WHITE PAPER

Serial Wire Debug—Ideal for Microcontrollers

Jean Anne Booth
Director of WW Stellaris Marketing and Customer-Facing Engineering

Sue Cozart
Applications Engineer

Introduction

Serial-wire brings robust debugging capability to low-cost micorocontrollers

Digital electronic equipment is becoming more sophisticated every day, so the programs running these processor-driven systems are becoming bigger and more intricate. Debugging these complex programs requires very capable facilities so the designer can track down problems and perfect the software.

The ARM® Cortex-M3™ processor core brought the performance and popularity of the ARM architecture to a compact size with deterministic real-time features, enabling Texas Instruments to build 32-bit Stellaris® microcontrollers (MCUs) that offer high performance and functionality at price points that are normally associated with much slower 8-bit MCUs. Serial Wire Debug (SWD) similarly brings fully capable debug and trace facilities to these MCUs while keeping chip and tool costs low, yet leaving the greatest number of pins available for system I/O.

Debugging on Stellaris Microcontrollers

The greater resources of 32-bit microcontrollers and the wide user base of the ARM architecture make code development easier for programmers in comparison to 8-bit and 16-bit offerings. High-level languages like C compile quickly and more compactly into 32-bit RISC code, limiting the errors of assembly language coding that are common with smaller MCUs. Debugging is still going to be necessary and Stellaris MCUs offer rich capabilities for locating problem code so it can be corrected. With the Cortex-M3 processor core, monitoring and debugging can take place in real time, without having to stop or slow the processor so internal resources can be interrogated or external instruments can keep up.

Stellaris MCUs have all of the debug and trace capabilities a high-end microcontroller needs. Program flow can be monitored with a rich set of hardware execution breakpoints and sophisticated watchpoints, vector catching, and meta trace facilities even with all interrupts enabled. Data in registers and memory can be read and written while the processor is running, and resource use can be verified. Code profiling can determine where the processor is spending its time, revealing when it is hung in an endless loop or where best to devote efforts to quicken program execution. Sections of code that never execute are exposed, freeing up valuable space for needed features. Programmers can verify that pulse-width modulators (PWMs) are firing as needed, stacks and heaps can be inspected for buried problems, exception returns can be confirmed, and status, condition codes, and branches can be validated. Breakpoints can be set to trigger on data values, address locations, and conditions on the chip. Detailed trace capabilities of the Stellaris microcontrollers are listed in Table 1. The MCU can be made to halt, single-step, and run with controls over the SWD connection as the operator desires. As needed, memory and on-chip registers can be written by the debugging tool.

Table 1. Serial Wire Trace Capabilities

Feature	Use
Time-stamped program counter	Supports profiling and coverage
Data reads and writes	Interject desired data
Peripheral register values	Monitor operation
Event counters	Evaluate clocks per instruction
Subroutine, interrupt and exception calls, and returns	Complete live change-of-flow
Timestamps and CPU cycle counts	Program execution profiling
Messages from OS/user code	Printf documentation of code flow

CoreSight—The Basis of Debugging

CoreSight is the comprehensive debugging and trace architecture which was developed by ARM in 2003. Some ARM9s and ARM11s use an early version of CoreSight, but the more recent popularity of the Cortex processors has really proven its advantages. CoreSight is made up of a number of components that each aid an external host's ability to see into the live operation of the chip in a highly efficient manner.

The components included in a given chip and the sophistication of the specific development tools in use determine the ultimate capabilities of real-time monitoring, control, and debugging of running programs. Most system designers make use of CoreSight through a development tool rather than by directly setting registers, following the protocol, and manipulating low-level signals, but a quick understanding of the architecture is instructive in highlighting the effectiveness of the design. Figure 1 illustrates the components used in Stellaris microcontrollers with their functionality described below.

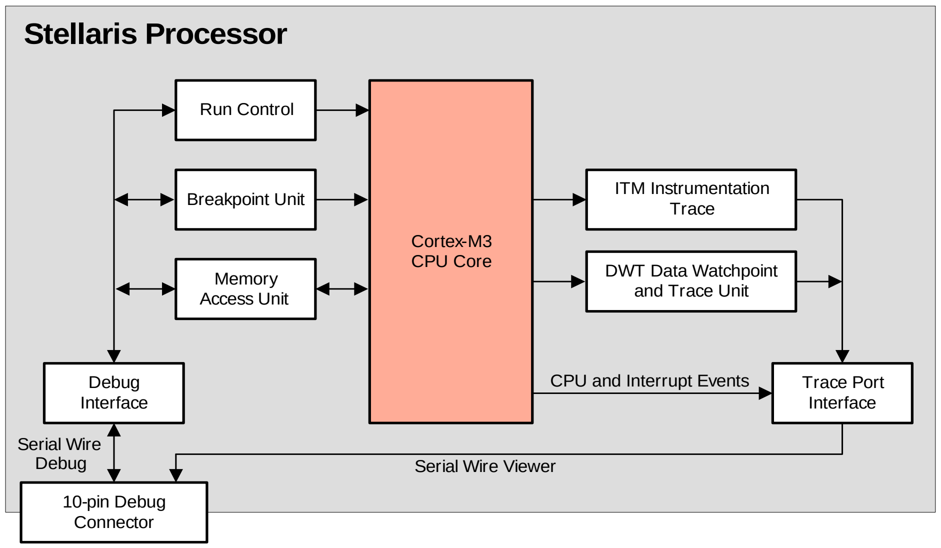

Figure 1. *Debug and Trace Components in Stellaris Cortex-M3 Microcontrollers*

The **Data Watchpoint and Trace** (DWT) macrocell keeps an eye on data accesses by the Cortex-M3 with four watchpoint comparators that can trigger an event at the debug host when a specified address or data value is put on an internal bus. The event can break the core flow or just be logged for transmission to the host. The program counter can be captured and saved. Selecting the action to be taken for a given event can be determined and programmed by the debug host. One very useful function of the DWT is to profile a program to determine where the processor is spending most of its time, possibly hung in an endless loop, and to monitor key data as it is being changed.

The **Instrumentation Trace Macrocell** (ITM) in the Stellaris MCU provides the fundamental trace information to an external monitor. Software running in the ARM core can use a single-cycle instruction to expressly send data to the ITM, which then transmits the data to the host tool. These *printf* -style messages can be used to communicate operating system, self-diagnostic, or application-specific information to the tool for monitoring progress and status of program execution.

It is very useful to have this back channel on which to send such console messages so that a traditional MCU peripheral like a UART or I/O port is not tied up with debugging data when it should be available for use by the running application. The ITM actually forwards trace and event data from the DWT to the external tools and can insert timestamp data unobtrusively that lets the tool determine the time between various events.

A **Flash Patch and Breakpoint** unit (FPB) can trigger a breakpoint when a matching address is hit, at which point the instruction or literal value can be replaced. Six instruction values and two literals can be set. For patching, a successful match can remap the address to a section of RAM, allowing different instructions or data to be inserted into the running program, even if the original is in

ROM or Flash memory. For breakpoints, it allows loading a breakpoint instruction into the upper and/or lower half word.

A key component of CoreSight is the Serial Wire Debug. This optimized two-wire physical interface utilizes a packet-based protocol to connect the chip-bound components with external debugging tools, offering complete debugging and trace capabilities to the programmer. And yet, SWD requires minimal silicon real estate and just two pins on the chip, consistent with a very low-cost microcontroller. Its advantages for Stellaris are best seen in light of alternatives that might have been used and an awareness of the needs for a successful 32-bit microcontroller.

Serial Wire Debug—Key to Cortex-M3 Debugging

ARM carefully considered the optimizations it made for the Cortex-M family of processor cores – cores destined for the smallest of processing applications. From the highly-efficient Thumb2 instructions to the debug and trace mechanisms available, these cores have to be very small in physical size and very thrifty in power consumption, delivering the best performance within those constraints. As the core is the heart of a microcontroller, the best in size and power is necessary. The core's small size is one of the reasons that Stellaris MCUs can replace 8-bit MCUs at equivalent prices.

Some features of the new CoreSight debug architecture are critical, but others require a lot of circuitry while providing little additional benefit for the smaller, more-contained chip implementations of the Cortex-M series. While exotic debugging facilities may sound attractive, cost is always a driving concern for MCU selection, so all of the features within the chip must be optimized, including debug support.

Microcontrollers Must Be Cost-Effective

Whether performing a dedicated function in a larger system or managing the entire application, small size and low cost are key decision-factors in a microcontroller. A careful balance must be made for every function included in an MCU.

The peripherals included on any one microcontroller are chosen according to the applications for which the MCU is expected to be used. If there are too many unused peripherals on a specific device, then another MCU is chosen to save chip, and therefore system, cost. This economic reality is reflected by the large array of MCUs offered by suppliers; the greater the variety available, the better the chance that just the right combination of performance, functions, and peripherals can be found for any system design. Because MCUs go into some very high-volume applications, every penny counts.

The cost of a semiconductor is highly influenced by the number of transistors used to implement the circuit. However, there is not purely a linear relationship of transistors to cost because there are interconnects and distances that complicate the issue. Adding a few more transistors may add nothing to the cost if they fit in just the right places, but doubling the transistors can more than double the silicon required in many circumstances.

More transistors and more complex circuitry affect more than just silicon area. Every transistor consumes power both in a quiescent state as well as when switching. Power domains can be switched off to conserve power, but when they are operating, transistors are going to draw power and generate heat. Both are undesirable and a good reason to minimize the transistor- and gate-count of MCUs.

Signal pins (or leads) are precious on a microcontroller. MCUs are packed with peripheral circuits specifically to sense the outside world through switches, analog signals, sensors, and communications channels and to drive outputs to motors, displays, networks, and ports. A number of signal lines are needed to support each peripheral and the more pins that are available, the more functions can be supported. Any function that can be served with fewer pins leaves room for other functions to be included within the same package footprint.

It should also be noted that debug facilities would typically be used rarely over the life of a piece of equipment. Once the design is final and the code is approved, there is usually no further need of the debug circuitry. Any transistors, signals, silicon space, or pins associated with debugging are likely to never be used again; yet a million chips shipped in production equipment still contain those resources, paid for, but idle. Keeping the circuits and pins to a minimum is clearly the best approach. A single pin used as a general-purpose I/O line is far more valuable in the end equipment than that pin being used for debugging during system design.

The package size of a microcontroller is also a key element of the chip. As systems become more portable and personal, size and shape are key convenience factors and selling points for the end equipment where small is usually better. Each signal pin on the package requires more space on the circuit board and in the equipment, and small physical size is a very important feature on many of today’s most attractive devices. Some 32-bit MCUs are as small as 28 pins, and devoting 15% of those pins to any single purpose, such as debugging, had better be justifiable. Multiplexing is often used to maximize pin utility, but that has downsides, too.

JTAG, the Old Standby Debug Interface

For debug communications, JTAG has evolved as a popular hardware interface between MCUs and external debugging tools. The origins of JTAG, however, have little to do with processor debuggers. JTAG (which stands for Joint Test Action Group) was established primarily to permit easy verification of the integrity of signals between multiple chips in electronic systems. JTAG helps narrow down signal flaws caused by shorts, opens, and cross-talk that can be difficult to detect with the small dimensions, multi-level pathways, hidden solder joints, environmental conditions, and numerous sources of noise found on modern circuit boards.

JTAG offers a channel into the back alleys of a chip, but is not the perfect transport mechanism for debugging. JTAG makes use of four pins (five when a reset is utilized) to connect to boundary scan, test, and debugging components on the chip. JTAG data transmission rates are slow by today’s frequency standards.

Serial Wire—Four Pins Good, Two Pins Better

Ultimately, there are really only two signals needed for a serial port to be used for debugging: the clock and the data signal (see Table 2). The more streamlined Serial Wire Debug technology implemented on the Stellaris MCU recommissions the TCK and TMS pins of JTAG, and uses a packet-based protocol that provides data rates more than double that which JTAG can offer in spite of the reduced pin count. The daisy-chain topology of JTAG is not used with SWD, but a multi-drop scheme allows multiple cores to still be accessed and coordinated by the host. While not as simple as daisy-chaining or the singular point-to-point connections, multi-drop allows individual cores to be completely disabled, as might be needed to conserve power, without impacting the remaining, still operating cores. Turn off a link in a daisy-chain and nothing downstream gets the message.

JTAG is still available on Stellaris MCUs for its boundary-scan test facility, but by using SWD, the two JTAG pins that are no longer needed are available for traditional microcontroller I/O (input/output) use. These are two more I/O lines that are not inhibited by the system running in a debug mode, perhaps aiding the tracking down of errant code. The less obtrusive the debug mode is to the target code, the more effective the problem-solving capability.

Real-time trace capability in Stellaris MCUs is made available using a third serial wire on Cortex-M3, the Serial Wire Output (SWO). The Instrumentation Trace Macrocell sends its data to the host over this line which often overlays the JTAG serial out pin (but that still totals two fewer pins than the five-pin JTAG port.) With the console messages being coordinated at the host with the program code, very effective trace debugging can take place.

Table 2. Signal Comparison between JTAG and Serial Wire Debug (SWD)

JTAG 1149.1		Serial Wire Debug	
Pin	**Purpose**	**Pin**	**Purpose**
TCK	Clock	SWCLK	Clock
TMS	State Machine Control	SWDIO	Data In & Out
TDI	Data In	—	—
TDO	Data Out	SWO[a]	—
TRST	Reset	—	—

a. SWO, a serial wire output, is used to aid trace.

ETM—Diminishing Returns

One additional trace component available for Cortex-M3 was not implemented in the Stellaris microcontroller family. The Embedded Trace Macrocell (ETM) is designed to provide explicit instruction-by-instruction trace information to the development tool.

This capability is especially useful in more complex microprocessors that make use of on-chip resources that are too dynamic for real-time determinism from the outside. For example, the variability of on-chip caches which are dependent upon the specific dynamics of the execution of a program loop with constantly changing data and inputs is impractical to replicate in an external tool at a reasonable price. However, MCUs do not utilize caches and so have little to gain from an ETM.

The original ETM put trace data out on 4-, 8-, and even 16-bit buses. The ETM designed for the Cortex-M3 holds this down to just 4 bits. But the ETM comes at a price in two areas: transistors and pins. The ETM requires more gates to implement and needs four pins to transfer information off the chip with fast drive pins and at high transfer rates. These are not huge numbers, but everything in an MCU is a tradeoff, and if saving two pins in the SWD interface is considered very positive, then using twice as many—four—to get generally-unneeded trace data off the chip would be very negative. Also, as noted elsewhere, multiplexing pins for debug signals makes debugging more intrusive to the main application program and the silicon real estate is being paid for with every production chip purchase. For the Stellaris microcontroller product line, the ETM was judged as more bulky than its function can justify for 99% of the applications, so it is not implemented. Many years of use in designers' hands has confirmed that to be a good decision.

The development tools which can make use of the ETM information are higher-end, more expensive tools, some described by one tool vendor as costing "more than a sane person can afford."[1] (Skrtic, Mike. IAR Systems.) In addition, the time period covered by the ETM information is very short.

In contrast, the ITM can provide significant trace information over the SWD and SWO lines with very modest, but very useful, development tools. A quick comparison of possible debug and trace facilities for the Stellaris microcontrollers is shown in Table 3.

Table 3. Stellaris MCUs Utilize the Most Efficient Debug and Trace Facilities

Component	ITM	ETM	JTAG	SWD	SWO
Features	*printf* msg	instr trace	test/ debug	debug	msg
Transfer Speed[a]	—	—	640 KBps	1,500 KBps	—
Pins Needed	1 (SWO)	4	5	2[b]	1[b]
Stellaris Feature	yes	no	yes	yes	yes

a. At 20 MHz

b. Shared with JTAG pins

1. Skrtic, Mike. *A Giant Leap or a Small Step? Making the Move from 8-bit to 32-bit.* IAR Systems. http://www.iar.com/website1/1.0.1.0/485/1/

Stellaris—Effective, Low-Cost Debug and Trace

The debug and trace facilities found in the Stellaris microcontrollers are true to the intent of the microcontrollers themselves: high performance at low cost. Serial Wire Debug is especially helpful in freeing up pins, increasing throughput, and enabling low-cost tools and chips. Debug components were eliminated that provide little additional value but consume more precious space and pins than can be justified for a compact MCU destined for price-sensitive applications. The result is an advanced 32-bit microcontroller family that has real-time debugging and trace support for the processor, memory, and peripherals that is optimal for the intricate application code that runs on the powerful chips, without requiring expensive development tools or driving up the cost of the production chips.

Texas Instruments • 108 Wild Basin, Suite 350 • Austin, TX 78746
http://www.ti.com/stellaris