ARM®

Student's Guide To Building a Low-cost Development Environment

Application Note 167

Released on: November 2006

Student's Guide To Building a Low-cost Development Environment
Application Note 167

Release Information

Table 1 lists the changes to this document.

Table 1 Change history

Date	Issue	Confidentiality	Change
August 2006	A	Non-Confidential	Initial issue
November 2006	B	Non-Confidential	Updated for compatibility with InfoCenter

Proprietary Notice

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

Product Status

The information in this document is final, that is for a developed product.

Web Address

`http://www.arm.com`

1 Introduction

This application note describes how to debug and run programs on an ARM-based development board in the Keil uVision3 IDE. Instead of the more traditional JTAG protocol converter such as Multi-ICE, this note focuses on using the H-JTAG debug agent with a JTAG programming dongle.

This application note contains the following sections:

1. *Obtaining the Necessary Materials* on page 4 describes the required materials and where to get them. This includes ARM-based development boards, the Keil uVision3 IDE, a JTAG programming dongle, and the H-JTAG debug agent.
2. *Connecting and Configuring the Hardware* on page 7 explains how to connect and configure the materials.
3. *Using the Keil uVision3 IDE* on page 12 describes how to launch and configure the H-JTAG debug agent for the Keil uVision3 debugger. It provides an overview of how to configure the Keil debugger and download code to a target board. The ARM Evaluator-7T board and the Keil MCB2100 board are used as examples.

2 Obtaining the Necessary Materials

This section details all of the materials necessary to download code to an ARM-based development board for debug in the Keil uVision3 IDE using the H-JTAG debug agent.

2.1 ARM-based Development Board

There are many ARM development boards available from suppliers. In theory, any ARM-based development board with a JTAG header can be debugged with the Keil uVision3 IDE using the method described in this application note.

If you do not already have an ARM development board, these links point to ARM development board resources:

```
http://www.keil.com/arm/
http://www.olimex.com
http://www.bluewatersys.com/
http://www.hitex.co.uk/
http://www.ashling.com/
http://www.spjsystems.com/
http://www.lpctools.com/
http://www.phytec.com/
http://www.nohau.com/
http://www.fudantech.com/
http://www.simtec.co.uk/
http://www.unicoi.com/
http://www.armkits.com/
http://store.earthlcd.com/
http://www.smxinfo.com/
http://www.embeddeddirect.nl/
http://www.cirrus.com/
http://microcontrollershop.com/
http://www.revely.com/
http://www.newmicros.com/
```

This application note describes how to use H-JTAG with the Keil uVision3 IDE.

It uses the ARM Evaluator-7T board and the Keil MCB2100 board, with code specific for each board as examples. You can use your own development board and code if you choose.

The ARM Evaluator-7T board and the Keil MCB2100 board are ARM platforms that enable you to download and debug software and attach additional I/O peripherals. The Samsung KS32C50100 microcontroller, built around the ARM7TDMI RISC processor core, drives the Evaluator-7T board. A Philips LPC2129 microcontroller, built around the ARM7TDMI-S RISC processor core, drives the Keil MCB2100 board.

For help with setup, specific hardware specifications, and programmer references. see the following resources:

- ARM Evaluator-7T Board User Guide
- Samsung KS32C50100 Chip User Manual
- Keil MCB2100 Board User Guide
- Philips LPC2129 Chip User Manual.

2.2 PC Running Keil uVision3 IDE

You can download the Keil RealView Microcontroller Development Kit Evaluation software from `http://www.keil.com/`. It contains the Keil uVision3 IDE. The evaluation version of the tools has a 16K limit on images, but comes in a license-free version.

Note

You can use either the full version or the evaluation version with this application note.

Because these Microcontroller Development Kit tools include full simulation models for peripherals around various microcontrollers in addition to compilers and debuggers, in many cases there is no requirement for hardware.

This note describes how to debug and download code to an actual development board. Simulation is not described in this application note.

2.3 H-JTAG Debug Agent

H-JTAG is a free JTAG debug agent for ARM microcontrollers. It supports various wiggler-compatible and SDT-compatible JTAG dongles, including some compatible user-defined interface dongles. ARM7 and ARM9 devices are debugged with the H-JTAG agent. H-JTAG requires Windows 9.X/NT/2000/XP. A download for installation is available from `http://hjtag.blogspot.com/`.

2.4 JTAG Programming Dongle

The H-JTAG debug agent requires a wiggler-compatible JTAG dongle, an SDT-JTAG dongle, or a compatible user-defined interface dongle. You can purchase a low-cost JTAG wiggler-compatible dongle from Olimex at: `http://www.olimex.com/`. The cost at the time of this writing is USD $19.95 + shipping.

The examples in this application note use the Olimex dongle. Although you can use any compatible JTAG dongle, the configurations can vary from those described for the Olimex dongle.

Note

When this Olimex dongle is paired with H-JTAG, it can only support ARM7 and ARM9 devices.

2.5 Parallel Cable Extension

ARM Ltd. recommends that you obtain a parallel cable extension because the ARM-JTAG dongle is only 8" in length. An example of a suitable cable is available from `http://www.business-supply.com` or most electronics suppliers.

Use this extension to place your board at a more comfortable distance from your PC while debugging it. This is a recommendation, not a requirement.

3 Connecting and Configuring the Hardware

To connect the hardware, perform these steps:

1. Connect the JTAG dongle.

   If you are using a parallel cable extension, connect one end to the parallel port of the PC and the other end to the JTAG dongle.

   If you are not using a parallel cable extension, plug the JTAG dongle directly into the parallel port of the PC.

2. Connect the JTAG ribbon of the dongle into the JTAG header of your development board.

3. Power your development board with a compatible power supply.

4. If not already in JTAG debug mode, switch the board into JTAG debug mode. Consult the user manual for your board to configure any necessary settings or switches.

Note

The ARM Evaluator-7T board does not have settings to put itself into JTAG debug mode.

Jumper J9 must be connected on the Keil MCB2100 board to enable JTAG debug mode.

4 Launching H-JTAG Server and Configuring for Keil

This section describes the settings required to launch the H-JTAG server.

4.1 Parallel port settings for H-JTAG

Ensure that the parallel port settings of H-JTAG are correct before you launch the H-JTAG server.

Verify the address of the parallel port of your PC using the System tray of the Control Panel in Windows.

If you are using Windows XP:

1. Click **Start → Control Panel → System → Hardware → Device Manager**.
2. Expand the **Ports** (**COM** and **LPT**) sub-tree.
3. Right-click on **Printer Port** (**LPT1**).
4. Click on **Properties** from the drop-down menu.
5. Click on the **Resources** tab.
   The I/O range setting is the address of the PC parallel port.

4.2 Connect the H-JTAG server to the ARM core

After installing the H-JTAG package, start the H-JTAG Server:

1. Double click the **H-JTAG** shortcut on the PC desktop to open the H-JTAG server window.
2. Click **Settings → Port Settings**.
   Ensure that the selected port address matches the port address of the PC and that it passes H-JTAG port test.
3. Click **OK** to enable the port settings.
4. Click **Operations → Detect Target** to connect the H-JTAG server to the ARM core on the development board.

If initially you can not connect to the H-JTAG server to the target, it might be necessary to change the JTAG settings of the H-JTAG server. Sections 4.3 and 4.4 describe this process. Resetting the board might also be required.

Figure 1 on page 9 shows the H-JTAG Server connected to the ARM7TDMI-based Evaluator-7T board.

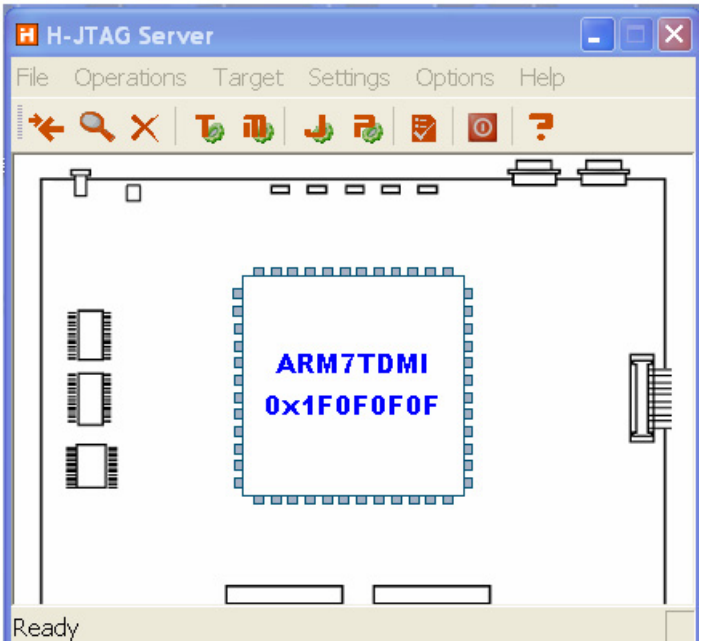

Figure 1 H-JTAG Server Connected to ARM Evaluator-7T

4.3 Using a compatible dongle

If you are using a wiggler-compatible JTAG dongle like the Olimex dongle, you must configure the JTAG settings for the H-JTAG Server:

1. Click **Settings → JTAG settings**.
2. Select the **Wiggler (Predefined) JTAG** selection.

If you are using an SDT-JTAG or user-defined interface you must select one of those options.

4.4 Debug and memory

By default, the Keil uVision3 debugger requests a system reset of the target so that it can begin executing from the ARM Reset exception vector address 0x0. This is preferred when debugging an MCU with on-chip flash that has a small amount of RAM.

If you want to use the Keil uVision3 debugger to run or debug a single assembly source file from a particular memory address without any startup code, you must disable the **nSRST** pin. Change the **Wiggler Pin Assignment** option to disable the pin. The case of the Evaluator-7T board is an example of this.

Before an ARM microcontroller can execute C programs, startup code is required to perform stack initialization and CPU setup. In Keil uVision3 C projects, the Startup.S assembler source file contains the CPU startup code. Keil provides different startup files for the different device variants. They are located in the folder `Keil\ARM\Startup`.

If you want to debug or run a complete multi-file Keil uVision3 C project with startup code, you must enable the **nSRST** pin. Figure 3 on page 11 shows an example of this on the MCB2100 board.

Figure 2 shows how to configure the JTAG settings for debugging or running a small piece of assembly code in RAM with the Keil uVision3 IDE using a wiggler-compatible JTAG dongle.

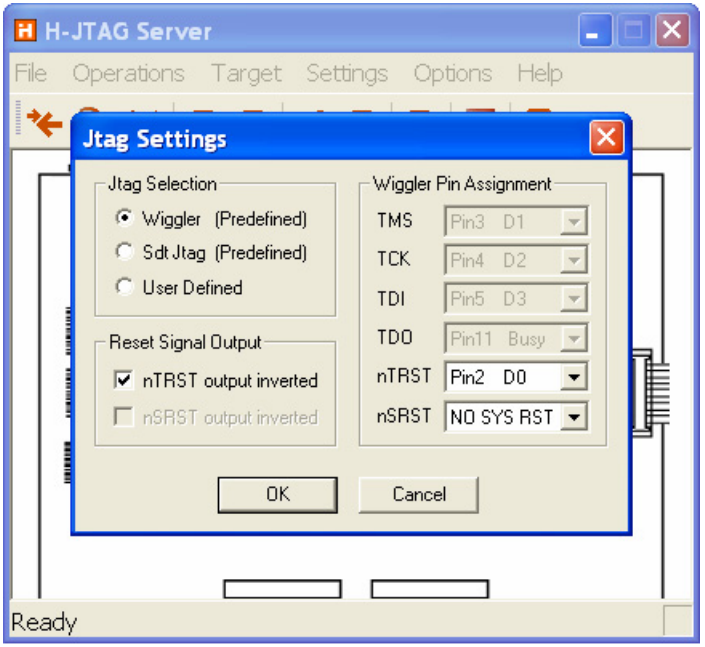

Figure 2 JTAG Settings configured for the Evaluator-7T example

Figure 3 on page 11 shows the JTAG settings for debugging or running a complete multi-file Keil uVision3 C project with startup code using a wiggler-compatible JTAG dongle.

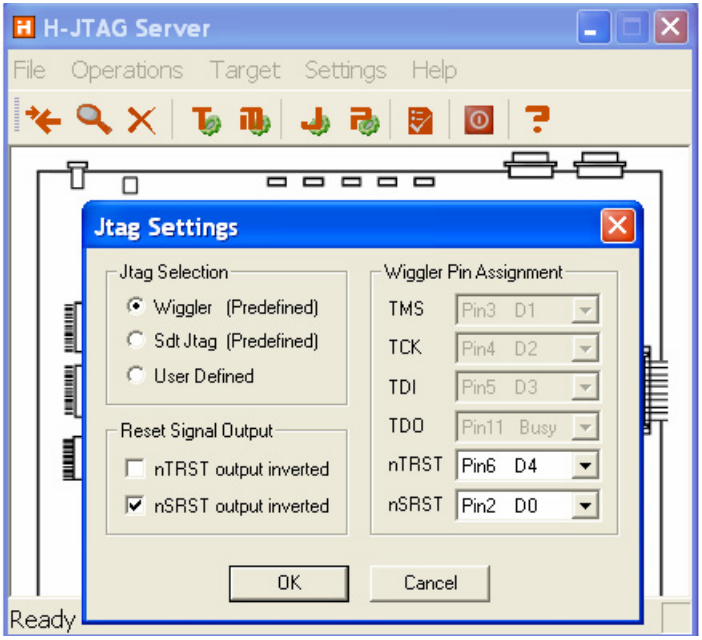

Figure 3 JTAG Settings configured for the MCB2100 example

When the settings are complete, click **OK** to enable the JTAG settings.

5 Using the Keil uVision3 IDE

After installing the Keil RealView Microcontroller Development Kit, double-click the **Keil uVision3** IDE icon from your desktop. The uVision3 IDE starts in Build Mode that enables you to create and maintain project files and generate your application.

5.1 Creating/Opening the Project

This section describes how to create or open an existing project.

Evaluator-7T Example

For the Evaluator-7T example, you must create a new project. To create a new project:

1. Click **Project → New Project**.
2. Navigate to the folder for the new project.
3. Name the folder and save it.

   For the purpose of the Evaluator-7T example, name your project EvalTest. The project is saved with the uVision `.Uv2` extension.
4. At the prompt, select a device for `Target 1`.

   `Target 1` is the current name of the target connection used in the project.

   For example, select the Samsung S3C4510B microcontroller because the Evaluator-7T board uses a variant of this. Alternatively, consult the descriptions and select the MCU you are using.
5. Click **OK**.
6. Click **File → New** to bring up a blank text file.
7. Copy and paste the following assembly code into the blank text file to add it as a source file to your project:

```
        AREA    EvalTest, CODE, AT 0x00008000
        ENTRY
        LDR     r1, =0x03FF5008    ; r1 = IOPDATA address
        LDR     r3, [r1]           ; r3 = memory[IOPDATA]
        LDR     r2, =0x10          ; r2 = LED offset
        AND     r3, r3, #0         ; r3 = 0x0 (light LED0)
Loop    LDR     r5, =0x0500000     ; r5 = delay counter
        STR     r3, [r1], #0       ; IOPDATA = r3
        B       Delay
Resume  ADD     r3, r3, r2         ; add offset
        ADDS    r4, r3, #-256      ; if r3 = 256
        BNE     Loop               ; done
```

```
        SWI     0x11                ; terminate program
Delay   SUBS    r5, r5, #1          ; delay loop
        BNE     Delay
        B       Resume
        END
```

The directives at the top of the code indicate the name of the block of code as well as the starting location of the code in the memory map of the target board. If you are using the Evaluator-7T, you must specify that the linker place all code and data in the memory range `0x8000 - 0x7FFFF`. The Evaluator-7T Board User Manual indicates this is the memory address range available as the download area for user code and data.

Note

If you are not using the Evaluator-7T, refer to the memory map information for your particular board to determine where to download your code and data.

The code displays the first 16 binary numbers to the four surface-mounted LEDs of the Evaluator-7T in ascending order.

It first loads register r1 with the memory mapped address of the **IOPDATA** register of the Samsung MCU. This register controls I/O devices on the Evaluator-7T board, including the four surface-mount LEDs. The contents of that address are read into register r3, cleared, and then written back to **IOPDATA** so the LEDs on the board initially show `0000`.

The system then runs a delay routine and the LEDs display on the board.

The system then increments register r3 by the offset and writes to **IOPDATA** to display the next binary number on the LEDs.

This continues until an undefined `SWI` instruction is executed that results in the display of the last binary number, `1111` and the termination of the program.

For more information on the programmer's model of the Evaluator-7T, see the ARM Evaluator-7T Board User Guide and the Samsung KS32C50100 User Manual.

8. Click **File → Save As**, name the file `Test.s`, and save it to a temporary location.
9. Expand the `Target 1` folders in the uVision3 IDE.
10. Right-click on **Source Group 1**.
11. Click **Add Files to Group → Source Group 1**.
12. Navigate to the location of `Test.s` and add it to the source group.

You add only one assembly file to **Source Group 1** in this example.

This procedure is similar if you are adding your own source files to your own project.

MCB2100 Example

The MCB2100 example requires that you open an existing example project. The example project is called `Blinky.Uv2`. It is located in the installation directory of the `Keil uVision3 IDE`. This project blinks the on-board LEDs. An on-board potentiometer, POT1, controls the speed at which the LEDs blink.

To open the project:

1. Click on **Project → Open Project**.
2. Navigate to the folder where the Keil tools are installed.

   For example: `Keil\ARM\Boards\Keil\MCB2100\Blinky`.
3. Click on `Blinky.Uv2` to open the project.

   The file `Abstract.txt`, shows this project has three different pre-configured target settings: `Simulator`, `MCB2100 RAM`, and `MCB2100 Flash`.

   For this example, select the `MCB2100 RAM` as the pre-configured target. This is the target used for debug.

   A small change to this configuration enables use of H-JTAG and a wiggler-compatible JTAG dongle. This is described in the section *Configuring Options for the Target*.

   This procedure is similar if you are opening a different project.

5.2 Configuring Options for the Target

Specify the target settings to use H-JTAG and a JTAG programming dongle, independent of the specific board and code used.

1. Click on **Project → Options for Target → your_target_name**.

   The value for **your_target_name** is the name of the current target. For example, in the case of the Evaluator 7T board, the value is `Target 1`, and in the case of the MCB2100 board, the value is `MCB2100 RAM`.

   Because you are using an actual hardware board, you must ensure the debugger is set for this board as well as H-JTAG and your JTAG programming dongle.
2. Click the **Debug** tab.
3. Select **Use: RDI Interface Driver**.

   You might need to select this from the drop-down menu if it is not already selected.

4. Click **Settings** and navigate to the location where the `H-JTAG.dll` driver is installed.

   This specifies the location of the H-JTAG driver and loads it. For example:

   `C:\Program Files\H-JTAG V0.2.1\H-JTAG.dll`.

5. Click **OK**.

Figure 4 shows an example of the Debug Options settings.

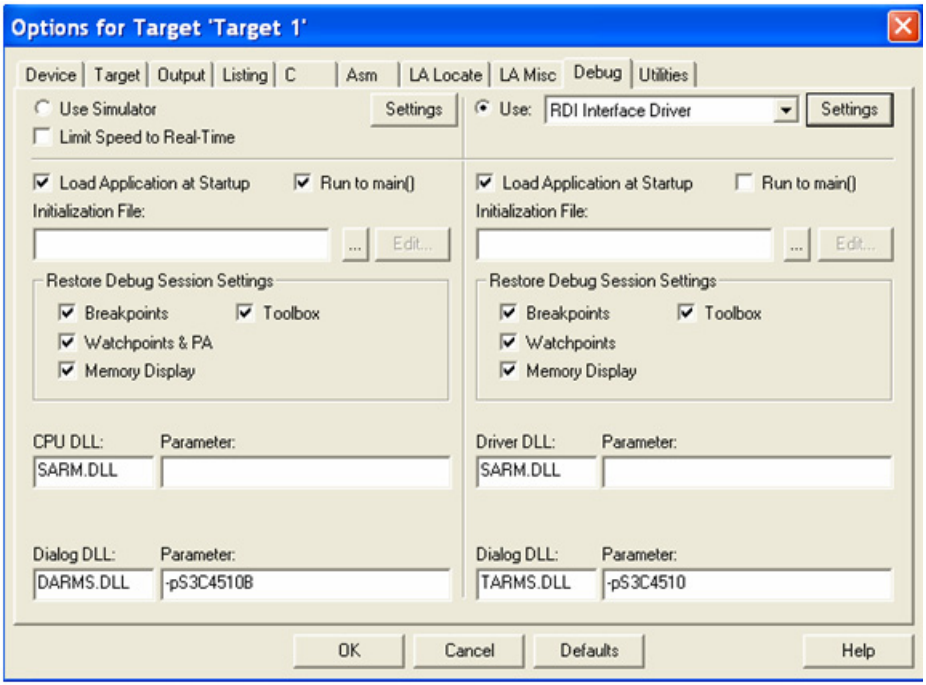

Figure 4 Debug Options for Target 1 in Evaluator-7T example

The parameter settings in Figure 4 show the selected device is the S3C4510. This is a variant of the Samsung KS32C50100 microcontroller used in the Evaluator-7T board. The parameter settings of your selected device display in the options for your target.

See the Keil uVision IDE User's Guide for more information on the Options for Target.

5.3 Building and Downloading Your Code for Debug

To compile, assemble and link your project:

Click **Project** → **Build Target**.

There is no expectation of warnings or errors generated if you are using the code in *Evaluator-7T Example* on page 12 with the Evaluator-7T board or the example project `Blinky.Uv2` with the `MCB2100` board.

To enter Debug Mode of the uVision IDE and start the debugger:

click **Debug → Start/Stop Debug Session**.

Note

If you are using the evaluation version of the Keil uVision IDE, a window indicates this. Click **OK** to continue.

The Keil uVision3 debugger downloads your code to your development board. If the program successfully downloads to the board, a blue status bar flashes in the bottom-left corner of the screen. The status bar flashes quickly and is easy to miss. An additional way to determine if the program successfully downloaded to the board is to inspect the memory.

To bring up the memory window:

Click **View → Memory Window**.

This displays your code in memory according to your target options, regardless of which board or code you are using.

The following procedures are similar if you are building and downloading your own unique code to a board other than the Evaluator-7T or MCB2100 boards.

Evaluator-7T Example

If you are using the Evaluator-7T and the code from *Creating/Opening the Project* on page 12 you can read the hex value of the first instruction residing at address `0x8000`. The hex value for the first instruction is:

```
30 10 9F E5.
```

Before you debug or run your code, ensure the program counter points to the correct memory address. For the case of the Evaluator-7T programming example, point the program counter to address 0x8000:

1. Click **View → Project Window**.
2. Click the **Regs** tab to view all of the registers.
3. Expand the **Current** registers.
4. Triple click on **R15 (PC)**.

5. Enter a new value for **R15(PC)**.

   Type in the new value `0x8000` and press **ENTER**.

   The program counter now holds the address `0x8000` and you are ready to debug or run the code.

To perform a quick run of your code:

Click **Debug → Run**.

If you are using the Evaluator-7T and code from *Creating/Opening the Project* on page 12, the binary numbers are displayed on the four surface-mounted LEDs in ascending order.

MCB2100 Example

If you are using the `MCB2100` and the `Blinky.Uv2` project, you can read the hex value of the first instruction of `main()` at address `0x40000230`. The hex value for the first instruction is `00 B5 0E 49`. You are now ready to debug or run the code.

To perform a quick run of your code:

Click **Debug → Run**.

If you are using the `MCB2100` and the `Blinky.Uv2` project, the surface-mounted LEDs blink, traversing through all eight individual on-board LEDs. You can adjust the traversal speed with the on-board potentiometer `POT1`.

See the Keil uVision3 IDE/Debugger User's Guide for more information on how to debug programs using the Keil uVision3 Debugger. This documentation is available from the web site: `http://www.keil.com/`.

6 Glossary of ARM Terms and Abbreviations

Debug Agent Tool that enables the physical connection between a target microprocessor and a debugger running on a host PC through one of the PC's I/O ports.

Dongle A physical device, attached to a PC's I/O port, that adds hardware capabilities.

ICE Acronym for **In-Circuit Emulator** (ICE).

An In-Circuit Emulator (ICE) is a device that aids the debugging of hardware and software. ARM processors from ARM7TDMI processor onwards have extra hardware called EmbeddedICE Logic to assist this process.

IDE Acronym for **Integrated Development Environment** (IDE).

A development environment offering facilities for automating image-building and file-management processes. The CodeWarrior IDE is an example.

JTAG Acronym for **Joint Test Action Group** (JTAG).

The name of the organization that developed standard IEEE 1149.1. This standard defines a boundary-scan architecture used for in-circuit testing of integrated circuit devices.

SWI Acronym for **Software Interrupt Instruction** (SWI).

This instruction (SWI) causes the core to enter Supervisor mode and take a Software Interrupt Exception.

Wiggler A low-cost parallel port interface used in the design, debug, and programming of microprocessor and microcontroller based embedded systems.