EN: This Datasheet/Document is presented by the manufacturer.

Please visit our website for pricing and availability at www.hestore.hu.

USER GUIDE

Atmel-ICE

The Atmel-ICE Debugger



Atmel-ICE is a powerful development tool for debugging and programming ARM® Cortex®-M based Atmel® SAM and Atmel AVR® microcontrollers with On-Chip Debug capability.

It supports:

- Programming and on-chip debugging of all Atmel AVR 32-bit microcontrollers on both JTAG and aWire interfaces
- Programming and on-chip debugging of all Atmel AVR XMEGA® family devices on both JTAG and PDI 2-wire interfaces
- Programming (JTAG and SPI) and debugging of all Atmel AVR 8-bit microcontrollers with OCD support on either JTAG or debugWIRE interfaces
- Programming and debugging of all Atmel SAM ARM Cortex-M based microcontrollers on both SWD and JTAG interfaces
- Programming (TPI) of all Atmel tinyAVR® 8-bit microcontrollers with support for this interface

Consult the supported devices list in the Atmel Studio User Guide for a full list of devices and interfaces supported by this firmware release.

Table of Contents

The Atmel-ICE Debugger 1

1. Introduction 4
   - 1.1. Introduction to the Atmel-ICE 4
   - 1.2. Atmel-ICE Features 4
   - 1.3. System Requirements 4

2. Getting Started with the Atmel-ICE 6
   - 2.1. Full Kit Contents 6
   - 2.2. Basic Kit Contents 6
   - 2.3. PCBA Kit Contents 7
   - 2.4. Spare Parts Kits 7
   - 2.5. Kit Overview 8
   - 2.6. Assembling the Atmel-ICE 8
   - 2.7. Opening the Atmel-ICE 10
   - 2.8. Powering the Atmel-ICE 12
   - 2.9. Connecting to the Host Computer 12
   - 2.10. USB Driver Installation 12
     - 2.10.1. Windows 12

3. Connecting the Atmel-ICE 13
   - 3.1. Overview: Connecting to AVR and SAM Target Devices 13
   - 3.2. Connecting to a JTAG Target 13
   - 3.3. Connecting to an aWire Target 14
   - 3.4. Connecting to a PDI Target 15
   - 3.5. Connecting to a debugWIRE Target 15
   - 3.6. Connecting to a SPI Target 16
   - 3.7. Connecting to a TPI Target 17
   - 3.8. Connecting to a SWD Target 17

4. On-Chip Debugging 19
   - 4.1. Introduction to On-Chip Debugging (OCD) 19
   - 4.2. Physical Interfaces 19
     - 4.2.1. JTAG 19
     - 4.2.2. aWire 21
     - 4.2.3. PDI Physical 22
     - 4.2.4. debugWIRE 22
     - 4.2.5. SPI 22
     - 4.2.6. TPI 23
     - 4.2.7. SWD 23
   - 4.3. Atmel OCD Implementations 23
     - 4.3.1. Atmel AVR UC3 OCD (JTAG and aWire) 23
     - 4.3.2. Atmel AVR XMEGA OCD (JTAG and PDI Physical) 24
     - 4.3.3. Atmel megaAVR OCD (JTAG) 24
     - 4.3.4. Atmel megaAVR / tinyAVR OCD (debugWIRE) 24
     - 4.3.5. ARM Coresight Components 24

5. Hardware Description 25
   - 5.1. LEDs 25
   - 5.2. Rear Panel 25
   - 5.3. Bottom Panel 25
   - 5.4. Architecture Description 26
     - 5.4.1. Atmel-ICE Mainboard 26
     - 5.4.2. Atmel-ICE Target Connectors 27
     - 5.4.3. Atmel-ICE target Connectors Part Numbers 27

6. Software Integration 28
   - 6.1. Atmel Studio 28
     - 6.1.1. Software Integration in Atmel Studio 28
     - 6.1.2. Programming Options 28
     - 6.1.3. Debug Options 28

7. Command Line Utility 30

8. Advanced Debugging Techniques 31
8.1. Atmel AVR UC3 Targets 31
8.1.1. EVTI / EVTO Usage 31
8.2. debugWIRE Targets 31
8.2.1. Software Breakpoints 31

9. Special Considerations 32
9.1. Atmel AVR XMEGA OCD 32
9.2. Atmel megaAVR OCD and debugWIRE OCD 32
9.2.1. Atmel megaAVR OCD (JTAG) 33
9.2.2. debugWIRE OCD 34
9.3. Atmel AVR UC3 OCD 35
9.4. SAM / Coresight OCD 35

10. Firmware Upgrade 37

11. Release History and Known issues 38
11.1. What's New 38
11.2. Firmware Release History 38
11.2.1. Atmel Studio 6.2 38
11.2.2. Atmel Studio 6.2 (beta)2 38
11.3. Known Issues Concerning the Atmel-ICE 38
11.3.1. Atmel AVR XMEGA OCD Specific Issues 38
11.3.2. Atmel megaAVR OCD and Atmel tinyAVR OCD Specific Issues 38
11.4. Device Support 38

12. Product Compliance 39
12.1. RoHS and WEEE 39
12.2. CE and FCC 39

13. Document Revisions 40

1. Introduction

1.1 Introduction to the Atmel-ICE

Atmel-ICE is a powerful development tool for debugging and programming ARM Cortex-M based Atmel SAM and Atmel AVR microcontrollers with On-Chip Debug capability.

It supports:

- Programming and on-chip debugging of all Atmel AVR UC3 microcontrollers on both JTAG and aWire interfaces
- 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
- Programming and debugging of all Atmel SAM ARM Cortex-M based microcontrollers on both SWD and JTAG interfaces
- Programming (TPI) of all Atmel tinyAVR 8-bit microcontrollers with support for this interface

1.2 Atmel-ICE Features

- Fully compatible with Atmel Studio
- Supports programming and debugging of all Atmel AVR UC3 32-bit microcontrollers
- Supports programming and debugging of all 8-bit AVR XMEGA devices
- Supports programming and debugging of all 8-bit Atmel megaAVR® and tinyAVR devices with OCD
- Supports programming and debugging of all SAM ARM Cortex-M based microcontrollers
- Target operating voltage range of 1.62V to 5.5V
- Draws less than 3mA from target VTref when using debugWIRE interface and less than 1mA for all other interfaces
- Supports JTAG clock frequencies from 32kHz to 7.5MHz
- Supports PDI clock frequencies from 32kHz to 7.5MHz
- Supports debugWIRE baud rates from 4kbit/s to 0.5Mbit/s
- Supports aWire baud rates from 7.5kbit/s to 7Mbit/s
- Supports SPI clock frequencies from 8kHz to 5MHz
- Supports SWD clock frequencies from 32kHz to 2MHz
- USB 2.0 high-speed host interface
- ITM serial trace capture at up to 3MB/s
- Supports 10-pin 50-mil JTAG connector with both AVR and Cortex pinouts. The standard probe cable supports AVR 6-pin ISP/PDI/TPI 100-mil headers as well as 10-pin 50-mil. An adapter is available to support 6-pin 50-mil, 10-pin 100-mil and 20-pin 100-mil headers. Several kit options are available with different cabling and adapters.

1.3 System Requirements

The Atmel-ICE unit requires that a front-end debugging environment Atmel Studio version 6.2 or later is installed on your computer.

The Atmel-ICE should be connected to the host computer using the USB cable provided, or a certified USB-micro cable.

2. Getting Started with the Atmel-ICE

2.1 Full Kit Contents

The Atmel-ICE full kit contains these items:

- Atmel-ICE unit
- USB cable (1.8m, high-speed, micro-B)
- Adapter board containing 50-mil AVR, 100-mil AVR/SAM and 100-mil 20-pin SAM adapters
- IDC flat cable with 10-pin 50-mil connector and 6-pin 100-mil connector
- 50-mil 10-pin mini squid cable with 10 x 100-mil sockets

Figure 2-1. Atmel-ICE Full Kit Contents



2.2 Basic Kit Contents

The Atmel-ICE basic kit contains these items:

- Atmel-ICE unit
- USB cable (1.8m, high-speed, micro-B)
- IDC flat cable with 10-pin 50-mil connector and 6-pin 100-mil connector

Figure 2-2. Atmel-ICE Basic Kit Contents

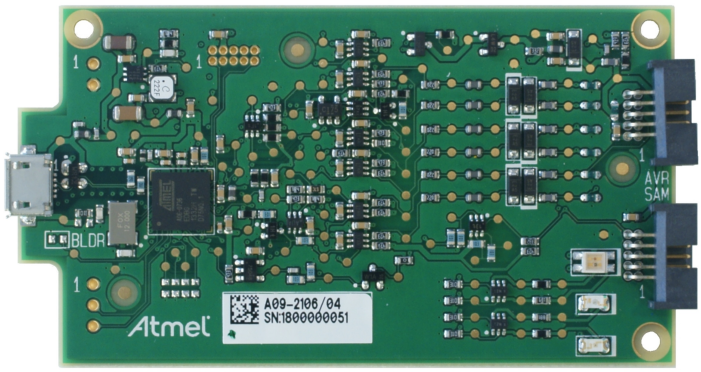


2.3 PCBA Kit Contents

The Atmel-ICE PCBA kit contains these items:

- Atmel-ICE unit without plastic encaptulation

Figure 2-3. Atmel-ICE PCBA Kit Contents



2.4 Spare Parts Kits

The following spare parts kits are available:

- Adapter kit
- Cable kit

Figure 2-4. Atmel-ICE Adapter Kit Contents



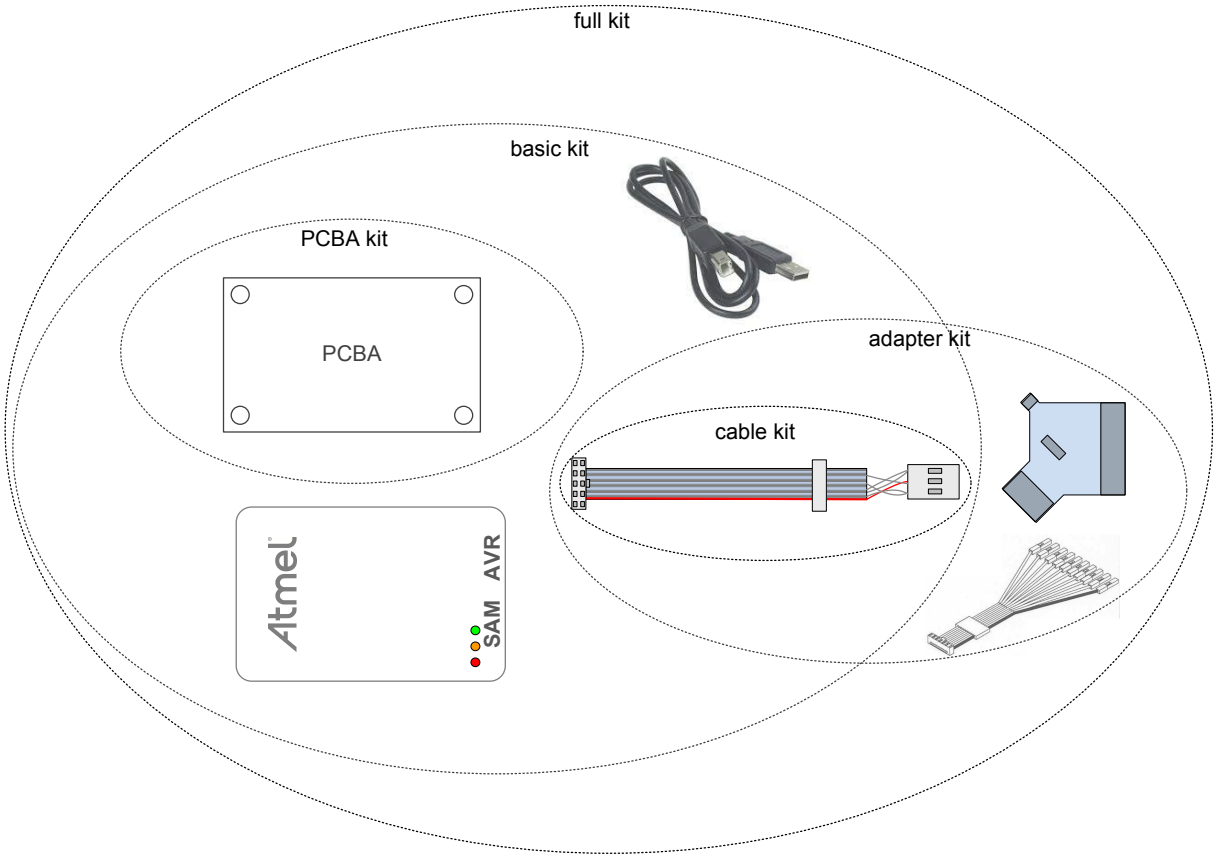
Figure 2-5. Atmel-ICE Cable Kit Contents



2.5 Kit Overview

The Atmel-ICE kit options are shown diagrammatically here:

Figure 2-6. Atmel-ICE Kit Overview



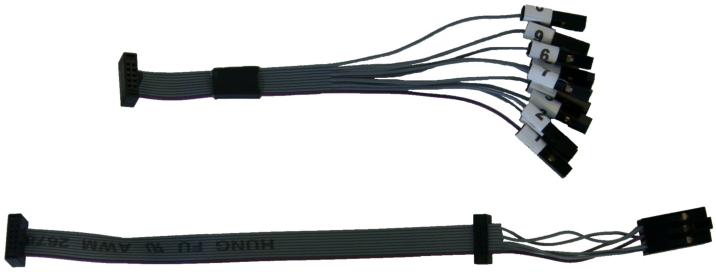
2.6 Assembling the Atmel-ICE

The Atmel-ICE unit is shipped with no cables attached. Two cable options are provided in the full kit:

- 50-mil 10-pin IDC flat cable with 6-pin ISP and 10-pin connectors

- 50-mil 10-pin mini-squid cable with 10 x 100-mil sockets

Figure 2-7. Atmel-ICE Cables



For most purposes, the 50-mil 10-pin IDC flat cable can be used, connecting either natively to its 10-pin or 6-pin connectors, or connecting via the adapter board. Three adapters are provided on one small PCBA. The following adapters are included:

- 100-mil 10-pin JTAG/SWD adapter
- 100-mil 20-pin SAM JTAG/SWD adapter
- 50-mil 6-pin SPI/debugWIRE/PDI/aWire adapter

Figure 2-8. Atmel-ICE Adapters



Note A 50-mil JTAG adapter has not been provided - this is because the 50-mil 10-pin IDC cable can be used to connect directly to a 50-mil JTAG header. For the part number of the component used for the 50-mil 10-pin connector, see “Atmel-ICE target Connectors Part Numbers” on page 27.

The 6-pin ISP/PDI header is included as part of the 10-pin IDC cable. This termination can be cut off if it is not required.

To assemble your Atmel-ICE into its default configuration, connect the 10-pin 50-mil IDC cable to the unit as shown below. Be sure to orient the cable so that the red wire (pin 1) on the cable aligns with the triangular indicator on the blue belt of the enclosure. The cable should connect upwards from the unit. Be sure to connect to the port corresponding to the pinout of your target - AVR or SAM.

Figure 2-9. Atmel-ICE Cable Connection



Figure 2-10. Atmel-ICE AVR Probe Connection

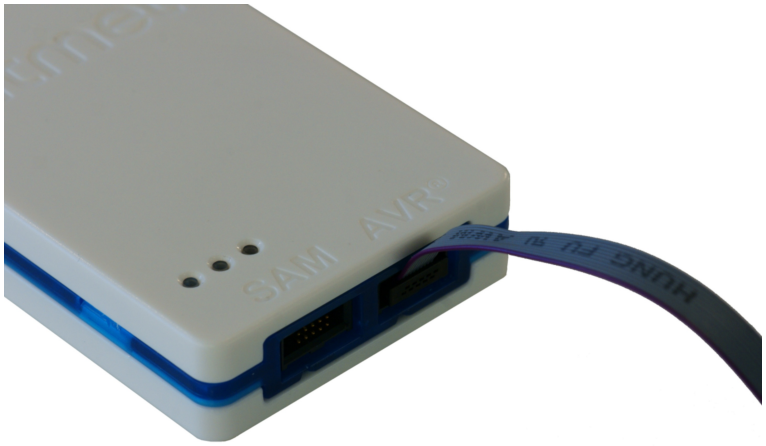
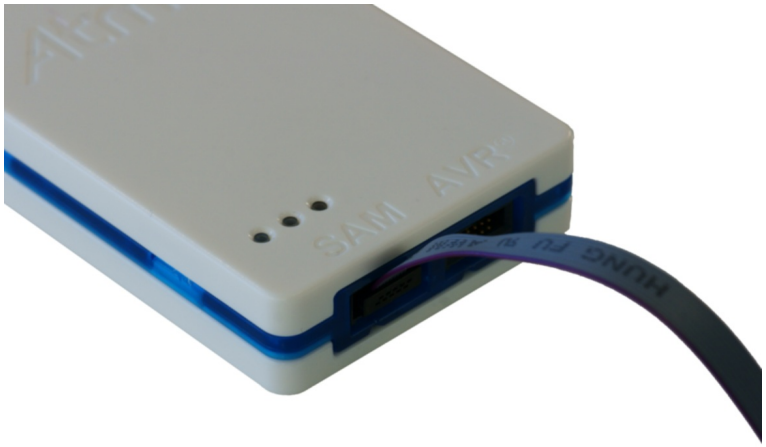


Figure 2-11. Atmel-ICE SAM Probe Connection



2.7 Opening the Atmel-ICE

Note For normal operation, the Atmel-ICE unit must not be opened. Opening the unit is done at your own risk. Anti-static precautions should be taken.

The Atmel-ICE enclosure consists of three separate plastic components - top cover, bottom cover and blue belt - which are snapped together during assembly. To open the unit, simply insert a large flat screwdriver into the openings in the blue belt, apply some inward pressure and twist gently. Repeat the process on the other snapper holes, and the top cover will pop off.

Figure 2-12. Opening the Atmel-ICE (1)



Figure 2-13. Opening the Atmel-ICE (2)



Figure 2-14. Opening the Atmel-ICE(3)



To close the unit again, simply align the top and bottom covers correctly, and press together firmly.

2.8 Powering the Atmel-ICE

The Atmel-ICE is powered by the USB bus voltage. It requires less than 100mA to operate, and can therefore be powered through a USB hub. The power LED will illuminate when the unit is plugged in. When not connected in an active programming or debugging session, the unit will enter low-power consumption mode to preserve your computer's battery. The Atmel-ICE cannot be powered down - it should be unplugged when not in use.

2.9 Connecting to the Host Computer

The Atmel-ICE communicates primarily using a standard HID interface, and does not require a special driver on the host computer. To use the advanced data gateway functionality of the Atmel-ICE, 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[1] for further information or to download the latest front-end software.

The Atmel-ICE must be connected to an available USB port on the host computer using the USB cable provided, or suitable USB certified micro cable. The Atmel-ICE contains a USB 2.0 compliant controller, and can operate in both full-speed and high-speed modes. For best results, connect the Atmel-ICE directly to a USB 2.0 compliant high-speed hub on the host computer using the cable provided.

2.10 USB Driver Installation

2.10.1 Windows

When installing the Atmel-ICE on a computer running Microsoft® Windows®, the USB driver is loaded when the Atmel-ICE is first plugged in.

Note Be sure to install the front-end software packages before plugging the unit in for the first time.

Once successfully installed, the Atmel-ICE will appear in the device manager as a "Human Interface Device".

[1] http://www.atmel.com/

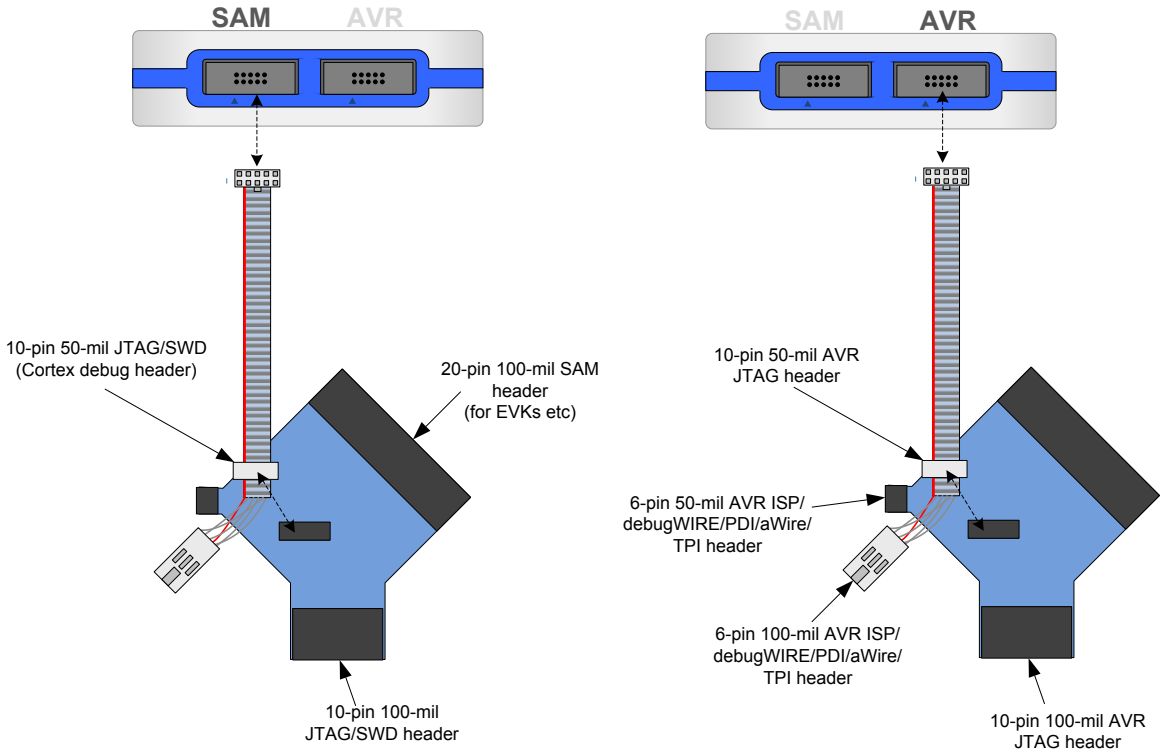
3. Connecting the Atmel-ICE

3.1 Overview: Connecting to AVR and SAM Target Devices

The Atmel-ICE probe has two 50-mil 10-pin JTAG connectors accessible on the front of the tool's enclosure. Both connectors are directly electrically connected, but conform to two different pinouts - the AVR JTAG header and the ARM Cortex Debug header. The connector should be selected based on the pinout of the target board, and not the target MCU type - for example a SAM device mounted in a AVR STK600 stack should use the AVR header.

Various cabling and adapters are available in the different Atmel-ICE kits. An overview of connection options is shown.

Figure 3-1. Atmel-ICE Connection Options



3.2 Connecting to a JTAG Target

The Atmel-ICE probe has two 50-mil 10-pin JTAG connectors accessible on the front of the tool's enclosure. Both connectors are directly electrically connected, but conform to two different pinouts - the AVR JTAG header and the ARM Cortex Debug header. The connector should be selected based on the pinout of the target board, and not the target MCU type - for example a SAM device mounted in a AVR STK600 stack should use the AVR header.

The recommended pinout for the 10-pin AVR JTAG connector is shown in Figure 4-2, “AVR JTAG Header Pinout” on page 20.

The recommended pinout for the 10-pin ARM Cortex Debug connector is shown in Figure 4-3, “SAM JTAG Header Pinout” on page 20.

Direct connection to a standard 10-pin 50-mil header

Use the 50-mil 10-pin flat cable (included in some kits) to connect directly to a board supporting this header type. Use the AVR connector port on the Atmel-ICE for headers layed out in the AVR pinout, and the SAM connector port for headers complying with the ARM Cortex Debug header pinout.

The pinouts for both 10-pin connector ports are shown below.

Connection to a standard 10-pin 100-mil header

Use a standard 50-mil to 100-mil adapter to connect to 100-mil headers. An adapter board (included in some kits) can be used for this purpose, or alternatively the JTAGICE3 adapter can be used for AVR targets.

Note The JTAGICE3 100-mil adapter cannot be used with the SAM connector port, since pins 2 and 10 (AVR GND) on the adapter are connected.

Connection to a custom 100-mil header

If your target board does not have a compliant 10-pin JTAG header in 50- or 100-mil, you can map to a custom pinout using the 10-pin "mini-squid" cable (included in some kits), which gives access to ten individual 100-mil sockets.

Connection to a 20-pin 100-mil header

Use the adapter board (included in some kits) to connect to targets with a 20-pin 100-mil header.

Table 3-1. Atmel-ICE JTAG Pin Description

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

3.3 Connecting to an aWire Target

The aWire interface only requires one data line in addition to Vcc and GND. On the target this line is the nRESET line, although the debugger uses the JTAG TDO line as the data line.

The recommended pinout for the 6-pin aWire connector is shown in Figure 4-5, “aWire Header Pinout” on page 22.

Connection to a 6-pin 100-mil aWire header

Use the 6-pin 100-mil tap on the flat cable (included in some kits) to connect to a standard 100-mil aWire header.

Connection to a 6-pin 50-mil aWire header

Use the adapter board (included in some kits) to connect to a standard 50-mil aWire header.

Connection to a custom 100-mil header

The 10-pin mini-squid cable should be used to connect between the Atmel-ICE AVR connector port and the target board. Three connections are required, as described in the table below.

Table 3-2. Atmel-ICE aWire Pin Mapping

Atmel-ICE AVR port pins	*Target pins*	*Mini-squid pin*	*aWire pinout*
Pin 1 (TCK)		1	
Pin 2 (GND)	GND	2	6

Atmel-ICE AVR port pins	Target pins	Mini-squid pin	aWire pinout
Pin 3 (TDO)	DATA	3	1
Pin 4 (VTG)	VTG	4	2
Pin 5 (TMS)		5	
Pin 6 (nSRST)		6	
Pin 7 (Not connected)		7	
Pin 8 (nTRST)		8	
Pin 9 (TDI)		9	
Pin 10 (GND)		0	

3.4 Connecting to a PDI Target

The recommended pinout for the 6-pin PDI connector is shown in Figure 4-6, “PDI Header Pinout” on page 22.

Connection to a 6-pin 100-mil PDI header

Use the 6-pin 100-mil tap on the flat cable (included in some kits) to connect to a standard 100-mil PDI header.

Connection to a 6-pin 50-mil PDI header

Use the adapter board (included in some kits) to connect to a standard 50-mil PDI header.

Connection to a custom 100-mil header

The 10-pin mini-squid cable should be used to connect between the Atmel-ICE AVR connector port and the target board. Four connections are required, as described in the table below.

Note

There is a difference from the JTAGICE mkII JTAG probe, where PDI_DATA is connected to pin 9. The Atmel-ICE is compatible with the pinout used by the JTAGICE3, AVR ONE! and AVR Dragon products.

Table 3-3. Atmel-ICE PDI Pin Mapping

Atmel-ICE AVR port pin	Target pins	Mini-squid pin	Atmel STK600 PDI pinout
Pin 1 (TCK)		1	
Pin 2 (GND)	GND	2	6
Pin 3 (TDO)	PDI_DATA	3	1
Pin 4 (VTG)	VTG	4	2
Pin 5 (TMS)		5	
Pin 6 (nSRST)	PDI_CLK	6	5
Pin 7 (Not connected)		7	
Pin 8 (nTRST)		8	
Pin 9 (TDI)		9	
Pin 10 (GND)		0	

3.5 Connecting to a debugWIRE Target

The recommended pinout for the 6-pin debugWIRE (SPI) connector is shown in Figure 4-7, “debugWIRE (SPI) Header Pinout” on page 22.

Connection to a 6-pin 100-mil SPI header

Use the 6-pin 100-mil tap on the flat cable (included in some kits) to connect to a standard 100-mil SPI header.

Connection to a 6-pin 50-mil SPI header

Use the adapter board (included in some kits) to connect to a standard 50-mil SPI header.

Connection to a custom 100-mil header

The 10-pin mini-squid cable should be used to connect between the Atmel-ICE AVR connector port and the target board. Three connections are required, as described in Table 3-4, “Atmel-ICE debugWIRE Pin Mapping” on page 16.

Although the debugWIRE interface only requires one signal line (RESET), Vcc 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.

Note 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 device 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 3-4. Atmel-ICE debugWIRE Pin Mapping

Atmel-ICE AVR port pin	*Target pins*	*Mini-squid pin*
Pin 1 (TCK)		1
Pin 2 (GND)	GND	2
Pin 3 (TDO)		3
Pin 4 (VTG)	VTG	4
Pin 5 (TMS)		5
Pin 6 (nSRST)	RESET	6
Pin 7 (Not connected)		7
Pin 8 (nTRST)		8
Pin 9 (TDI)		9
Pin 10 (GND)		0

3.6 Connecting to a SPI Target

The recommended pinout for the 6-pin SPI connector is shown in Figure 4-8, “SPI Header Pinout” on page 23.

Connection to a 6-pin 100-mil SPI header

Use the 6-pin 100-mil tap on the flat cable (included in some kits) to connect to a standard 100-mil SPI header.

Connection to a 6-pin 50-mil SPI header

Use the adapter board (included in some kits) to connect to a standard 50-mil SPI header.

Connection to a custom 100-mil header

The 10-pin mini-squid cable should be used to connect between the Atmel-ICE AVR connector port and the target board. Six connections are required, as described in the table below.

Note The SPI interface is effectively disabled when the debugWIRE enable fuse (DWEN) is programmed, even if 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 probable 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.

Table 3-5. Atmel-ICE SPI Pin Mapping

Atmel-ICE AVR port pins	*Target pins*	*Mini-squid pin*	*SPI pinout*
Pin 1 (TCK)	SCK	1	3
Pin 2 (GND)	GND	2	6
Pin 3 (TDO)	MISO	3	1
Pin 4 (VTG)	VTG	4	2
Pin 5 (TMS)		5	
Pin 6 (nSRST)	/RESET	6	5
Pin 7 (Not connected)		7	
Pin 8 (nTRST)		8	
Pin 9 (TDI)	MOSI	9	4
Pin 10 (GND)		0	

3.7 Connecting to a TPI Target

The recommended pinout for the 6-pin TPI connector is shown in Figure 4-9, “TPI Header Pinout” on page 23.

Connection to a 6-pin 100-mil TPI header

Use the 6-pin 100-mil tap on the flat cable (included in some kits) to connect to a standard 100-mil TPI header.

Connection to a 6-pin 50-mil TPI header

Use the adapter board (included in some kits) to connect to a standard 50-mil TPI header.

Connection to a custom 100-mil header

The 10-pin mini-squid cable should be used to connect between the Atmel-ICE AVR connector port and the target board. Six connections are required, as described in Table 3-6, “Atmel-ICE TPI Pin Mapping” on page 17.

Table 3-6. Atmel-ICE TPI Pin Mapping

Atmel-ICE AVR port pins	*Target pins*	*Mini-squid pin*	*TPI pinout*
Pin 1 (TCK)	CLOCK	1	3
Pin 2 (GND)	GND	2	6
Pin 3 (TDO)	DATA	3	1
Pin 4 (VTG)	VTG	4	2
Pin 5 (TMS)		5	
Pin 6 (nSRST)	/RESET	6	5
Pin 7 (Not connected)		7	
Pin 8 (nTRST)		8	
Pin 9 (TDI)		9	
Pin 10 (GND)		0	

3.8 Connecting to a SWD Target

The ARM SWD interface is s subset of the JTAG interface, making use of TCK and TMS pins, which means that when connecting to an SWD device, the 10-pin JTAG connector can technically be used. The ARM JTAG and AVR JTAG connectors are however not pin-compatible, so this depends upon the layout of the target board in use. When using STK600 or a board making use of the AVR JTAG pinout, the AVR connector port on the Atmel-ICE must be used. When connecting to a board which makes use of the ARM JTAG pinout, the SAM connector port on the Atmel-ICE must be used.

The recommended pinout for the 10-pin Cortex Debug connector is shown in Figure 4-10, “Recommended ARM SWD/JTAG Header Pinout” on page 23.

Connection to a 10-pin 50-mil Cortex header

Use the flat cable (included in some kits) to connect to a standard 50-mil Cortex header.

Connection to a 10-pin 100-mil Cortex-layout header

Use the adapter board (included in some kits) to connect to a 100-mil Cortex-pinout header.

Connection to a 20-pin 100-mil SAM header

Use the adapter board (included in some kits) to connect to a 20-pin 100-mil SAM header.

Connection to a custom 100-mil header

The 10-pin mini-squid cable should be used to connect between the Atmel-ICE AVR or SAM connector port and the target board. Six connections are required, as described in the table below.

Table 3-7. Atmel-ICE SWD Pin Mapping

Name	AVR port pin	SAM port pin	Description
SWDCLK	1	4	Serial Wire Debug Clock.
SWDIO	5	2	Serial Wire Debug Data Input/Output.
SWO	3	6	Serial Wire Output (optional- not implemented on all devices).
nSRST	6	10	Reset.
VTG	4	1	Target voltage reference.
GND	2, 10	3, 5	Ground.

4. On-Chip Debugging

4.1 Introduction to On-Chip Debugging (OCD)

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

The Atmel-ICE is not a traditional *Emulator*. Instead, the Atmel-ICE interfaces with the internal On-Chip Debug system inside the target 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 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 realisable with a traditional *emulator*.

Run Mode

When in Run mode, the execution of code is completely independent of the Atmel-ICE. The Atmel-ICE will continuously monitor the target device 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 target 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 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 "Atmel OCD Implementations" on page 23 .

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 "Atmel OCD Implementations" on page 23.

For further information on the considerations and restrictions when using an OCD system, see "Special Considerations" on page 32.

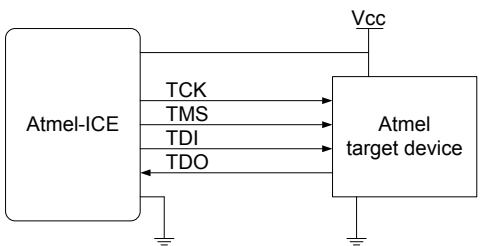
4.2 Physical Interfaces

The Atmel-ICE supports several hardware interfaces as described in the following sections.

4.2.1 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 and SAM devices have extended this functionality to include full Programming and On-Chip Debugging support.

Figure 4-1. JTAG Interface Basics



When designing an application PCB which includes an Atmel AVR with the JTAG interface, it is recommended to use the pinout as shown in Figure 4-2, “AVR JTAG Header Pinout” on page 20. The Atmel-ICE can connect to both 100-mil and 50-mil variants of this pinout.

Figure 4-2. AVR JTAG Header Pinout

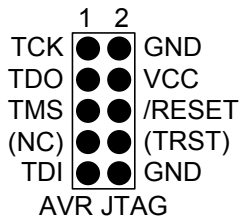


Table 4-1. AVR JTAG Pin Description

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

Tip

Remember to include a decoupling capacitor between pin 4 and GND.

When designing an application PCB which includes an Atmel SAM with the JTAG interface, it is recommended to use the pinout as shown in Figure 4-3, “SAM JTAG Header Pinout” on page 20. The Atmel-ICE can connect to both 100-mil and 50-mil variants of this pinout.

Figure 4-3. SAM JTAG Header Pinout

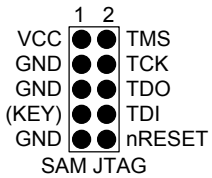


Table 4-2. SAM JTAG pin description

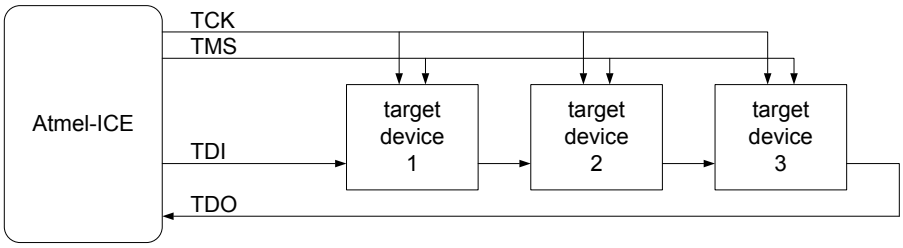
Name	Pin	Description
TCK	4	Test Clock (clock signal from the Atmel-ICE into the target device).
TMS	3	Test Mode Select (control signal from the Atmel-ICE into the target device).
TDI	8	Test Data In (data transmitted from the Atmel-ICE into the target device).
TDO	6	Test Data Out (data transmitted from the target device into the Atmel-ICE).
nRESET	10	Reset (optional) Used to reset the target device. Connecting this pin is recommended since it allows the Atmel-ICE to hold the target device in a reset state, which can be essential to debugging in certain scenarios.

Name	Pin	Description
VTG	1	Target voltage reference. The Atmel-ICE samples the target voltage on this pin in order to power the level converters correctly. The Atmel-ICE draws less than 3mA from this pin in debugWIRE mode and less than 1mA in other modes.
GND	3, 5, 9	Ground. All must be connected to ensure that the Atmel-ICE and the target device share the same ground reference.
KEY	7	Connected internally to TRST pin on the AVR connector. Recommended as not connected.

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 4-4, “JTAG Daisy-Chain” on page 21.

Figure 4-4. 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 Atmel-ICE probe
- All devices must be operating on the same target voltage. VTG on the Atmel-ICE 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 Atmel-ICE 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 Atmel-ICE TDO pin
- "Instruction bits before" and "after" refers to the sum total 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 256 bits. The number of devices in the chain is limited to 15 before and 15 after

Tip Daisy chaining example: TDI → ATmega1280 → ATxmega128A1 → ATUC3A0512 → TDO

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

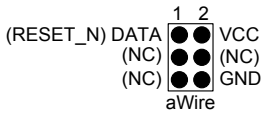
- Devices before: 1
- Devices after: 1
- Instruction bits before: 4 (8-bit AVR devices have 4 IR bits)
- Instruction bits after: 5 (32-bit AVR devices have 5 IR bits)

4.2.2 aWire

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

When designing an application PCB which includes an Atmel AVR with the aWire interface, it is recommended to use the pinout as shown in Figure 4-5, “aWire Header Pinout” on page 22. The Atmel-ICE ships with both 100-mil and 50-mil adapters supporting this pinout.

Figure 4-5. 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.

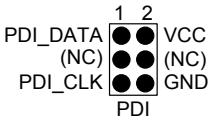
The aWire interface can be used as both a programming and debugging interface, all features of the OCD system available through the 10-pin JTAG interface can also be accessed using aWire.

4.2.3 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 4-6, “PDI Header Pinout” on page 22 should be used. One of the 6-pin adapters provided with the Atmel-ICE kit can then be used to connect the Atmel-ICE probe to the application PCB.

Figure 4-6. PDI Header Pinout

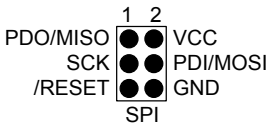


4.2.4 debugWIRE

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 4-7, “debugWIRE (SPI) Header Pinout” on page 22 should be used.

Figure 4-7. debugWIRE (SPI) Header Pinout



Note

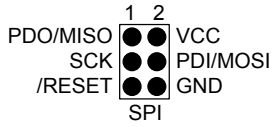
The debugWIRE interface can not be used as a programming interface. This means that the SPI interface must also be available (as shown in Figure 4-8, “SPI Header Pinout” on page 23) in order to program the target.

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.

4.2.5 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 4-8, "SPI Header Pinout" on page 23 should be used.

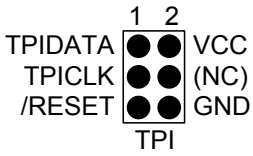
Figure 4-8. SPI Header Pinout



4.2.6 TPI

TPI is a programming-only interface for some AVR ATtiny devices. It is not a debugging interface, and these devices to not have OCD capability. When designing an application PCB which includes an AVR with the TPI interface, the pinout shown in Figure 4-9, "TPI Header Pinout" on page 23 should be used.

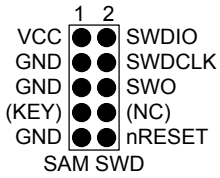
Figure 4-9. TPI Header Pinout



4.2.7 SWD

The ARM SWD interface is a subset of the JTAG interface, making use of TCK and TMS pins. The ARM JTAG and AVR JTAG connectors are however not pin-compatible, so when designing an application PCB which uses a SAM device with SWD or JTAG interface, it is recommended to use the ARM pinout shown in Figure 4-10, "Recommended ARM SWD/JTAG Header Pinout" on page 23. The SAM connector port on the Atmel-ICE can connect directly to this pinout.

Figure 4-10. Recommended ARM SWD/JTAG Header Pinout



The Atmel-ICE is capable of streaming UART-format ITM trace to the host computer. Trace is captured on the TRACE/SWO pin of the 10-pin header (JTAG TDO pin). Data is buffered internally on the Atmel-ICE and is sent over the HID interface to the host computer. The maximum reliable data rate is about 3MB/s.

4.3 Atmel OCD Implementations

4.3.1 Atmel AVR UC3 OCD (JTAG and aWire)

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

- Unlimited number of user program breakpoints (using BREAK)
- Real-time program counter branch tracing, data trace, process trace (not supported by Atmel-ICE)

For special considerations regarding this debug interface, see “Atmel AVR UC3 OCD” on page 35.

For more information regarding the UC3 OCD system, consult the AVR32UC Technical Reference Manuals, located on www.atmel.com/uc3[1].

4.3.2 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” on page 32.

4.3.3 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 “Atmel megaAVR OCD (JTAG)” on page 33.

4.3.4 Atmel megaAVR / tinyAVR OCD (debugWIRE)

The debugWIRE OCD is a specialised 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 “Atmel megaAVR OCD (JTAG)” on page 33.

4.3.5 ARM Coresight Components

Atmel ARM Cortex-M based microcontrollers implement Coresight™ compliant OCD components. The features of these components can vary from device to device. For further information consult the device's datasheet.

[1] http://www.atmel.com/uc3

5. Hardware Description

5.1 LEDs

The Atmel-ICE top panel has three LEDs which indicate the status of current debug or programming sessions.



Table 5-1. LEDs

LED	Function	Description
Left	Target power	GREEN when target power is OK. Flashing indicates a target power error. Does not light up until a programming/debugging session connection is started.
Middle	Main power	RED when main-board power is OK.
Right	Status	GREEN when the target is running. ORANGE when target is stopped.

5.2 Rear Panel

The rear panel of the Atmel-ICE houses the micro-B USB connector.



5.3 Bottom Panel

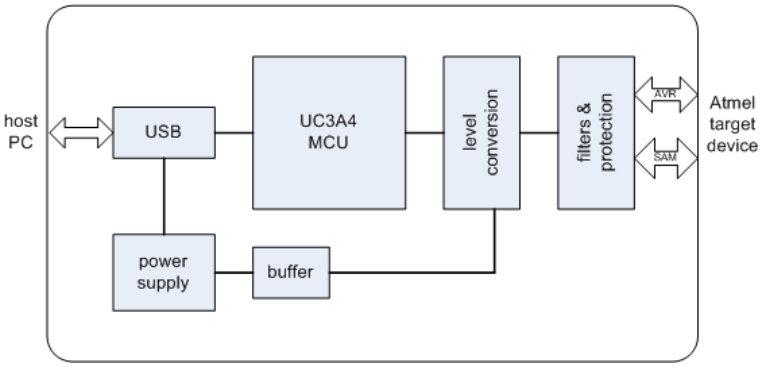
The bottom panel of the Atmel-ICE has a sticker which shows the serial number and date of manufacture. When seeking technical support, include these details.



5.4 Architecture Description

The Atmel-ICE architecture is shown in the block diagram in Figure 5-1, “Atmel-ICE block diagram” on page 26.

Figure 5-1. Atmel-ICE block diagram



5.4.1 Atmel-ICE Mainboard

Power is supplied to the Atmel-ICE from the USB bus, regulated to 3.3V by a step-down switchmode regulator. The VTG pin is used as a reference input only, and a separate power supply feeds the variable-voltage side of the on-board level converters At the heart of the Atmel-ICE mainboard is the Atmel AVR UC3 microcontroller AT32UC3A4256, which runs at between 1MHz and 60MHz depending on the tasks being processed. The microcontroller includes an on-chip USB 2.0 high-speed module, allowing high data throughput to and from the debugger.

Communication between the Atmel-ICE and the target device is done through a bank of level converters that shift signals between the target's operating voltage and the internal voltage level on the Atmel-ICE. Also in the signal path are zener overvoltage protection diodes, series termination resistors, inductive filters and ESD protection diodes. All signal channels can be operated in the range 1.62V to 5.5V, although the Atmel-ICE hardware can not drive out a higher voltage than 5.0V. Maximum operating frequency varies according to the target interface in use.

5.4.2 Atmel-ICE Target Connectors

The Atmel-ICE does not have an active probe. A 50-mil IDC cable is used to connect to the target application either directly, or through the adapters included in some kits. For more information on the cabling and adapters, see section Assembling the Atmel-ICE

5.4.3 Atmel-ICE target Connectors Part Numbers

In order to connect the Atmel-ICE 50-mil IDC cable directly to a target board, any standard 50-mil 10-pin header should suffice. It is advised to use keyed headers to ensure correct orientation when connecting to the target, such as those used on the adapter board included with the kit.

The part number for this header is: FTSH-105-01-L-DV-K-A-P from SAMTEC.

6. Software Integration

6.1 Atmel Studio

6.1.1 Software Integration in Atmel Studio

Atmel Studio is an Integrated Development Environment (IDE) for writing and debugging Atmel AVR and Atmel SAM applications in Windows environments. Atmel Studio provides a project management tool, source file editor, simulator, assembler and front-end for C/C++, programming, emulation and on-chip debugging.

Atmel Studio version 6.2 or later must be used in conjunction with the Atmel-ICE.

6.1.2 Programming Options

Atmel Studio supports programming of Atmel AVR and Atmel SAM ARM devices using the Atmel-ICE. The programming dialog can be configured to use JTAG, aWire, SPI, PDI, TPI or SWD modes, according to the target device selected.

When configuring the clock frequency, different rules apply for different interfaces and target families:

- SPI programming makes use of the target clock. Configure the clock frequency to be lower than one fourth the frequency at which the target device is currently running
- JTAG programming on Atmel megaAVR devices is clocked by the programmer. This means that the programming clock frequency is limited to the maximum operating frequency of the device itself. (Usually 16MHz)
- AVR XMEGA programming on both JTAG and PDI interfaces is clocked by the programmer. This means that the programming clock frequency is limited to the maximum operating frequency of the device itself. (Usually 32MHz)
- AVR UC3 programming on JTAG interface is clocked by the programmer. This means that the programming clock frequency is limited to the maximum operating frequency of the device itself. (Limited to 33MHz)
- AVR UC3 programming on aWire interface is clocked by the programmer. The optimal frequency is given by the SAB bus speed in the target device. The Atmel-ICE debugger will automatically tune the aWire baud rate to meet this criteria. Although it's usually not necessary the user can limit the maximum baud rate if needed (e.g. in noisy environments)
- SAM device programming on SWD interface is clocked by the programmer. The maximum frequency supported by Atmel-ICE is 2MHz. The frequency should not exceed the target CPU frequency times 10, $f = 10 \times f_{sysclk}$.

6.1.3 Debug Options

When debugging an Atmel AVR device using Atmel Studio, the 'Tool' tab in the project properties view contains some important configuration options. Those options which need further explanation are:

- **Target Clock Frequency**

  Accurately setting the target clock frequency is vital to achieve reliable debugging of Atmel megaAVR device over the JTAG interface. This setting should be less than one fourth of the lowest operating frequency of your AVR target device in the application being debugged. See “Atmel megaAVR OCD (JTAG)” on page 33 for more information.

  Debug sessions on debugWIRE target devices are clocked by the target device itself, and thus no frequency setting is required. The Atmel-ICE will automatically select the correct baud rate for communicating at the start of a debug session. However, if you are experiencing reliability problems related to a noisy debug environment, it is possible to force the debugWIRE speed to a fraction of its "recommended" setting.

  Debug sessions on AVR XMEGA target devices can be clocked at up to the maximum speed of the device itself (usually 32MHz).

  Debug sessions on AVR UC3 target devices over the JTAG interface can be clocked at up to the maximum speed of the device itself (limited to 33MHz). However the optimal frequency will be slightly below the current SAB clock on the target device.

Debug sessions on UC3 target devices over the aWire interface will be automatically tuned to the optimal baud rate by the Atmel-ICE itself. However, if you are experiencing reliability problems related to a noisy debug environment, it is possible to force the aWire speed below a configurable limit.

Debug sessions on SAM target devices over the SWD interface can be clocked at up to ten times the CPU clock (but limited to 2MHz max).

- **Preserve EEPROM**

  Select this option to avoid erasing the EEPROM during reprogramming of the target before a debug session.

- **Use external reset**

  If your target application disables the JTAG interface, the external reset must be pulled low during programming. Selecting this option avoids repeatedly being asked whether to use the external reset.

7. Command Line Utility

Atmel Studio comes with a command line utility called atprogram that can be used to program targets using the Atmel Atmel-ICE. During the Atmel Studio installation a shortcut called Atmel Studio 6.2 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 6.2/atbackend/.

To get more help on the command line utility type the command: **atprogram --help**.

8. Advanced Debugging Techniques

8.1 Atmel AVR UC3 Targets

8.1.1 EVTI / EVTO Usage

The EVTI and EVTO pins are not accessible on the Atmel-ICE. However, they can still be used in conjunction with other external equipment.

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 synchronisation messages. Not used by the Atmel-ICE

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. Not used by the Atmel-ICE

8.2 debugWIRE Targets

8.2.1 Software Breakpoints

The debugWIRE OCD is drastically scaled down when compared to the Atmel megaAVR (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 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 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 behaviour. 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 minimise 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 minimise the number of FLASH page write operations
- If possible, avoid placing breakpoints on double-word instructions

9. Special Considerations

9.1 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.

I/O modules in stopped mode

In contrast to 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 (IO, 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:

- Go into the Atmel-ICE Options in the Tools menu 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

9.2 Atmel megaAVR OCD and debugWIRE OCD

IO 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 will 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 (eg: 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 registeraccess

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 (eg: 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

9.2.1 Atmel megaAVR OCD (JTAG)

Software breakpoints

Since it contains an early version of the 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 synchronisation 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 VCC changes. Be conservative when specifying the target clock frequency.

See “Debug Options” on page 28 for details on how to set the target clock frequency using the software front-end.

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-ICE 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.

Note Be sure to check the "use external reset" checkbox in both the programming dialog and debug options dialog in order to allow the Atmel-ICE to assert the RESET line and re-enable the JTAG interface on devices which are running code which disables the JTAG interface by setting the JTD bit.

IDR events

When the application program writes a byte of data to the OCDR register of the AVR device being debugged, the Atmel-ICE reads this value out and displays it in the message window of the software front-end. The IDR register is polled every 50ms, 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 Atmel-ICE may still poll the device as the target voltage drops below the AVR’s minimum operating voltage.

9.2.2 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, the RSTDISBL fuse unprogrammed! 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 unprogram 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 unprogram 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, and may result in hardware damage to the Atmel-ICE
- Any stabilising 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.

9.3 Atmel AVR UC3 OCD

JTAG interface

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-ICE can enable the JTAG interface.

aWire interface

The baud rate of aWire communications depends upon the frequency of the system clock, since data must be synchronised between these two domains. The Atmel-ICE 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. Automatic detection will still work, but a ceiling value will be imposed on the results.

Any stabilising 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 (10kΩ or higher) on this line is recommended.

Shutdown sleep mode

Some AVR UC3 devices have an internal regulator that can be used in 3.3V supply mode with 1.8V regulated I/O lines. This means that the internal regulator powers both the core and most of the I/O. The Atmel-ICE does not support the Shutdown sleep mode were this regulator is shut off. In other words this sleep mode cannot be used during debugging. If it is a requirement to use this sleep mode during debugging, use an Atmel AVR ONE! debugger instead.

9.4 SAM / Coresight OCD

Some SAM devices include an ERASE pin which is asserted to perform a complete chip erase and unlock devices on which the security bit is set. This pin is NOT routed to any debug header, and thus the Atmel-ICE is unable to unlock a device. In such cases the user should perform the erase before starting a debug session.

JTAG interface

The RESET line should always be connected so that the Atmel-ICE can enable the JTAG interface.

SWD interface

The RESET line should always be connected so that the Atmel-ICE can enable the SWD interface.

10. Firmware Upgrade

For information on how to upgrade the firmware, see the Atmel Studio user guide in *Atmel Studio (USER GUIDE)*.

11. Release History and Known issues

11.1 What's New

Atmel-ICE is new!

11.2 Firmware Release History

11.2.1 Atmel Studio 6.2

Table 11-1. New in this Release

Release platform	Atmel Studio 6.2 (final)
Firmware version	1.13
New features	None
Fixes	• Fixed oscillator calibration command • Improved debugWIRE reliability

11.2.2 Atmel Studio 6.2 (beta)2

Table 11-2. New in this Release

Release platform	Atmel Studio 6.2 (beta)
Firmware version	1.09
New features	First release of Atmel-ICE
Fixes	N/A

11.3 Known Issues Concerning the Atmel-ICE

11.3.1 Atmel AVR XMEGA OCD Specific Issues

- For the ATxmegaA1 family, only revision G or later is supported

11.3.2 Atmel megaAVR OCD and Atmel tinyAVR OCD Specific Issues

- Cycling power on ATmega32U6 during a debug session may cause a loss of contact with the device

11.4 Device Support

For a full device support table for all Atmel Tools, see the "Supported Devices" in *Atmel Studio (USER GUIDE)*.

12. Product Compliance

12.1 RoHS and WEEE

The Atmel-ICE (all kits) and its accessories are manufactured in accordance to both the RoHS Directive (2002/95/EC) and the WEEE Directive (2002/96/EC).

12.2 CE and FCC

The Atmel-ICE 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.

13. Document Revisions

Document revision	Date	Comment
42330A	06/2014	Initial document for release.





Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**

© 2014 Atmel Corporation. / Rev.: 42330A-MCU-07/2014

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, AVR®, AVR Studio®, megaAVR®, tinyAVR®, XMEGA®, and others are registered trademarks or trademarks of Atmel Corporation in U.S. and other countries. ARM®, ARM Connected®, Cortex® logo and others are the registered trademarks or trademarks of ARM Ltd. Windows® is a registered trademark of Microsoft Corporation in the 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.