Programmers and Debuggers

AVR JTAGICE mkII

USER GUIDE

The Atmel AVR JTAGICE mkII Debugger

The Atmel® AVR® JTAGICE mkII supports On-Chip Debugging and programming on all Atmel AVR 8- and 32-bit microcontrollers and processors with On-Chip Debug capability.

Supported interfaces are:

- JTAG (AVR 32-bit, AVR XMEGA®, and megaAVR® devices)
- PDI (AVR XMEGA devices) *Hardware revision 01 only
- debugWIRE (megaAVR and tinyAVR® devices)
- SPI (megaAVR and tinyAVR devices)
- aWire (AVR 32-bit devices) *Hardware revision 01 only

Table of Contents

1. Introduction

1.1. Atmel JTAGICE mkII Features

- Fully compatible with Atmel Studio, AVR32 Studio, and AVR Studio® 4 and later
- Supports debugging of all Atmel AVR 8- and 32-bit microcontrollers with OCD
- Supports programming of all 8- and 32-bit AVR devices with OCD
- Exact Electrical Characteristics
- Emulates Digital and Analog On-Chip Functions
- Software Breakpoints (*not ATmega128[A])
- Program Memory Breakpoints
- Supports Assembler and HLL Source Level Debugging
- Programming Interface to flash, EEPROM, fuses, and lockbits (not debugWIRE)
- USB 1.1 and RS232 Interface to PC for Programming and Control
- Regulated Power Supply for 9-15V DC Power
- Can be powered from the USB bus
- Target operating voltage range of 1.65V to 5.5V

1.2. System Requirements

The Atmel JTAGICE mkII unit requires that a front-end debugging environment (Atmel Studio, AVR32 Studio, or AVR Studio 4.9 or later) and associated utilities are installed on your computer. For system requirements of these packages, consult www.atmel.com.

The JTAGICE mkII unit must be connected to the host computer using either the USB or RS-232 cable provided. Functionality of the two connection options is identical.

Note: Atmel Studio does not support RS-232 serial communications.

The JTAGICE mkII unit may in addition be connected to a 9-15V DC external power source. A cable is included in the kit. If the external power supply is connected, USB power will not be used.

1.3. Hardware Revisions

Hardware revision 0 does NOT support the PDI or aWire interface, and have serial numbers starting with A0... as shown here.

Hardware revision 1 supports both PDI and aWire. Serial numbers start with A09-0041 or B0... as shown here.

Revision 1 also has a green LED inside the encapsulation, which light up when a USB connection is made.

Revision 1 units are also fully RoHS compliant.

2. Getting started

2.1. Kit Contents

The Atmel AVR JTAGICE mkII kit contains these items:

- Atmel AVR JTAGICE mkII unit with probe
- USB cable (1.8m)
- RS-232 serial cable
- DC power supply cable
- 10-pin (JTAG) to 6-pin (SPI) probe adapter cable
- 10-wire multicolor custom connector cable ("squid")
- Spare 30-lead flex cable
- AVR Technical Library DVD
- PDI adapter for Atmel AVR XMEGA microcontrollers

Figure 2-1. JTAGICE mkII Kit Contents

2.2. Powering the Atmel AVR JTAGICE mkII

The Atmel AVR JTAGICE mkII is able to operate using an external power supply providing 9-15V DC or it can be powered directly from the USB bus. An internal switch will default select the power from the external power supply. However, if this is not connected, or the external power supply drops below a usable level the power will be taken from the USB (if connected).

Although any polarity will work, the preferred polarity of the DC jack is negative-centre due to the power switch grounding. When powering up the JTAGICE mkII, the power LED should illuminate immediately. If the LED does not light up, check that an adequate power supply is being used.

Note: If the JTAGICE mkII is powered from the USB only, it's required that the USB can deliver minimum 500mA (Self Powered Hub).

When the JTAGICE mkII is properly connected to the target and the host PC, the power can be turned on. It's recommended to power up the JTAGICE mkII before the target is powered, to avert the possibility of current flowing from the target into the unpowered JTAGICE mkII.

Two LEDs indicate emulator power and target power respectively as shown in the Hardware descriptions section.

2.3. Connecting to the Host Computer

Before connecting up the Atmel AVR JTAGICE mkII 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 JTAGICE mkII can connect to the host PC through a USB cable or serial cable (to COM port on the PC).

2.4. Serial Port Connection

AVR Studio 5 and Atmel Studio does not support RS-232 serial communications.

2.5. USB Driver Installation

USB drivers for the Atmel JTAGICE mkII are installed with the Atmel Studio, AVR Studio 4 and later, or AVR32 Studio front-end software.

2.5.1. Windows

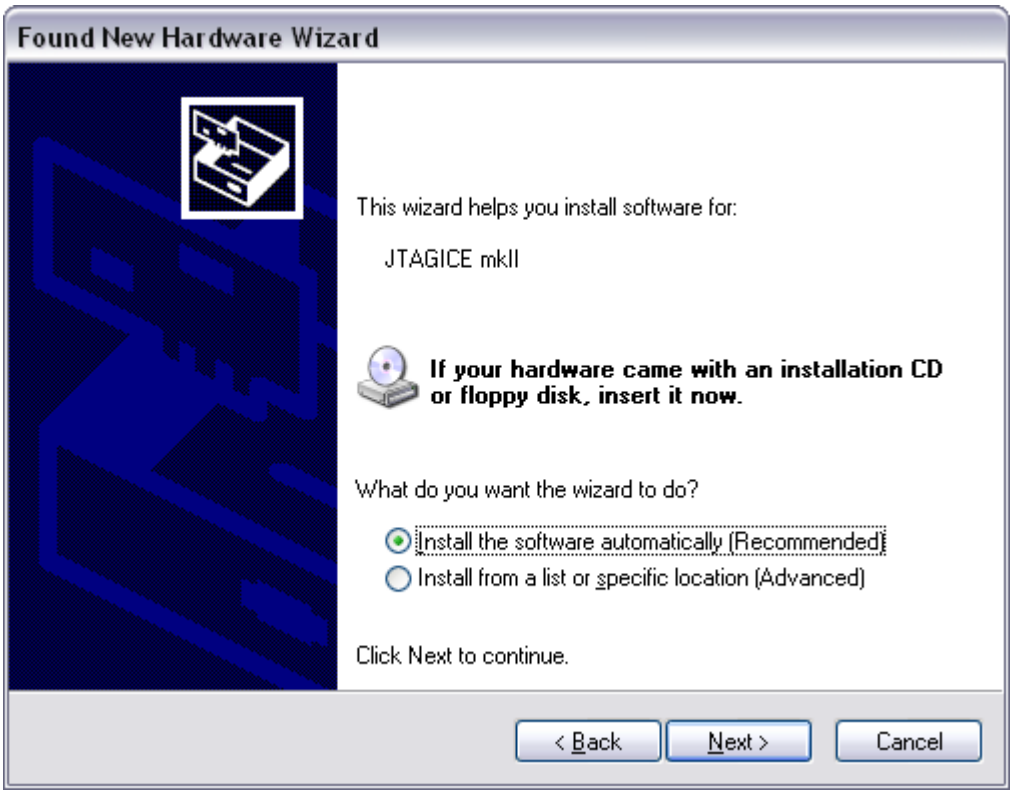
When installing the JTAGICE mkII on a computer running Microsoft® Windows®, the USB driver is loaded when the unit is first powered up.

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 2-2. Installing the JTAGICE mkII USB Driver

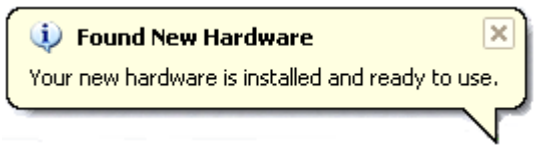
Figure 2-3. Installing the JTAGICE mkII USB Driver



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

Once successfully installed, the JTAGICE mkII will appear in the device manager as a "Jungo" device.

Your JTAGICE mkII is now ready to use.



2.6. Debugging

The simplest way to get started with your Atmel AVR JTAGICE mkII using Atmel Studio is to build one of the many example projects from ASF. See the for more information.

3. Connecting the Atmel JTAGICE mkII

3.1. Connecting to a JTAG Target

The Atmel JTAGICE mkII probe has two target 10-pin connectors that supports all debugging and programming interfaces. The pinout on each connector is identical. For JTAG debugging, the 10-pin connector can be used directly. For other interfaces, an adapter is required.

3.1.1. Using the JTAG 10-pin Connector

The pinout for the 10-pin JTAG connector is shown in Figure 4-2 JTAG Header Pinout.

Be sure to use the correct orientation of the 10-pin header when connecting the JTAGICE mkII to the target application PCB. A 50-mil stand-off adapter is available from Atmel, if required.

For more information on the JTAG physical interface, see Physical Interfaces: JTAG.

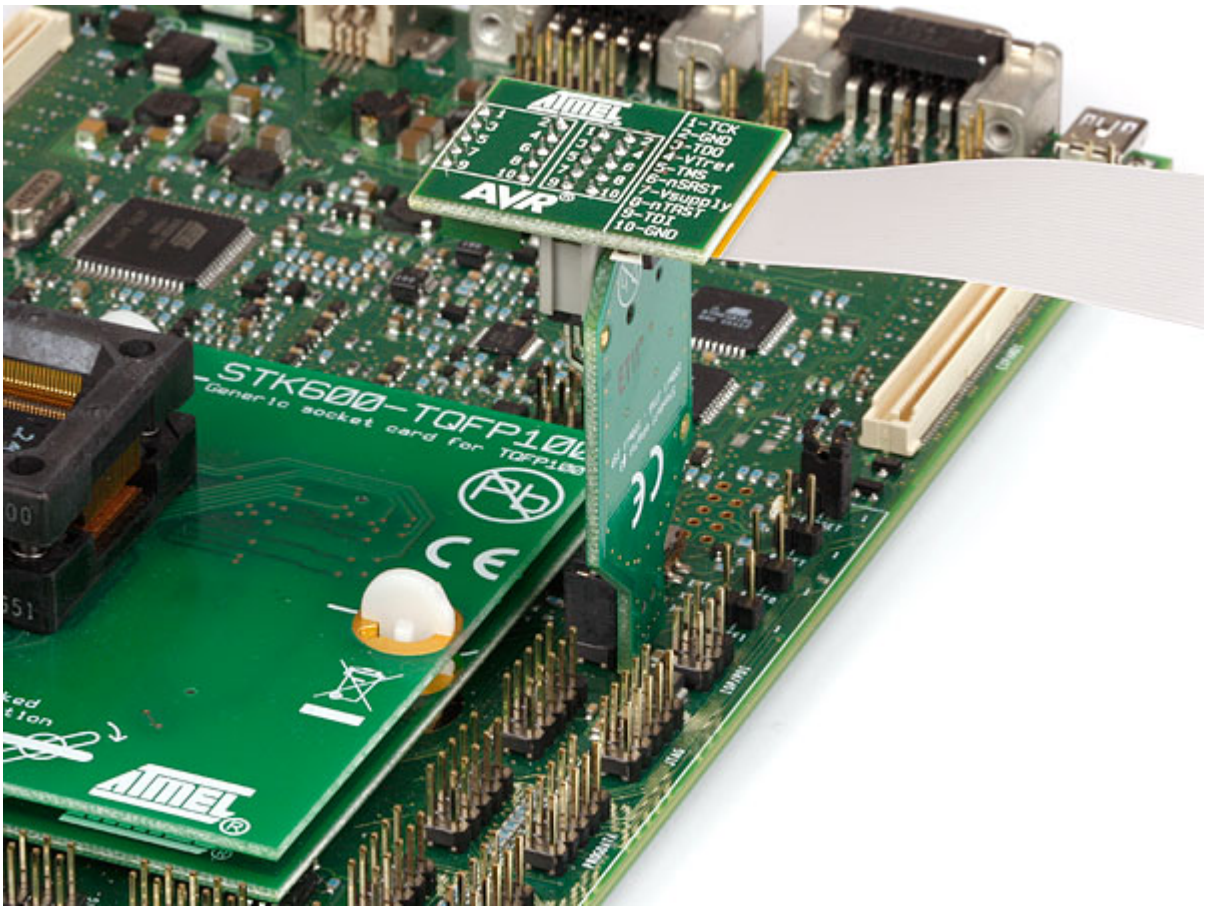
3.2. Connecting to a PDI Target

The pinout for the 6-pin PDI connector is shown in Figure 4-5 PDI Header Pinout.

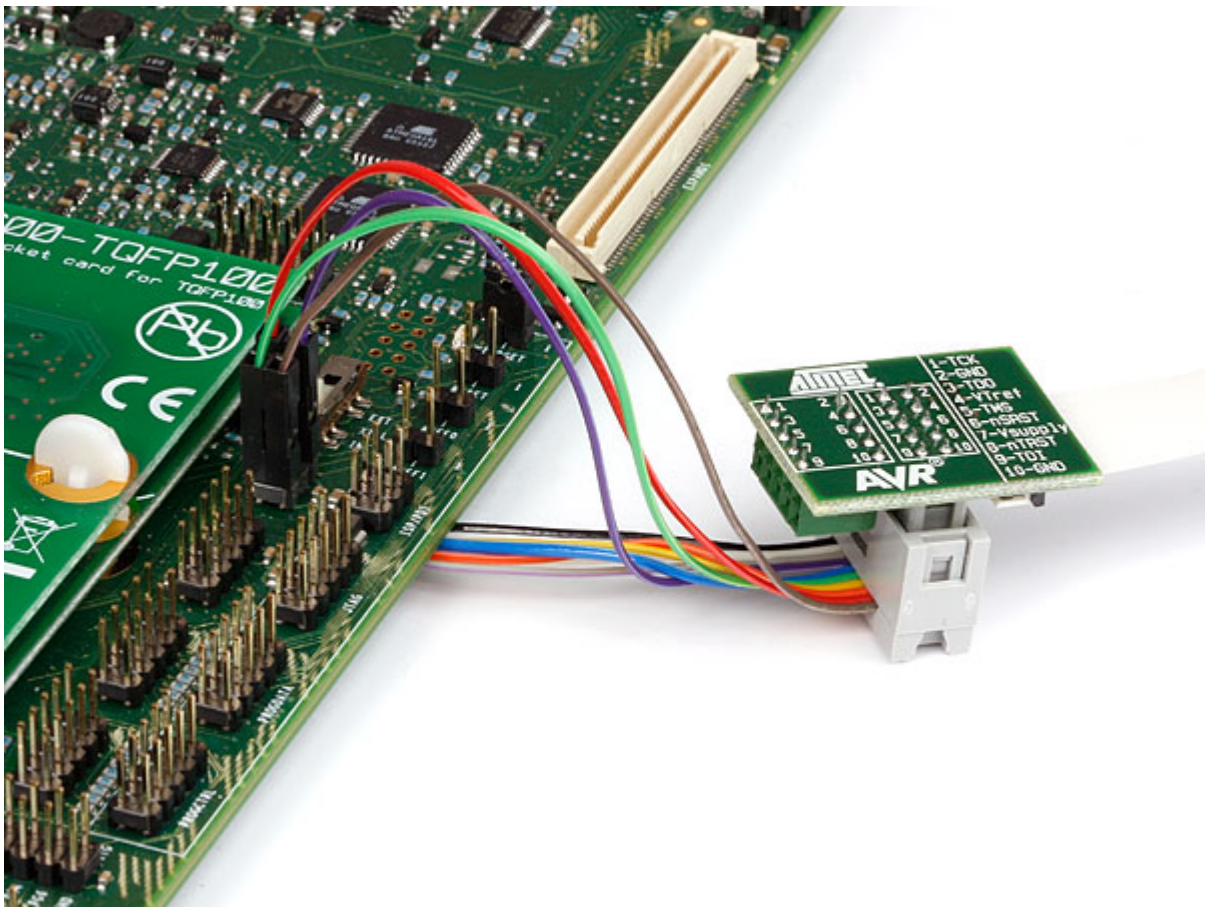
Note: Atmel AVR JTAGICE mkII units with hardware revision 0 do NOT have PDI capabilities. For more information on which hardware revision your unit is, see Hardware Revisions.

If your unit is revision 0, then PDI programming and debugging is not possible using this hardware. The JTAG interface (if available on your target device) does however provide the same functionality as PDI, although it uses I/O pins on the target device.

The pinout shown above is supported natively by the Atmel STK®600 as well as all future Atmel AVR XMEGA capable tools. In order to use the JTAGICE mkII with this pinout, it is necessary to make use of the XMEGA PDI adapter for JTAGICE mkII, which is available from your local Atmel representative.



Alternatively, the PDI interface can be connected using the multicolored "squid" cable, which is shipped with the JTAGICE mkII kit.



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

Note: The PDI_DATA is connected to pin 9 on the 10-pin JTAG connector compared to the normal pin 3 connection used on most tools.

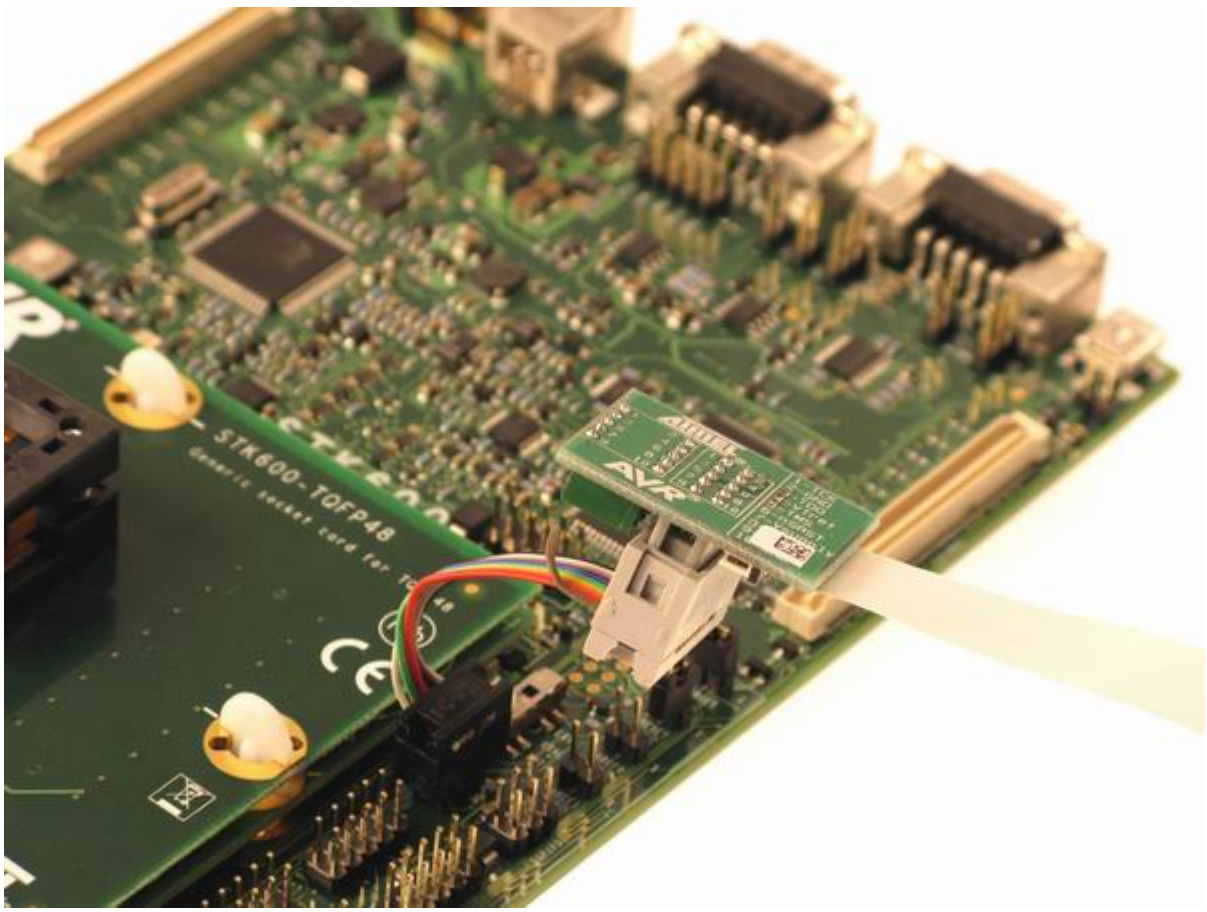
Table 3-1. Connecting to PDI Using the Squid Cable

JTAGICE mkII probe	*Target pins*	*Squid cable colors*	*STK600 PDI pinout*
Pin 1 (TCK)		Black	
Pin 2 (GND)	GND	White	6
Pin 3 (TDO)		Grey	
Pin 4 (VTref)	VTref	Purple	2
Pin 5 (TMS)		Blue	
Pin 6 (nSRST)	PDI_CLK	Green	5
Pin 7 (Not connected)		Yellow	
Pin 8 (nTRST)		Orange	
Pin 9 (TDI)	PDI_DATA	Red	1
Pin 10 (GND)		Brown	

3.3. Connecting to a debugWIRE Target

The pinout for the 6-pin debugWIRE (SPI) connector is shown in Figure 4-6 debugWIRE (SPI) Header Pinout.

Be sure to use the correct orientation of the 6-pin header when connecting the Atmel AVR JTAGICE mkII to the target application PCB.



Although the debugWIRE interface only requires 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 (DWEN fuse) can be enabled and disabled using SPI programming.

Some precautions regarding the RESET line must be taken to ensure proper communication over the debugWIRE interface. If there is a pull-up resistor on the RESET line, this resistor must be larger than 10kΩ. The JTAGICE mkII has an internal RESET pullup. Any capacitive load on the RESET line should be removed. Any other logic connected to the RESET line should also be removed.

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 3-2. Connecting to SPI Using the Squid Cable

JTAGICE mkII probe	*Target pins*	*Squid cable colors*	*SPI pinout*
Pin 1 (TCK)	SCK	Black	3
Pin 2 (GND)	GND	White	6
Pin 3 (TDO)	MISO	Grey	1
Pin 4 (VTref)	VTref	Purple	2
Pin 5 (TMS)		Blue	
Pin 6 (nSRST)	RESET	Green	5
Pin 7 (Vsupply)		Yellow	
Pin 8 (nTRST)		Orange	
Pin 9 (TDI)	MOSI	Red	4
Pin 10 (GND)		Brown	

3.4. Connecting to an aWire Target

The pinout for the 6-pin aWire connector is shown in Figure 4-4 aWire Header Pinout

Note: Atmel AVR JTAGICE mkII units with hardware revision 0 do NOT have aWire capabilities. For more information on which hardware revision your unit is, see Hardware Revisions.

If your unit is revision 0, then aWire programming and debugging is not possible using this hardware. The JTAG interface (if available on your target device) does however provide the same functionality as aWire, although it uses I/O pins on the target device.

The pinout shown above is the recommended pinout for aWire, and will be supported natively by future aWire capable tools. The JTAGICE mkII requires that the 10-pin multicolored "squid" cable be used to map to this pinout.

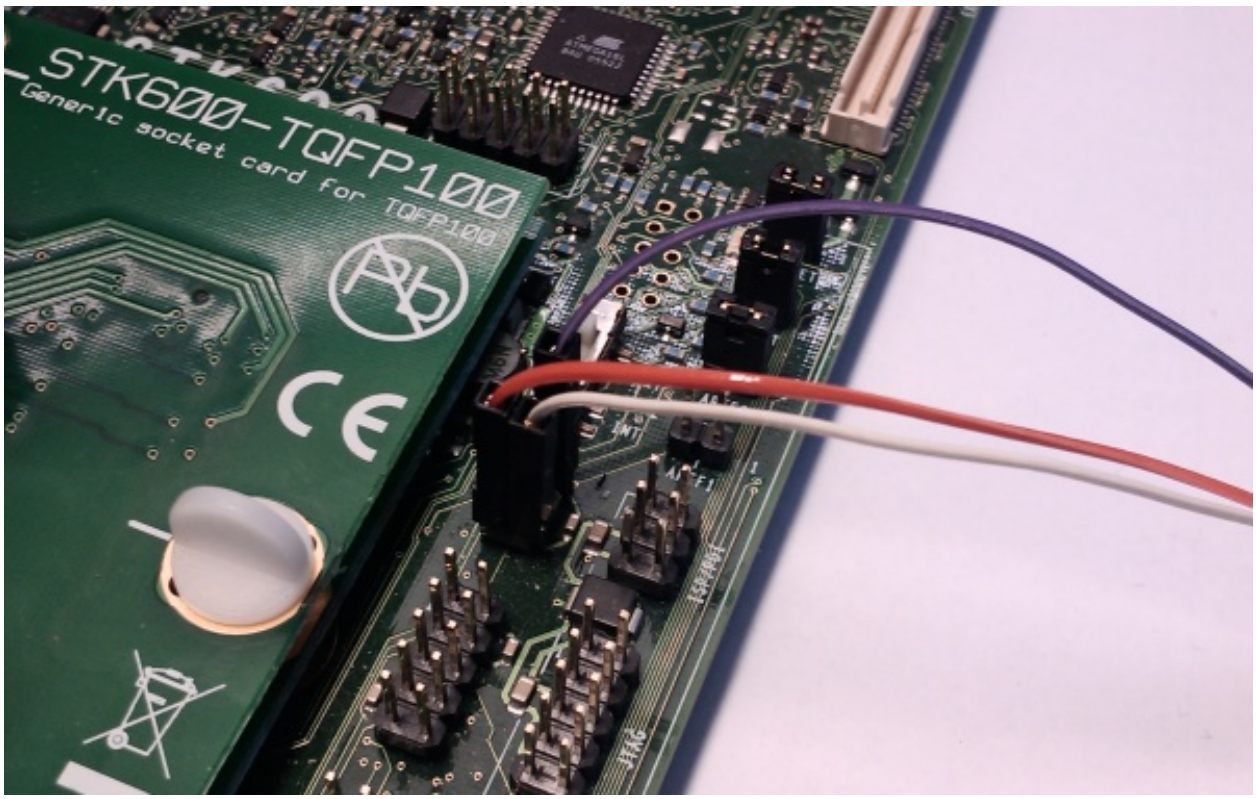


Table 3-3. Connecting to aWire Using the Squid Cable

JTAGICE mkII probe	Target pins	Squid cable colors	aWire pinout
Pin 1 (TCK)		Black	
Pin 2 (GND)	GND	White	6
Pin 3 (TDO)		Grey	
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)	aWire	Red	1
Pin 10 (GND)		Brown	

3.5. Connecting to an SPI Target

The pinout for the 6-pin SPI connector is shown in Figure 4-7 SPI Header Pinout.

Be sure to use the correct orientation of the 6-pin header when connecting the Atmel AVR JTAGICE mkII to the target application PCB.

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. It is HIGHLY ADVISED to simply let Atmel Studio handle setting and clearing of the DWEN fuse.

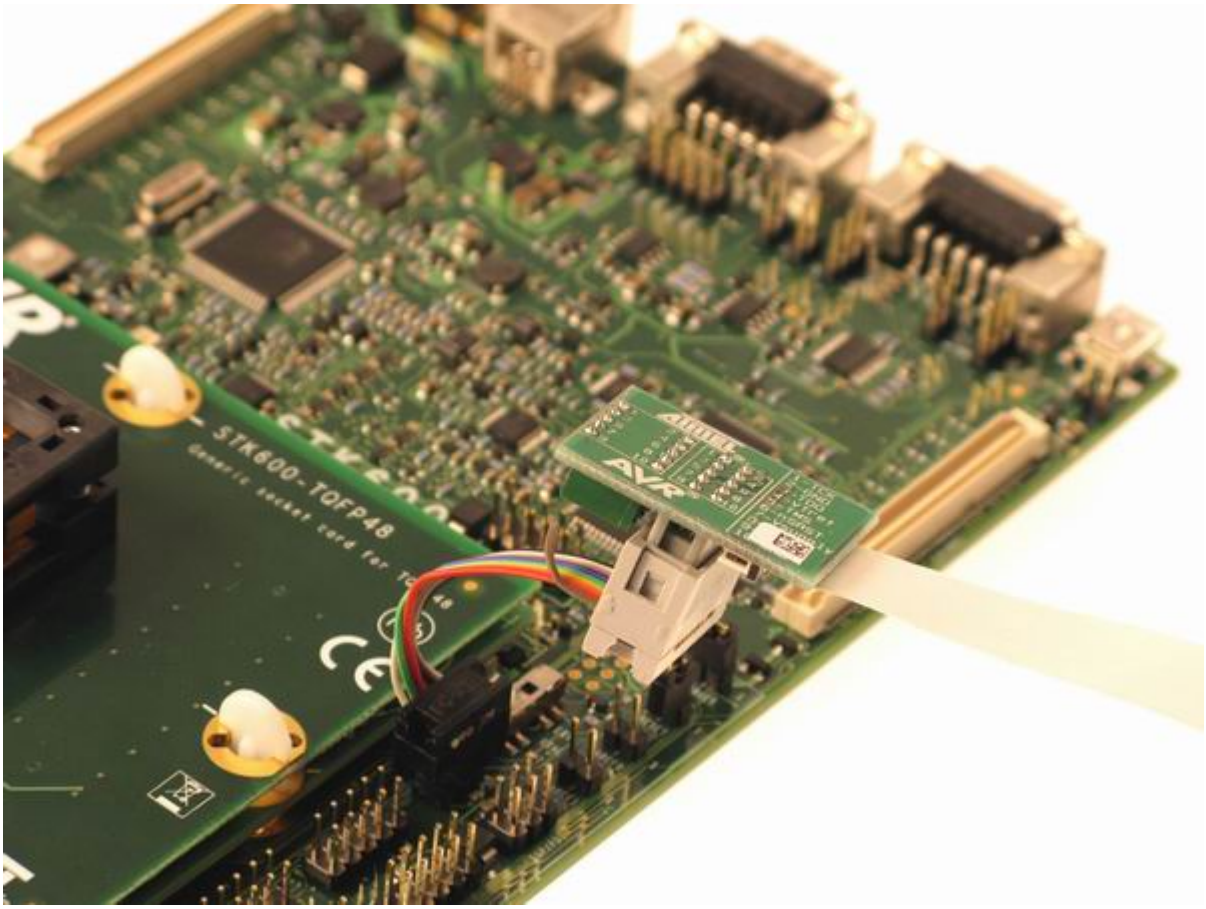


Table 3-4. Connecting to SPI Using the Squid Cable

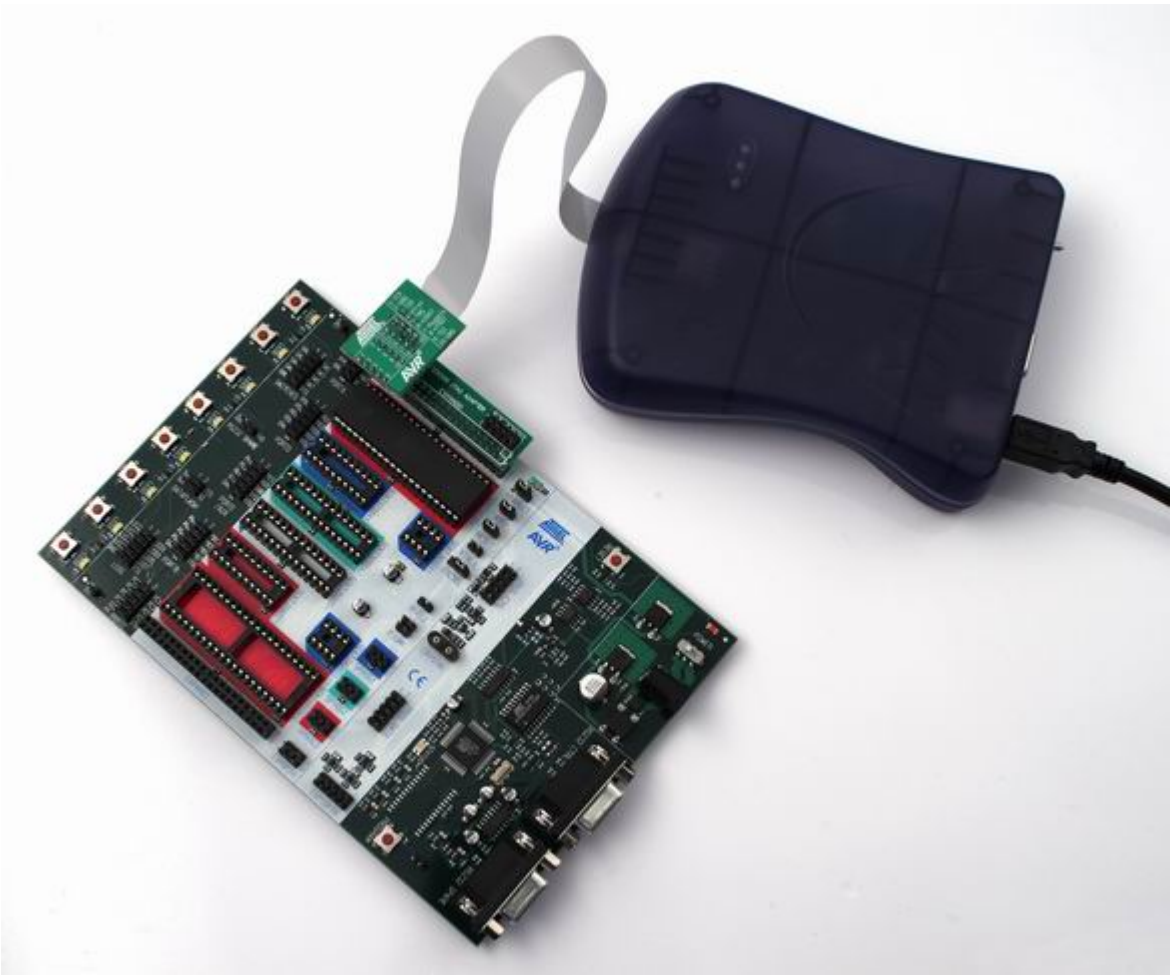
JTAGICE mkII probe	Target pins	Squid cable colors	SPI pinout
Pin 1 (TCK)	SCK	Black	3
Pin 2 (GND)	GND	White	6
Pin 3 (TDO)	MISO	Grey	1
Pin 4 (VTref)	VTref	Purple	2
Pin 5 (TMS)		Blue	
Pin 6 (nSRST)	RESET	Green	5
Pin 7 (Vsupply)		Yellow	
Pin 8 (nTRST)		Orange	
Pin 9 (TDI)	MOSI	Red	4
Pin 10 (GND)		Brown	

3.6. Using the Atmel JTAGICE mkII with Atmel STK500

The Atmel STK500 starter kit can be used to house Atmel AVR devices to which the Atmel AVR JTAGICE mkII can connect through JTAG, debugWIRE, and SPI interfaces.

When connecting to a JTAG target, simply use the ATSTK500_JTAG_ADAPTER shown here:

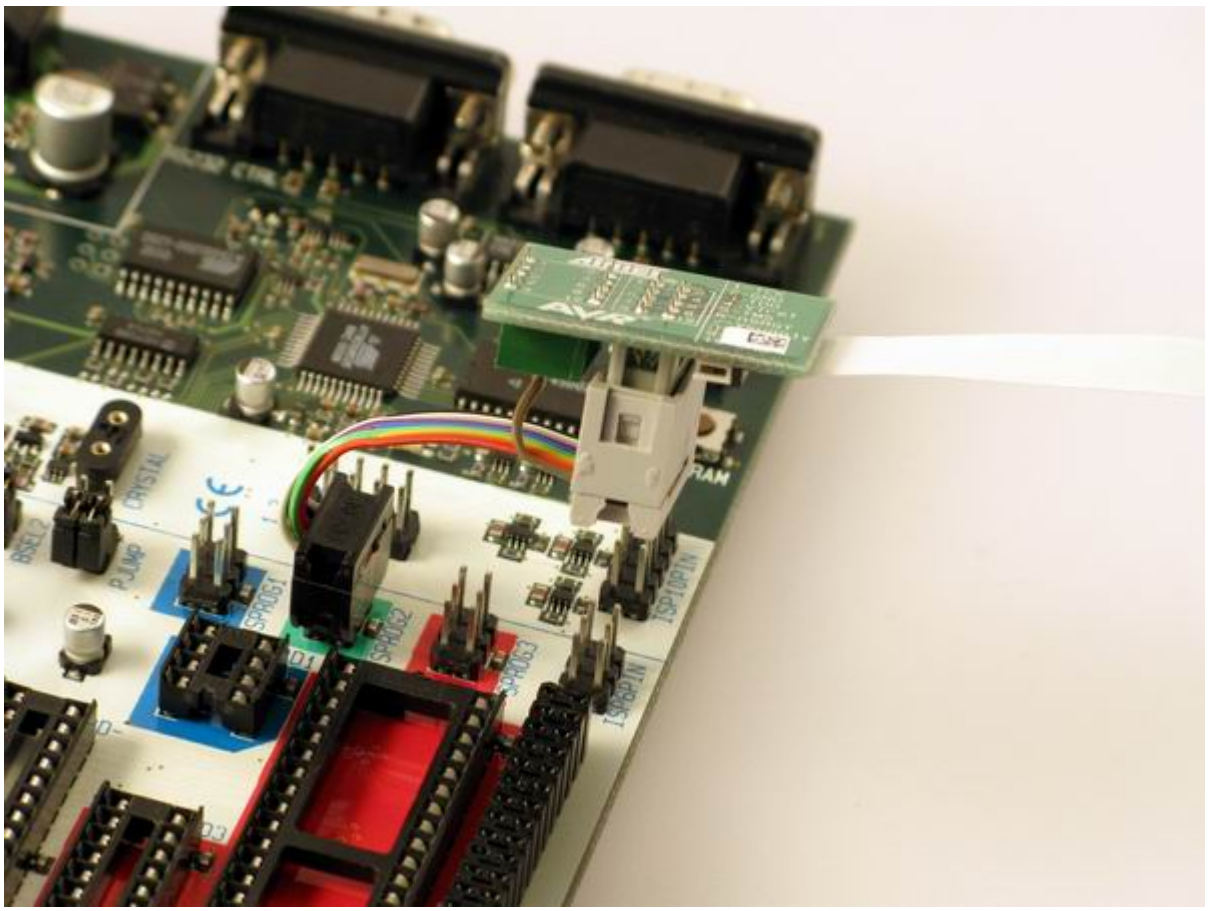
The STK500 JTAG Adapter, that ships with some STK500 (and earlier JTAGICE mkII kits), can be used to simplify the connection to the STK500 for AVR devices with JTAG that mates with socket SCKT3100A3 and SCKT3000D3 on the STK500.



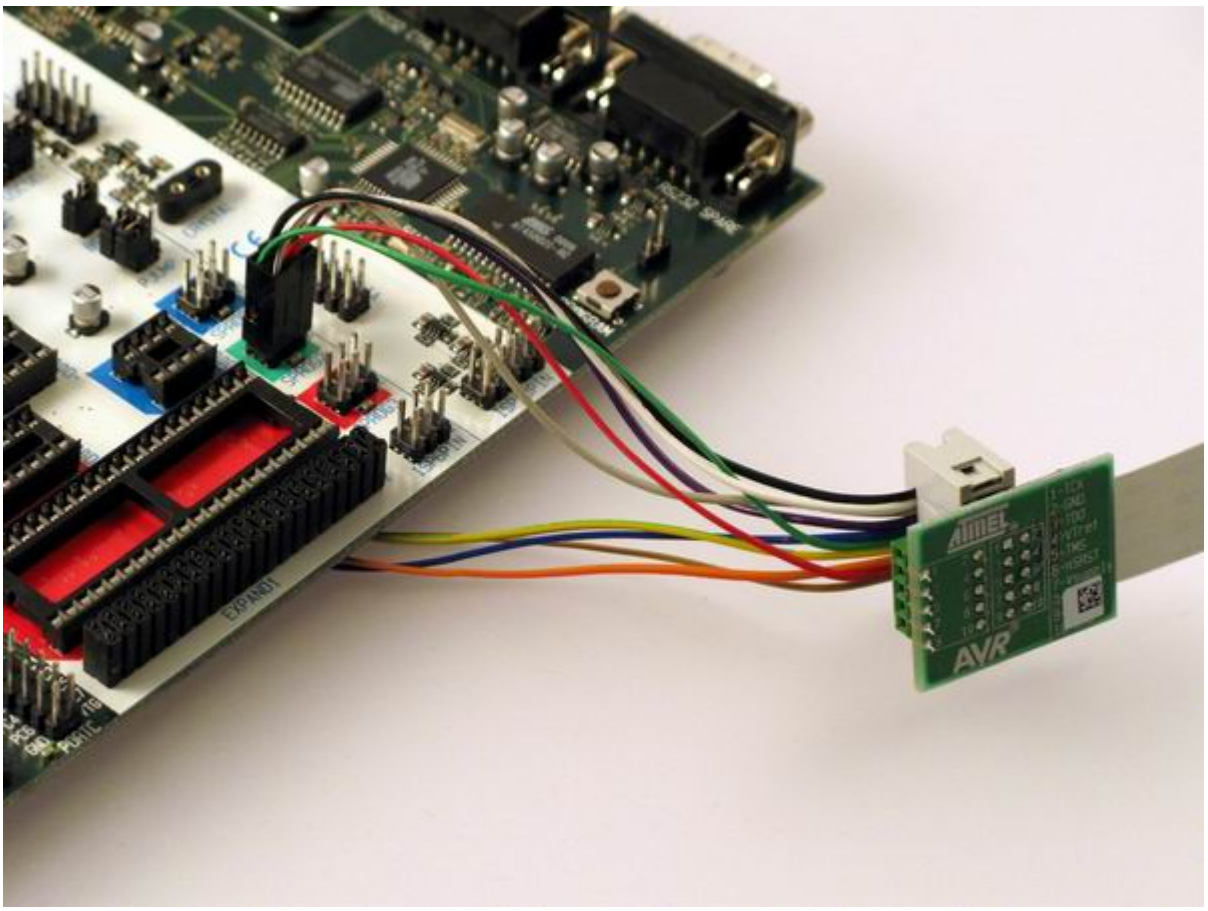
Note: Add-on cards for the STK500 like e.g. STK501/502 usually have a dedicated JTAG connector.

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 10-pin to 6-pin ribbon cable. When using the debugWIRE interface, be sure to remove the STK500's RESET jumper to allow the reset line to be driven as required.



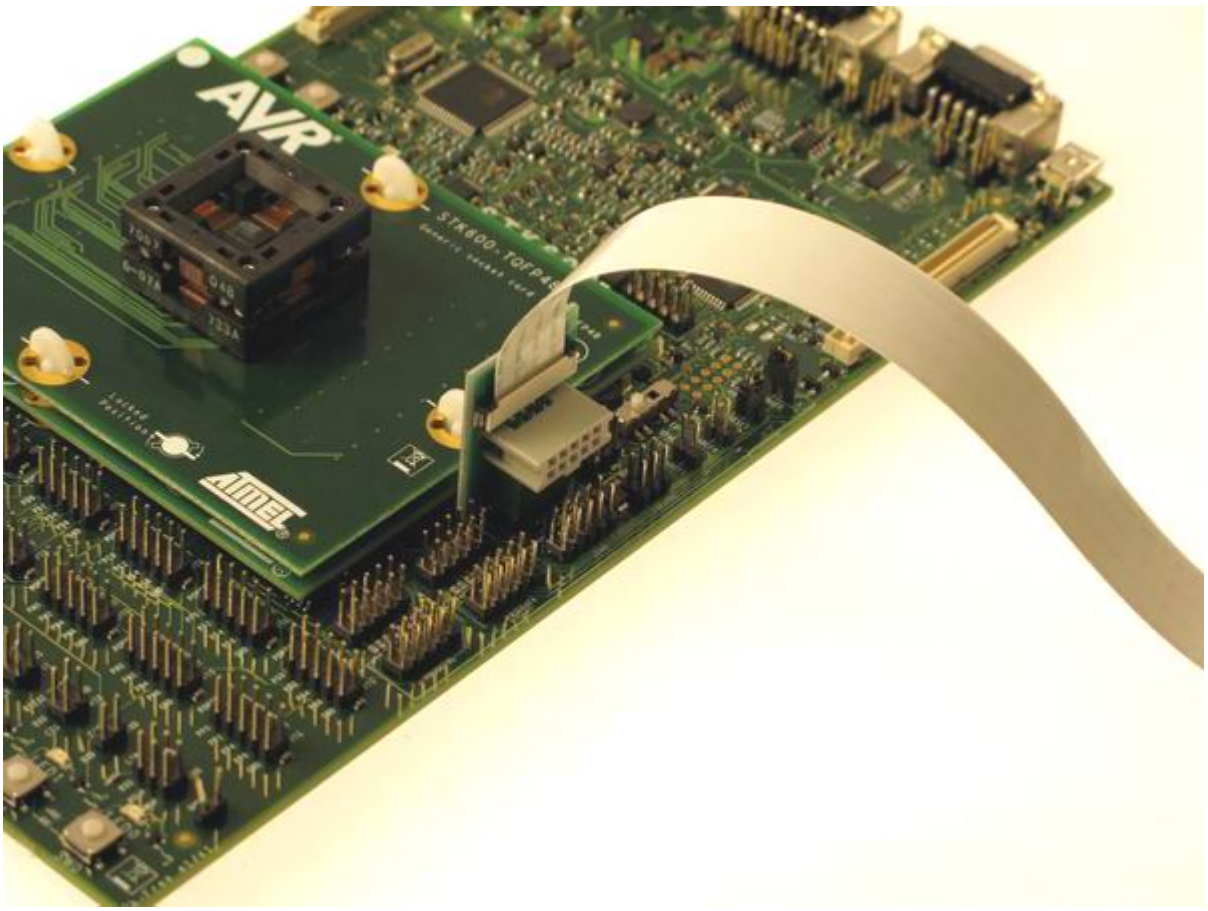
Alternatively, the JTAGICE mkII can be connected to any target interface using the 10-pin "squid" cable (provided).



When debugging devices using the debugWIRE interface on the STK500, be sure to connect a clock to the device (unless using internal RC) - this often requires some straps for Atmel tinyAVR devices. Also, be sure that the RESET signal is correctly strapped, and that the RESET jumper on the STK500 is removed.

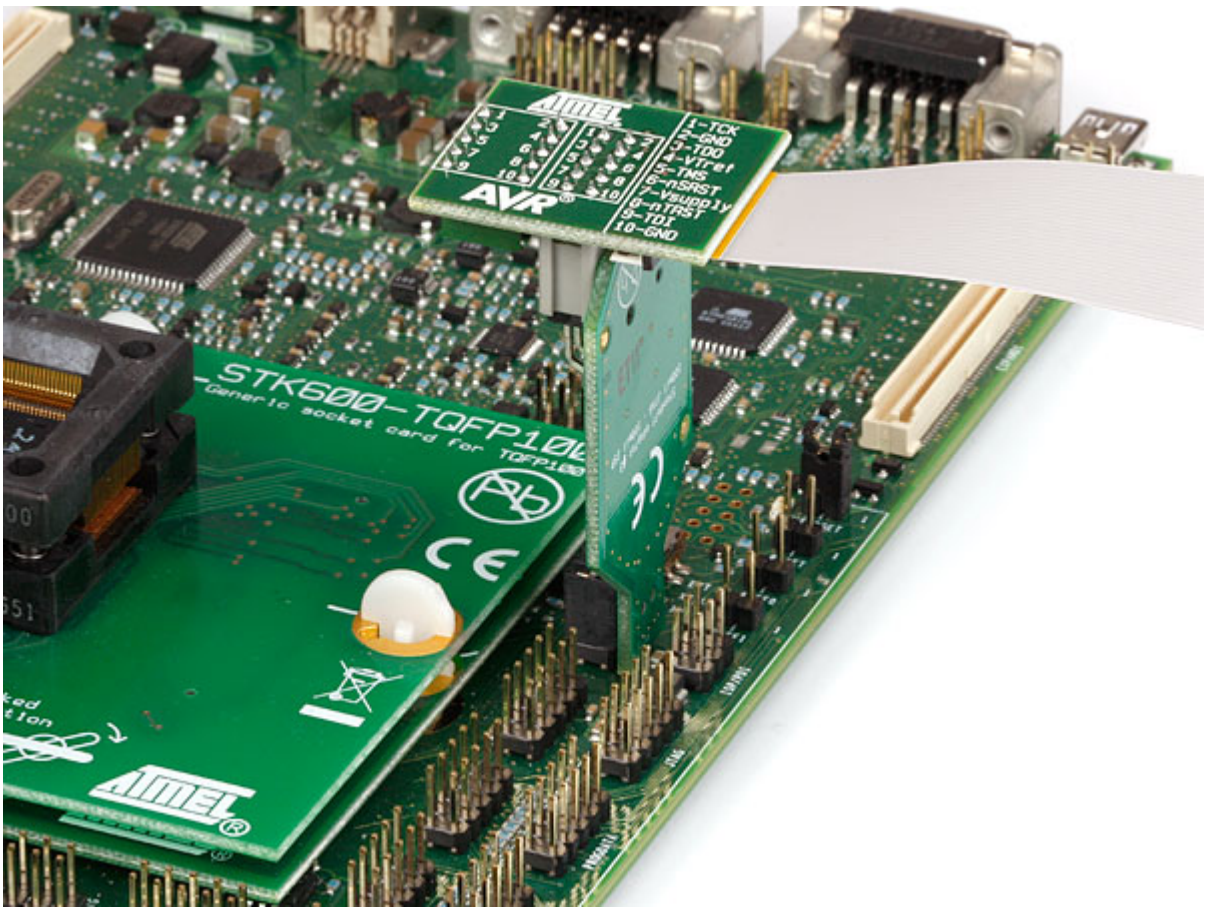
3.7. Using the Atmel JTAGICE mkII with Atmel STK600

The Atmel STK600 starter kit can be used to house Atmel AVR devices to which the Atmel JTAGICE mkII can connect through the JTAG, PDI, aWire, debugWIRE, and SPI interfaces.

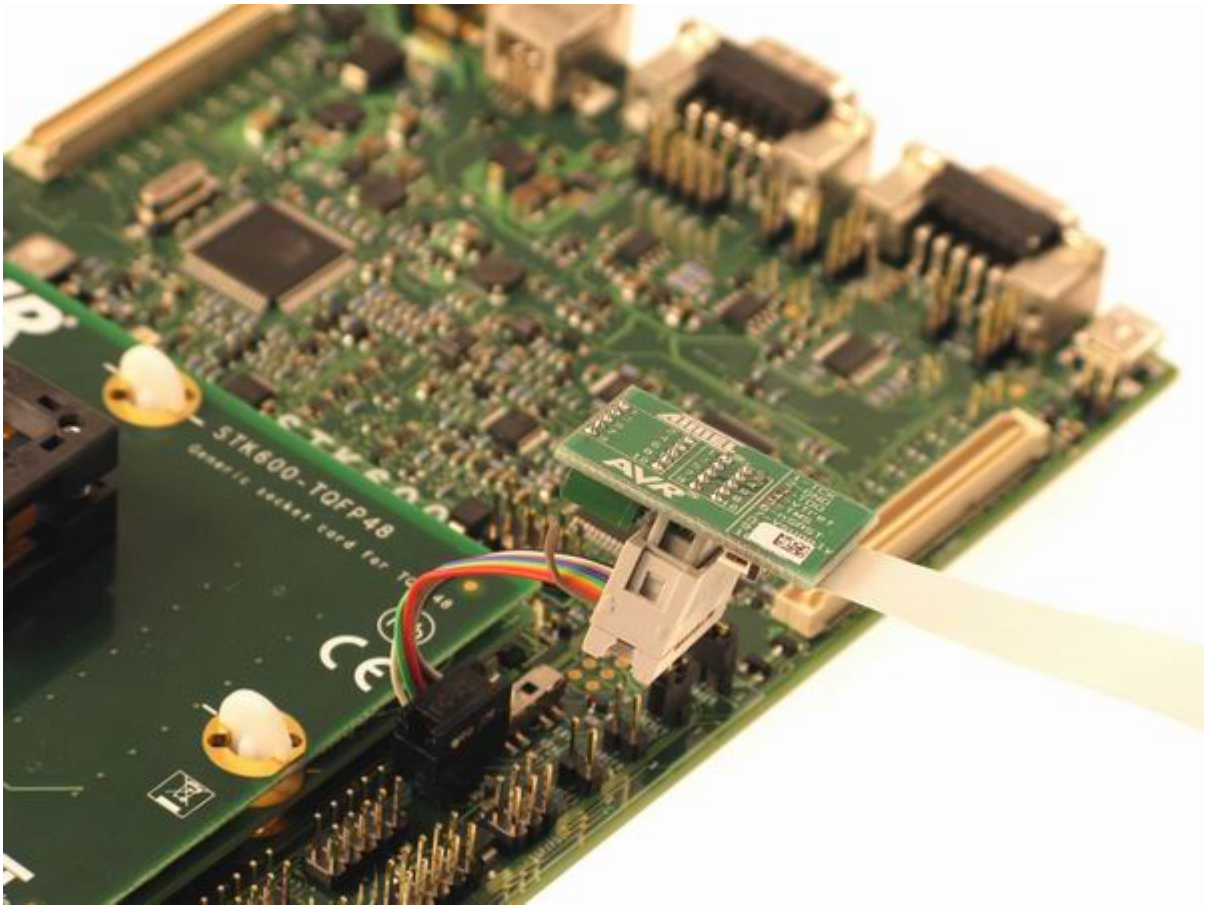


When connecting to a JTAG target, simply connect the JTAGICE mkII probe to the inner JTAG pin header on the STK600.

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



When connecting to a debugWIRE or SPI target, simply use the 10-pin to 6-pin 100mil ribbon cable (provided) to connect to the SPI/PDI connector.



When connecting to an aWire target, the 10-pin multicolored "squid" cable (provided) must be used to connect to the specified pinout. See section Connecting aWire.

4. On-Chip Debugging

4.1. 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 JTAGICE mkII is not a traditional *Emulator*. Instead, the JTAGICE mkII 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 JTAGICE mkII. The JTAGICE mkII 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.

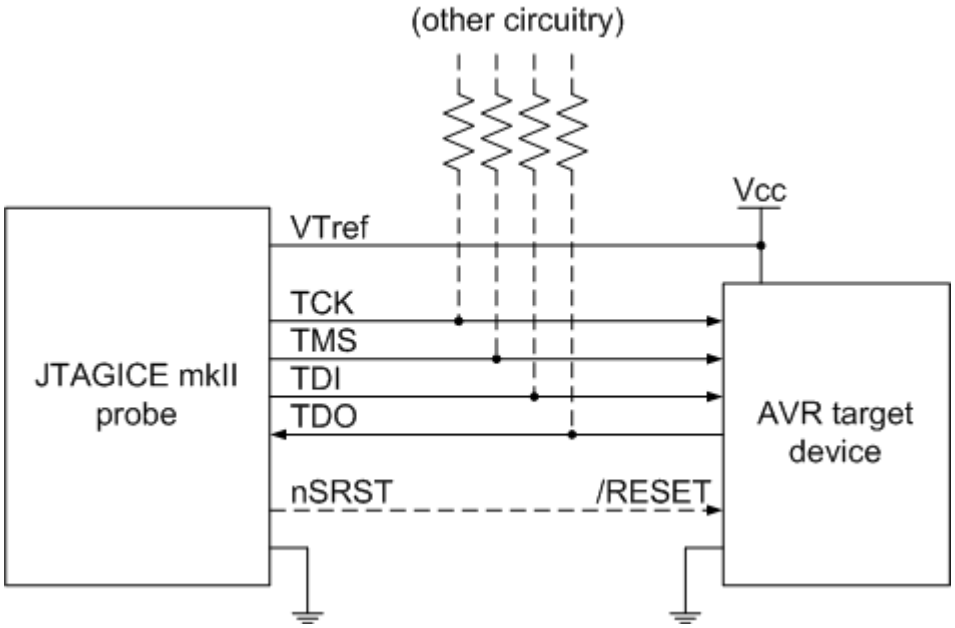
4.2. Physical Interfaces

The Atmel AVR JTAGICE mkII supports several hardware interfaces as described in the sections that follow.

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 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 AVR with the JTAG interface, it is recommended to use the pinout as shown in Figure 4-2 JTAG Header Pinout. The JTAGICE mkII 100-mil probe connectors support this pinout.

Figure 4-2. JTAG Header Pinout

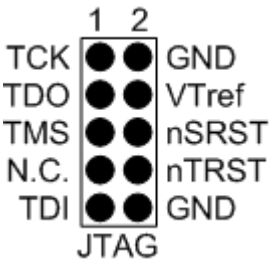


Table 4-1. JTAG Pin Description

Name	Pin	Description
TCK	1	Test Clock (clock signal from the JTAGICE mkII into the target device)
TMS	5	Test Mode Select (control signal from the JTAGICE mkII into the target device)
TDI	9	Test Data In (data transmitted from the JTAGICE mkII into the target device)
TDO	3	Test Data Out (data transmitted from the target device into the JTAGICE mkII)
nTRST	8	Test Reset (optional, only on some AVR devices). Used to reset the JTAG TAP controller.

Name	Pin	Description
nSRST	6	Source Reset (optional). Used to reset the target device. Connecting this pin is recommended since it allows the JTAGICE mkII to hold the target device in a reset state, which can be essential to debugging in certain scenarios - for example if the JTD bit is set by the application firmware, disabling the JTAG interface. The nSRST pin has an internal pullup resistor in the JTAGICE mkII.
VTref	4	Target voltage reference. The JTAGICE mkII samples the target voltage on this pin in order to power the level converters correctly. The JTAGICE mkII draws less than 1mA from this pin.
GND	2, 10	Ground. Both must be connected to ensure that the JTAGICE mkII and the target device share the same ground reference

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

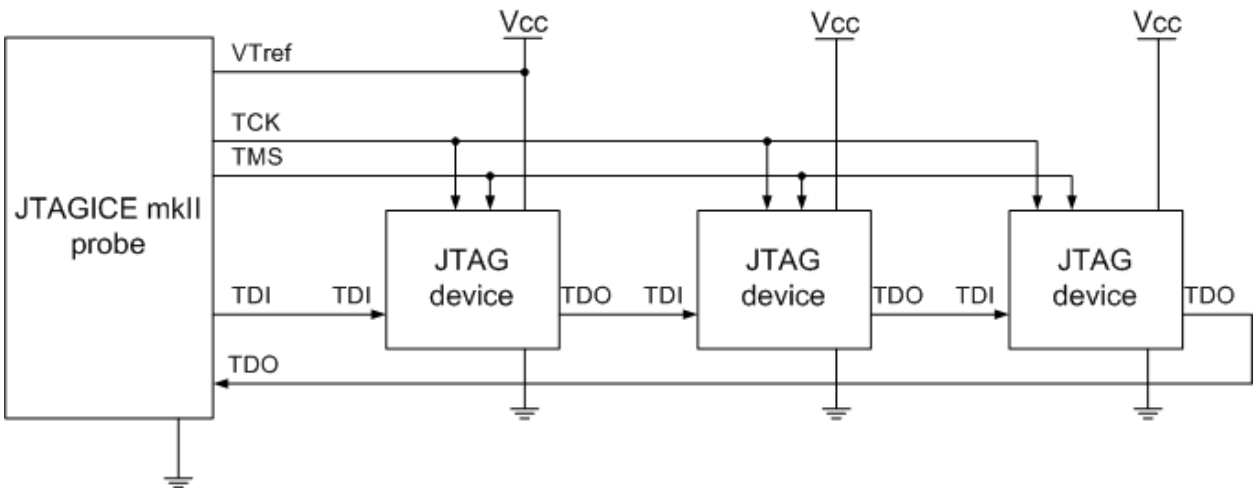
Note: The JTAGICE mkII cannot be powered by the target. V_{SUPPLY} (pin 7) should be left as NOT CONNECTED.

When external circuitry shares the JTAG debug lines on the target application, series resistors should be used to avoid driver contention, as shown in Figure 4-1 JTAG Interface Basics. The value of the resistors should be chosen so that the external circuitry and the AVR do not exceed their maximum ratings (i.e. sink or source too much current). 1kΩ is a commonly used value.

It is recommended to disconnect any analog filters on these lines (which should be on the 'outside' of the resistors) during a JTAG session, since these elements are discharged by the JTAG signals, possibly causing false logic levels influenced by the residual voltage in the capacitor. If the filters cannot be disconnected, it is then recommended to apply target V_{CC} directly to the capacitor during a session to hold the voltage stable. Be sure to use a large enough resistor between the capacitor and the JTAG line when doing this!

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 4-3 JTAG Daisy-chain.

Figure 4-3. 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 JTAGICE mkII probe

- All devices must be operating on the same target voltage level. VTref on the JTAGICE mkII probe must be connected only to V_{CC} on the first device in the chain.
- TMS and TCK are connected in parallel; TDI and TDO are connected in a serial chain.
- NSRST on the JTAGICE mkII 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 JTAGICE mkII 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 + instruction bits after) is limited to a maximum of 32 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 four IR bits)

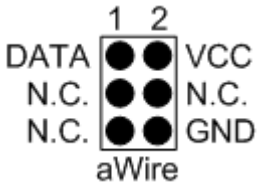
Instruction bits before: 5 (Atmel AVR 32-bit microcontrollers have five IR bits)

4.2.2. aWire Physical

aWire is a single-pin interface for programming and debugging of low-pin-count AVR 32-bit devices using the RESET pin. All features of the OCD system available through the JTAG interface can also be accessed using aWire.

When designing an application PCB which includes an AVR with the aWire interface, the pinout shown in Figure 4-4 aWire Header Pinout should be used. While future aWire capable tools will support this pinout, the Atmel JTAGICE mkII requires that the 10-pin "squid" cable is used to map to this pinout.

Figure 4-4. aWire Header Pinout

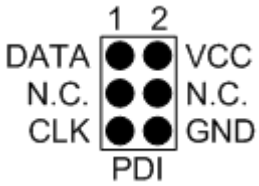


4.2.3. PDI Physical

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 AVR with the PDI interface, the pinout shown in Figure 4-5 PDI Header Pinout should be used. The 6-pin adapter provided with the Atmel AVR JTAGICE mkII kit can then be used to connect the JTAGICE mkII probe to the application PCB.

Figure 4-5. PDI Header Pinout



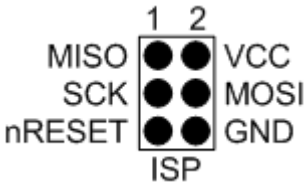
Note:
The pinout shown above is compatible with the Atmel STK600, Atmel AVR ONE!, AVR Dragon™, and future tools with PDI interface. The JTAGICE mkII supports this pinout using the AVR XMEGA PDI adapter for JTAGICE mkII (shipped with the kit or available from your local Atmel representative.)

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 AVR with the debugWIRE interface, the pinout shown in Figure 4-6 debugWIRE (SPI) Header Pinout should be used.

Figure 4-6. debugWIRE (SPI) Header Pinout



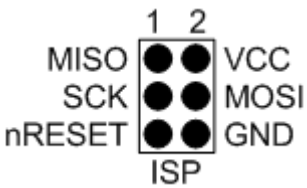
Note:
The debugWIRE interface cannot be used as a programming interface. This means that the SPI interface must also be available (as shown in Figure 4-7 SPI Header Pinout) in order to program the target.

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 wired-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 4-7 SPI Header Pinout should be used.

Figure 4-7. SPI Header Pinout



4.3. Atmel AVR 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 watch-points
- Hardware breakpoints can be combined to give break on ranges

For special considerations regarding this debug interface, see Special Considerations.

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

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 instruction)
- No limitation on system clock frequency

For special considerations regarding this debug interface, see Special Considerations.

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 (1 is reserved)
- Hardware breakpoints can be combined to form data breakpoints
- Unlimited number of program breakpoints (using BREAK instruction) (except ATmega128[A])

For special considerations regarding this debug interface, see Special Considerations.

4.3.4. Atmel megaAVR/tinyAVR OCD (debugWIRE)

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

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

For special considerations regarding this debug interface, see Special Considerations.

5. Hardware Description

5.1. Physical Dimensions

- Main unit: 140mm x 110mm x 30mm
- Probe cable: 180mm
- Probe: see description for the probe.

5.2. LEDs

The Atmel AVR JTAGICE mkII three LEDs which indicate the status of current debug or programming sessions.

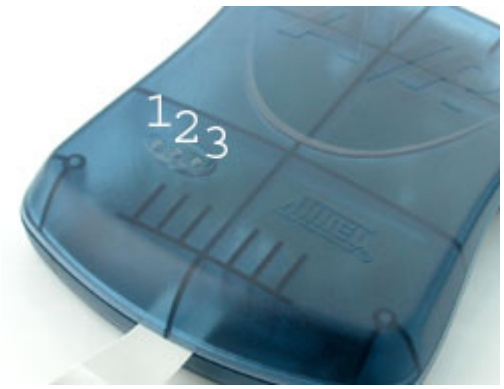


Table 5-1. LEDs

LED	Position	Description
Target power	1	GREEN when target board power is ON
JTAGICE mkII power	2	RED when the JTAGICE mkII unit is powered. Flashing indicates that the USB hub has not allocated the required power allowance.
Status	3	GREEN: Data transfer. Flashing green indicates target running. ORANGE: Firmware upgrade or initialization. RED: Idle, not connected. NONE: Idle, connected.

5.3. Rear Panel

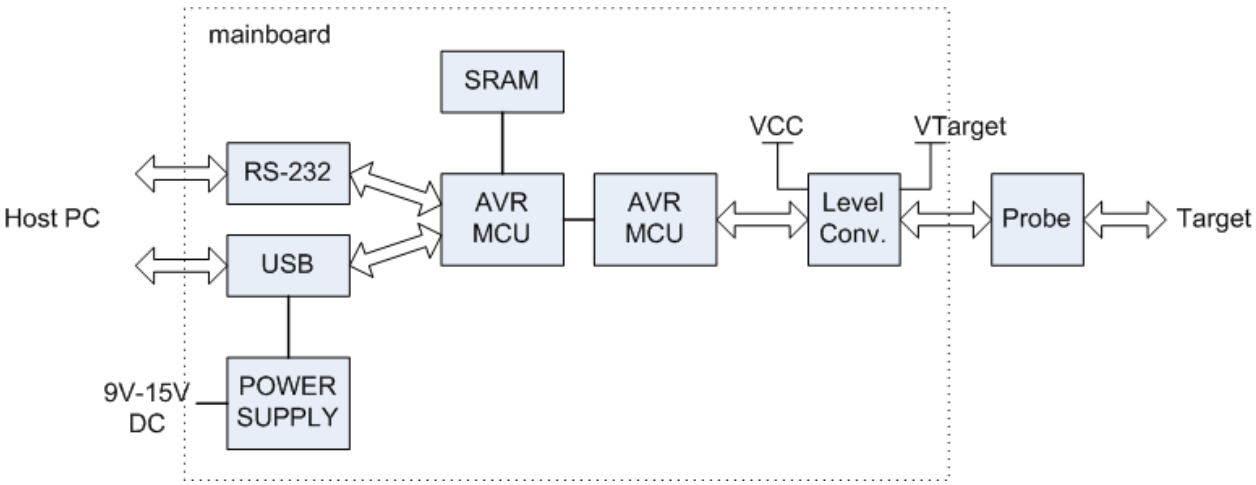
The rear panel of the Atmel AVR JTAGICE mkII houses the DC jack, power switch, USB, and RS-232 connectors.

The serial number is shown on a label on the underside of the unit.

5.4. Architecture Description

The Atmel AVR JTAGICE mkII architecture is shown in the block diagram in Figure 5-1 JTAGICE mkII Block Diagram.

Figure 5-1. JTAGICE mkII Block Diagram

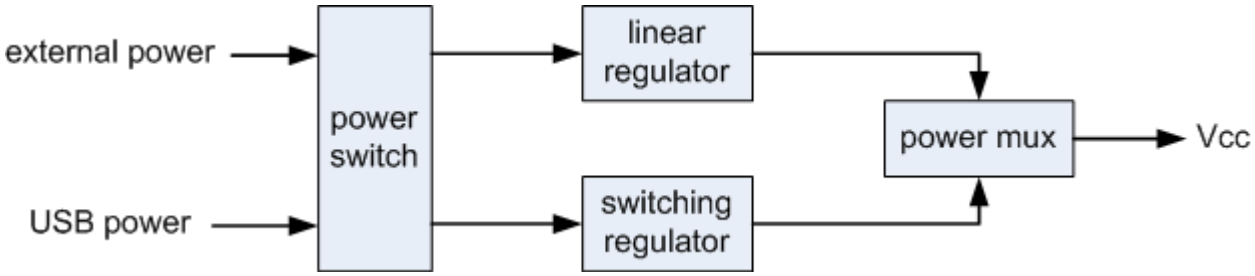


5.4.1. Power Supply

The Atmel AVR JTAGICE mkII power supply is implemented as shown in the figure below. The power switch found on the back of the JTAGICE mkII turns on both the External and USB power, an internal

switch will select which power source to use. The external power supply is default selected if it provides sufficient power.

Note:
The JTAGICE mkII cannot be powered from the target application.

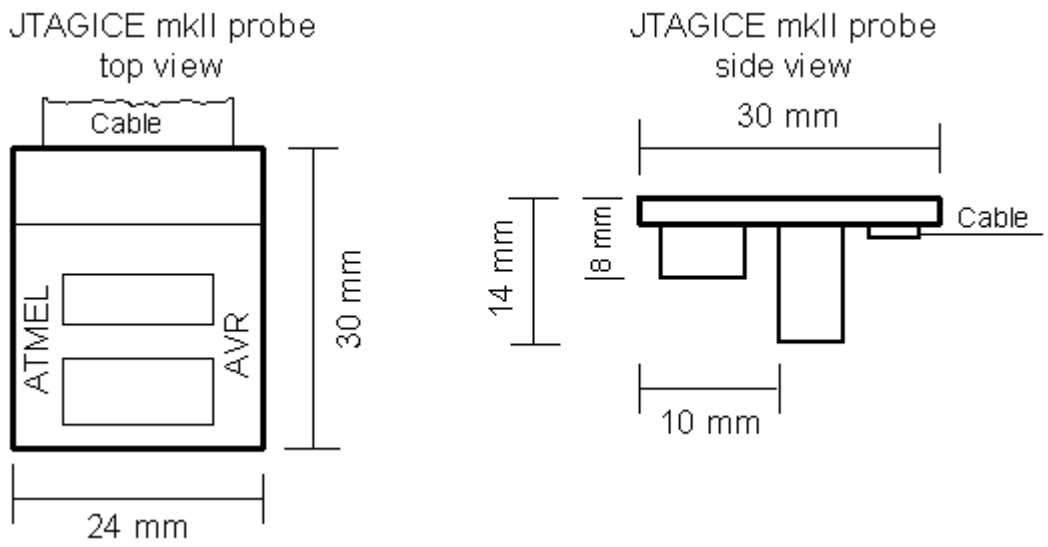


5.4.2. Level Converters

The purpose of the Level Converters is to provide successful communication with target boards running at voltages different than the Atmel AVR JTAGICE mkII itself. The level converters are designed to support target voltages from 1.65V to 5.5V.

5.4.3. Probe

The Target Adapter is pictured below. The 20-wire flex-cable connects the Target Adapter to the Atmel AVR JTAGICE mkII. The Target Adapter has two 10-pin connectors that have identical pin-out, and signals. Use the one that best fits the target board. Only one of the connectors should be connected at any given time.



For further information on how to connect the probe to the target application, see section Connecting the JTAGICE mkII.

6. Software Integration

6.1. Atmel Studio

6.1.1. Atmel Studio

Atmel Studio is an integrated development environment (IDE) for developing applications for Atmel AVR 8- and 32-bit microcontrollers based on the Visual Studio® Shell. For further information on what Atmel Studio has to offer, see the Atmel Studio help.

The Atmel AVR JTAGICE mkII is also compatible with AVR Studio 4 and AVR32 Studio.

6.1.2. Atmel Studio Programming GUI

The Atmel AVR JTAGICE mkII can be used to program Atmel AVR devices using a GUI environment which is part of the Atmel Studio IDE.

6.1.3. Programming Options

Atmel Studio supports programming of Atmel AVR devices using the Atmel AVR JTAGICE mkII. The programming dialog can be configured to use JTAG, aWire, SPI, or PDI modes, according to the target device selected.

The clock frequency can only be configured for the SPI interface. For JTAG, aWire, and PDI interfaces, programming is done independently of the target's clock, so no configuration is required.

6.1.4. Debug Options

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

- **Target clock frequency**

  Target clock frequency: Accurately setting the target clock frequency is vital to achieve reliable debugging of Atmel megaAVR devices over the JTAG interface. This setting should reflect the lowest operating frequency of your AVR target device in the application being debugged. See the special considerations section 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 AVR JTAGICE mkII will automatically select the correct baud rate for communicating at the start of a debug session.

  When debugging using the aWire interface the JTAGICE mkII will automatically tune the baud to the optimal value and no user settings are required.

  Atmel AVR XMEGA target devices will be clocked at maximum speed for JTAG and a constant 1MHz clock for PDI. No synchronization is required for XMEGA devices so the clock is not configurable.
- **Preserve EEPROM**

  Select this option to avoid erasing the EEPROM during reprogramming of the target before a debug session.
- **Always activate external reset when reprogramming device**

  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 JTAGICE mkII. 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
```

8. Special Considerations

8.1. 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.

The Atmel AVR JTAGICE mkII does not offer a variable clock rate for Atmel AVR XMEGA targets.

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 (0x10) is automatically used. This value can't be changed by the user.

I/O modules in stopped mode

Unlike most Atmel megaAVR devices, in XMEGA the I/O modules are stopped in stop mode. This means that USART transmissions will be interrupted, and 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 10kΩ or higher, or be removed altogether. Any reset capacitors should be removed. Other external reset sources should be disconnected.

8.2. 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 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 (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 be safely 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.

8.3. 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. Setting the target clock frequency too high will cause failure of a debug session shortly after programming completes. This may be accompanied by several spurious SLEEP, WAKEUP, or IDR messages being displayed. 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 the software integration section 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 AVR JTAGICE mkII 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 JTAGICE mkII reads this value out and displays it in the message window of the software front-end. The IDR registers is 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 JTAGICE mkII may still poll the device as the target voltage drops below the AVR's minimum operating voltage.

8.4. 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 it 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.

8.5. Atmel AVR UC3 OCD

JTAG interface

On some Atmel AVR UC3 microcontrollers 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 JTAGICE mkII 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.

aWire interface

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

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 JTAGICE mkII 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. Troubleshooting

9.1. Troubleshooting Guide

Table 9-1. Troubleshooting Guide

Problem	Possible causes	Solution
JTAG debugging starts, then suddenly fails.	1. The Atmel AVR JTAGICE mkII is not sufficiently powered 2. The JTAG Disable bit in the MCUCSR register has been inadvertently written by the application 3. Synchronization is lost.	1. If the JTAGICE mkII is powered from the USB only, it's required that the USB can deliver 500mA 2. Hold reset low to regain control and change the code so that the JTAG Disable bit is not written. 3. Power cycle the JTAGICE mkII and target board. Decreasing the communication speed between the PC and the JTAGICE mkII may be required.
After Using the JTAGICE mkII to download code to the device, the emulator no longer works	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	1. Program the JTAG ENABLE fuse. 2. Close the Programming interface, then enter emulation mode.
JTAGICE mkII 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 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.
OCD fuse is disabled, but using the JTAGICE mkII, OCD is still possible	The JTAGICE mkII will automatically program the OCD fuse if it is disabled	This is correct operation

Problem	Possible causes	Solution
Some I/O registers are not updated correctly in Atmel Studio I/O view	When non-intrusive read back is not possible, the JTAGICE mkII 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 Special considerations section for information about which registers are affected by this.
Debugging ATmega169 with the Atmel STK500 and Atmel STK502 does not work when using external clock	The TOSC switch on the STK502 is in the TOSC position	Set the switch to the XTAL position on the STK502 board
Sometimes after using software breakpoints, the target application "freezes" and will not run correctly	The JTAGICE mkII debugging session was not closed properly, and BREAK instructions are still present in the flash area	Make sure that the JTAGICE mkII debugging session is closed properly, or reprogram the flash with the correct hex file
debugWIRE Emulation start out OK, then suddenly it fails	1. The JTAGICE mkII is not sufficiently powered. 2. Drivers or active capacitance is disturbing the communication over the RESET-line. 3. Synchronization is lost.	1. If the JTAGICE mkII is powered from the USB only, it's required that the USB can deliver 500mA. 2. Remove all capacitance or drivers on the RESET-line. Make sure that a any pull-up resistor is larger than 10kΩ. 3. Power cycle the JTAGICE mkII and target board. Decreasing the communication speed between the PC and the JTAGICE mkII may be required.
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 section Special considerations "Connecting to Target through the debugWIRE Interface". Use command line software to re-enable SPI interface.

Problem	Possible causes	Solution
Neither SPI nor debugWIRE connection works	The SPI and debugWIRE interface 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.
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.

10. Firmware Upgrade

For information on how to upgrade the firmware, see the Atmel Studio user guide.

11. Release history and known issues

11.1. What's New

Table 11-1. New in this Release

Firmware versions	Master: 7.26; Slave: 7.26
Studio release	Atmel Studio 6.2 SP1
Notes	Fixed Status LED on Sign off

11.2. Firmware Release History (Atmel Studio)

Table 11-2. Previous Releases

Firmware versions	Master: 7.25; Slave: 7.25
Studio release	Atmel Studio 6.2
Notes	Fixed oscillator calibration

Firmware versions	Master: 7.24; Slave: 7.24
Studio release	Atmel Studio 6.1 SP2
Notes	• Fixed accessing locked PDI parts • Fixed launch issues for megaAVR

Firmware versions	Master: 7.20; Slave: 7.20
Studio release	AVR Studio 5.1
Notes	• Improved debugWIRE single-stepping performance • Support for software breakpoints on large (>320kB) Atmel AVR XMEGA devices • aWire auto-baud calculation improvements • Fixed XMEGA flash page programming error (seen at low voltages) • Chip erase timeout corrected for newer XMEGA devices • Support for high SUT values on XMEGA devices

Firmware versions	Master: 7.12; Slave: 7.12
Studio release	5.0 public release
Notes	

Firmware versions	Master: 7.11; Slave: 7.11
Studio release	5.0 public beta 2
Notes	Improved aWire speed

Firmware versions	Master: 7.06; Slave: 7.06
Studio release	5.0 public beta 1
Notes	None

11.3. Known Issues

Known issues in their respective categories are described in the following sections.

11.3.1. General

- Single stepping GCC-generated code in source-level may not always be possible. Set optimization level to lowest for best results, and use the disassemble view when necessary.

11.3.2. Hardware Related

- Always switch off the target application power before switching off the Atmel AVR JTAGICE mkII. Never leave a powered-down JTAGICE mkII connected to a powered application as current may leak from the application and result in damage to the emulator.
- If the target application uses the JTAG pins for general purpose I/O, the JTAGICE mkII can still be used to program the target via the JTAG pins (provided that the JTAG enable fuse is set.) However, be sure to connect the RESET pin of the target device to the nSRST pin of the emulator. Without this connection, the target application cannot be prevented from running after programming. If the application drives the JTAG pins as outputs, there will be signal contention with the emulator, which may result in damage to the emulator and/or target.

11.3.3. Atmel AVR XMEGA Related

- Stepping through code accessing EEPROM in memory-mapped mode may cause corruption of byte 0 in each EEPROM page.

11.3.4. JTAG (mega) Related

- When target power is lost, restored, or an external reset is applied, some spurious event messages may appear
- For projects containing self modifying code: When uploading a project, the target AVR enters RUN mode for some milliseconds before resetting to the reset vector. This means that the target application may have altered the state of the target Atmel AVR Flash and EEPROM. Workaround: The application must be written so that the self modifying code is not unintentionally accessed.

11.3.5. debugWIRE Related

- debugWIRE communication is lost when part code forces the part into reset. BOD, WDT, or other reset sources cause the part to lose debugWIRE communication. High voltage programming is required to get communication re-established.
- If a breakpoint is set at the last address of program memory, it is not possible to continue executing after reaching that breakpoint. Stepping or run will not cause the part to run. The instruction at the breakpoint will not be executed after breaking once.

- Stepping over the end of a non-terminated source code will cause the dW to produce error messages. This could be because the part contains code here, which has not been erased. Note that flash pages which are not used are not erased when starting debugging.
- Do not change the target voltage or frequency during a debug session. The debug session must be terminated first.
- OSCCAL and CLKPR register cannot be written in the application during debugging. This will cause the Atmel AVR JTAGICE mkII to lose synchronization and Atmel Studio will not be able to communicate with the JTAGICE mkII. Restart the debug session to fix this problem.
- The PRSPI bit (Power Reduction Serial Peripheral Interface bit) in parts with PRR register (Power Reduction Register) must not be written to 1. If this bit is written to 1 it will disable the clock to the debugWIRE module and all communication between the JTAGICE mkII and the debugWIRE interface will stop. Currently this bug is present in ATmega48/88/168.
- Inserting too many breakpoints may cause communication to be lost at very low target clock frequencies. When inserting or removing breakpoints, for each Flash page containing a modified breakpoint, the JTAGICE mkII must read, modify, and write the entire Flash page to the target device. When running at very low clock frequencies (kHz range), this may cause Atmel Studio to timeout. Workaround: insert breakpoints in groups, single stepping between inserts.
- When running off a 128kHz clock source, do not set the CLKDIV8 fuse. This will cause the debug session to fail since the interface speed is too low. The recommended minimum clock speed for successful debugging is 128kHz.
- Setting the CLKDIV8 fuse can cause connection problems when using debugWIRE. For best results, leave this fuse un-programmed during debugging.

11.3.6. Common

- When editing I/O bits in the I/O view: in order to clear a flag, which is "cleared by writing a one to its bit location", first clear it, then set it immediately. After it is set, it will automatically be cleared by the target device on its next cycle. Also note that any flags which are set in a register will be cleared by editing the register, since the set flags are written back to the register, and thus get automatically cleared.
- Single stepping a SLEEP instruction does not put the part into sleep mode. Use Run mode instead of single stepping.
- It is not possible to edit the target flash content using the Program Memory View in Atmel Studio, or by editing flash constants in the Watch Window
- When using USB, it is not possible to use one Atmel AVR JTAGICE mkII in a debug session and another one for programming from the same instance of Atmel Studio. Use separate instances of Atmel Studio when you need to debug a part and also have the programming interface available.
- It is not possible to upgrade a JTAGICE mkII over USB if it is connected via a bus-powered USB hub
- Be aware that the On-chip Debug system is disabled when any Lock bits are set, as a security feature
- Do no set the compatibility fuse while debugging with JTAGICE mkII for parts with this fuse bit
- On some devices, breaking execution on one of the two instructions immediately after an LPM instruction seems to corrupt the flash displayed in the dis-assembly view. Workaround: Do not single step LPM code, and do not insert breakpoints immediately after LPM instructions.

12. Revision History

Doc. Rev.	Date	Comments
42710A	04/2016	Initial document release.

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