

Auto industry drives embedded boundary-scan debugging



Photo courtesy Infineon Technologies

BOUNDARY-SCAN TECHNIQUES CONTINUE TO EVOLVE TO TACKLE THE DEVELOPMENT AND DEBUGGING OF DEEPLY EMBEDDED SYSTEMS. THE AUTOMOTIVE INDUSTRY'S SPECIAL REQUIREMENTS ARE ENCOURAGING MAJOR SEMICONDUCTOR AND TEST-TOOL VENDORS TO WORK TOGETHER FOR A NEW GLOBAL DEBUGGING STANDARD.

At a glance.....23

For more information24

AS *EDN EUROPE* REPORTED in its last issue, boundary-scan techniques continue to develop from their original pc-board test environment to meet evolving needs, such as in-system configuration and system-on-chip testing (**Reference 1**). Yet another area in which

boundary-scan debugging enjoys widespread application is the automotive industry. Today's deeply embedded in-vehicle systems frequently comprise networked components, such as engine-management units and active safety controls, that require increasingly capable processors to provide power-train supervision. Complex devices with multi-

ple cores and deep on-chip memories are becoming more difficult to emulate and debug, compromising the automotive industry's notoriously short design cycles—especially among the interdisciplinary development teams that tackle today's vehicle development projects. In-system programmability is also critical to automotive development because as

much as 70% of a vehicle's final system-calibration data relies on iterative in-vehicle tests.

Debugging embedded processors in real time traditionally requires a logic analyser with a chip-specific preprocessor, an in-circuit emulator, or both. Such equipment can provide a programmer's view of chip resources and typically comprises a general-purpose base unit with dedicated pods that emulate the target processor in FPGA-based hardware. But more and more, today's processor-clock speeds and inaccessible packaging techniques limit or preclude external techniques. It's also very difficult to trace on-chip memory accesses because these cycles don't appear at the processor's pins—although some semiconductor vendors produce special bond-out versions of the target microcontroller that provide access to otherwise hidden core nodes. Instruction pipelines, concurrent execution, and predictive branching provide further debugging challenges. To address such concerns, Motorola started adding on-chip monitoring to its CPU32-series cores to pioneer BDM (background-debug mode) some 10 years ago. Other semiconductor vendors have since followed suit with proprietary on-chip hardware, often employing the processor's IEEE 1149.1 test interface. Generically, such OCD (on-chip-debugging) techniques link the chip's internal resources with a host PC's development environment to make chip activity visible and to furnish debugging controls.

These similar proprietary OCD implementations are an obvious candidate for a standardisation effort to rationalise the debugging interface and promote toolchain development. Recognising this need, semiconductor developers Hitachi, Infineon, and Motorola joined with automotive-tool developers Agilent Technologies, Robert Bosch, and ETAS to form the Nexus 5001 Forum back in April 1998. Working under the umbrella of the IEEE Industry Standards and Technology Organisation, the Nexus Forum released its first specification in September 1999. Such a standardisation pace had been hitherto unknown in traditional IEEE circles. Today, the Nexus Forum's member list reads like a global directory of semiconductor vendors and software-development toolmakers. You can download the Forum's IEEE-ISTO

AT A GLANCE

- ▶ System-on-chip processors and today's packages mandate on-chip debug systems.
- ▶ Automotive applications demand nonintrusive access to calibration constants.
- ▶ Motorola's pioneering background-debug-mode serves low-to-midrange applications.
- ▶ Today's complex processors require parallel debugging access to track on-chip events.
- ▶ IEEE-ISTO 5001-99 harmonises proprietary on-chip-debug mechanisms.

5001-99 specification for free at www.ieee-isto.org.

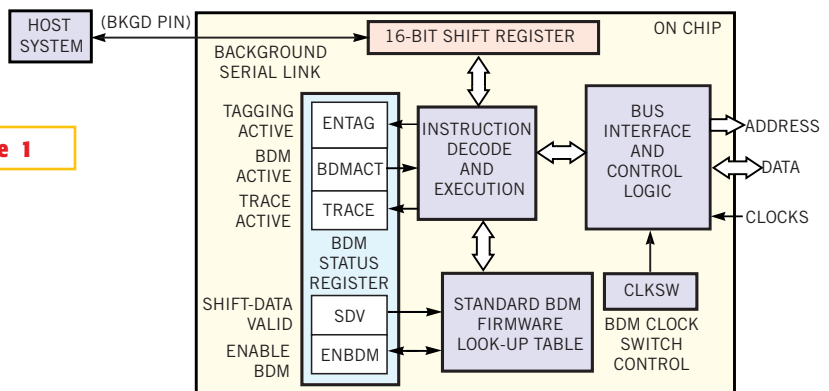
OCD STARTS WITH A ROM MONITOR

Providing further impetus for 5001-99, proprietary OCD implementations vary enormously in their capabilities and rarely feature common physical interfaces. Written into the CPU32's microcode in chips such as the 68331, Motorola's original BDM emulates a traditional ROM monitor and includes a similar command set. Debugging commands include read and write data, address, and system-control registers; read and write memory or blocks of memory; flush and refill the pipeline before resuming execution; reset the processor's peripherals; and call user patch-code. The original hardware-debugging interface is via a 10-pin header that carries seven control sig-

nals to serially load or unload data, to provide clock and breakpoint control, and to freeze and reset the processor. A comparable BDM implementation appears in later Motorola microcontrollers, such as the growing 68HC12 family that powers many automotive control units (Figure 1). Sampling now, the latest family member is the MC9S12DP256; it augments typical microcontroller peripherals with features such as 256 kbytes of flash memory and as many as five CAN (controller-area-network) interfaces.

In the 68HC12, the BDM interface reduces to just two active pins in a six-pin header: the BKGD (background) serial-communications line and hardware-reset signal. The BKGD line uses a clocking scheme that resolves bidirectional communications that are—from the processor's view—asynchronous from host to processor and synchronous from processor to host. The remaining header pins carry the supply voltages and the flash-memory programming voltage to permit in-system programming. The 68HC12's debugging mechanisms divide into hardware and microcoded functions that lie outside of the processor core to minimise intrusiveness. The hardware functions don't require the processor to be running BDM to allow nonintrusive access to on-chip memory during normal program operation, but the debug hardware may steal processor clock cycles if the memory address is off-chip. The microcoded functions require BDM to be active and typically use the processor's dead cycles to execute debugging commands, stealing clock cycles only if no dead cycles appear within 128 cycles of receiving a command.

Figure 1



NOTE: BDM=BACKGROUND-DEBUG MODE.

Motorola's BDM (background-debug mode) provides full access to on-chip resources that include an embedded monitor via a single-wire serial link.

Microcoded functions include read and write the processor's register set, execute a user instruction and return to BDM (trace mode), enable instruction "tagging," and resume normal processing. You activate BDM by sending a debugging command that enables the debugging microcode. The processor can then enter BDM via a debug-hardware command, by encountering breakpoints, or via the enter-background-mode instruction. When BDM mode is active, the 68HC12 maps its debugging-control and microcode registers to the top of its 64-kbyte memory space. These registers reflect the status and type of BDM instruction that's executing and provide temporary storage for data and commands.

As for operation with a traditional emulator, the 68HC12's debugging system includes hardware-breakpoint capabilities to halt instruction execution on address or address-and-data combinations. You can typically set breakpoints to halt the processor before or after an instruction executes and have the breakpoint trigger a software interrupt or force BDM. Like most contemporary processors, the CPU12 core uses queued instruction pipelining that accelerates execution speed but obscures program-flow analysis. You can reconstruct program flow from debugging events or, if you're using an external logic analyser, monitor two dedicated device pins that provide time-multiplexed information about instruction execution and data movements. But because instruction execution begins before becoming visible to a logic analyser or debugging environment, the CPU12's designers included a separate tagging mechanism that tracks instruction status. When a tagged instruction reaches the head of the

queue, the processor suspends execution and activates BDM. You can then execute the instruction and observe the results. Together, the 68HC12's debugging mechanisms provide tools such as P&E Microsystems' less-than-\$500 development environment with powerful but low-cost access to the processor's resources. Running on a PC host, the P&E tool uses a parallel-port based "wiggler" to provide the BDM interface. P&E's tools also support BDM-enabled Motorola targets, including the 68HC16, 68HC3xx, ColdFire, MCore, and PowerPC families.

DEBUG-PORT REMAPPING SPEEDS I/O

As is the case for traditional RS-232 based ROM monitors, the main restrictions on serial-debugging performance are due to I/O bandwidth between the chip and the host PC. Accelerator cards, such as P&E's Lightning adapter or Ethernet-based interfaces, such as Abatron's debuggers, can communicate 10 times faster than a parallel-port-based "wiggler." Abatron quotes maximum host-to-target speeds of 2 Mbps and debug-code loading at as fast as 320 Kbytes/sec for its BDI2000 interface. This accelerated serial throughput suits midrange debugging tasks with complex processors such as Infineon Technologies' TC1775 microcontroller, but faster throughput requires additional parallel access to on-chip resources.

Currently in its first generation and with features such as a DSP-like multiplier-accumulator, a separate peripheral-control processor, twin-CAN communication links, and direct calibration support, Infineon's TC1775 targets automotive use. The 329-pin BGA package mandates the OCDS (on-chip-debugging-support) system, which uses a com-

bination of JTAG pins and device-pin mapping to accelerate I/O performance during debugging. Accordingly, the TC1775's debugging system illustrates some of the techniques that you can expect from 5001-99-standard-compliant chips. The TC1775's debug system comprises support for the main TriCore processor and its peripheral-control processor, a trace module for the TriCore processor, and an extended JTAG-based debugger interface that Infineon calls Cerberus (Figure 2). The main TriCore processor partitions debugging into event-generation mechanisms and subsequent debugging actions. The idea is to integrate a simple but powerful debugging system and rely upon external support for more complex actions. Ewald Liess, Infineon's applications manager for microcontroller products, explains, "The driving philosophy behind the OCDS support is that the complete architectural state of the system is visible from the TriCore's FPI (flexible-peripheral-interconnect) bus." In other words, he notes, you can access every piece of architectural state in the TC1775 through a mapping into the FPI's address space.

The TC1775's hardware-breakpoint-generation logic responds to on-chip events or to an external break command from an emulator. You can set the hardware-breakpoint logic to trigger upon instruction execution or data read/writes at an address or within an address range. Trigger-event-specifier registers provide combinational trigger capabilities to generate a single trigger from multiple events, such as code- and data-space-protection violations. Specifier registers within the chip's OCDS control-register set control debug actions when external and software breaks occur and, upon instruction, move

FOR MORE INFORMATION...

For more information on products such as those discussed in this article, go to our information-request page at www.edn-info.com. When you contact any of the following manufacturers directly, please let them know you read about their products in *EDN Europe*.

Abatron

www.abatronag.ch
Enter No. 442

Ashling Microsystems

www.ashling.com
Enter No. 445

Infineon Technologies

www.infineon.com
Enter No. 448

P&E Microcomputer Systems

www.pemicro.com
Enter No. 450

Robert Bosch

www.bosch.de
Enter No. 451

Agilent Technologies

www.agilent.com
Enter No. 443

ETAS

www.etas.de
Enter No. 446

Motorola Semiconductor

www.mot-sps.com
Enter No. 449

Samtec

www.samtec.com
Enter No. 452

AMP

www.amp.com
Enter No. 444

Hitachi Semiconductor

www.hitachi-eu.com
Enter No. 447

SUPER INFO NUMBER

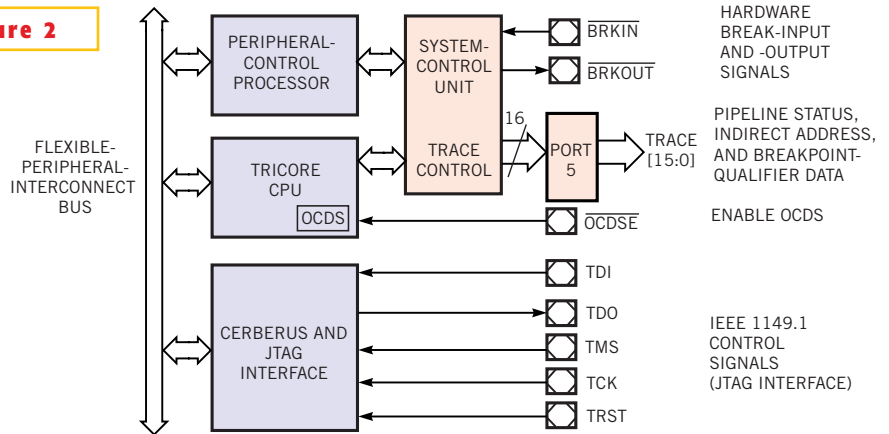
For more information on the products available from all of the vendors listed in this box, Enter No. 453 at www.edn-info.com.

between the processor's protected core-special-function registers and its general-purpose registers. Debug actions include halting execution; entering the debug monitor; entering software-debug mode; and asserting the external break pin to synchronise external hardware, such as an emulator. Debugging actions can generally occur immediately before or after a trigger event. A debug-status register provides overall control.

The TC1775 augments the five 1149.1-standard device pins with emulator hardware-control signals that comprise OCDS enable, break input, and break output. A trace command optionally remaps the microcontroller's 16-bit I/O port 5 to carry trace information. An emulator can use this information to reconstruct TriCore or peripheral-control-processor program flow in real time or offline. Debugging the peripheral-control processor relies upon a dedicated "debug" instruction that the debug tool inserts at critical code segments, reflecting traditional code-instrumentation practice. This debug instruction triggers in response to condition codes becoming true to store the current instruction address and I/O channel number. Optionally, you can halt channel execution or generate a synchronisation signal on a device pin.

Program-flow information also facilitates software-development tools, such as profilers and code-coverage analysers. Each cycle, the TC1775's processor core transmits state information that comprises 5 bits of pipeline status, 8 bits of indirect-program-counter information, and 3 bits that describe breakpoint qualifiers. An emulator synchronises with program execution by examining the

Figure 2



NOTE: OCDS=ON-CHIP-DEBUGGING SUPPORT.

Infineon's first-generation TriCore processor includes an on-chip-debug system with capabilities similar to those of 5001-99 Class-2 devices.

pipeline-status codes and waiting for an indirect branch, such as a return-from-subroutine instruction. A four-entry FIFO decouples the processor's 32-bit program counter from the 8-bit program-counter bus that outputs indirect-branch-target addresses, one byte at a time on sequential processor cycles. The FIFO rarely overruns; Infineon's analysis shows that overruns can happen only following several back-to-back jump-indirect instructions. You will typically not encounter this condition. A \$400 TC1775 starter kit is available now.

FOUR CLASSES POWER 5001 DEBUGGING

OCD's main limitations lie with the complexity and capability of the test logic, available buffer-memory depth for capturing results, and communication speed with the external development environment. To balance these issues and meet the needs of a range of processor ar-

chitectures, the 5001-99 specification divides debugging performance into four compliance classes (Table 1). The minimum Class-1 capability is similar to Motorola's BDM, and the highest level Class 4 capability extends the real-time program-tracing and on-the-fly memory-access capabilities of chips such as the TC1775 to include data acquisition and memory substitution. In an automotive context, on-the-fly memory access is essential to provide a mechanism for changing calibration constants, for example while an engine is running. Data-acquisition capabilities make visible on-chip data values, such as an engine-management unit's calibration constants. In a system that supports it, memory substitution can follow an exception or reset event to redirect internal processor accesses to external program and data memory via an optional Nexus auxiliary port.

TABLE 1—5001-99'S FOUR CLASSES

IEEE-ISTO 5001-1999 Class	Class 1	Class 2	Class 3	Class 4
Trace features	None	Adds ownership trace and program trace via auxiliary ports	Adds data-write trace and read/write memory on the fly via auxiliary ports	Allows tracing to be triggered by a watchpoint via auxiliary ports
Debug-communication method	Half-duplex communication	Communication may be full-duplex using auxiliary port	Communication may be full-duplex using auxiliary port	Communication may be full-duplex using auxiliary port
Runtime control	Supports runtime-control features using IEEE-1149.1 interface	Supports runtime-control features using 1149.1 interface or auxiliary port	Supports runtime-control features using 1149.1 interface or auxiliary port	Supports runtime-control features using 1149.1 interface or auxiliary port
Auxiliary port	None	Allows port sharing; the auxiliary port may share slow I/O port pins	Allows port sharing with high-speed I/O ports	Allows port sharing with high-speed I/O ports
Data acquisition	None	None	Supports data acquisition	Supports data acquisition
Memory substitution	None	None	None	Supports memory substitution (fetching or reading data over the IEEE-ISTO 5001-1999 auxiliary port); optional memory substitution triggering on a watchpoint

To interface with the OCD hardware, 5001-99-compliant chips include the NRR (Nexus-recommended-register) set, which supports as many as 127 OCD registers to provide control and status information for as many as 32 embedded-processor cores. Physically, the specification includes the 1149.1 interface and provides for an optional parallel interface connector to accelerate I/O speed. Class-1 devices include the minimum 5001-99 features and require “connector A,” a dual-row, 20-pin AMP System-50 header that carries the four-wire 1149.1 interface; additional reset and voltage-reference pins provide a hardware reset and establish the debug interface’s signaling levels. Optionally, connector A includes the following signals to improve flexibility:

- CLOCKOUT, the target processor’s system clock;
- EVTI (event-in input), to halt the processor or to transmit program- and data-synchronisation messages from the processor to the debug tool;
- EVTO (event-out output), to provide precise breakpoint timing information for the debug tool;
- RDY, a handshake signal to speed information transfers;
- TRST, the 1149.1 optional test-reset input; and
- VENDOR, a device-specific I/O signal.

The specification also defines “connector B,” a dual-row, 30-pin header that carries a superset of connector A’s signals comprising two MDI (message-data-input) and four MDO (message-data-output) pins. Debugging tools can use connector B in the 1149.1-only mode, an auxiliary-port mode, or in combination. Connector B’s typical use is for static debugging via the 1149.1 port with the auxiliary port providing runtime trace support. The maximum-capability “connector C” dispenses with the 1149.1 interface to support applications that require a wider data pathway. The main additional pin function is an optional 16-bit I/O port that can employ port replacement, meaning that you can remap low- or high-speed I/O pins for the debugging tool’s use during debugging sessions. This pin-multiplexing strategy reduces the number of device-package pins necessary to accommodate complex I/O and debugging functions. The recommended connector-C hardware is a four-

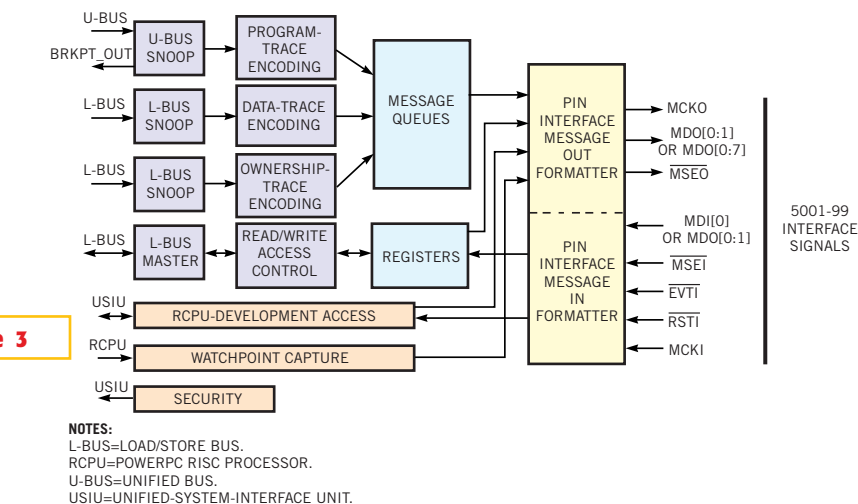


Figure 3

Motorola’s MPC565 is the first microcontroller to include a 5001-99 Class-3 interface.

row, 80-pin Samtec header on a 0.05-in. matrix.

The 5001-99 hardware interface provides debug control via the standard’s API. Implemented as a standard set of header files, the API is a two-layer model that comprises a target abstraction layer and an emulator-hardware abstraction layer. The chip vendor provides the target-abstraction-layer information that implements the Nexus debug semantics using the device under test’s OCD system. Target-to-tool communication occurs via the emulator’s hardware-abstraction layer, which also abstracts communication between external hosts (such as a PC) and the emulator hardware. Software tools build on the target abstraction layer, typically using vendor-specific APIs to the tool set’s application layer. At the lowest level, the Nexus API provides a uniform interface to debug tasks such as entering and leaving debug mode, reading and writing registers and memory, halting on breakpoints, and single-stepping instructions. The API includes powerful features, such as the ability to read and write memory and to monitor program flow and data reads while the processor runs in real time, letting you change calibration constants while the program runs. For applications such as event tracking, a “watch-point” function sends a message to the debugging tool rather than halting execution as for a breakpoint. The 5001-99 API also provides a program-ownership mechanism that’s essential for tracking program flow within an RTOS environment.

Infineon’s Lies observes that the company’s first-generation TC1775’s OCDS implementation models 5001-99’s Class-2 requirements. Future TriCore OCDS implementations will meet higher level 5001-99 requirements, so that standard production parts will offer a level of access to data tracing equal to that of dedicated bond-out processors. And last December, Motorola announced the industry’s first truly 5001-99-compliant device with its MPC565 PowerPC embedded processor (Figure 3). The MPC565 includes a Class-3 debug port that tools from Ashling Microsystems support. (Another Nexus member, Ashling also developed the first tools to perform real-time tracing for Infineon’s TriCore architecture.) The preferred connector that Motorola includes on its MPC565 development board is a 40-pin derivative of the 80-pin connector C that supports eight message-data outputs and two message-data inputs. Although not yet recognised as a 5001-approved standard, this connector configuration allows the standard 40-MHz MPC565 to output data at maximum speeds of 40 Mbytes/sec and input messages at 5 Mbytes/sec; a 56-MHz version of the processor is also available. □

You can reach Contributing Editor David Marsh at forncett@compuserve.com.

REFERENCE

1. Marsh, David, “Simple boundary-scan techniques tackle sophisticated systems,” *EDN Europe*, July 2001, pg 34.