller to be introduced was the MPC565 from Motorola Semiconductor (now Freescale Semiconductor). The MPC565 was announced in Detroit in October 2000 at the bi-annual Convergence Conference.

Processor Support

Nexus is currently supported by microcontrollers from several semiconductor companies and runs the gamut of applications from automotive powertrain controllers to video processors and hard disk controllers. Table 1.3 lists all of the processors that support Nexus at

2. Hiware was later bought by Metrowerks, which is now owned by Freescale Semiconductor.

the time of this book publication. Moving into the future, more processors will be added to this list.

TABLE 1.3 Current Nexus Microcontrollers

Manufacturer	Core Type		Target Market (Applications)
Alphamosaic	DSP	SC01, SC02	Video and Multimedia Application Processor
Freescale Semiconductor	PowerPC RISC RCPU Core	MPC531, MPC533, MPC535, MPC561, MPC562, MPC563, MPC564, MPC565, MPC566	Automotive Powertrain
	PowerPC Book E e200	MPC5553, MPC5554	Automotive Powertrain
	ARM7	MAC7101, MAC7104, MAC7111, MAC7112, MAC7114, MAC7121, MAC7122, MAC7124, MAC7134	Automotive Dashboard Applications
	ARM9	Custom	Handheld Personal Digital Assistants
	DSP	Custom	StarCore
	eTPU	MPC5553, MPC5554, MCF5232[a], MCF5233, MCF5234, MCF5235	Enhanced Timing Processor Unit
	MCore	Custom	Automotive Body and Control
National Semiconductor	CompactRISC CR16C		Wireless Baseband Solutions
ST Microelectronics	Super 10 Megacell	ST10R301, ST10R302, ST10R303, Custom	Custom hard disk controllers Automotive
	MMDSP+	?	?

a. The MCF5232, MCF5233, MCF5234, and MCF5235 support the Nexus Auxiliary Output port for trace and the Nexus development support registers for the eTPU, but are accessed via the Coldfire Background Debug Port.

What is Nexus

Nexus is a debug standard that defines different levels of on-chip feature requirements, pin interfaces, protocols, and even connectors. The standard defines both static and dynamic debug features. The message protocol is based on packets of information.

Static debug features are the typical run control and stop mode debug features, such as modifying registers, loading memory, start program execution, and set and use breakpoints. In other words, they are features that require the target processor to be stopped for access and provide a means of stopping the processor via breakpoints set while stopped.

Dynamic features are more advanced. Dynamic features allow access to and real-time information about a target while it is executing code. These features are required for the more advanced system-on-a-chip microcontrollers.

Nexus Classes

Nexus provides a standard method of determining the debug capabilities of different MCUs, by defining four different classes or levels of functionality. Each class has a set of required features that must be supported and a second set of optional features that are not required, but could be available. In some cases MCU vendors may add features from a higher class, but may not support all of the features of the higher class. This is commonly referred to as a "+" level of features. Class 2+ would mean that the MCU supports all Class 2 features plus it supports some features that are required for either Class 3 or Class 4, but does not support all features of a higher Class. By defining a standard support level, it becomes easy to define the capabilities of a given processor.

Nexus Class 1. Nexus Class 1 offers the minimum level of debug capability (basically static debug) and control of the target system, including run control (starting, stopping, breakpoints), reading and modification of memory registers. This is basically the same functionality of the traditional Freescale BDM interface or the common JTAG connections. Class 1 requires that all static debug features be supported as well as some dynamic features.

TABLE 1.4 Class 1 (and higher) Static Debug Features

Development Feature	Class 1	Class 2	Class 3	Class 4
Read/write user registers in debug mode	√	√	√	√
Read/write user memory in debug mode	√	√	√	√
Enter a debug mode from reset	√	√	√	√
Enter a debug mode from user mode	√	√	√	√
Exit a debug mode to user mode	√	√	√	√
Single step instruction in user mode and re-enter debug mode	√	√	√	√
Stop program execution on instruction/data breakpoint and enter debug mode (minimum 2 breakpoints)	√	√	√	√

In addition, Class 1 functionality requires that some dynamic debug features be supported.

TABLE 1.5 Dynamic Debug Features

Development Feature	Class 1	Class 2	Class 3	Class 4
Breakpoint/Watchpoints - The ability to set breakpoints or watchpoints.	√[a]	√	√	√
Device ID Message - Device Identification	√	√	√	√

a. Since Class 1 does not require an Auxiliary port, Watchpoints are signaled via the Event out (EVTO) pin.

√ = Required Feature

Nexus Class 2. Nexus Class 2 requires that all Class 1 features be supported and adds support for Watchpoints, Ownership Trace, and Program Trace. In most cases, an Auxiliary port is required, however there is a methodology defined to embed Auxiliary port messages into a JTAG stream.

TABLE 1.6 Class 2 (and higher) Dynamic Debug Features

Development Feature	Class 1	Class 2	Class 3	Class 4
Watchpoint Message - Ability to send out an event occurrence when watchpoint matches	—	√	√	√
Ownership Trace - Monitor process ownership while processor runs in real-time	—	√	√	√
Program Trace - Monitor program flow while processor runs in real-time (logical address)	—	√	√	√

√ = Required Feature
— = Not required to be supported

Nexus Class 3. Nexus Class 3 requires all Class 2 features and adds the capability of performing data trace.

TABLE 1.7 Class 3 (and higher) Dynamic Debug Features

Development Feature	Class 1	Class 2	Class 3	Class 4
Data Trace (Writes Only) - Monitor data writes while processor runs in real-time	—	—	√	√
Read/Write Access - Read/write memory locations while program runs in real-time	—	—	√	√
Data Trace - Monitor data reads while processor runs in real-time	—	—	O	O
Port Replacement/Sharing - LSIO port replacement and HSIO port sharing	—	—	O	O
Data Acquisition - Transmit data values for acquisition by tool	—	—	O	O

√ = Required Feature
O = Optional Feature
— = Not required to be supported

Nexus Class 4. Nexus Class 4 requires all Class 3 features and adds some very advanced features such as memory substitution and the capability of enabling or disabling trace upon a watchpoint occurrence.

TABLE 1.8 **Class 4 Dynamic Debug Features**

Development Feature	Class 1	Class 2	Class 3	Class 4
Memory Substitution - Program execution (instruction/data) from Nexus port for reset or exceptions	—	—	—	√
Development Control and Status - Ability to start ownership, program or data trace upon watchpoint occurrence	—	—	—	√
Development Control and Status - Ability to start memory substitution upon watchpoint occurrence or upon program access of device-specific address	—	—	—	O

√ = Required Feature
O = Optional Feature
— = Not required to be supported

Hardware Interfaces

Microcontrollers have previously had debug interfaces that were not standardized. JTAG, while a common interface used for debug, is not even standardized. The original proposal that eventually became the Nexus Standard proposed a more standardized input and output interface, an Auxiliary Input Port and an Auxiliary Output Port. Unfortunately, by the time the Nexus consortium was founded, JTAG had gotten a strong toe hold as a standard debug interface for run control. The Nexus Standard, therefore defines two types of interfaces: the Auxiliary Only port that incorporates both an Auxiliary Input Port and the Combined JTAG/Auxiliary Port.

The Auxiliary Only Port allows for Control, Program and Data Trace, and Read/Write Access messaging all to be performed using the same type of packet based protocol.

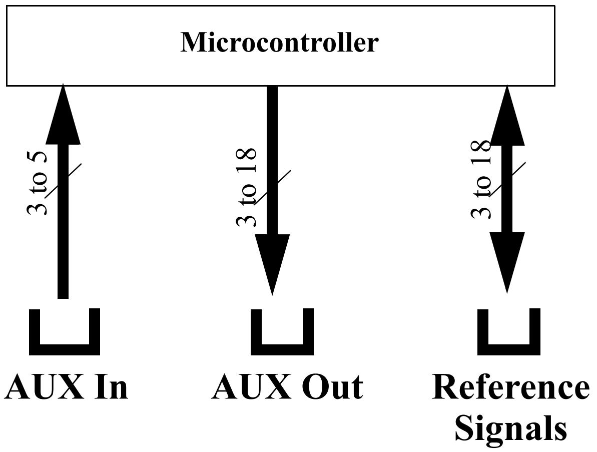

FIGURE 1.3. **Auxiliary (AUX) only Port**

The Combined JTAG/Auxiliary Port allows for JTAG to be used for commands and responses to commands, including read/write messages, but Program, Data, Ownership, and Watchpoint Trace Messages are output on the Auxiliary Output port. The combined JTAG/Aux port also allows for an Event Input pin ($\overline{\text{EVTI}}$).

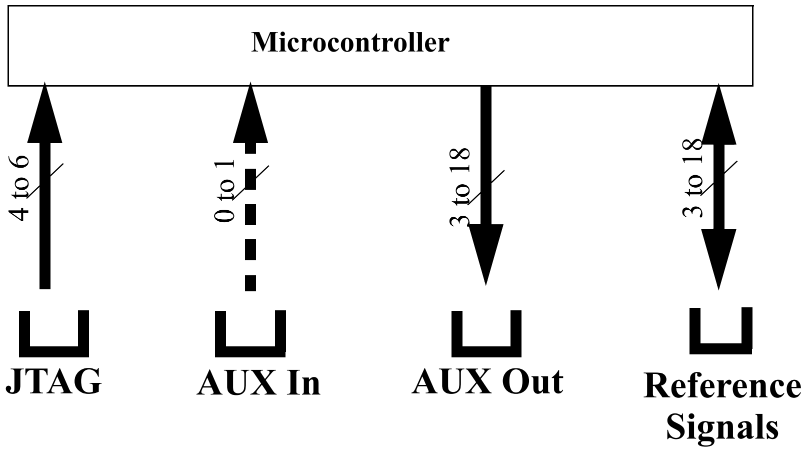

FIGURE 1.4. **Combined JTAG/Auxiliary (AUX) Port**

APPENDIX D

About the Author

The author graduated from the University of Houston in December 1980 where he built his first single board computer, a 2 MHz Z80 with 1K of SRAM, 2K of ePROM, a hex keypad, and six 7-segment LEDs. He later expanded it to 6K of SRAM, a serial port connected to a Lear Siegler ADM3a terminal, and a home-brew, self-written, hand assembled monitor. One of his senior projects was to design and build a successive approximation Analog to Digital converter from basic glue logic. The hardest part of the project ended up being not the A-to-D converter, but the requirement to display the result (volts and tenths of a volt - 0 to 5.0 volts) on Binary Coded Decimal LEDs. This exercise was to show how much easier it was to use a microcontroller to do the same task using a parallel port interfaced to a Digital to Analog Converter and a comparator and perform most of the task in software. It is much simpler to convert a binary number into decimal with software than it was with logic.

Upon graduating from college, he immediately went to work for Motorola Semiconductor in January of 1981, relocating to Austin, Texas. For his first 18 years at Motorola, he worked in Product Engineering where he primarily supported new product introductions of Telecommunication Devices such as 300 band frequency Shift Keying (FSK) modems, 1200 and 2400 differential phase shift keying (DPSK) modems, analog filters, CMOS op-amps, µ-law and a-law codecs, and two generations of pre-ISDN voice/data transceivers, including everything from test development, characterization, and specification development, to device definitions, applications support, and IC design. Eventually he drifted into the job of being the technical liaison supporting custom telecommunication devices for a large, major customer.

After a reorganization, his group was redefined to support custom microcontrollers, which led to his support of the MPC555 for one of its first OEM customers. During his work on the MPC555, it became clear that he spent most of his time supporting application development, which led to his transfer to the Applications organization for the MPC500 family of microcontrollers.

During the development of the MPC56x devices (the first commercially available MCU's that support Nexus) he became involved with the IEEE-ISTO 5001 consortium. Along with Rich Collins (also of Motorola/Freescale), he co-chaired the Hardware Technical subcommittee of the 5001/Nexus consortium. It was during this time that the IEEE-ISTO 5001-1999 was updated to the IEEE-ISTO 5001-2003 standard.

When not working at Freescale Semiconductor, Inc., the author lives on a small ranch, in the middle of freaking nowhere (somewhere near Pedernales Falls State Park), with 7 cats, and untold numbers of deer, armadillos, skunks, jack rabbits, wild turkeys, raccoons, ring-tail cats, and foxes. He enjoys digital photography and travelling with his lovely wife, Mary Jo. This autobiography was written while overlooking the Pacific Ocean in Ixtapa, Mexico. While most of this book was written out on the ranch, parts were written in less glamorous locales of Detroit, San Francisco, Ashville (North Carolina), and even a few pages in a hospital while recovering from minor surgery.