

The Atmel AVR ONE! Debugger



The Atmel® AVR ONE! is a powerful development tool for on-chip debugging and programming of all Atmel AVR® 8- and 32-bit MCU devices. It supports:

- Programming and on-chip debugging of all AVR UC3 microcontrollers and processors on both JTAG and aWire interfaces
- On-chip debugging with Nexus AUX trace of all AVR UC3 microcontrollers and processors with AUX port
- Programming and on-chip debugging of all AVR XMEGA[®] family devices on both JTAG and PDI 2-wire interfaces
- Programming (JTAG and SPI) and debugging of all Atmel megaAVR[®] and Atmel tinyAVR[®] microcontrollers with OCD support on both JTAG or debugWIRE interfaces

Table of Contents

The Atmel AVR ONE! Debugger.....	1
1. Introduction.....	5
1.1. Introduction to the Atmel AVR ONE!.....	5
1.2. Atmel AVR ONE! Features.....	6
1.3. System Requirements.....	7
2. Release History, New Features.....	8
2.1. What's New.....	8
2.2. Firmware Release History.....	8
3. Known Issues.....	14
3.1. General.....	14
3.2. Atmel AVR XMEGA OCD Specific Issues.....	14
3.3. Atmel megaAVR OCD and Atmel tinyAVR OCD Specific Issues.....	14
3.4. Atmel AVR 32-bit Microcontroller Specific Issues.....	14
4. Getting Started.....	15
4.1. Kit Contents.....	15
4.2. Powering the Atmel AVR ONE!.....	16
4.3. Connecting to the Host Computer.....	16
4.4. USB Driver Installation.....	16
4.4.1. Windows.....	16
4.5. Programming and Debugging.....	18
5. Connecting the Atmel AVR ONE!.....	19
5.1. Connecting to a JTAG Target.....	19
5.1.1. Using the JTAG Mictor Connector.....	19
5.1.2. Using the JTAG 10-pin Connector.....	19
5.2. Connecting to an aWire Target.....	20
5.3. Connecting to a PDI Target.....	22
5.4. Connecting to a debugWIRE Target.....	23
5.5. Connecting to an SPI Target.....	24
5.6. Using the Atmel AVR ONE! with Atmel STK500.....	25
5.7. Using the Atmel AVR ONE! with Atmel STK600.....	28
6. On-chip Debugging.....	31
6.1. Introduction to On-chip Debugging (OCD)	31
6.2. Physical Interfaces.....	31
6.2.1. JTAG.....	32
6.2.2. Auxiliary (AUX) Physical (including JTAG).....	33
6.2.3. aWire.....	35
6.2.4. PDI Physical.....	35
6.2.5. debugWIRE.....	36
6.2.6. SPI.....	36

6.3.	Atmel AVR OCD Implementations.....	36
6.3.1.	Atmel AVR UC3 OCD (JTAG and aWire Physical).....	36
6.3.2.	Atmel AVR XMEGA OCD (JTAG and PDI Physical).....	37
6.3.3.	Atmel megaAVR OCD (JTAG).....	37
6.3.4.	Atmel megaAVR / tinyAVR OCD (debugWIRE).....	37
7.	Atmel AVR ONE! Hardware Description.....	38
7.1.	LEDs.....	38
7.2.	Rear Panel.....	39
7.3.	Probe.....	39
7.4.	Architecture Description.....	40
7.4.1.	Atmel AVR ONE! Main-board.....	40
7.4.2.	Atmel AVR ONE! Probe.....	40
8.	Software Integration.....	42
8.1.	Atmel Studio.....	42
9.	Command Line Utility.....	43
10.	Advanced Debugging Techniques.....	44
10.1.	Atmel AVR 32-bit Microcontrollers.....	44
10.1.1.	EVTI/EVTO Usage.....	44
10.2.	Atmel megaAVR Targets.....	44
10.2.1.	I/O Debug Register (IDR).....	44
10.3.	debugWIRE Targets.....	45
10.3.1.	Software Breakpoints.....	45
11.	Special Considerations.....	46
11.1.	Atmel AVR XMEGA OCD.....	46
11.2.	Atmel megaAVR OCD and debugWIRE OCD.....	47
11.3.	Atmel megaAVR OCD (JTAG).....	48
11.4.	debugWIRE OCD.....	49
11.5.	Atmel AVR UC3 OCD.....	50
11.6.	Atmel AVR UC3 Shutdown Mode.....	50
12.	Troubleshooting.....	51
12.1.	Self-test.....	51
12.1.1.	Connecting.....	51
12.1.2.	Launching.....	52
12.1.3.	How to use the Results for Diagnosis.....	52
12.2.	Troubleshooting Guide.....	52
13.	Product Compliance.....	55
13.1.	RoHS and WEEE.....	55
13.2.	CE and FCC.....	55
14.	Revision History.....	56

1. Introduction

1.1. Introduction to the Atmel AVR ONE!

The Atmel AVR ONE! is a powerful development tool for on-chip debugging and programming of all Atmel AVR 8- and 32-bit microcontrollers. It supports:

- Programming and on-chip debugging of all AVR UC3 microcontrollers and processors on both JTAG and aWire interfaces
- On-chip debugging with Nexus AUX trace of all UC3 microcontrollers and processors with AUX port
- Programming and on-chip debugging of all AVR XMEGA family devices on both JTAG and PDI 2-wire interfaces
- Programming (JTAG and SPI) and debugging of all AVR 8-bit microcontrollers with OCD support on both JTAG or debugWIRE interfaces

To see which devices are currently supported read the Atmel Studio release notes/readme.



1.2. Atmel AVR ONE! Features

- Fully compatible with Atmel Studio, AVR32 Studio, AVR Studio[®] 4, and AVR Studio 5
- Supports programming and debugging of all Atmel AVR UC3 and AVR XMEGA devices, and all Atmel megaAVR and Atmel tinyAVR devices with OCD
- Supports AUX trace on all UC3 devices with AUX port
- On-board 128MB DDR-SDRAM for use as circular, linear, or elasticity trace buffer
- Target operating voltage range of 1.65V to 5.5V
- Draws less than 1mA from target V_{Tref} during operation
- Supports communications clock frequency from 32kHz to 33MHz
- USB 2.0 high-speed host interface
- Supports both MICTOR-38 connector and 10-pin JTAG connectors, as well as aWire, PDI, SPI, and debugWIRE interfaces using an adapter



1.3. System Requirements

The Atmel AVR ONE! unit requires that a front-end debugging environment (AVR32 Studio or AVR Studio 4.15 or later, or Atmel Studio) and associated utilities are installed on your computer. For system requirements of these packages, consult www.atmel.com.

The AVR ONE! unit must be connected to the host computer using the USB cable provided. To achieve streaming trace readout from the AVR ONE!, the USB port on the computer must be USB 2.0 high-speed compliant. If the computer only supports USB 1.1 (or "full-speed" USB), the AVR ONE! can still be used for debugging (Note: restrictions apply, see [Known issues](#)), but trace will be limited to operating in buffer mode.

The AVR ONE! unit must be connected to a 12V external power source, which is included with the kit.

2. Release History, New Features

2.1. What's New

New in this release

Release platform	Atmel Studio 6.0
Firmware versions	MCU: 5.21 (0x0515) AVR32 image: 4.1 (0x0401) XMEGA image: 3.2 (0x0302) TMEGA image: 2.1 (0x0201)
New features	None
Fixes	Minor internal bug fixes

2.2. Firmware Release History

Table 2-1. Previous Releases

Release platform	AVR Studio 5.1
Firmware versions	MCU: 5.20 (0x0514) AVR32 image: 4.1 (0x0401) XMEGA image: 3.2 (0x0302) TMEGA image: 2.1 (0x0201)
New features	<ul style="list-style-type: none">• Support for high SUT values on Atmel AVR XMEGA devices
Fixes	<ul style="list-style-type: none">• aWire auto-baud calculation improvements• Fixed Atmel AVR XMEGA flash page programming error (seen at low voltages)• Improved debugWIRE single-stepping performance
Release platform	AVR Studio 5.0
Firmware versions	MCU: 5.13 (0x050D) AVR32 image: 4.0 (0x0400) XMEGA image: 3.0 (0x0300) TMEGA image: 2.0 (0x0200)
New features	None
Fixes	Fixed Atmel AVR XMEGA software reset handling

Release platform	AVR Studio 5.0 beta
Firmware versions	MCU: 5.11 (0x050B) AVR32 image: 4.0 (0x0400) XMEGA image: 3.0 (0x0300) TMEGA image: 2.0 (0x0200)
New features	None
Fixes	Improved aWire speed

Release platform	AVR Studio 5 beta
Firmware versions	MCU: 0x0506 AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200
New features	None
Fixes	Added support for AVR Studio 5 frontend

Release platform	AVR Studio 4.18 SP3	AVR32 Studio 2.6.0
Firmware versions	MCU: 0x040F AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
Firmware versions	MCU: 0x0410 AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
New features	None	None
Fixes	Added support for special HVE MUL instructions	<ul style="list-style-type: none"> • Added support for shutdown mode (Atmel AVR UC3 L) • Fixed device disconnecting when no FPGA image is loaded (on certain Linux® versions)

Release platform	AVR Studio 4.18 SP2	AVR32 Studio 2.5.0
Firmware versions	MCU: 0x040C AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
Firmware versions	MCU: 0x040C AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
New features	None	None
Fixes	Fixed run from software breakpoint at double word instruction.	None
Release platform	AVR Studio 4.18 SP1 PP1	AVR32 Studio 2.4.0
Firmware versions	MCU: 0x040B AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
Firmware versions	MCU: 0x040B AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
New features	None	None
Fixes	None	Fixed EVTO sensing on 10-pin JTAG connector.
Release platform	AVR Studio 4.18 SP1	AVR32 Studio 2.4.0
Firmware versions	MCU: 0x040B AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	

Firmware versions	MCU: 0x040B AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
New features	None	None
Fixes	Fixed Atmel AVR XMEGA EEPROM read/write when EEPROM is memory mapped.	Fixed EVTO sensing on 10-pin JTAG connector.
Release platform	AVR32 Studio 2.3.1	
Firmware versions	MCU: 0x0409 AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
Firmware versions	MCU: 0x0409 AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	
New features		
Fixes	Fixed issues regarding connection to targets running at voltages above 3.3V.	Fixed issues regarding connection to targets running at voltages above 3.3V.
Release platform	AVR Studio 4.18	AVR32 Studio 2.3
Firmware versions	MCU: 0x0408 AVR32 image: 0x0301 XMEGA image: 0x0201 TMEGA image: 0x0107	
Firmware versions	MCU: 0x0408 AVR32 image: 0x0400 XMEGA image: 0x0300 TMEGA image: 0x0200	

New features	Support for longer JTAG daisy chains. Max. daisy chain is now 240 instruction register bits.	<ul style="list-style-type: none"> Support for longer JTAG daisy chains. Max. daisy chain is now 240 instruction register bits. Support for oscillator calibration for XMEGA
Fixes	<ul style="list-style-type: none"> Improved support for debugging targets running at low clock frequencies Fixed bug resulting in lost hardware breakpoint when a software breakpoint is removed Fixed software breakpoint masking when reading flash in debug mode for Atmel megaAVR JTAG 	Improved aWire baud rate setting.

Release platform	AVR Studio 4.17
Firmware versions	MCU: 0x0310 AVR32 image: 0x0301 XMEGA image: 0x0201 TMEGA image: 0x0107
New features	Support for Atmel tinyAVR and Atmel megaAVR devices
Fixes	Fixed reset line loading problem. Atmel AVR ONE! will not load the target reset line anymore as long as its power is switched on.

Release platform	AVR32 Studio 2.2
Firmware versions	MCU: 0x0306 AVR32 image: 0x0300 XMEGA image: 0x0200 TMEGA image: 0x0103
New features	Support for aWire interface
Fixes	Fixed reset line loading problem. Atmel AVR ONE! will not load the target reset line anymore as long as its power is switched on.

Release platform	AVR Studio 4.16
Firmware versions	MCU: 2.0B AVR32 image: 1.09 XMEGA image: 1.04

New features	None	
Fixes	<ul style="list-style-type: none"> • Fixed JTAG programming problem with JTAG clock set to 8MHz • Fixed software breakpoint bug resulting in ghost breakpoints • Fixed single stepping bug for Atmel AVR XMEGA (step out) 	
Release platform	AVR Studio 4.15	AVR32 Studio 2.1.0
Firmware versions	MCU: 2.07 MCU: 2.06 AVR32 image: 1.09 XMEGA image: 1.03	
New features	Atmel AVR XMEGA Family support	
Fixes	None, this is a new release	Minor bug fixes
Release platform	AVR32 Studio 2.0	
Firmware versions	MCU: 1.01 AVR32 image: 1.01	
New features	Atmel AVR 32-bit microcontrollers support	
Fixes	None	

3. Known Issues

3.1. General

- Firmware upgrade will not work when using USB 1.1 or USB 2.0 Full Speed. Until a fix is published in a future GNU Toolchain AVR Studio 4 release, the workaround is to use a USB 2.0 host to perform the upgrade.
- When the Atmel AVR ONE! is connected to the target, but is not powered, it will passively load the RESET line. This may cause the target device to be held in reset unintentionally. Always power up the emulator when it is connected to a target device. The RESET will also be loaded slightly when performing a firmware upgrade, and when reconfiguring the FPGA when changing target family types. In some circumstances this may cause the target device to be reset.
- Daisy chain auto detection won't work when an ATmega128 is a part of the daisy chain except if the ATmega128 is the first device in the chain. This is due to the JTAG instruction IDCODE not working correctly in this device. For more information, see the Errata section in the ATmega128 data sheet.
- The AVR ONE! firmware released by AVR Studio 4.15 and later only works with AVR32 Studio version 2.1 and later. If you have upgraded your AVR ONE! from AVR Studio 4.15 and need to work with AVR32 Studio version 2.0, the firmware can be downgraded by manually initializing an upgrade in AVR32 Studio. When you want to work with AVR Studio 4 again, a firmware upgrade is automatically initiated.

3.2. Atmel AVR XMEGA OCD Specific Issues

- For the ATxmegaA1 family, only revision G or later is supported
- For the Atmel AVR XMEGA family the LiveDebug support on JTAG sometimes fails (target is being reset). PDI is the preferred interface for LiveDebug. When detaching, the target may be left in stopped mode, requiring an external reset.

3.3. Atmel megaAVR OCD and Atmel tinyAVR OCD Specific Issues

- Cycling power on ATmega32U6 during a debug session may cause a loss of sync with the device
- Single stepping GCC-generated code in source-level may not always be possible. Set optimization level to lowest for best results, and use the dis-assembly view when necessary.
- Setting the target clocks (debugging and programming clocks for JTAG) too low, typically below 100kHz or running a debugWIRE target at slow frequencies might result in errors in Atmel Studio due to timeouts being too short to handle the time it takes to complete some commands in the Atmel AVR ONE! emulator when using these low frequencies. This problem increases as the target device memories increase in size and as the number of active software breakpoints increases.

3.4. Atmel AVR 32-bit Microcontroller Specific Issues

- NONE

4. Getting Started

4.1. Kit Contents

The Atmel AVR ONE! kit contains these items:

- AVR ONE! unit with probe
- USB cable (1.8m, high-speed)
- EU and US mains power cables
- Power supply
- AVR ONE! test adapter with 10-pin cable attached
- Stand-off adapters (10-pin 100-mil, 6-pin 100-mil, 10-pin 50-mil, 6-pin 50-mil)
- 38-pin MICTOR connector samples
- 10-pin squid cable
- Atmel AVR Technical Library DVD
- Getting Started Guide

Figure 4-1. Atmel AVR ONE! Kit Contents



4.2. Powering the Atmel AVR ONE!

The Atmel AVR ONE! must be powered by an external power supply (provided) which is capable of supplying 15W at 12V DC. The polarity of the DC jack is positive-centre. When powering up the AVR ONE!, the power LED should illuminate immediately. If the LED does not light up, check that the correct power supply is being used, or check that the polarity is positive-centre if a different power supply is being used.

4.3. Connecting to the Host Computer

Before connecting up the Atmel AVR ONE! for the first time, be sure to install the USB driver on the host computer. This is done automatically when installing the front-end software provided free by Atmel. See www.atmel.com for further information or to download the latest front-end software.

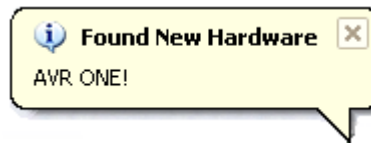
The AVR ONE! must be connected to an available USB port on the host computer using the USB cable provided. The AVR ONE! contains a USB 2.0 compliant controller, and can operate in both full-speed (Note: restrictions apply, see [Known issues](#)) and high-speed modes, although the streaming trace feature is not available when operating at only full-speed. For best results, connect the AVR ONE! directly to the host computer (not via external hubs) and use a USB 2.0 high-speed certified cable (provided).

4.4. USB Driver Installation

4.4.1. Windows

When installing the Atmel AVR ONE! on a computer running Microsoft® Windows®, the USB driver is loaded when the AVR ONE! is first powered up.

Note: Be sure to install the front-end software packages before powering up for the first time!

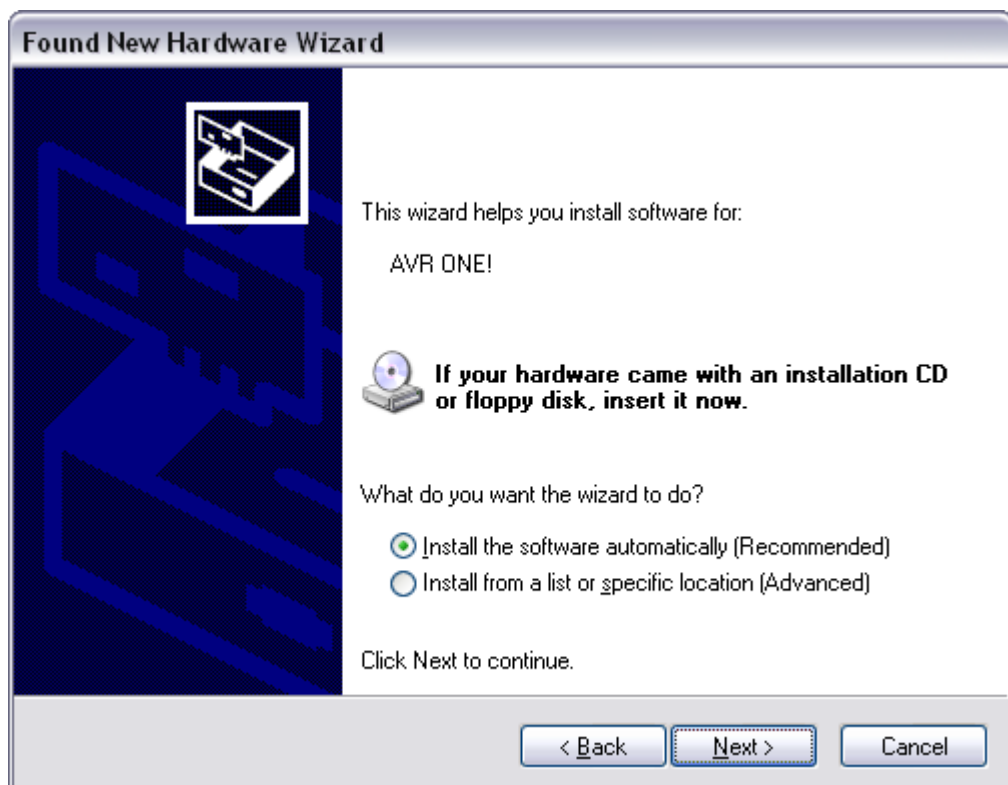


Proceed with the default ("recommended") options through the New Hardware Wizard.

Figure 4-2. Installing the AVR ONE! USB Driver



Figure 4-3. Installing the AVR ONE! USB Driver

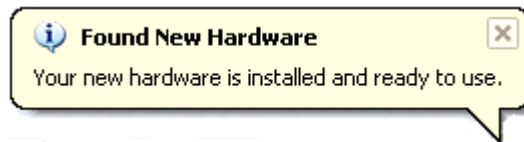


If not automatically detected, point the wizard to the device driver (provided by Jungo) called avrone.inf, which is stored in the <windows_root>\inf folder.

Once successfully installed, the AVR ONE! will appear in the device manager as a "Jungo" device.



Your AVR ONE! is now ready to use.



4.5. Programming and Debugging

The simplest way to get started with your Atmel AVR ONE! using Atmel Studio is to build one of the many example projects included with Atmel Studio.

5. Connecting the Atmel AVR ONE!

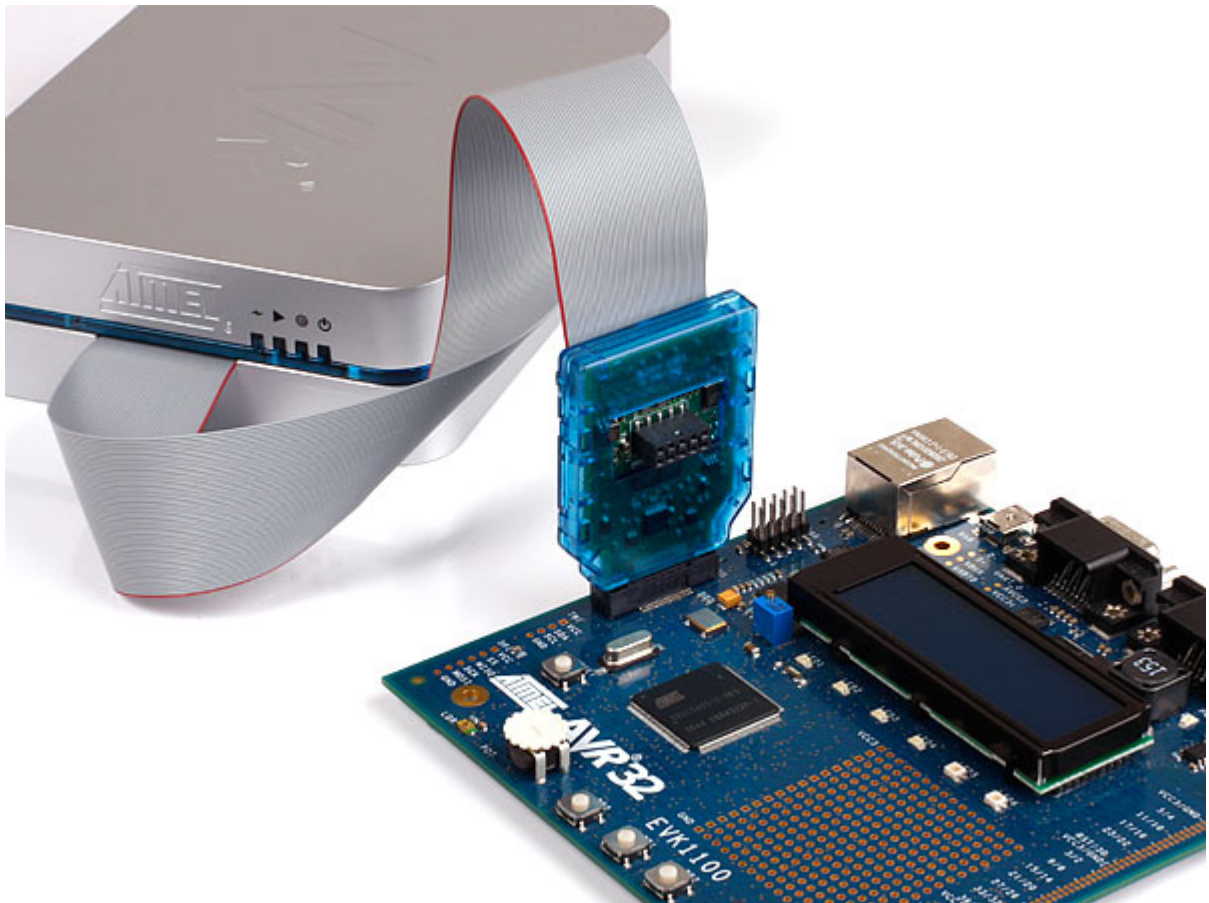
5.1. Connecting to a JTAG Target

The Atmel AVR ONE! probe has two target connectors that support JTAG debugging and programming. When debugging a target device with AUX trace, the 38-pin Nexus (Mictor) connector should be used. For JTAG debugging without AUX trace, or programming, either the Mictor or the 10-pin connector can be used.

5.1.1. Using the JTAG Mictor Connector

The pinout for the Mictor 38-pin connector is shown in [Figure 6-4 Mictor Connector Pinout](#).

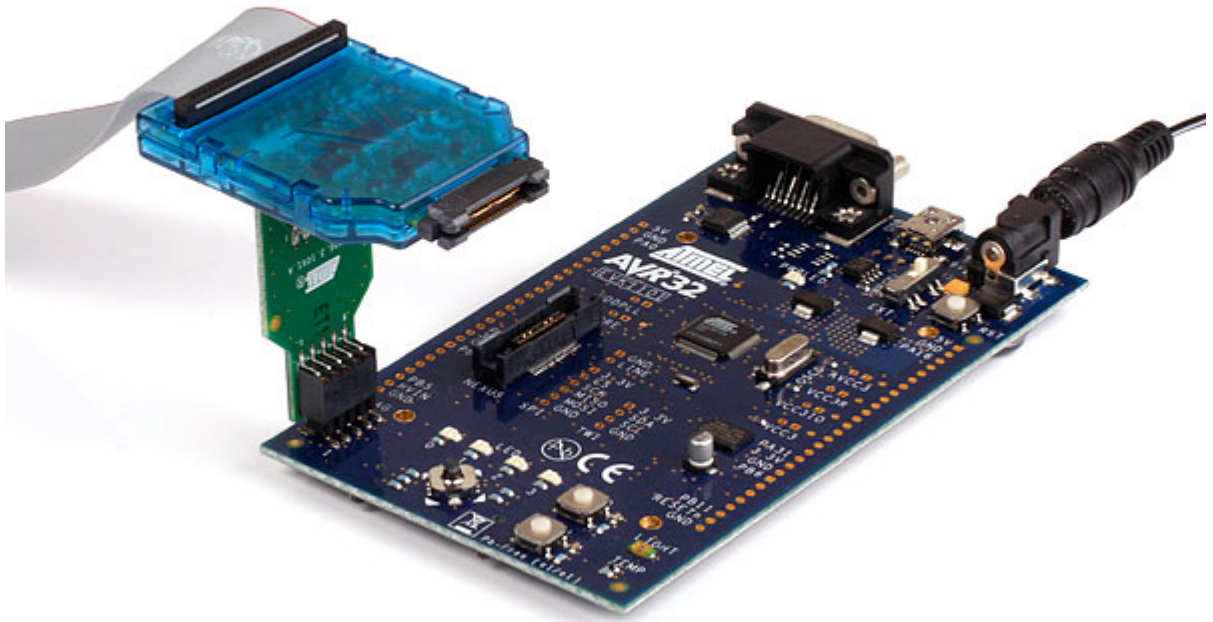
Simply insert the Atmel AVR ONE! probe connector into the target application's Mictor connector and apply pressure until a connection is secured. The socket is polarized to prevent incorrect mating orientation.



5.1.2. Using the JTAG 10-pin Connector

The pinout for the 10-pin JTAG connector is shown in [Figure 6-2 JTAG Header Pinout](#).

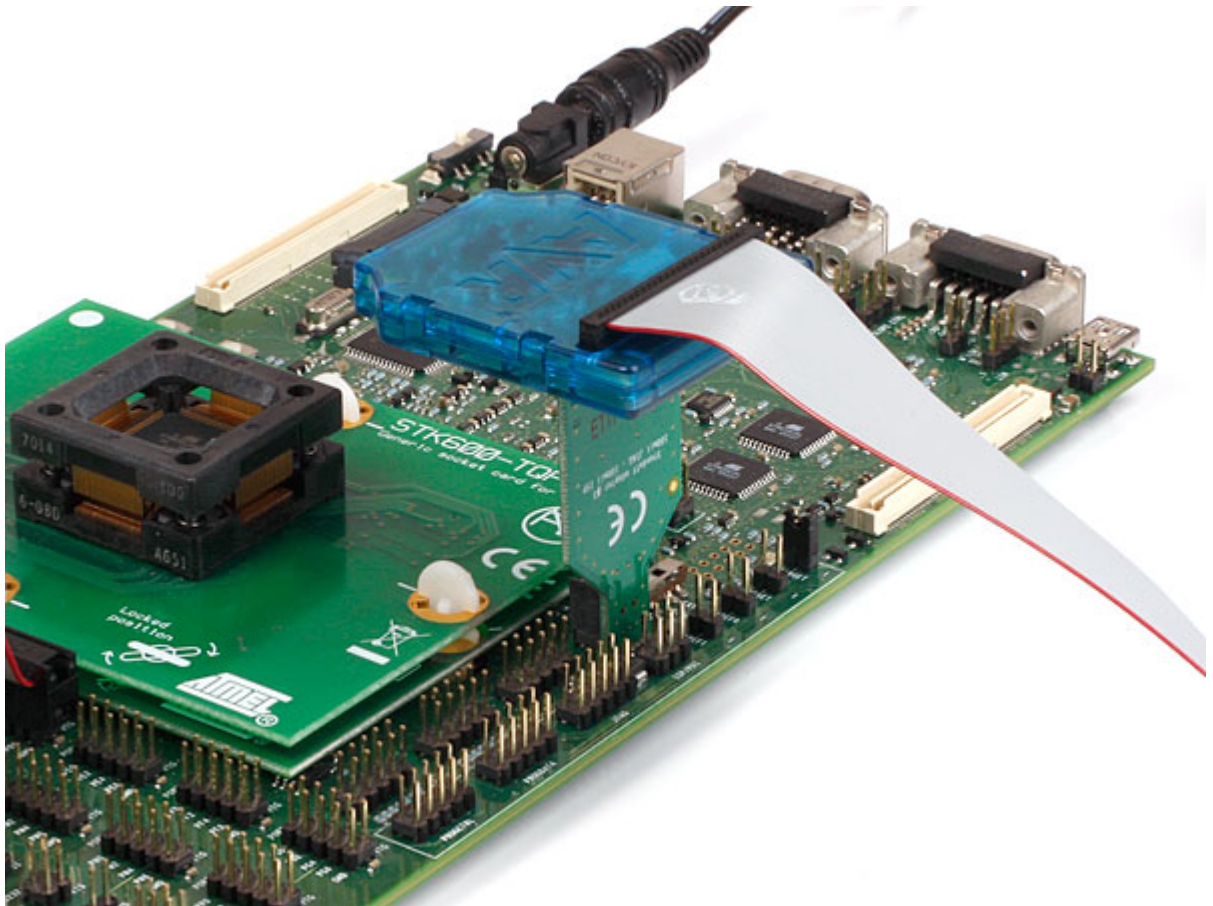
Be sure to use the correct orientation of the 10-pin header when connecting the Atmel AVR ONE! to the target application PCB. The stand-off adapters (provided) can be used to connect the AVR ONE! probe to both 100-mil and 50-mil target application connectors.



5.2. Connecting to an aWire Target

The Atmel AVR ONE! can interface with the Atmel AVR UC3 L family of devices using the single-wire 'aWire' interface. The recommended pinout for the aWire interface is shown in [Figure 6-5 aWire Header Pinout](#).

Be sure to use the correct orientation of the 6-pin header when connecting the AVR ONE! to the target application PCB. The stand-off adapters (provided) can be used to connect the AVR ONE! probe to both 100-mil and 50-mil target application connectors.



The aWire interface only requires one data line in addition to V_{CC} and GND. The recommended 6-pin pinout is based on existing AVR interfaces, as well as the resources on the AVR ONE! debugger itself. When connecting to a target that does not have the standard 6-pin header, you can use the squid cable between the AVR ONE! 10-pin JTAG connector on the probe and the target board. Three connections are required, as described in the table below.

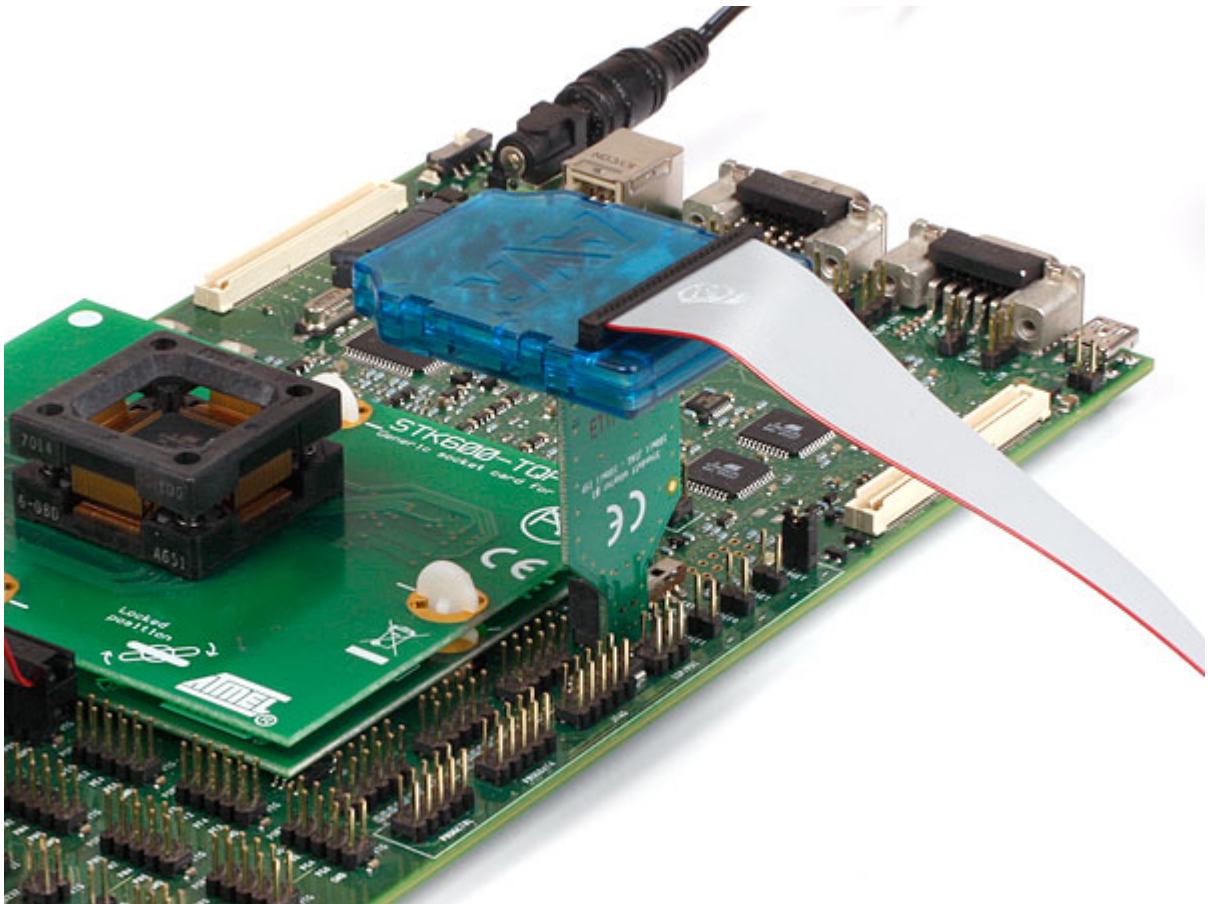
Table 5-1. Connecting to aWire using the Squid Cable

<i>AVR ONE! JTAG probe</i>	<i>Target pins</i>	<i>Squid cable colors</i>	<i>aWire pinout</i>
Pin 1 (TCK)		Black	
Pin 2 (GND)	GND	White	6
Pin 3 (TDO)	DATA	Grey	1
Pin 4 (VTref)	VTref	Purple	2
Pin 5 (TMS)		Blue	
Pin 6 (nSRST)		Green	
Pin 7 (Not connected)		Yellow	
Pin 8 (nTRST)		Orange	
Pin 9 (TDI)		Red	
Pin 10 (GND)		Brown	

5.3. Connecting to a PDI Target

The pinout for the 6-pin PDI connector is shown in [Figure 6-6 PDI Header Pinout](#).

Be sure to use the correct orientation of the 6-pin header when connecting the Atmel AVR ONE! to the target application PCB. The stand-off adapters (provided) can be used to connect the AVR ONE! probe to both 100-mil and 50-mil target application connectors.



When connecting to a target that does not have the standard 6-pin header, you can use the squid cable between the AVR ONE! 10-pin JTAG connector on the probe and the target board. 4 connections are required, and the table below describes where to connect them.

Note: There is a difference from the JTAGICE mkII JTAG probe, where PDI_DATA is connected to pin 9.

Table 5-2. Connecting to PDI using the Squid Cable

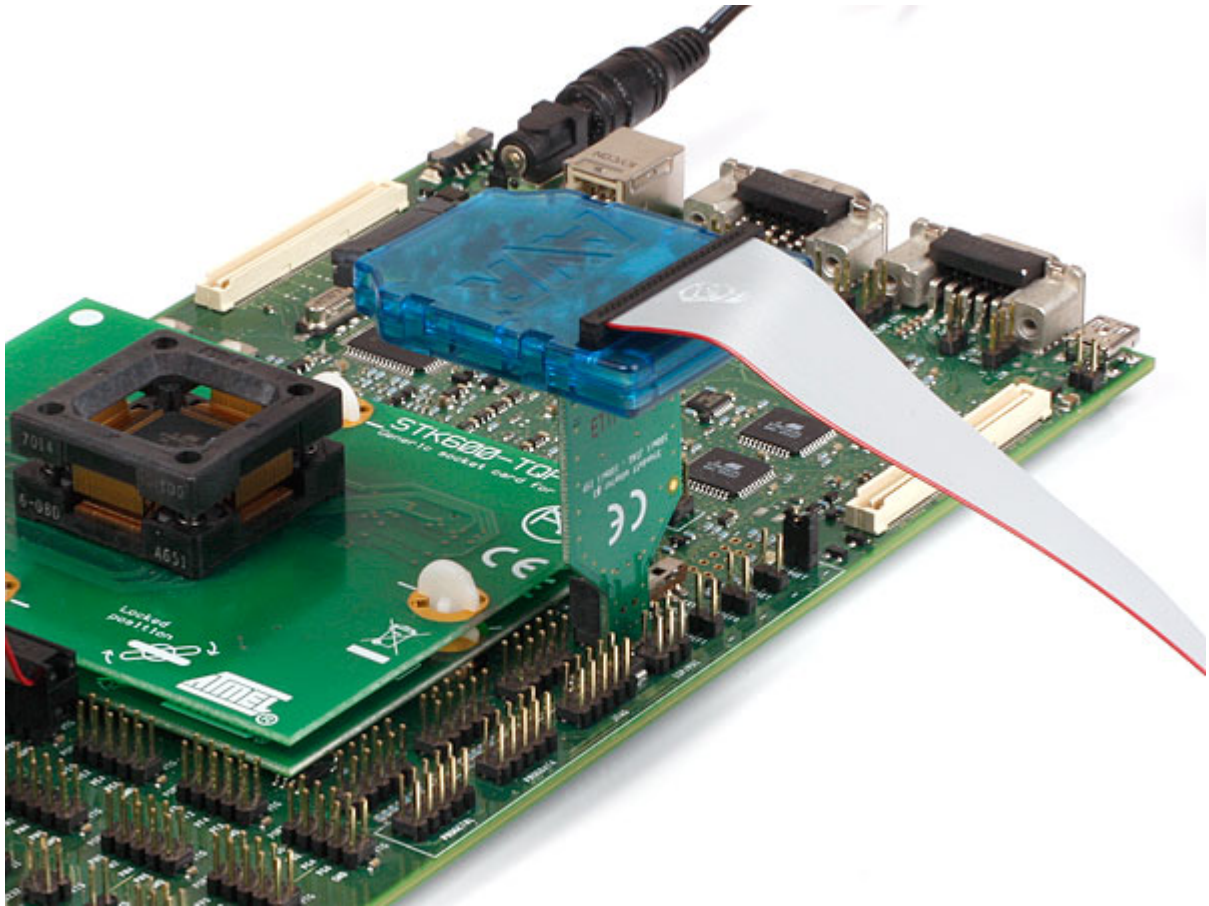
<i>AVR ONE! JTAG probe</i>	<i>Target pins</i>	<i>Squid cable colors</i>	<i>Atmel STK600 PDI pinout</i>
Pin 1 (TCK)		Black	
Pin 2 (GND)	GND	White	6
Pin 3 (TDO)	PDI_DATA	Grey	1
Pin 4 (VTref)	VTref	Purple	2
Pin 5 (TMS)		Blue	
Pin 6 (nSRST)	PDI_CLK	Green	5

AVR ONE! JTAG probe	Target pins	Squid cable colors	Atmel STK600 PDI pinout
Pin 7 (Not connected)		Yellow	
Pin 8 (nTRST)		Orange	
Pin 9 (TDI)		Red	
Pin 10 (GND)		Brown	

5.4. Connecting to a debugWIRE Target

The pinout for the 6-pin debugWIRE (SPI) connector is shown in [Figure 6-7 debugWIRE \(SPI\) Header Pinout](#).

Be sure to use the correct orientation of the 6-pin header when connecting the Atmel AVR ONE! to the target application PCB. The stand-off adapters (provided) can be used to connect the AVR ONE! probe to both 100-mil and 50-mil target application connectors.



Although the debugWIRE interface requires only one signal line (RESET), V_{CC} and GND to operate correctly, it is advised to have access to the full SPI connector so that the debugWIRE interface can be enabled and disabled using SPI programming.

When the DWEN fuse is enabled the SPI interface is overridden internally in order for the OCD module to have control over the RESET pin. The debugWIRE OCD is capable of disabling itself temporarily (using the button on the debugging tab in the properties dialog in Atmel Studio), thus releasing control of the RESET line. The SPI interface is then available again (only if the SPIEN fuse is programmed), allowing

the DWEN fuse to be un-programmed using the SPI interface. If power is toggled before the DWEN fuse is un-programmed, the debugWIRE module will again take control of the RESET pin. It is HIGHLY ADVISED to simply let Atmel Studio handle setting and clearing of the DWEN fuse!

It is not possible to use the debugWIRE Interface if the lockbits on the target AVR are programmed. Always be sure that the lockbits are cleared before programming the DWEN fuse and never set the lockbits while the DWEN fuse is programmed. If both the debugWIRE enable fuse (DWEN) and lockbits are set, one can use High Voltage Programming to do a chip erase, and thus clear the lockbits. When the lockbits are cleared the debugWIRE Interface will be re-enabled. The SPI Interface is only capable of reading fuses, reading signature, and performing a chip erase when the DWEN fuse is un-programmed.

Table 5-3. Connecting to debugWIRE using the Squid Cable

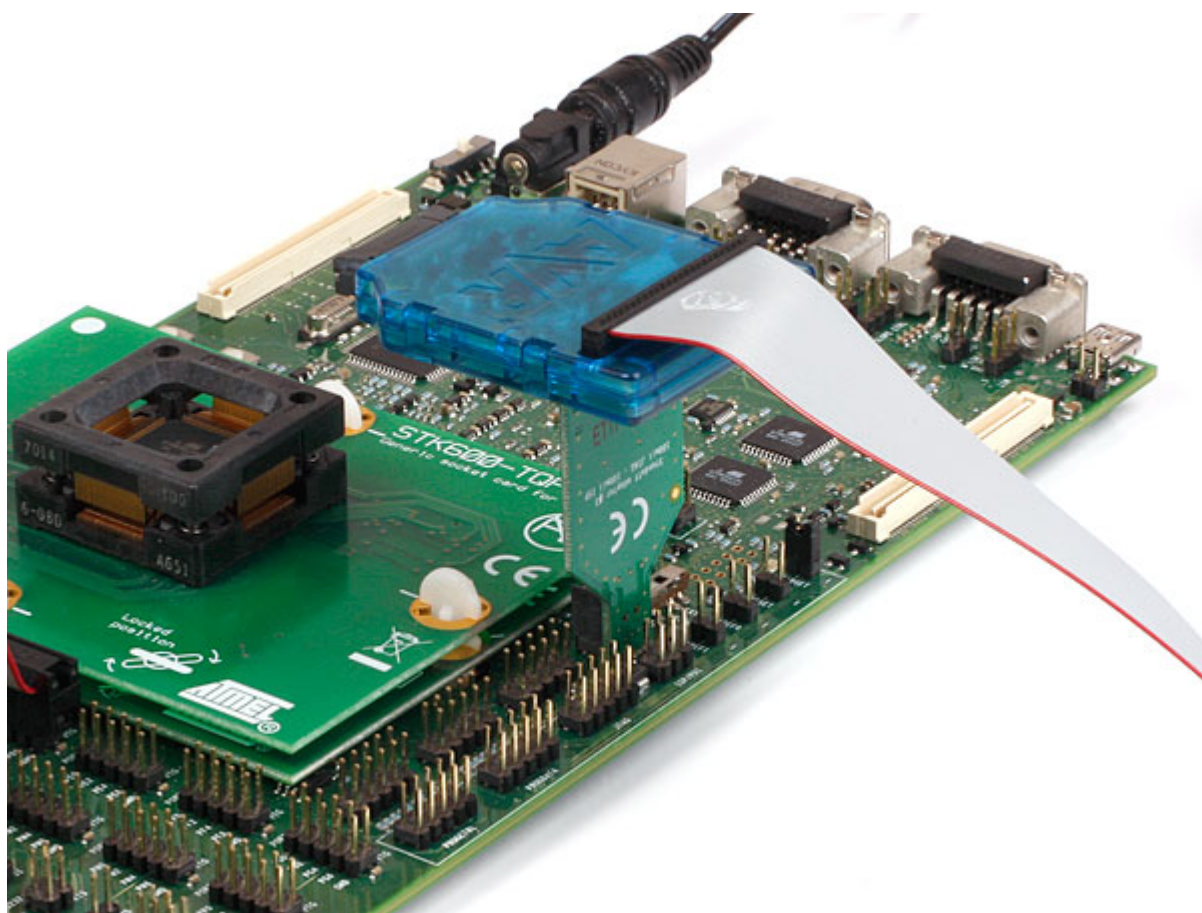
<i>AVR ONE! JTAG probe</i>	<i>Target pins</i>	<i>Squid cable colors</i>
Pin 1 (TCK)		Black
Pin 2 (GND)	GND	White
Pin 3 (TDO)		Grey
Pin 4 (VTref)	VTref	Purple
Pin 5 (TMS)		Blue
Pin 6 (nSRST)	RESET	Green
Pin 7 (Not connected)		Yellow
Pin 8 (nTRST)		Orange
Pin 9 (TDI)		Red
Pin 10 (GND)		Brown

5.5. Connecting to an SPI Target

The pinout for the 6-pin SPI connector is shown in [Figure 6-8 SPI Header Pinout](#).

Be sure to use the correct orientation of the 6-pin header when connecting the Atmel AVR ONE! to the target application PCB. The stand-off adapters (provided) can be used to connect the AVR ONE! probe to both 100-mil and 50-mil target application connectors.

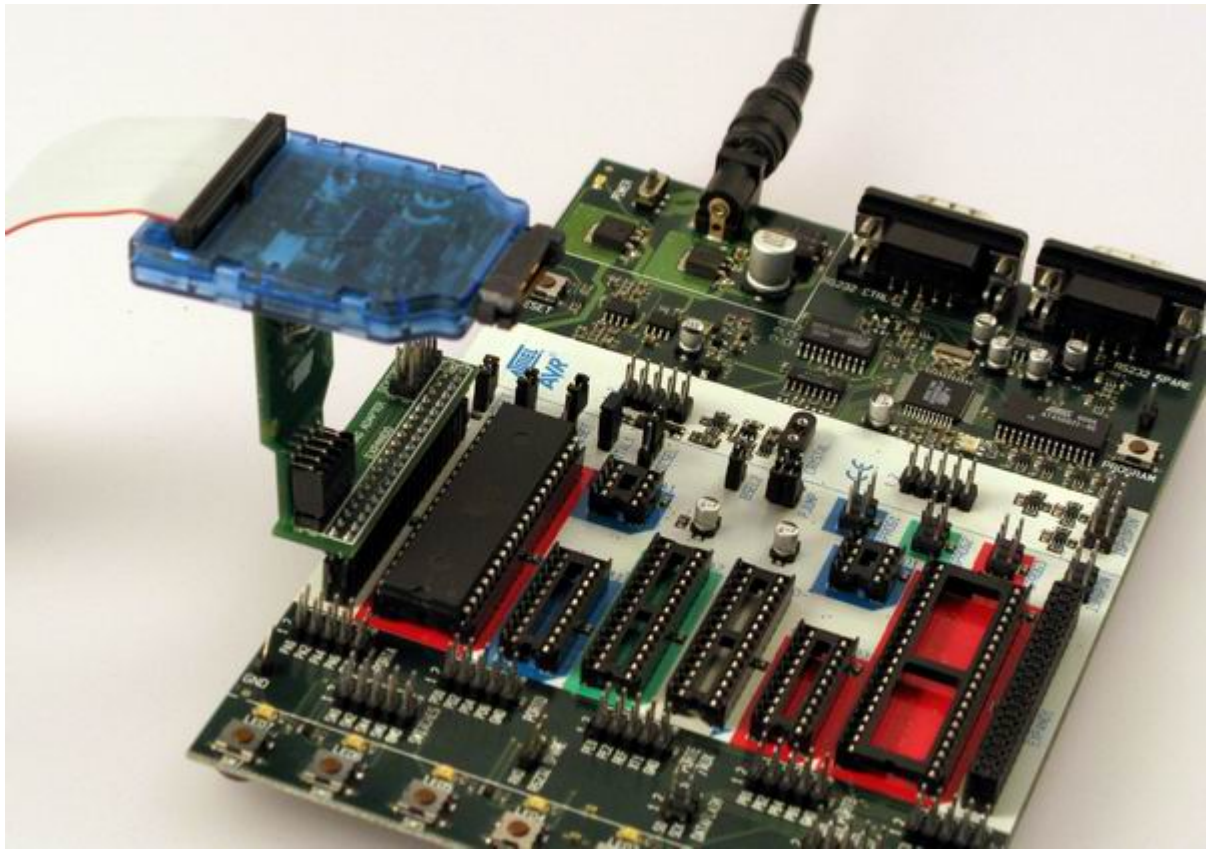
Note: The SPI interface is effectively disabled when the debugWIRE enable fuse (DWEN) is programmed, even if the SPIEN fuse is also programmed. To re-enable the SPI interface, the 'disable debugWIRE' command must be issued while in a debugWIRE debugging session. Disabling debugWIRE in this manner requires that the SPIEN fuse is already programmed. If Atmel Studio fails to disable debugWIRE, it is probably that the SPIEN fuse is NOT programmed. If this is the case, it is necessary to use a high-voltage programming interface to program the SPIEN fuse. It is HIGHLY ADVISED to simply let Atmel Studio handle setting and clearing of the DWEN fuse!



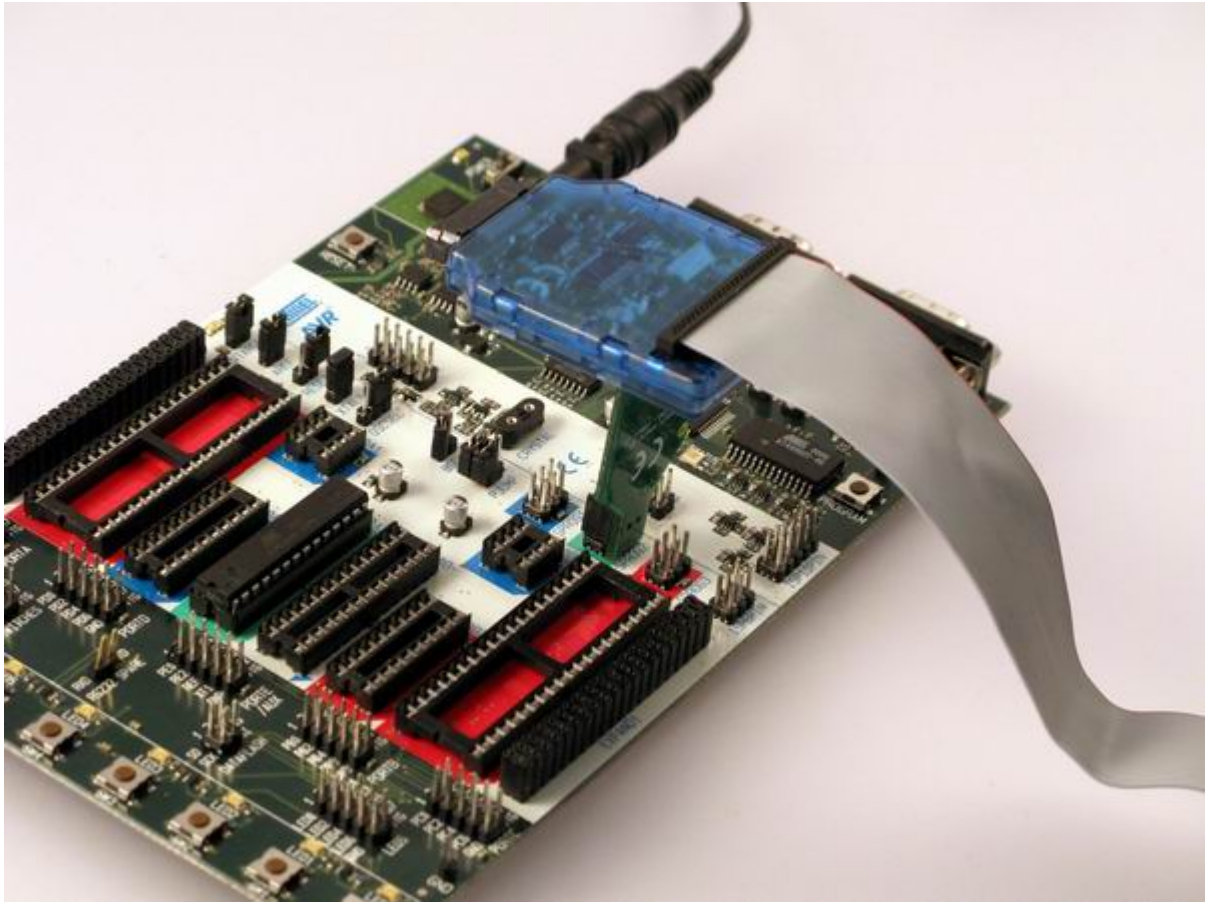
5.6. Using the Atmel AVR ONE! with Atmel STK500

The Atmel STK[®]500 starter kit can be used to house Atmel AVR devices to which the AVR ONE! can connect through JTAG, debugWIRE, and SPI interfaces.

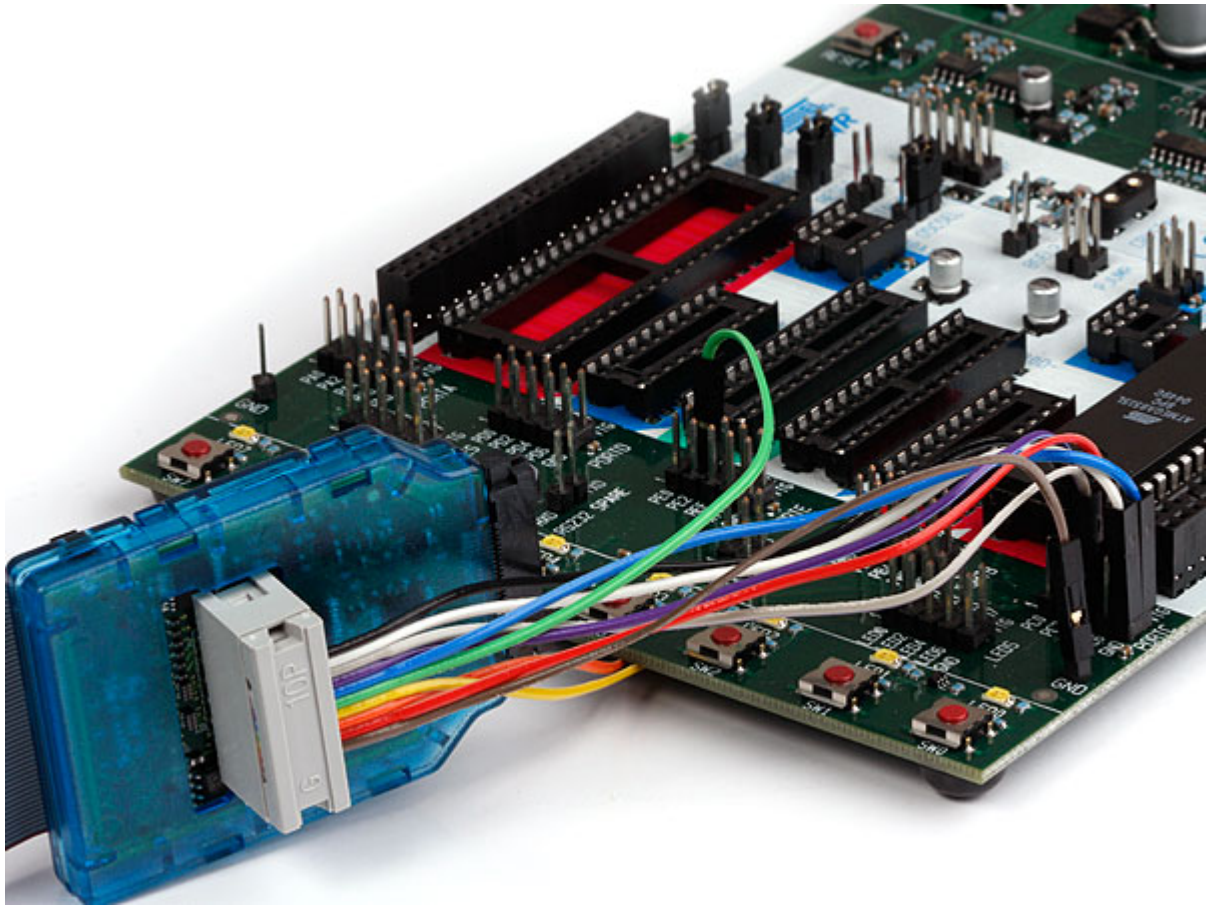
When connecting to a JTAG target, simply use the ATSTK500_JTAG_ADAPTER. If you do not have an STK500 JTAG adapter available, the 10-pin multicolored "squid" cable can also be used to connect directly to the device's JTAG port on PORTC[5::2] of the STK500.



Connecting to debugWIRE and SPI targets is done using the same stand-off adapter. When using the debugWIRE interface, be sure to remove the STK500's RESET jumper to allow the reset line to be driven as required.

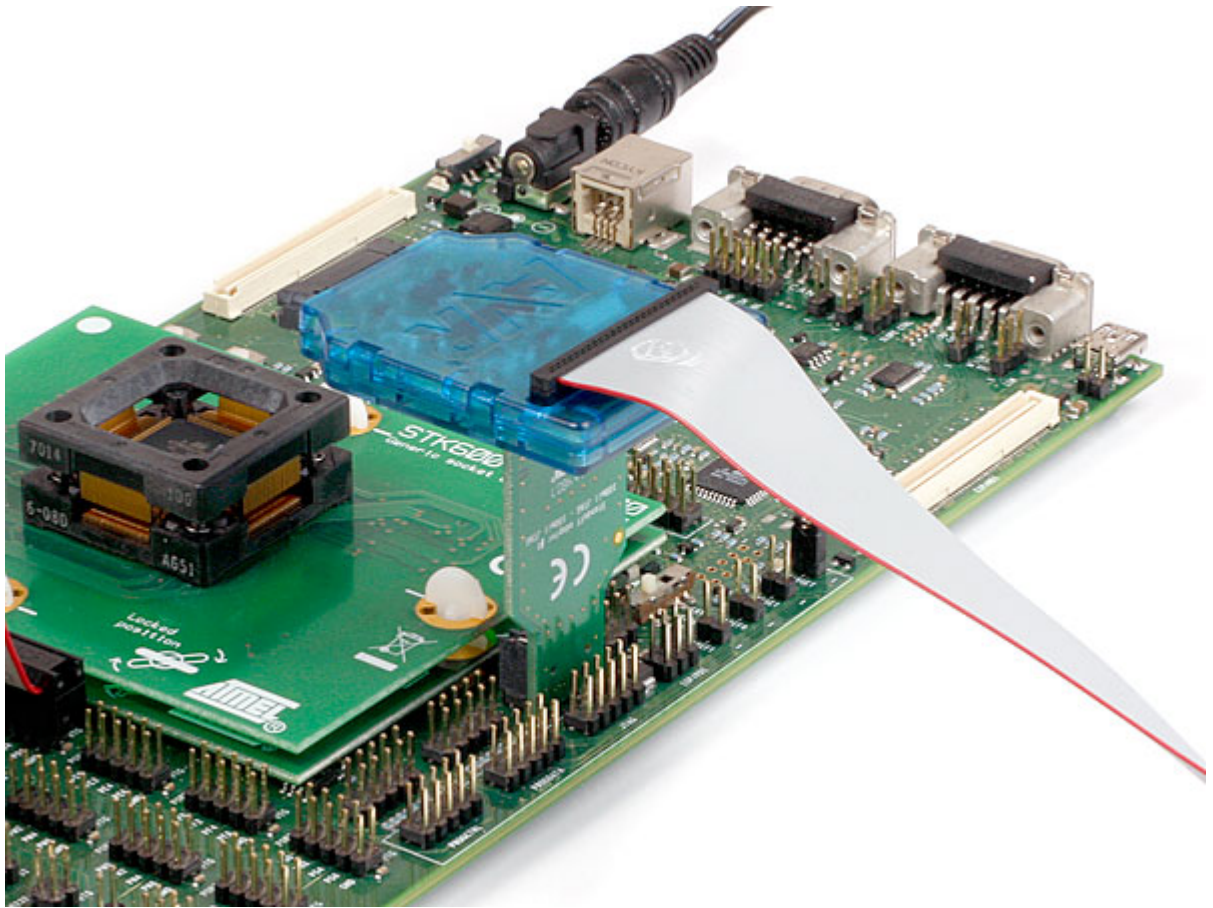


Alternatively, the AVR ONE! can be connected to any target interface using the 10-pin "squid" cable (provided).

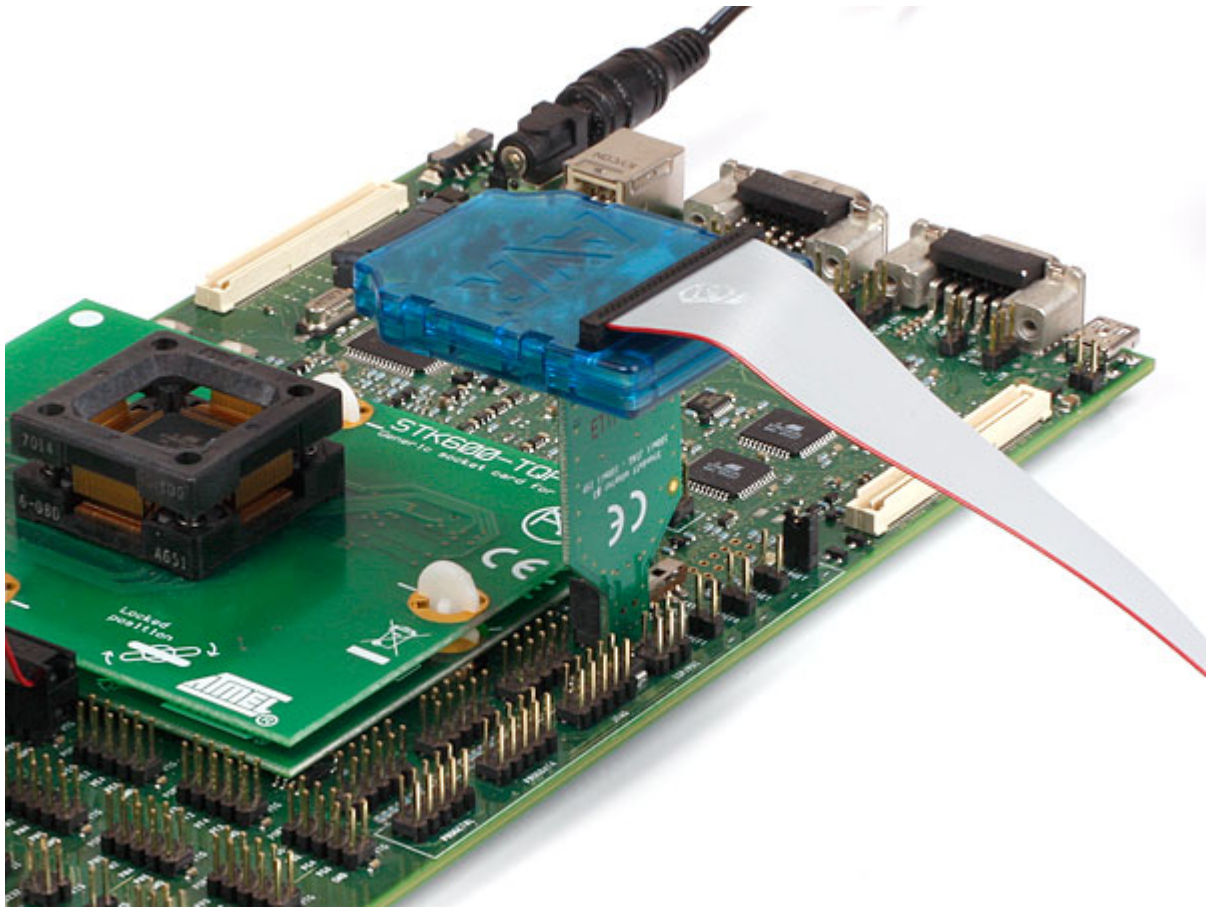


5.7. Using the Atmel AVR ONE! with Atmel STK600

The Atmel STK600 starter kit can be used to house Atmel AVR devices to which the AVR ONE! can connect through the JTAG interface.



When connecting to a JTAG target, simply use the 10-pin 100mil stand-off adapter (provided) to connect to the JTAG connector on the STK600.



When connecting to a PDI, debugWIRE, or SPI target, simply use the 6-pin 100mil stand-off adapter (provided) to connect to the SPI/PDI connector.

6. On-chip Debugging

6.1. Introduction to On-chip Debugging (OCD)

A traditional *Emulator* is a tool which tries to imitate the exact behavior of a target device. The closer this behavior is to the actual device's behavior, the better the emulation will be.

The Atmel AVR ONE! is not a traditional *Emulator*. Instead, the AVR ONE! interfaces with the internal On-chip Debug system inside the target Atmel AVR device, providing a mechanism for monitoring and controlling its execution. In this way the application being debugged is not *emulated*, but actually executed on the real AVR target device.

With an OCD system, the application can be executed whilst maintaining exact electrical and timing characteristics in the target system – something not technically realizable with a traditional *emulator*.

Run Mode

When in Run mode, the execution of code is completely independent of the AVR ONE!. The AVR ONE! will continuously monitor the target AVR to see if a break condition has occurred. When this happens the OCD system will interrogate the device through its debug interface, allowing the user to view the internal state of the device.

Stopped Mode

When a breakpoint is reached, program execution is halted, but all I/O will continue to run as if no breakpoint had occurred. For example, assume that a USART transmit has just been initiated when a breakpoint is reached. In this case the USART continues to run at full speed completing the transmission, even though the core is in stopped mode.

Hardware Breakpoints

The AVR OCD module contains a number of program counter comparators implemented in hardware. When the program counter matches the value stored in one of the comparator registers, the OCD enters stopped mode. Since hardware breakpoints require dedicated hardware on the OCD module, the number of breakpoints available depends upon the size of the OCD module implemented on the AVR target. Usually one such hardware comparator is 'reserved' by the debugger for internal use. For more information on the hardware breakpoints available in the various OCD modules, see the [OCD implementations](#) section.

Software Breakpoints

A software breakpoint is a BREAK instruction placed in program memory on the target device. When this instruction is loaded, program execution will break and the OCD enters stopped mode. To continue execution a "start" command has to be given from the OCD. Not all AVR devices have OCD modules supporting the BREAK instruction. For more information on the software breakpoints available in the various OCD modules, see the [OCD implementations](#) section.

For further information on the considerations and restrictions when using an OCD system, see the [Special Considerations](#) section.

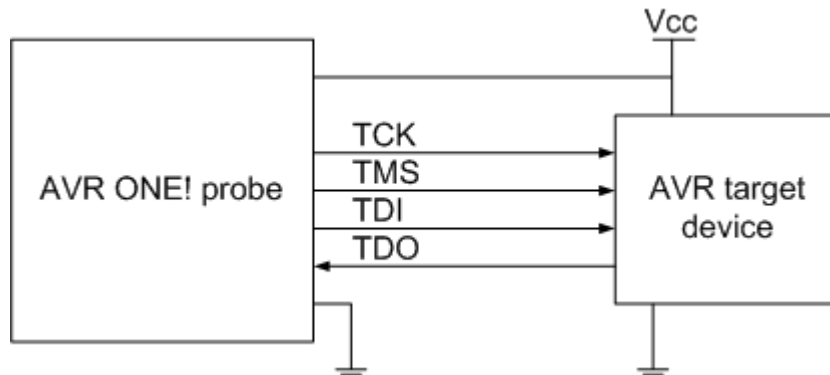
6.2. Physical Interfaces

The Atmel AVR ONE! supports several hardware interfaces as described in the sections that follow.

6.2.1. JTAG

The JTAG interface consists of a 4-wire Test Access Port (TAP) controller that is compliant with the IEEE® 1149.1 standard. The IEEE standard was developed to provide an industry-standard way to efficiently test circuit board connectivity (Boundary Scan). Atmel AVR devices have extended this functionality to include full Programming and On-Chip Debugging support.

Figure 6-1. JTAG Interface Basics



When designing an application PCB which includes an AVR with the JTAG interface, it is recommended to use the pinout as shown in [Figure 6-2 JTAG Header Pinout](#). The Atmel AVR ONE! ships with both 100-mil and 50-mil adapters supporting this pinout.

Figure 6-2. JTAG Header Pinout

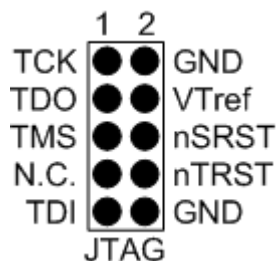


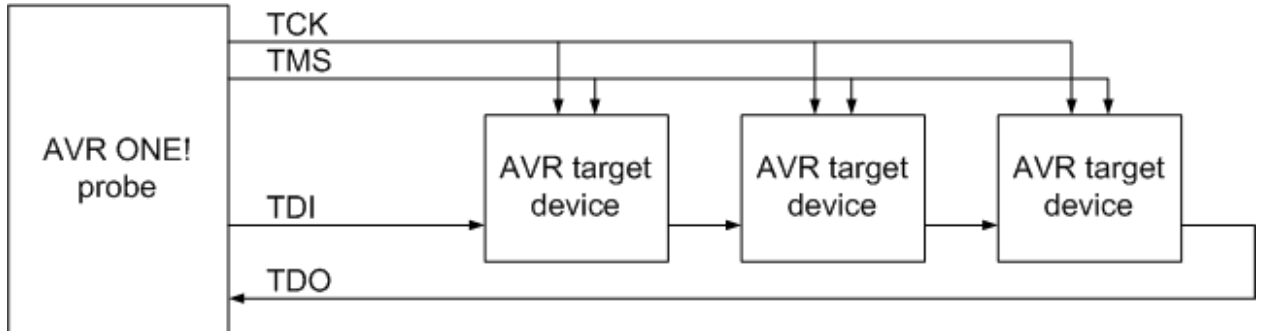
Table 6-1. JTAG Pin Description

Name	Pin	Description
TCK	1	Test Clock (clock signal from the AVR ONE! into the target device)
TMS	5	Test Mode Select (control signal from the AVR ONE! into the target device)
TDI	9	Test Data In (data transmitted from the AVR ONE! into the target device)
TDO	3	Test Data Out (data transmitted from the target device into the AVR ONE!)
nTRST	8	Test Reset (optional, only on some AVR devices). Used to reset the JTAG TAP controller.
nSRST	6	Source Reset (optional) Used to reset the target device. Connecting this pin is recommended since it allows the AVR ONE! to hold the target device in a reset state, which can be essential to debugging in certain scenarios.
VTref	4	Target voltage reference. The AVR ONE! samples the target voltage on this pin in order to power the level converters correctly. The AVR ONE! draws less than 1mA from this pin.
GND	2, 10	Ground. Both must be connected to ensure that the AVR ONE! and the target device share the same ground reference.

Tip: remember to include a decoupling capacitor between pin 4 and GND.

The JTAG interface allows for several devices to be connected to a single interface in a daisy-chain configuration. The target devices must all be powered by the same supply voltage, share a common ground node, and must be connected as shown in [Figure 6-3 JTAG Daisy-chain](#).

Figure 6-3. JTAG Daisy-chain



When connecting devices in a daisy-chain, the following points must be considered:

- All devices must share a common ground, connected to GND on the AVR ONE! probe
- All devices must be operating on the same target voltage. VTref on the AVR ONE! probe must be connected to this voltage.
- TMS and TCK are connected in parallel; TDI and TDO are connected in a serial chain
- NSRST on the AVR ONE! probe must be connected to RESET on the devices if any one of the devices in the chain disables its JTAG port
- "Devices before" refers to the number of JTAG devices that the TDI signal has to pass through in the daisy chain before reaching the target device. Similarly "devices after" is the number of devices that the signal has to pass through after the target device before reaching the AVR ONE! TDO pin.
- "Instruction bits before" and "after" refers to the total sum of all JTAG devices' instruction register lengths which are connected before and after the target device in the daisy chain
- The total IR length (instruction bits before + Atmel AVR IR length + instruction bits after) is limited to a maximum of 240 bits

Daisy chaining example: TDI -> ATmega1280 -> ATxmega128A1 -> ATUC3A0512 -> TDO.

In order to connect to the Atmel AVR XMEGA device, the daisy chain settings are:

Devices before: 1

Devices after: 1

Instruction bits before: 4 (AVR devices have 4 IR bits)

Instruction bits before: 5 (AVR UC3 devices have 5 IR bits)

6.2.2. Auxiliary (AUX) Physical (including JTAG)

When debugging Atmel AVR target devices that feature an auxiliary port, it is recommended to use the 38-pin connector, which provides access to both JTAG and AUX ports. The AUX port facilitates advanced debugging features such as program trace.

The pinout of the 38-pin connector is shown in [Figure 6-4 Mictor Connector Pinout](#) and listed in [Table 6-2 Mictor Connector Pinout](#).

The Mictor Connector is available from Tyco Electronics (part number 2-5767004-2).

Figure 6-4. Mictor Connector Pinout

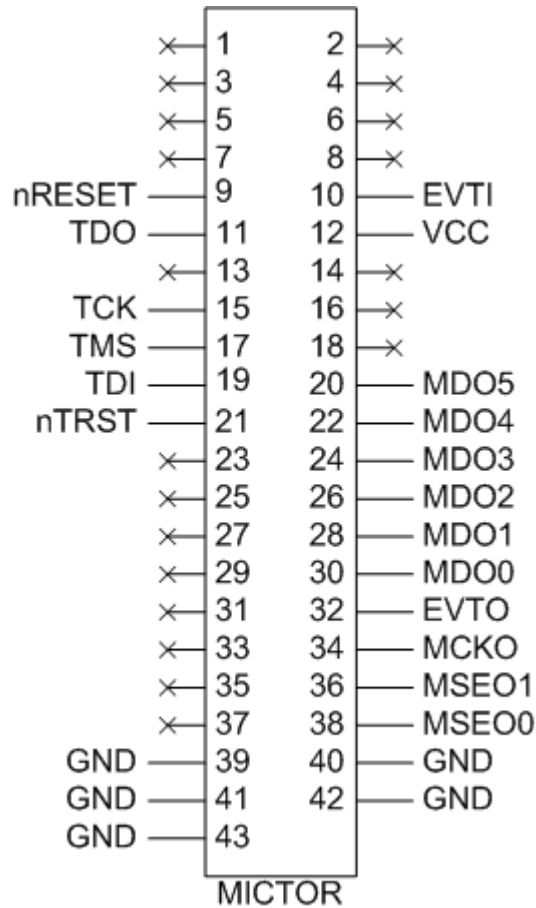


Table 6-2. Mictor Connector Pinout

Name	Pin	Description
TCK	15	Test Clock
TMS	17	Test Mode Select
TDI	19	Test Data In
TDO	11	Test Data Out
nTRST	21	Test Reset
nRESET	9	Source Reset
EVTI	10	Event In
EVTO	32	Event Out
MCKO	34	Message Clock Out
MSEO0	38	Message Start/End Out [0]
MSEO1	36	Message Start/End Out [1]
MDO0	30	Message Data Out [0]
MDO1	28	Message Data Out [1]

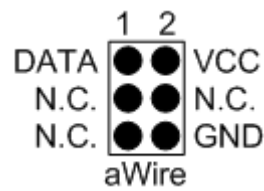
Name	Pin	Description
MDO2	26	Message Data Out [2]
MDO3	24	Message Data Out [3]
MDO4	22	Message Data Out [4]
MDO5	20	Message Data Out [5]
VREF	12	Target Voltage Reference
GND	39-43	Ground

6.2.3. aWire

The aWire interface makes use the /RESET wire of the Atmel AVR device to allow programming and debugging functions. A special enable sequence is transmitted by the Atmel AVR ONE!, which disables the default /RESET functionality of the pin.

When designing an application PCB, which includes an AVR with the aWire interface, it is recommended to use the pinout as shown in [Figure 6-5 aWire Header Pinout](#). The AVR ONE! ships with both 100-mil and 50-mil adapters supporting this pinout.

Figure 6-5. aWire Header Pinout



Tip: Since aWire is a half-duplex interface, a pull-up resistor on the /RESET line in the order of 47k is recommended to avoid false start-bit detection when changing direction.

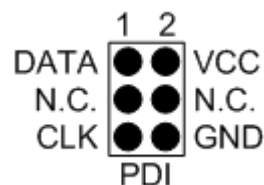
The aWire interface can be used as both a programming and debugging interface, all features of the OCD system available through the 10-pin JTAG interface can also be accessed using aWire.

6.2.4. PDI Physical

The Program and Debug Interface (PDI) is an Atmel proprietary interface for external programming and on-chip debugging of a device. PDI Physical is a 2-pin interface providing a bi-directional half-duplex synchronous communication with the target device.

When designing an application PCB, which includes an Atmel AVR with the PDI interface, the pinout shown in [Figure 6-6 PDI Header Pinout](#) should be used. The 6-pin standoff adapter provided with the AVR ONE! kit can then be used to connect the AVR ONE! probe to the application PCB.

Figure 6-6. PDI Header Pinout

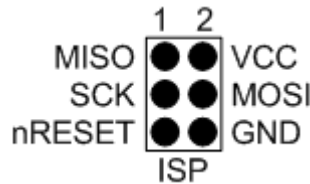


6.2.5. debugWIRE

The debugWIRE interface was developed by Atmel for use on low pin-count devices. Unlike the JTAG interface which uses four pins, debugWIRE makes use of just a single pin (RESET) for bi-directional half-duplex asynchronous communication with the debugger tool.

When designing an application PCB, which includes an Atmel AVR with the debugWIRE interface, the pinout shown in [Figure 6-7 debugWIRE \(SPI\) Header Pinout](#) should be used.

Figure 6-7. debugWIRE (SPI) Header Pinout



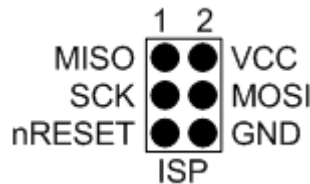
Note: The debugWIRE interface can not be used as a programming interface. This means that the SPI interface must also be available (as shown in [Figure 6-8 SPI Header Pinout](#)) in order to program the target.

When the debugWIRE enable (DWEN) fuse is programmed and lock-bits are un-programmed, the debugWIRE system within the target device is activated. The RESET pin is configured as a wire-AND (open-drain) bi-directional I/O pin with pull-up enabled and becomes the communication gateway between target and debugger.

6.2.6. SPI

In-System Programming uses the target Atmel AVR's internal SPI (Serial Peripheral Interface) to download code into the flash and EEPROM memories. It is not a debugging interface. When designing an application PCB, which includes an AVR with the SPI interface, the pinout shown in [Figure 6-8 SPI Header Pinout](#) should be used.

Figure 6-8. SPI Header Pinout



6.3. Atmel AVR OCD Implementations

6.3.1. Atmel AVR UC3 OCD (JTAG and aWire Physical)

The Atmel AVR UC3 OCD system is designed in accordance with the Nexus 2.0 standard (IEEE-ISTO 5001™-2003), which is a highly flexible and powerful open on-chip debug standard for 32-bit microcontrollers. It supports the following features:

- Nexus compliant debug solution
- OCD supports any CPU speed
- Six program counter hardware breakpoints
- Two data breakpoints
- Breakpoints can be configured as watchpoints
- Hardware breakpoints can be combined to give break on ranges

- Real-time program counter branch tracing, data trace, process trace

For special considerations regarding this debug interface, see [Special Considerations](#).

For more information regarding the UC3 OCD system, consult the [AVR32UC Technical Reference Manual](#), located on www.atmel.com/uc3.

6.3.2. Atmel AVR XMEGA OCD (JTAG and PDI Physical)

The Atmel AVR XMEGA OCD is otherwise known as PDI (Program and Debug Interface). Two physical interfaces (JTAG and PDI Physical) provide access to the same OCD implementation within the device. It supports the following features:

- Complete program flow control
- One dedicated program address comparator or symbolic breakpoint (reserved)
- Four hardware comparators
- Unlimited number of user program breakpoints (using BREAK)
- No limitation on system clock frequency

For special considerations regarding this debug interface, see [Special Considerations](#).

6.3.3. Atmel megaAVR OCD (JTAG)

The Atmel megaAVR OCD is based on the JTAG physical interface. It supports the following features:

- Complete program flow control
- Four program memory (hardware) breakpoints (one is reserved)
- Hardware breakpoints can be combined to form data breakpoints
- Unlimited number of program breakpoints (using BREAK) (except ATmega128[A])

For special considerations regarding this debug interface, see [Special Considerations](#).

6.3.4. Atmel megaAVR / tinyAVR OCD (debugWIRE)

The debugWIRE OCD is a specialized OCD module with a limited feature set specially designed for Atmel AVR devices with low pin-count. It supports the following features:

- Complete program flow control
- Unlimited number of User Program Breakpoints (using BREAK)
- Automatic baud configuration based on target clock

For special considerations regarding this debug interface, see [Special Considerations](#).

7. Atmel AVR ONE! Hardware Description




7.1. LEDs


The Atmel AVR ONE! front panel has four LEDs which indicate the status of current debug or programming sessions.

Figure 7-1. AVR ONE! LED Location



Table 7-1. LEDs

LED	Icon	Description
Main power		RED when main-board power is OK
Target power		GREEN when target power is OK. Flashing indicates a target power error. Does not light up unless a debug session connection is attempted.
Target run		GREEN when the target is running. ORANGE when target is stopped.

LED	Icon	Description
Communication		GREEN when a session with the host computer is active
Blue belt		The belt is lit up whenever the FPGA inside the debugger is loaded. It has no functional significance at the moment. Note that the belt will not lit up directly after an upgrade, since the FPGA won't be programmed until Atmel Studio connects to it.

7.2. Rear Panel

The rear panel of the Atmel AVR ONE! houses the DC jack, power switch, and USB connector. A sticker on the upper section shows the serial number and date of manufacture. When seeking technical support, include these details.

Figure 7-2. AVR ONE! Rear Panel



7.3. Probe

The Atmel AVR ONE! main unit comes with a probe with the following capabilities:

- 10-pin connector: Target voltage range, 1.65 - 5.5V; Interface frequencies, 32kHz - 33MHz.
- MICTOR38 connector: Target voltage range, 1.65 - 3.6V; AUX interface frequencies, up to 200MHz.

An alternative probe can be purchased separately. This probe looks the same as the original probe but has the following capabilities:

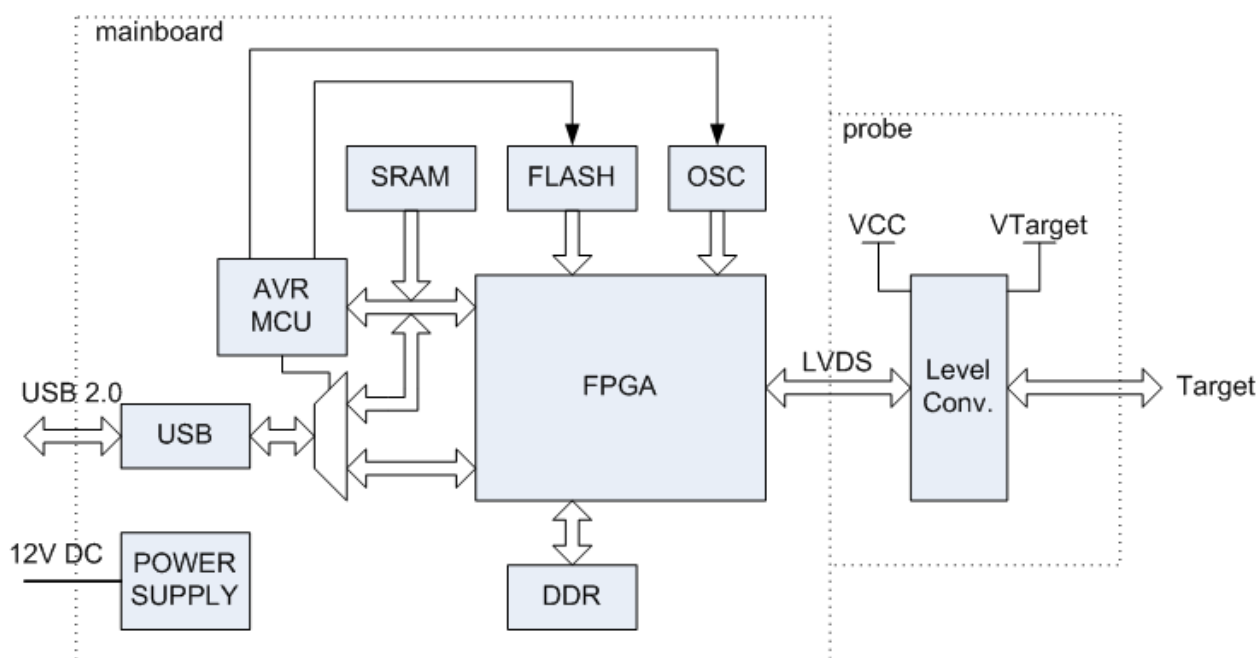
- 10-pin connector: Target voltage range, 1.65 - 5.5V; Interface frequencies, 32kHz - 33MHz.
- MICTOR38 connector: Target voltage range, 1.65 - 5.5V; AUX interface frequencies, up to 75MHz.

The AVR ONE! firmware will automatically detect which probe is connected and refuse to connect to a 5.5V target through the MICTOR38 connector when using the original 3.3V probe. It is worth noting that the JTAG 10-pin connector has the same capabilities on both probes. It is only the MICTOR38 connector that differs.

7.4. Architecture Description

The Atmel AVR ONE! architecture is shown in the [Figure 7-3 AVR ONE Block Diagram](#).

Figure 7-3. AVR ONE Block Diagram



7.4.1. Atmel AVR ONE! Main-board

Power is supplied via the 12V DC connector. The USB port is used for host communications only, and does not draw current from the host. At the heart of the Atmel AVR ONE! main-board is the ATmega1280 AVR microcontroller, which is coupled to an FPGA for target interface signal processing. The target interface is clocked by an external clock generator capable of providing a frequency in the range of approximately 1kHz to 64MHz. A small external SRAM is connected to the AVR MCU, and is used to store symbolic information during a debug session, while the larger and faster DDR-SDRAM is used by the FPGA as a trace message buffer only.

7.4.2. Atmel AVR ONE! Probe

Communication between the Atmel AVR ONE! main-board and the probe is done by LVDS signalling over the probe cable. The LVDS transceivers are connected to level translators that shift signals between the target's operating voltage and the internal voltage level on the AVR ONE!. The 'external' side of the level translators is connected physically to the target connector. The level translators are not powered directly from V_{Tref} , but from a buffered power source, which matches the V_{Tref} voltage. The JTAG channels can be

operated in the range 1.65V to 5.5V up to 33MHz, while the AUX channels are capable of operating at up to 200MHz at 3.6V.

For best results it is recommended to correctly terminate the high-speed AUX signals on the target application PCB.

For further information on how to connect the probe to the target application, see section [Connecting the AVR ONE!](#).

8. Software Integration

8.1. Atmel Studio

The Atmel AVR ONE! can be used in conjunction with Atmel Studio for programming and debugging of all Atmel AVR 8- and 32-bit microcontrollers.

For more information, consult the Atmel Studio user guide.



9. Command Line Utility

Atmel Studio comes with a command line utility called `atprogram` that can be used to program targets using the AVR ONE!. During the Atmel Studio installation a shortcut called "Atmel Studio 7.0. Command Prompt" were created in the Atmel folder on the Start menu. By double clicking this shortcut a command prompt will be opened and programming commands can be entered. The command line utility is installed in the Atmel Studio installation path in the folder `Atmel/Atmel Studio 7.0/atbackend/`.

To get more help on the command line utility type the command:

```
atprogram --help
```

10. Advanced Debugging Techniques

10.1. Atmel AVR 32-bit Microcontrollers

10.1.1. EVTI/EVTO Usage

The EVTI and EVTO pins provide features which can be used to ease debugging in certain scenarios. EVTI is used to signal an event INTO the target device, while EVTO is an event signaled OUT FROM the target device. The 38-pin MICTOR connector contains these two signals to allow the Atmel AVR ONE! access to these pins on the target device. EVTO can also be connected to pin 7 on the 10-pin JTAG connector for the AVR ONE! to be able to read it. Note that EVTO will only be available on the JTAG 10-pin connector if the target voltage is below 3.6 volts for probe hardware revision 0. In order to use these pins externally during debugging, it is recommended to disconnect them from the AVR ONE! MICTOR connector on the application board.

EVTI can be used for the following purposes:

- The target can be forced to stop execution in response to an external event. If the Event In Control (EIC) bits in the DC register are written to 0b01, high-to-low transition on the EVTI pin will generate a breakpoint condition. EVTI must remain low for one CPU clock cycle to guarantee that a breakpoint is triggered. The External Breakpoint bit (EXB) in DS is set when this occurs.
- Generating trace synchronization messages. This is a Nexus compliant feature which is not required by Atmel AVR 32-bit microcontrollers. A high-to-low transition on EVTI can be configured (using the EIC bits in DC) to generate trace synchronization messages. The AVR ONE! will ignore these messages since they are not required for trace reconstruction.

EVTO can be used for the following purposes:

- Indicating that the CPU has entered debug mode. Setting the EOS bits in DC to 0b01 causes the EVTO pin to be pulled low for one CPU clock cycle when the target device enters debug mode. This signal can be used as a trigger source for an external oscilloscope.
- Indicating that the CPU has reached a breakpoint or watchpoint. By setting the EOC bit in a corresponding Breakpoint/Watchpoint Control Register, breakpoint or watchpoint status is indicated on the EVTO pin. The EOS bits in DC must be set to 0xb10 to enable this feature. The EVTO pin can then be connected to an external oscilloscope in order to examine watchpoint timing.
- Generating trace timing signals. The AVR target can be configured to toggle every time a message is added to the trace transmit queue. This can be used to acquire more accurate timing information for trace output. This feature is currently not supported by the AVR ONE!

10.2. Atmel megaAVR Targets

10.2.1. I/O Debug Register (IDR)

The OCD debugger makes use of the memory mapped OCDR register to access the internals of the Atmel AVR target device while it is in stopped mode. When in run mode, the application running on the target can write a value to this register. The OCD system will then signal this to the debugger, which then fetches the data and passes it to the GUI front-end, where it is displayed. The application can thus give primitive debug messages to the debugger.

Note: The IDR value is polled at a fixed interval (100ms) so writing to it at a higher frequency than this will not yield reliable results.

Note: If the AVR target device experiences a power out condition while it is being debugged, spurious IDR messages may result. This because the debugger continues to pull the AVR as the voltage drops below its minimum operating voltage.

10.3. debugWIRE Targets

10.3.1. Software Breakpoints

The debugWIRE OCD is drastically scaled down when compared to the Mega (JTAG) OCD. This means that it does not have any program counter breakpoint comparators available to the user for debugging purposes. One such comparator does exist for purposes of Run-To-Cursor and single-step operations, but user breakpoints are not supported in the hardware.

Instead, the debugger must make use of the Atmel AVR BREAK instruction. This instruction can be placed in FLASH, and when it is loaded for execution it will cause the AVR CPU to enter stopped mode. To support breakpoints during debugging, the debugger must insert a BREAK instruction into FLASH at the point at which the users requests a breakpoint. The original instruction must be cached for later replacement. When single stepping over a BREAK instruction, the debugger has to execute the original cached instruction in order to preserve program behavior. In extreme cases, the BREAK has to be removed from FLASH and replaced later. All these scenarios can cause apparent delays when single stepping from breakpoints, which will be exacerbated when the target clock frequency is very low.

It is thus recommended to observe the following guidelines, where possible:

- Always run the target at as high a frequency as possible during debugging. The debugWIRE physical interface is clocked from the target clock.
- Try to minimize on the number of breakpoint additions and removals, as each one require a FLASH page to be replaced on the target
- Try to add or remove a small number of breakpoints at a time, to minimize the number of FLASH page write operations
- If possible, avoid placing breakpoints on double-word instructions

11. Special Considerations

11.1. Atmel AVR XMEGA OCD

OCD and clocking

When the MCU enters stopped mode, the OCD clock is used as MCU clock. The OCD clock is either the JTAG TCK if the JTAG interface is being used, or the PDI_CLK if the PDI interface is being used.

SDRAM refresh in stopped mode

When the OCD is in stopped mode, the MCU is clocked by the PDI or JTAG clock, as described in the paragraph above. Since nothing is known of this frequency by the debugger or OCD, a low refresh period (0x20) is automatically used. This value can't be changed by the user, and it might not fit with all combinations of OCD clock frequency and SDRAM devices. If SDRAM problems are observed in stopped mode, try to adjust the OCD clock frequency.

I/O modules in stopped mode

Different from earlier Atmel megaAVR devices, in XMEGA the I/O modules are stopped in stop mode. This means that USART transmissions will be interrupted, timers (and PWM) will be stopped.

Hardware breakpoints

There are four hardware breakpoint comparators - two address comparators and two value comparators. They have certain restrictions:

- All breakpoints must be of the same type (program or data)
- All data breakpoints must be in the same memory area (I/O, SRAM, or XRAM)
- There can only be one breakpoint if address range is used

Here are the different combinations that can be set:

- Two single data or program address breakpoints
- One data or program address range breakpoint
- Two single data address breakpoints with single value compare
- One data breakpoint with address range, value range or both

Atmel Studio will tell you if the breakpoint can't be set, and why. Data breakpoints have priority over program breakpoints, if software breakpoints are available.

External reset and PDI physical

The PDI physical interface uses the reset line as clock. While debugging, the reset pullup should be 10k or more or be removed. Any reset capacitors should be removed. Other external reset sources should be disconnected.

Debugging with sleep for ATxmegaA1 rev H and earlier

There was a bug in the early versions of the ATxmegaA1 family that prevented the OCD to be enabled while the device was in certain sleep modes. There are two methods to use to get back on the debugging:

- Enter the Atmel AVR ONE! options dialog and enable "Always activate external reset when reprogramming device"
- Perform a chip erase

The sleep modes that trigger this bug are:

- Power-down
- Power-save
- Standby
- Extended standby

The consequence of this bug is that attaching to a target using LiveDebug will not work when the target is in these sleep modes. The programming dialog is not affected by this bug.

11.2. Atmel megaAVR OCD and debugWIRE OCD

I/O Peripherals

Most I/O peripherals will continue to run even though the program execution is stopped by a breakpoint. Example: If a breakpoint is reached during a UART transmission, the transmission will be completed and corresponding bits set. The TXC (transmit complete) flag will be set and be available on the next single step of the code even though it normally would happen later in an actual device.

All I/O modules will continue to run in stopped mode with the following two exceptions:

- Timer/Counters (configurable using the software front-end)
- Watchdog Timer (always stopped to prevent resets during debugging)

Single Stepping I/O access

Since the I/O continues to run in stopped mode, care should be taken to avoid certain timing issues. For example, the code:

```
OUT PORTB, 0xAA
```

```
IN TEMP, PINB
```

When running this code normally, the TEMP register would not read back 0xAA because the data would not yet have been latched physically to the pin by the time it is sampled by the IN operation. A NOP instruction must be placed between the OUT and the IN instruction to ensure that the correct value is present in the PIN register.

However, when single stepping this function through the OCD, this code will always give 0xAA in the PIN register since the I/O is running at full speed even when the core is stopped during the single stepping.

Single stepping and timing

Certain registers need to be read or written within a given number of cycles after enabling a control signal. Since the I/O clock and peripherals continue to run at full speed in stopped mode, single stepping through such code will not meet the timing requirements. Between two single steps, the I/O clock may have run millions of cycles. To successfully read or write registers with such timing requirements, the whole read or write sequence should be performed as an atomic operation running the device at full speed. This can be done by using a macro or a function call to execute the code, or use the run-to-cursor function in the debugging environment.

Accessing 16-bit Registers

The Atmel AVR peripherals typically contain several 16-bit registers that can be accessed via the 8-bit data bus (e.g.: TCNTn of a 16-bit timer). The 16-bit register must be byte accessed using two read or write operations. Breaking in the middle of a 16-bit access or single stepping through this situation may result in erroneous values.

Restricted I/O register access

Certain registers cannot be read without affecting their contents. Such registers include those which contain flags, which are cleared by reading, or buffered data registers (e.g.: UDR). The software front-end will prevent reading these registers when in stopped mode to preserve the intended non-intrusive nature of OCD debugging. In addition, some registers cannot safely be written without side-effects occurring - these registers are read-only. For example:

- Flag registers, where a flag is cleared by writing '1' to any bit. These registers are read-only.
- UDR and SPDR registers cannot be read without affecting the state of the module. These registers are not accessible.

11.3. Atmel megaAVR OCD (JTAG)

Software breakpoints

Since it contains an early OCD module, ATmega128[A] does not support the use of the BREAK instruction for software breakpoints.

JTAG clock

The target clock frequency must be accurately specified in the software front-end before starting a debug session. For synchronization reasons, the JTAG TCK signal must be less than one fourth of the target clock frequency for reliable debugging. When programming via the JTAG interface, the TCK frequency is limited by the maximum frequency rating of the target device, and not the actual clock frequency being used.

When using the internal RC oscillator, be aware that the frequency may vary from device to device and is affected by temperature and V_{CC} changes. Be conservative when specifying the target clock frequency.

JTAGEN and OCDEN fuses

The JTAG interface is enabled using the JTAGEN fuse, which is programmed by default. This allows access to the JTAG programming interface. Through this mechanism, the OCDEN fuse can be programmed (by default OCDEN is un-programmed). This allows access to the OCD in order to facilitate debugging the device. The software front-end will always ensure that the OCDEN fuse is left un-programmed when terminating a session, thereby restricting unnecessary power consumption by the OCD module. If the JTAGEN fuse is unintentionally disabled, it can only be re-enabled using SPI or PP programming methods.

If the JTAGEN fuse is programmed, the JTAG interface can still be disabled in firmware by setting the JTD bit. This will render code un-debuggable, and should not be done when attempting a debug session. If such code is already executing on the Atmel AVR device when starting a debug session, the Atmel AVR ONE! will assert the /RESET line while connecting. If this line is wired correctly, it will force the target AVR device into reset, thereby allowing a JTAG connection.

If the JTAG interface is enabled, the JTAG pins cannot be used for alternative pin functions. They will remain dedicated JTAG pins until either the JTAG interface is disabled by setting the JTD bit from the program code, or by clearing the JTAGEN fuse through a programming interface.

IDR events

When the application program writes a byte of data to the OCDR register of the AVR device being debugged, the AVR ONE! reads this value out and displays it in the message window of the software front-end. The IDR registers are polled every 100ms, so writing to it at a higher frequency will NOT yield reliable results. When the AVR device loses power while it is being debugged, spurious IDR events may be reported. This happens because the AVR ONE! may still poll the device as the target voltage drops below the AVR's minimum operating voltage.

11.4. debugWIRE OCD

The debugWIRE communication pin (dW) is physically located on the same pin as the external reset (/RESET). An external reset source is therefore not supported when the debugWIRE interface is enabled.

The debugWIRE Enable fuse (DWEN) must be set on the target device in order for the debugWIRE interface to function. This fuse is by default un-programmed when the Atmel AVR device is shipped from the factory. The debugWIRE interface itself cannot be used to set this fuse. In order to set the DWEN fuse, SPI mode must be used. The software front-end handles this automatically provided that the necessary SPI pins are connected. It can also be set using SPI programming from the Atmel Studio programming dialog.

- Either:

Attempt to start a debug session on the debugWIRE part. If the debugWIRE interface is not enabled, Atmel Studio will offer to retry, or attempt to enable debugWIRE using SPI programming. If you have the full SPI header connected, debugWIRE will be enabled, and you will be asked to toggle power on the target - this is required for the fuse changes to be effective.

- or:

Open the programming dialog in SPI mode, and verify that the signature matches the correct device. Check the DWEN fuse to enable debugWIRE.

Note: It is important to leave the SPIEN fuse programmed and the RSTDISBL fuse un-programmed! Not doing this will render the device stuck in debugWIRE mode, and high-voltage programming will be required to revert the DWEN setting.

To disable the debugWIRE interface, use high-voltage programming to un-program the DWEN fuse. Alternately, use the debugWIRE interface itself to temporarily disable itself, which will allow SPI programming to take place, provided that the SPIEN fuse is set.

Note: If the SPIEN fuse was NOT left programmed, Atmel Studio will not be able to complete this operation, and high-voltage programming must be used.

- During a debug session, select the 'Disable debugWIRE and Close' menu option from the 'Debug' menu. DebugWIRE will be temporarily disabled, and Atmel Studio will use SPI programming to un-program the DWEN fuse.

Having the DWEN fuse programmed enables some parts of the clock system to be running in all sleep modes. This will increase the power consumption of the AVR while in sleep modes. The DWEN Fuse should therefore always be disabled when debugWIRE is not used.

When designing a target application PCB where debugWIRE will be used, the following considerations must be made for correct operation:

- Pull-up resistors on the dW(/RESET) line must not be smaller (stronger) than 10kΩ. The pull-up resistor is not required for debugWIRE functionality, since the debugger tool provides this.
- Connecting the /RESET pin directly to VCC will cause the debugWIRE interface to fail
- Any stabilizing capacitor connected to the /RESET pin must be disconnected when using debugWIRE, since they will interfere with correct operation of the interface
- All external reset sources or other active drivers on the /RESET line must be disconnected, since they may interfere with the correct operation of the interface

Never program the lock-bits on the target device. The debugWIRE interface requires that lock-bits are cleared in order to function correctly.

11.5. Atmel AVR UC3 OCD

On some Atmel AVR UC3 devices the JTAG port is not enabled by default. When using these devices it is essential to connect the RESET line so that the Atmel AVR ONE! can enable the JTAG interface.

Any stabilizing capacitor connected to the /RESET pin must be disconnected when using aWire since it will interfere with correct operation of the interface. A weak external pullup on this line is recommended.

The baud rate of aWire communications depends upon the frequency of the system clock, since data must be synchronized between these two domains. The AVR ONE! will automatically detect that the system clock has been lowered, and re-calibrate its baud rate accordingly. The automatic calibration only works down to a system clock frequency of 8kHz. Switching to a lower system clock during a debug session may cause contact with the target to be lost.

If required, the aWire baud rate can be restricted by setting the aWire clock parameter in the toolchain. Automatic detection will still work, but a ceiling value will be imposed on the results.

11.6. Atmel AVR UC3 Shutdown Mode

Some Atmel AVR UC3 devices, like UC3 L, supports a special sleep mode called Shutdown. When using this mode the device must be powered by 3.3V on VDDIN and then an internal regulator delivers 1.8V to the core. In addition this the 1.8V can be used to power the VDDIO. When the part goes into Shutdown mode it turns off the internal regulator resulting in both the core and most of the I/O to be powered down. To wake the part up again it must be reset by an external reset on the reset pin. The reset pin is powered by VDDIN while the JTAG lines are powered by VDDIO. For more information on the Shutdown mode, refer to the data sheet of the device being used. Some special considerations are required when debugging targets in such a configuration.

aWire

When debugging using aWire the reset line is the only signal line used. VTref (pin 4 in the Atmel AVR ONE! JTAG connector) must be connected to VDDIN (3.3V) for aWire to work.

JTAG

When debugging using JTAG it is important to also wire the reset line nSRST, pin 6 in the AVR ONE! JTAG connector, so that the AVR ONE! is able to wake the target up from the Shutdown mode. VTref, pin 4 in the AVR ONE! JTAG connector, must be connected to VDDIO (1.8V). The problem when using JTAG is that the reset pin on the target is powered by VDDIN @ 3.3V, while on the AVR ONE! probe it is powered by VTref @ 1.8 V. This configuration might sound dangerous to the hardware but the AVR ONE! only drives the reset line low, never high, so there should not be any contention. However, there is a strong pullup on the AVR ONE! probe pulling the reset line to 1.8V. This pullup is stronger than the internal pullup in the UC3 target so the reset line will stay at about 2V. To avoid that the target reads this as a low value on reset some action must be taken. There are two options, either add an external 3.3kΩ pullup on the reset line or lower VDDIN on the target to 2.2V or lower.

12. Troubleshooting

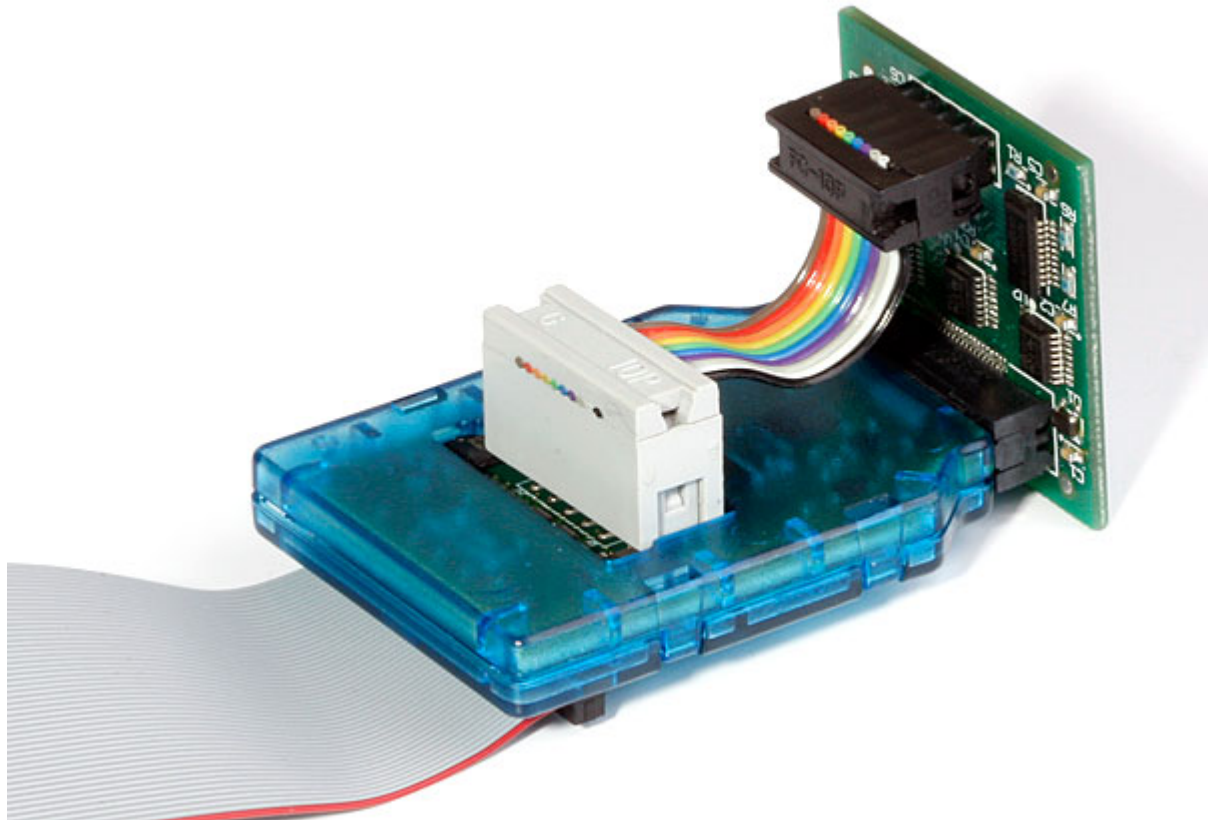
12.1. Self-test

The Atmel AVR ONE! is shipped with a self-test adapter with an IDC cable. If for any reason it is suspected that the AVR ONE! cable and probe are not functioning correctly, a self-test can be run. The self-test adapter is a loopback adapter that enables the on-board MCU to perform diagnostics on the target communication circuitry. This includes the probe, the cable, parts of the main board, and implicitly the test adapter itself.

12.1.1. Connecting

To run the test, the test adapter and cable must be connected to the probe as shown in [Figure 12-1 Connecting the AVR ONE! Test Adapter](#). The edge connector mates directly onto the Mictor38 socket on the test adapter. Press firmly to ensure a good connection. The male part of the test adapter cable is connected to the probe. Note that although the hole for the cable is keyed so that the connector will only insert fully in the correct orientation, it is still possible to insert it halfway if using the incorrect orientation. The female part of the cable mates with the pin-header on the test adapter. Make sure this is mated correctly.

Figure 12-1. Connecting the AVR ONE! Test Adapter



12.1.2. Launching

The self-test can either be launched from the command line (not yet implemented) or from Atmel Studio. The self-test is started from Atmel Studio by opening the Available Tools View (View > Available Atmel AVR Tools) and then right click the Atmel AVR ONE! that needs to be tested and select Self Test on the drop down menu.

12.1.3. How to use the Results for Diagnosis

If the self-test finds errors, it will tell you whether both the MICTOR38 connector and the 10-pin connector or only the MICTOR38 connector contains failing signals. Double-check your connections and retry the test if this happens. The connections may be loose, or in the case of the JTAG connector; it may be connected incorrectly onto the pin-header. If the error message persist, it is likely that there is a hardware problem somewhere. New probes and cables can be ordered separately from the Atmel Store (store.atmel.com).

The most likely source of hardware errors are after connecting any of the signals in the Mictor38 or 10-pin connector on the probe to voltages exceeding the maximum limits of the Atmel AVR ONE! probe hardware. The maximum ratings for the probe hardware are given in [Atmel AVR ONE! Features](#).

Note: The self-test can only give a hint of what is wrong. It cannot tell exactly which component is failing. The main purpose of the self-test is to clarify whether the hardware is working or not.

12.2. Troubleshooting Guide

Table 12-1. Troubleshooting Guide

Target type/ family	Problem	Possible causes	Solution
N/A	Power LED does not light up	DC supply voltage is insufficient or of incorrect polarity	Check that you are using the correct power supply provided with the kit. If you are using another power supply, ensure that it has a center-positive connector.
Atmel megaAVR and Atmel AVR XMEGA	JTAG debugging starts, then suddenly fails	The JTAG Disable bit in the MCUCSR register has been inadvertently written by the application	Hold reset low to regain control and change the code so that the JTAG Disable bit is not written
Atmel megaAVR and Atmel AVR XMEGA	After using the Atmel AVR ONE! to download code to the device, the emulator no longer works	<ol style="list-style-type: none">1. The JTAG ENABLE fuse has been disabled.2. The programming interface is still active. It is not possible to use both OCD and programming at the same time.	<ol style="list-style-type: none">1. Program the JTAG ENABLE fuse.2. Close the Programming interface, then enter emulation mode.

Target type/ family	Problem	Possible causes	Solution
Atmel megaAVR, Atmel AVR XMEGA, and Atmel tinyAVR	Atmel AVR ONE! is detected by Atmel Studio or other software front-end, but it will not connect to target device	JTAG: JTAG ENABLE Fuse is not programmed debugWIRE: DWEN Fuse is not programmed	JTAG: Use an other programming interface to program the JTAG ENABLE Fuse debugWIRE: Use an other programming interface to program the DWEN Fuse
Atmel megaAVR, Atmel AVR XMEGA, and Atmel AVR UC3	JTAG Debugging and programming is unstable, or does not work at all	For some target board configurations there might be some ringing on the interface lines	Add series resistors on the JTAG lines, especially TCK, but series resistors could be useful on TMS and TDI too. A value of about 68Ω will in most cases be suitable. Note that you might have to reduce the JTAG clock frequency after adding the series resistors.
N/A	Atmel Studio gives a message that no voltage is present	1. No power on target board. 2. Vtref not connected. 3. Target voltage too low.	1. Apply power to target board. 2. Make sure your JTAG connector includes the Vtref signal. 3. Make sure the target power supply is able to provide enough power.
Atmel megaAVR	OCD fuse is disabled, but using the Atmel AVR ONE!, OCD is still possible	The AVR ONE! will automatically program the OCD fuse if it is disabled	This is correct operation
Atmel megaAVR and Atmel tinyAVR	Some I/O registers are not updated correctly in Atmel Studio I/O view	When non-intrusive read back is not possible, the Atmel AVR ONE! will not update this location in the Atmel Studio I/O view	Read this I/O location into a temporary register, and view it there during debugging. See the chapter "Special Considerations" for information about which registers affected by this.
Atmel tinyAVR and Atmel megaAVR with debugWIRE support	SPI programming after a debugWIRE session is not possible	When the debugWIRE Interface is enabled the SPI Interface is disabled	Re-enable the SPI Interface as described in the section "Connecting to Target through the debugWIRE Interface". Use command line software to re-enable SPI interface.

Target type/ family	Problem	Possible causes	Solution
Atmel tinyAVR and Atmel megaAVR with debugWIRE support	Neither SPI nor debugWIRE connection works	The SPI and debugWIRE interfaces are disabled. DebugWIRE will not work if the lockbits are programmed.	Connect to target with High Voltage Programming. Enable SPI or debugWIRE and clear lockbits if using debugWIRE.
Atmel megaAVR, Atmel tinyAVR, Atmel AVR XMEGA, and Atmel AVR UC3	Error messages, or other strange behavior when using debugWIRE or JTAG	Target is running outside Safe Operation Area. Maximum frequency vs. V_{CC} .	Make sure the target is running within the Safe Operation Area as described in the chapter "Electrical Characteristics" in the datasheet for the actual part. Lower the frequency and/or increase the voltage.

13. Product Compliance

13.1. RoHS and WEEE

The AVR ONE! and all accessories are manufactured in accordance to both the RoHS Directive (2002/95/EC) and the WEEE Directive (2002/96/EC).

13.2. CE and FCC

The AVR ONE! unit has been tested in accordance to the essential requirements and other relevant provisions of Directives:

- Directive 2004/108/EC (class B)
- FCC part 15 subpart B
- 2002/95/EC (RoHS, WEEE)

The following standards are used for evaluation:

- EN 61000-6-1 (2007)
- EN 61000-6-3 (2007) + A1(2011)
- FCC CFR 47 Part 15 (2013)

The Technical Construction File is located at:

```
Atmel Norway  
Vestre Rosten 79  
7075 Tiller  
Norway
```

Every effort has been made to minimise electromagnetic emissions from this product. However, under certain conditions, the system (this product connected to a target application circuit) may emit individual electromagnetic component frequencies which exceed the maximum values allowed by the abovementioned standards. The frequency and magnitude of the emissions will be determined by several factors, including layout and routing of the target application with which the product is used.

14. Revision History

Doc Rev.	Date	Comments
32222A	06/2016	Initial document release.



Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, AVR®, AVR Studio®, megaAVR®, STK®, tinyAVR®, XMEGA®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. Windows® is a registered trademark of Microsoft Corporation in U.S. and other countries. Other terms and product names may be trademarks of others.

DISCLAIMER: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

SAFETY-CRITICAL, MILITARY, AND AUTOMOTIVE APPLICATIONS DISCLAIMER: Atmel products are not designed for and will not be used in connection with any applications where the failure of such products would reasonably be expected to result in significant personal injury or death ("Safety-Critical Applications") without an Atmel officer's specific written consent. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Atmel products are not designed nor intended for use in military or aerospace applications or environments unless specifically designated by Atmel as military-grade. Atmel products are not designed nor intended for use in automotive applications unless specifically designated by Atmel as automotive-grade.