Serial Wire Viewer (SWV) for ARM Cortex-M3 Processors
ARM Real-Time Trace aids debugging.

Version 2.1

By Robert Boys robert.boys@arm.com
ARM, Ltd. Sunnyvale, California

Introduction

Two objectives of ARM are to provide developers with advanced and powerful microcontrollers with appropriate and useful debugging facilities. These objectives go hand-in-hand to provide the tools to give developers the ability to debug and test their projects as effectively as possible.

ARM has expanded the EmbeddedICE™ debugging modules found in ARM7™, ARM9™ and ARM11™ cores with new CoreSight™ on-chip debug and trace technology. CoreSight is used in Cortex®, ARM9 and ARM11 processors but is most common in Cortex processors, especially the Cortex-M3.

In this article we will look at how over the years debugging power increased, then decreased and how ARM has brought it back. Our focus will be on the Serial Wire Viewer (SWV) component of CoreSight as found in Cortex™-M3 microprocessors. This is also called Real-Time Trace: SWV. We will discuss how it works, why it is useful and how and where it is used.

The Good Old Days

Using full in-circuit emulators (ICEs), MCU debugging power was at its peak culminating during the late 1990s. These powerful machines offered stop, go and single-step debugging plus trace memories with complex triggers, filters and many, often unlimited, hardware breakpoints.

With the relatively slow speed of the 8051 and other similar devices, in-circuit emulators used a mixture of production chips and special, costly bond-out chips. Bond-out chips required the formation of a dedicated chip design team by the semiconductor manufacturer. This team obviously would be better utilized in developing parts that could actually be sold to customers.

Debugging via the JTAG port

As chips with higher pin counts, more efficient bus cycles, more single chip applications and especially higher speeds began appearing in the market, ICEs were unable to keep up.

Manufacturers, including ARM, started putting debug modules inside their production chips. These were accessed through the JTAG interface which was already present on many chips and used for board integrity testing. Today, JTAG is present on most MCUs. Figure 1 shows the Keil™ ULINK™-ME USB-JTAG adapter on the MCBSTM32™ evaluation board.

These debug modules provided basic stop, go and single-step debugging with hardware breakpoints. Unfortunately, most of the advanced ICE debugging power associated with trace memory was lost when outside access was cut off from the internal address and data busses. This, plus the extreme speeds of some of these devices severely curtailed the use of full ICEs. Developers' debugging power was set back at least ten years to almost the early days of debug monitors.

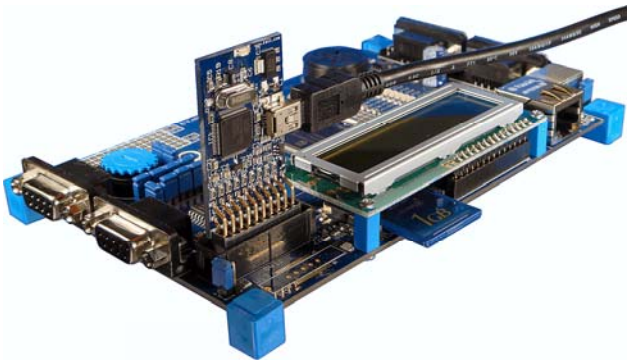

Figure 1: Keil USB-JTAG adapter on STM32 board

In order to bring internal debugging data to the outside world, solutions such as simulators and debugging code were promoted. These function quite well, but are somewhat intrusive to CPU time and code space. They usually consume a UART port. Simulators are quite effective but will eventually need "hardware in the loop" products to interface to customer hardware. With more sophisticated real-time systems, it is not always possible to stop the CPU or steal cycles to examine internal states.

Developers had to use very ingenious methods to debug their projects successfully and this caused significant time delays, inefficiencies and missed bugs. There was a real need to get data about the MCU out of the chip and in real-time to the developer at an affordable price. This data included program counter values for change of program flow status, data reads and writes plus internal register values. ARM provides excellent solutions to these problems.

Serial Wire Viewer (SWV) for ARM Cortex-M3 Processors
ARM Real-Time Trace aids debugging.

Version 2.1

Real-Time Trace ETM™

In the late 1990s ARM developed ETM (Embedded Trace Macrocell) to provide crucial trace information. This data is output on a 4, 8 or 16 bit dedicated trace bus and is captured by a special trace debugger tool. This external tool needs very fast hardware to capture all the data at the speeds ARM™ cores are capable of running.

While ETM is effective, there is a need for a lower cost method of gathering trace information without the need for this expensive hardware. Not all ARM devices have the optional ETM implemented.

Recent Cortex-M3 processors from NXP, Toshiba and STMicroelectronics provide a 4 bit ETM port as well as the Serial Wire Viewer. In general, for these Cortex-M3 devices, ETM is used to support PC information while SWV is used to provide read and write data as well as exception events.

CoreSight Real-Time Trace SWV

ARM, for its Cortex family, uses the new CoreSight set of debugging modules. CoreSight includes all debugging modules such as JTAG debugging, ETM and most significantly adds Serial Wire Debug (SW-DB) and Serial Wire Viewer (SWV). Not all Cortex MCUs will have ETM, but nearly all will have both JTAG and SW-DB debugging technologies. Cortex – M3 processors with SWV are currently available from Luminary, STMicroelectronics, NXP and Toshiba. More devices are planned from various manufacturers.

Serial Wire Debug (SW-DB) offers the same debugging capabilities as JTAG but with fewer pins. SW-DB uses only two wires while JTAG uses 4 to 7 to connect to the CoreSight debugging modules. A smaller connector can therefore be used with SW-DB. SW-DB and JTAG offers stop, go and single-step debugging, up to 8 hardware breakpoints and up to 4 watchpoints. The Keil ULINK2 USB-JTAG adapter works with both these debugging methods.

Serial Wire Viewer (SWV) is similar to a one bit ETM port. It needs only one pin and this pin on the MCU is called Serial Wire Output (SWO). These three pins (SW-DB and SWO) share the same connector as the JTAG port.

Figure 2 illustrates the various components of a typical Cortex processor.

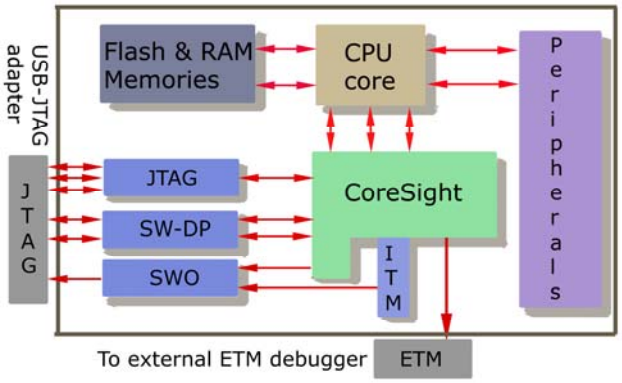

Figure 2: Cortex Debugging Diagram

What Serial Wire Viewer can do for you

The Serial Wire Viewer provides these types of information:

1) Program counter values. (sampled)
2) Data read and write cycles.
3) Variable and peripheral values.
4) Event counters.
5) Exception entry and return.
5) Timestamps and CPU cycles used.

No debugging code in the user program or cycle stealing are needed. User code can also send data out the SWO with the ITM using a one cycle write instruction. ITM is described later.

The main difference between SWV and ETM is that ETM has a higher throughput of data. SWV can provide considerable trace information with no investment in expensive hardware tools.

The RealView® MDK-ARM® (Microcontroller Development Kit) provides SWV support for no additional charge. MDK uses µVision®3 from Keil as its IDE (Integrated Development Environment. See www.keil.com for more information. Keil provides ETM support using a tight integration with the Signum Systems JTAGJetTrace adapter and the Segger J-Trace.

Serial Wire Viewer (SWV) for ARM Cortex-M3 Processors
ARM Real-Time Trace aids debugging.

Version 2.1

Variables

µVision can display variables in several familiar formats including graphs in real time without any intrusiveness. Variables are transmitted out the SWO line by the SWV macrocell which requires no CPU cycles or resources other than the three serial pins described above. Global and static variables plus structures are easily displayed. Local variables must first be converted to global or static before they can be viewed. Figure 3 shows the output from the A/D converter global variable AD_DbgVal and the Systick handler in real-time.

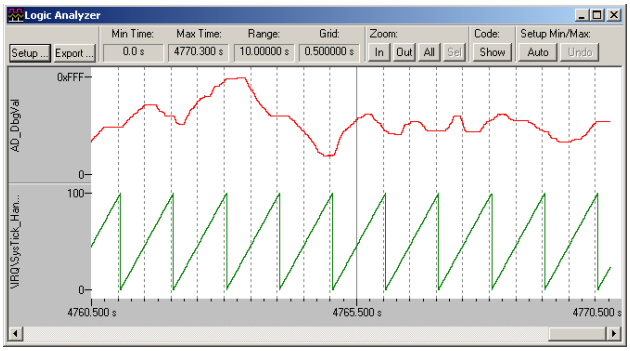

Figure 3: Keil µVision Logic Analyzer

Peripheral Registers

It is possible to extract the values of peripheral registers in real-time. Whenever a read or write is performed on a peripheral address by a global or static variable, a trace event is created and can be displayed in a graph as in Figure 3 or in the Watch and Memory windows.

Program Counter (PC) Sample

It is possible to display program counter values. This is useful for program flow change, profile analysis and determining where the CPU might be caught in an infinite loop and which instruction sent it there. Profile Analysis gives an indication where the CPU is spending its time. Clearly, the single serial wire port is not capable of providing every program counter value because of the high speed ARM cores can run at. A statistical sampling indication of performance analysis is possible and is sufficiently adequate for profile analysis. Figure 4 shows the value of the PC as well as timestamps in CPU in both cycles and seconds.

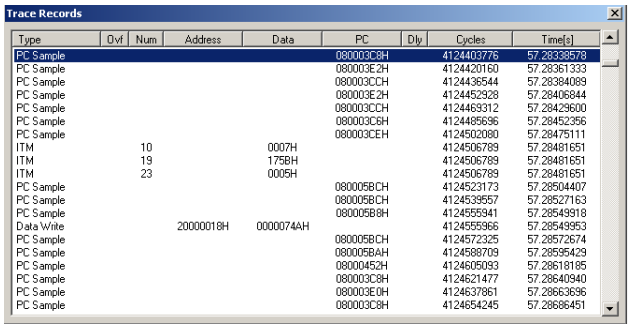

Figure 4: PC Samples & ITM Trace Records

Data Read and Write Frames

Read and write data frames can be displayed giving the address of the responsible instruction, the data value transferred, the data address and timestamps in both core cycles and seconds. Figure 5 shows a series of data writes showing these attributes. Data reads are similarly displayed.

Note the small filter window showing the types of events that can be displayed.

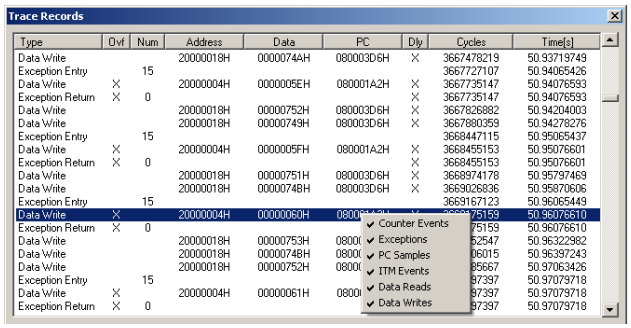

Figure 5: Data Writes & Exception examples

ITM: Instrumentation Trace Macrocell

The ITM is useful for user programs to write data out the SWO port. This data can be displayed on either the Trace Records or Serial Wire Viewer windows. Sample user code to write to the ITM is shown below. This sometimes referred to as “printf debugging”. This method is marginally intrusive to the user program.

Sample code to use ITM follows: Essentially you merely write to the specific ITM address and CoreSight automatically sends your data out the SWO port and uVision displays it in the ITM window shown in Figure 6 and normally uses 1 CPU write cycle.

Define a pointer:

```
#define ITM_Port32(n)   (*((volatile unsigned long *)(0xE0000000+4*n)))
```

Write to the ITM port:

```
int SendChar (int ch)  {/* Write to SWO*/
  while (ITM_Port32(0) == 0);
  ITM_Port8(0) = ch;
  return (ch);
}
```

Three ITM records are displayed in Figure 4. Figure 6 shows the results of the user code displayed in the Serial Wire Viewer window.

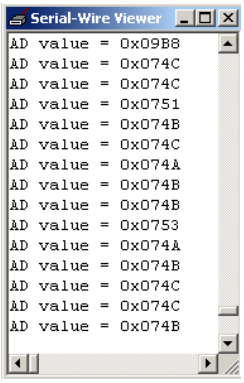

Figure 6: Serial Wire Viewer Window

Trace Event Counters

These are the types of events traced:

1) # of CPU clock cycles (32 bit).
2) # of folded instructions count (8 bits).
3) # of instruction cycles (8 bit).
4) # of cycles processor is sleeping (8 bit).
5) Total cycles spent in interrupts (8 bit).
6) Total cycles spent in load/store operations.

Each time one of these counters rolls over, an event is generated and fed to the Event Counters window. These counters can be used to measure total elapsed time, provide information on system performance and many other uses.

Figure 7 shows the number of times these counters have rolled over. They can be cleared by clicking on the "0".

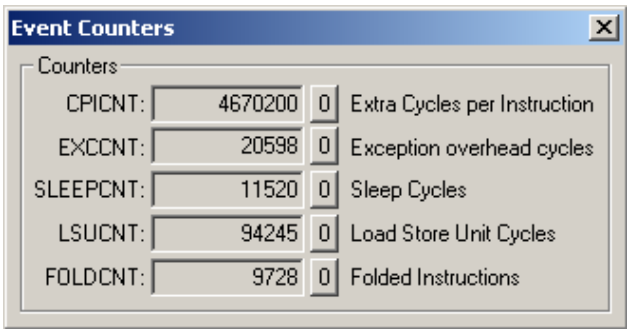

Figure 7: Event Counters

Exception Trace Dialog:

SWV will capture various exceptions' return and exit. These are timestamped and the exception number is displayed. Figure 8 shows the Exception Trace window where the exception SysTick (15) has occurred 921 times. Information about exceptions will be displayed in this window and also repeated in the Trace Records window as shown in Figure 5 where exceptions 0 and 15 are displayed.

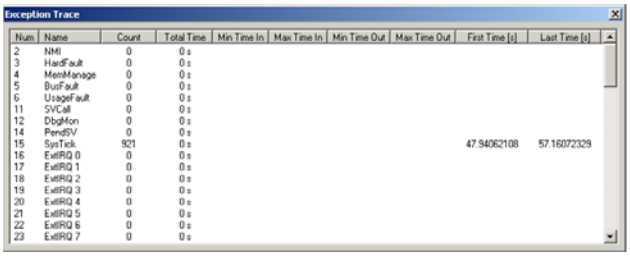

Figure 8: Exception Trace Window

Usefulness of the Trace

Trace, either ETM or SWV, adds significant power to debugging efforts. Problems that may take hours, days or even weeks in big projects can often be found in a fraction of these times with a trace. Especially useful is where the bug occurs a long time before the consequences are seen or where the state of the system disappears with a change in scope. Usually, using trace helps find bugs without stopping the program.

In the more complicated environments of today, it is becoming more difficult to find bugs easily and in a timely fashion. Tracing can help find nastier bugs easier than traditional stop and go debugging.

Serial Wire Viewer (SWV) for ARM Cortex-M3 Processors
ARM Real-Time Trace aids debugging.

Version 2.1

These are the types of problems that can be found with a quality trace:

1) Pointer problems.
2) Illegal instructions and data aborts (such as misaligned writes).
3) Code overwrites – writes to Flash, unexpected writes to peripheral registers (SFRs), corrupted stack.
4) Out of bounds data. Uninitialized variables and arrays.
5) Stack overflows. What causes the stack to grow bigger than it should ?
6) Runaway programs: your program has gone off into the weeds and you need to know what instruction caused this.
7) Communication protocol and timing issues. System timing problems.
8) Where is the program spending all its time ? A Profile Analyzer will tell you.

In addition, interesting items such as variable contents, status of the RTOS such as task number, semaphores and other information can be viewed in real time to send information to the developer.

Information about the state of the KEIL RTX RTOS running on Cortex-M3 processors is shown in real-time using Serial Wire Viewer. It is not necessary to stop the processor to view this type of data. The displayed information changes "on-the-fly". See Figure 9 below.

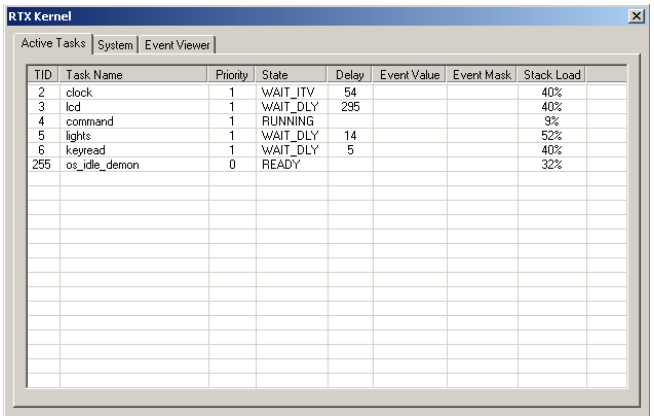

Figure 9: RTX Viewer in Real-Time

How can I try this out myself ?

All you need is an evaluation board with a STM32, NXP, Toshiba or Luminary processor, a Keil ULINK2 (or ME) and a copy of RealView MDK. MDK has several examples included and all will compile with the evaluation version of the tools. See www.keil.com to obtain the software.

If you have another Cortex-M3 processor not listed here that has Serial Wire Viewer technology implemented, you can easily view it working with Keil MDK. Please contact Keil technical support if you experience any issues.

Lab exercises for the ST and Luminary processors are also available on the Keil website.

Conclusion

The Serial Wire Viewer provides a low cost method of obtaining information from inside the MCU at low cost due to ARM CoreSight technology and Keil's µVision3.

For heavy duty trace debugging such as looking for nasty bugs, the ETM will still be used but to find occasional problems using trace debugging, the Serial Wire Viewer is hard to beat for both performance, ease of use and cost. As well as being readily available.