Programmers and Debuggers

AVR Dragon

USER GUIDE

The Atmel AVR Dragon Debugger

With the Atmel® AVR® Dragon, Atmel has set a new standard for low-cost development tools. AVR Dragon™ supports all programming modes for the Atmel AVR device families. It also includes full debugging support for most Atmel AVR devices.

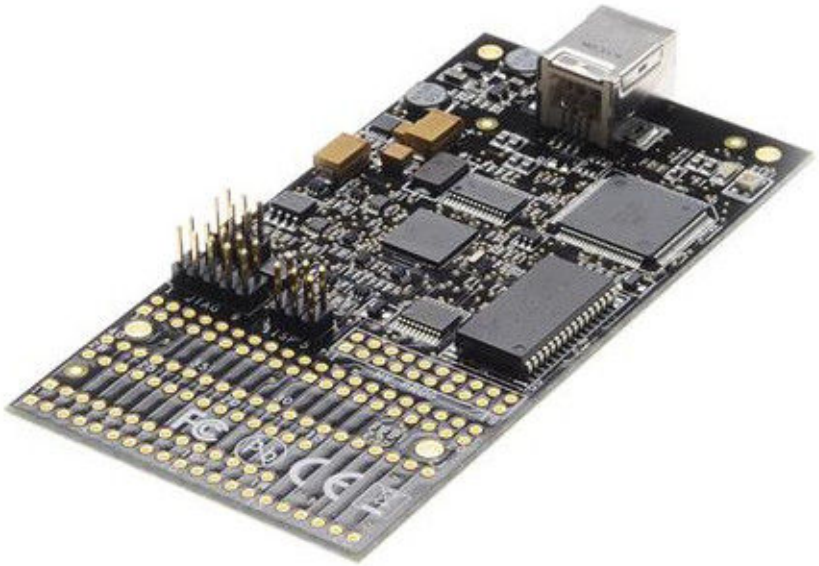


Table of Contents

1. Introducing AVR Dragon

Figure 1-1. Front side

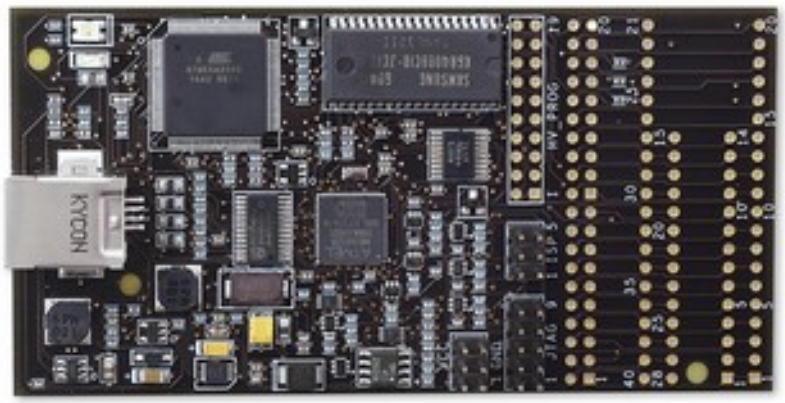
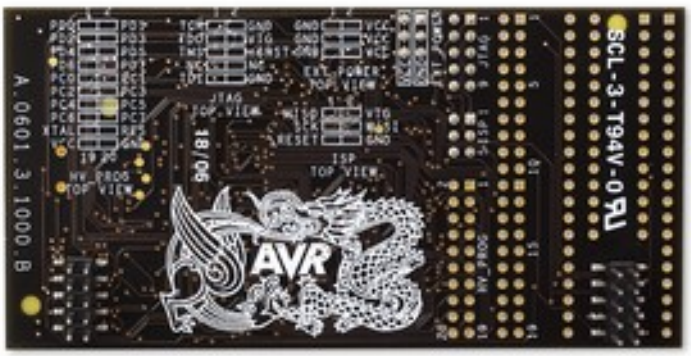


Figure 1-2. Back side



With the **Atmel AVR Dragon**, Atmel has set a new standard for low-cost development tools. AVR Dragon supports all programming modes for the Atmel AVR device families. It also includes full debugging support for most AVR devices.

At a fraction of the price traditionally associated with this kind of featured tool, the AVR Dragon will fulfill all your programming and debugging needs. The flexible and secure firmware upgrade feature allows the software front-end to easily upgrade the AVR Dragon to support new devices.

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

New devices will be added through Atmel Studio updates on a regular basis.

1.1. Supported Protocols

Currently the following protocols are supported:

1.1.1. Programming Interfaces

- SPI programming (SPI)

- High Voltage Serial Programming (HVSP)
- Parallel Programming (PP)
- JTAG Programming (JTAG)
- PDI Programming (PDI)
- aWire Programming (aWire)

1.1.2. Debugging Interfaces

- JTAG (JTAG)
- debugWIRE (dW)
- PDI (PDI)
- aWire (aWire)

1.2. Overview

Atmel AVR Dragon can be used with an external target board. However, the onboard prototype area allows simple programming and debugging without any additional hardware besides strapping cables. See the Using the Atmel AVR Prototype Area section for a description on how to use this.

AVR Dragon is powered by the USB cable, and can also source an external target with up to 300mA (from the VCC connector, 5V) when programming or debugging. For more information on technical details, read the AVR Dragon Requirements section. If the target is already powered by an external power source, the AVR Dragon will adapt and level convert all signals between the target and the AVR Dragon.

Note:
If the target board is powered by an external power source, no connection should be made between the VCC connector and the external board.

AVR Dragon is fully supported by Atmel Studio (hereafter called the software front-end). This allows the AVR Dragon firmware to be easily updated to support new devices and protocols. When connecting the AVR Dragon, the software front-end will automatically check the firmware and prompt the user if an updated firmware is available.

2. Known Issues

- JTAG communication with packages in PDIP which have the CKOUT fuse enabled and running above 3.5V may be unstable if there is a long wiring from the Atmel AVR Dragon to the PDIP AVR.
- **High voltage programming issue, all targets:** Parallel Programming and High Voltage Serial Programming might not work if the startup time is set to 0ms (SUT fuses). The problem gets worse if the CKDIV8 fuse is not set.
- **ATtiny84 Programming issue:** Parallel Programming may fail on the ATtiny84 if both the DWEN fuse and any of the external clock fuses are enabled at the same time. A workaround is to use the Atmel STK®500 platform for Parallel Programming of this part.
- **ATtiny26 Programming issue:** Parallel Programming on ATtiny26 is unstable with the AVR Dragon. A workaround is to use the STK500 platform for Parallel Programming of this part.
- In order to set SPI frequency, AVR Dragon needs to sense target voltage. See the troubleshooting guide.
- **XMEGA® PDI issues:** XMEGA PDI mode on AVR Dragon does NOT work for the following XMEGA devices: A3/D3 - revisions B, C, and E, or A1 (up to revision K).

3. Getting Started

3.1. Before Starting

Important:
Read this section before connecting the Atmel AVR Dragon to the computer or target.

Important:
Install the Atmel Studio software front-end and the USB driver before connecting AVR Dragon to your PC.

Follow these simple steps to get started using the AVR Dragon:

1. Download the latest version of Atmel Studio.
2. Install the software front-end and the USB driver.
3. Connect the AVR Dragon to the computer, and auto-install new hardware (AVR Dragon) on the computer.
4. Start the software front-end.
5. Connect AVR Dragon to the target.

3.1.1. USB Setup

In order to use the Atmel AVR Dragon it is required to install the USB driver first (comes with the software front-end). Do not connect the AVR Dragon to the computer before running the USB Setup in order to follow the procedures described in Software and USB Setup.

3.1.2. Unpacking the Atmel AVR Dragon

The box contains:

- Atmel AVR Dragon tool
- Internet link to Software (http://www.atmel.com/avrdragon)

There is no CD shipped with the AVR Dragon. The only way of getting the software is by downloading it directly from the Internet.

You will also need: (not included)

- PC with free USB connector or a USB HUB capable of delivering 500mA
- USB Cable
- Latest Atmel Studio - minimal requirement 4.12 SP2 (Link: http://www.atmel.com/avrdragon)
- 6/10 pin Header Connector (or similar cables to connect the AVR Dragon to the target board)

3.1.3. System Requirements

The minimum hardware and software requirements are:

1. Pentium (Pentium II and above is recommended).
2. Windows® 98, Windows ME, Windows 2000, or Windows XP.
3. 128 MB RAM.

4. AVR Studio® 4.12 with Service Pack 3 or Atmel Studio.
5. USB port, self-powered (500mA required).
6. Internet Connection for Software download.

Note:
Windows 95 and Windows NT does not support USB, hence cannot be used with AVR Dragon.

3.1.4. Hints

- Always power up the Atmel AVR Dragon first before connecting to or powering up the target
- AVR Dragon needs to sense the target voltage at pin 2 on the SPI(ISP) header or pin 4 on the JTAG header
- This also applies when using the High Voltage interface
- The High Voltage interface is set to 5V. Make sure the target board are running at 5V before using this interface off board.
- VCC header is set to 5V, and can source max. 300mA
- If AVR Dragon is used for programming/debugging targets on the Atmel STK500, the RESET jumper on the STK500 must be removed

3.2. Software and USB Setup

3.2.1. Software and USB Setup

In order to use the Atmel AVR Dragon it is required to install the USB driver. Do not connect the AVR Dragon to the computer before running the USB Setup. USB driver installation is done during the software front-end installation.

Note:
AVR Dragon requires AVR Studio 4.12 with Service Pack 3 or later, or Atmel Studio. Latest version of the Atmel Studio can be found at: http://www.atmel.com/atmelstudio.

Start the Atmel Studio installation. To install the USB driver, check the Install/Upgrade USB Driver checkbox, and the USB Driver will automatically be installed.

3.2.2. Install New Hardware on the Computer

When Atmel Studio and USB driver installation is finished, attach the USB cable to both the PC and the Atmel AVR Dragon. (The AVR Dragon is powered from the USB.) If it is the first time the AVR Dragon is connected to the computer, the box below will appear:

If running Windows XP you need to click "Next" a couple of times. Wait until the installation process completes by itself. It may take from a few seconds up to a few minutes depending on the computer and operating system.

If the USB driver is correctly installed and AVR Dragon is connect to the PC, the green LED next to the USB connector will be lit.

If the software front-end for some reason can't detect the AVR Dragon after the USB setup, try to restart the computer in order to get the driver properly loaded.

3.2.3. Install USB Driver after Atmel Studio is Installed

The USB driver can be installed even after Atmel Studio has been installed by following these steps:

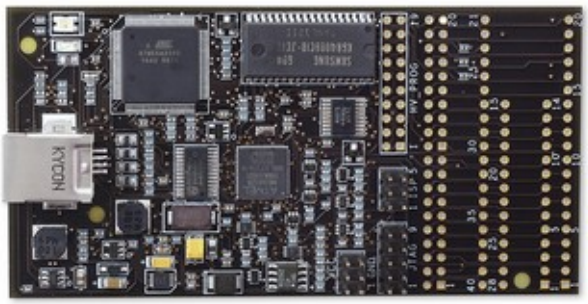
1. Open "**Control Panel**" on the PC (Windows 95 and Windows NT does not support USB).
2. Select "**Add or Remove Programs**".

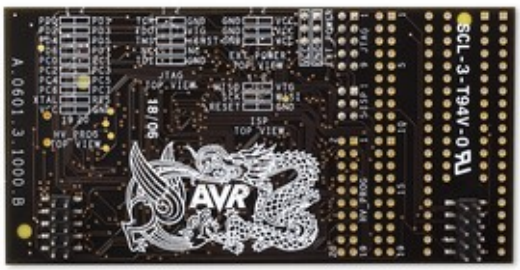
3. Select "**Atmel Studio**" or "**Atmel Studio**" in the list of programs.
4. Click on the "**Change**" button.
5. Select "**Modify**".
6. Select "**Install/upgrade USB Driver**".

The USB driver is now properly installed on the PC.

Note:
The Atmel AVR Dragon requires a USB port that can deliver 500mA (self-powered USB hub).

3.3. Board Description / Headers

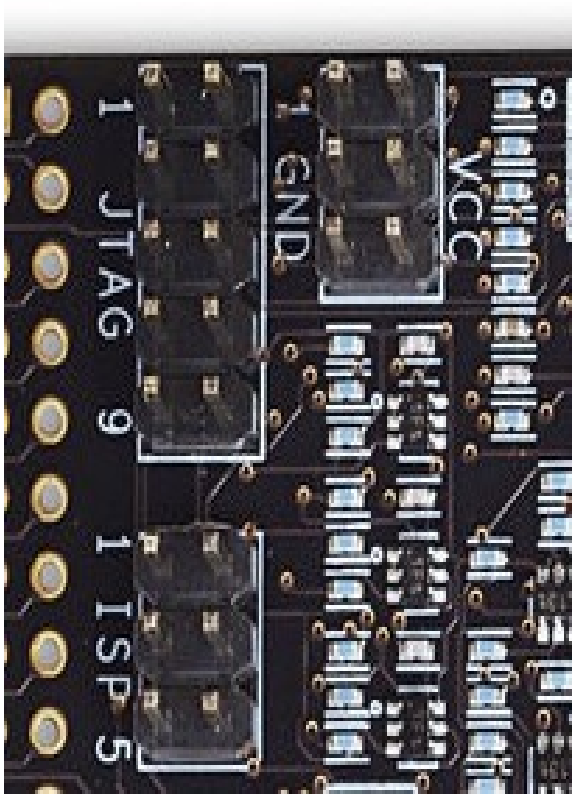




3.3.1. Headers

Out of the box, the Atmel AVR Dragon has the following three header connectors mounted:

- SPI(ISP) Header - Used for SPI(ISP) programming and debugWIRE OCD.
- JTAG Header - Used for JTAG programming and JTAG OCD.
- VCC Header - Used for powering Devices placed in the prototype area, or to power external target boards (max. 300mA). Set to fixed 5V.



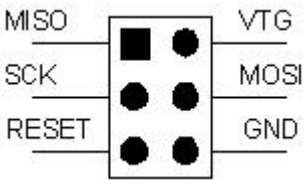
The following headers are not mounted:

- HV_PROG header
- EXPAND header
- 40-pin DIP socket
- 28-pin DIP socket

3.3.1.1. SPI (ISP) Header (mounted)

This 6-pin header uses the standard AVR SPI (ISP) pinout for easy connection to external targets. The signals are level-converted to allow communication with targets running at any voltage between 1.8 and 5.5V.

Note that the target voltage must be applied to pin 2 on the SPI (ISP) header for the Atmel AVR Dragons level converters.



3.3.1.2. JTAG Header (mounted)

The 10-pin JTAG header is a standard pinout JTAG connector. When connecting the Atmel AVR Dragon JTAG header to an external target, the signals are level converted to match the target board voltage. This is done automatically. Note that the AVR Dragon will not power the target through the JTAG interface. The

target needs to be powered through a dedicated power supply or by powering it using the VCC connector (5.0V max. 300mA). AVR Dragon needs to sense the target voltage on pin 4 on the JTAG connector.

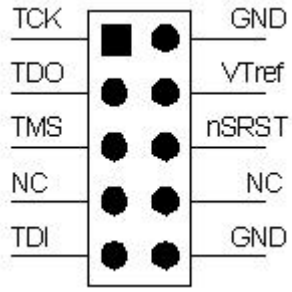
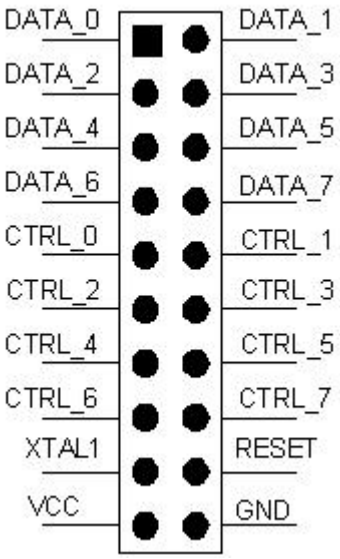


Table 3-1. JTAG Header Pinout and Description

Pin	Signal	I/O	Description
1	TCK	Output	Test Clock, clock signal from AVR Dragon to target JTAG port
2	GND	-	Ground
3	TDO	Input	Test Data Output, data signal from target JTAG port to AVR Dragon
4	VTref	Input	Target reference voltage. VDD from target used to control level-converters.
5	TMS	Output	Test Mode Select, mode select signal from AVR Dragon to target JTAG port
6	nSRST	In/Out-put	Open collector output from adapter to the target system reset. This pin is also an input to the adapter so that the reset initiated on the target may be reported to the AVR Dragon.
7	-	-	Not connected
8	-	-	Not Connected
9	TDI	Output	Test Data Input, data signal from AVR Dragon to target JTAG port
10	GND	-	Ground

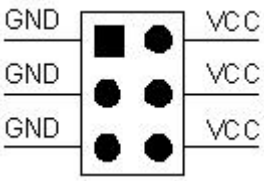
3.3.1.3. HV_PROG Header (not mounted)

The HV_PROG connector contains all signals required to do HVSP or PP programming. The signals on this connector is not level-converted, and should only be connected to the EXPAND connector on the AVR Dragon. You could damage both your target and the Atmel AVR Dragon if you try to do HVSP or PP on an external target board. The signal levels on the HV_PROG header are 5V.



The figure above shows the general pinout of the HV_PROG header. The High Voltage programming pin mapping is not the same for all parts. See the HVSP Description or PP Description chapters for more information on the pinout of this header.

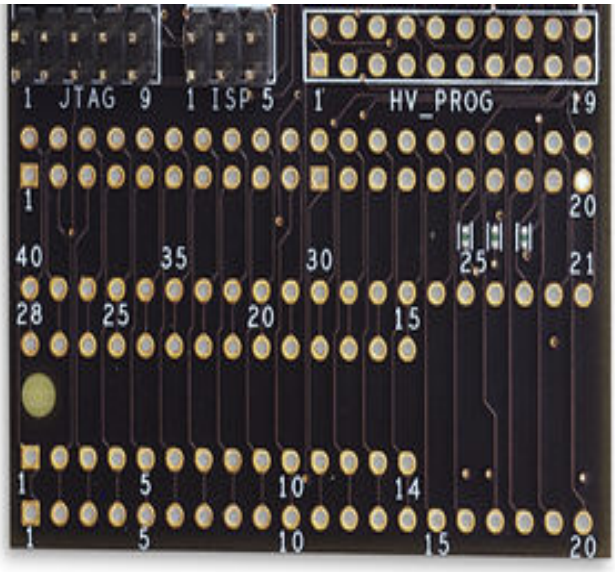
3.3.1.4. VCC Header (mounted)



The VCC Header contains 5.0 Volt VCC and GND that must be used to power the target device placed in the prototype area of the Atmel AVR Dragon board. The voltage can also be used to power an external target board, but it is important that the current consumption is less than 300mA. Note that the AVR Dragon current sourcing capabilities are also limited by the amount of current the Host USB controller can deliver.

Note:
If the current consumption excedes 300mA, then this may lead to errors or disconnects during programming or debugging. If the AVR Dragon starts misbehaving try to remove the load by applying external power to your circuitry and removing the VCC header connections.

3.3.1.5. EXPAND Header (not mounted)



The expand connector is directly mapped to the 28- and 40-pin DIP sockets. Pin 1 on the connector - is pin one on both the 28 and the 40pin DIP socket. When doing either programming or debugging on-board, the appropriate signals should be routed from the SPI(ISP), JTAG, VCC, and HV_PROG headers to the correct pins on the EXPAND connector. Read the Using the Atmel AVR Dragon Prototype Area section for more information on how to use this function.

3.3.1.6. Status LEDs

Two LEDs show the status of the Atmel AVR Dragon. Check the Troubleshooting Guide to check for solutions if there are any errors.

Table 3-2. LEDs Pinout and Description

LED #	Color	Description
2	Green	Indicates USB traffic
1	Red	Idle, not connected to the software front-end
	Dark	Idle, connected to the software front-end
	Green	Data Transfer
	Yellow	Firmware Upgrade or Initialization

4. Connecting the Atmel AVR Dragon

4.1. Connecting to Target through the JTAG Interface

A minimum of six wires is required to connect Atmel AVR Dragon to the target board. These Signals are TCK, TDO, TDI, TMS, VTref, and GND.

Optional line is the nSRST. The nTRST signal is not used, and is reserved for compatibility with other equipment.

nSRST is used to control and monitor the target reset line. This is however not necessary for correct debugging. But if the application code sets the JTD bit in the MCUCSR, the JTAG Interface will be disabled. For the AVR Dragon to reprogram the target AVR, it will need to have control of the Reset Pin.

The figures in Connecting Atmel AVR Dragon to Target Board shows which JTAG lines should be connected to the target AVR to ensure correct operation. To avoid drive contention on the lines it is recommended that series resistors are placed between the JTAG lines and external circuitry. The value of the resistors should be chosen so that the external circuitry and the AVR device do not exceed their maximum ratings (i.e. sinks or sources to much current).

4.1.1. Connecting Atmel AVR Dragon to Target Board

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. Connecting the JTAG connector to external target

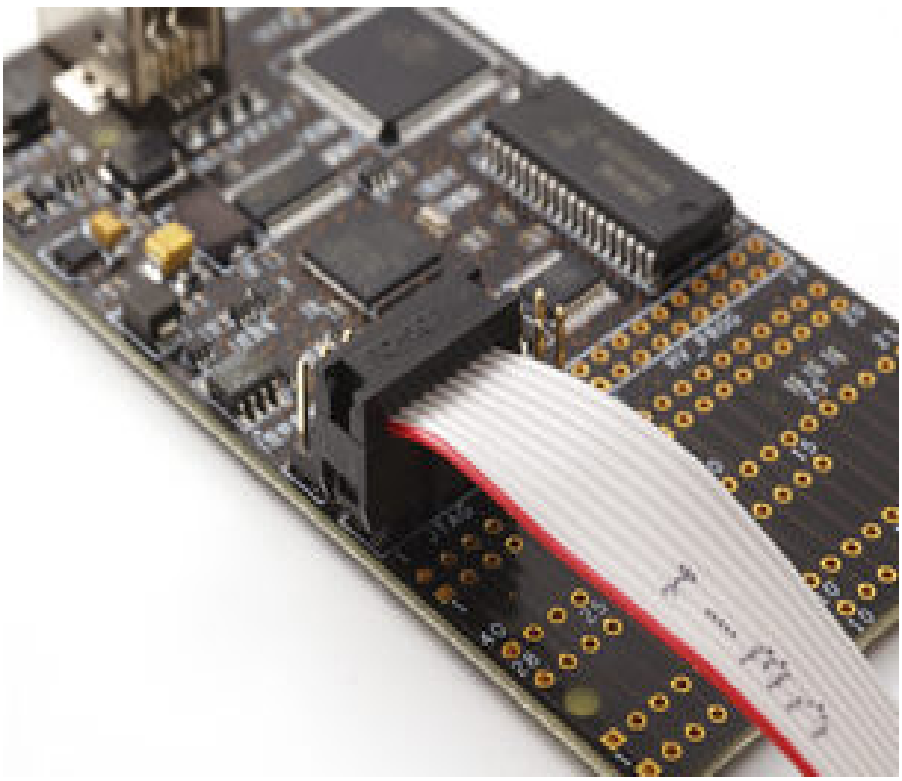


Figure 4-2. Connections needed to access external targets through JTAG interface

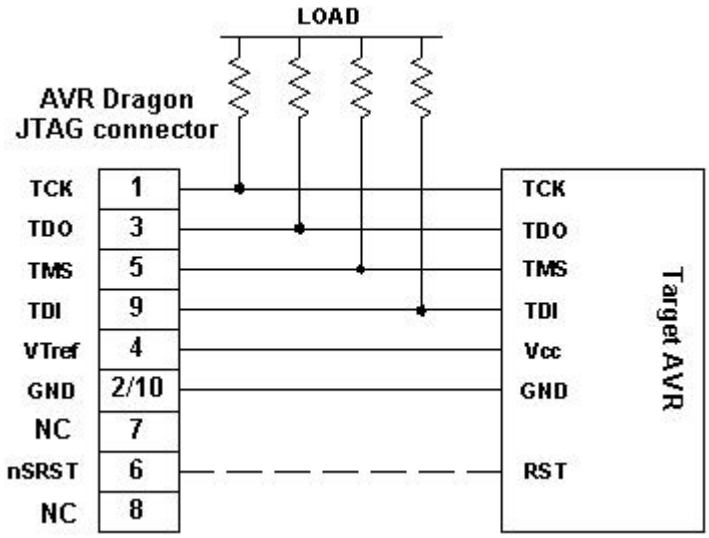


Figure 4-3. JTAG connector pinout

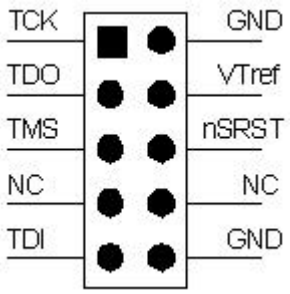


Table 4-1. JTAG Pin Description

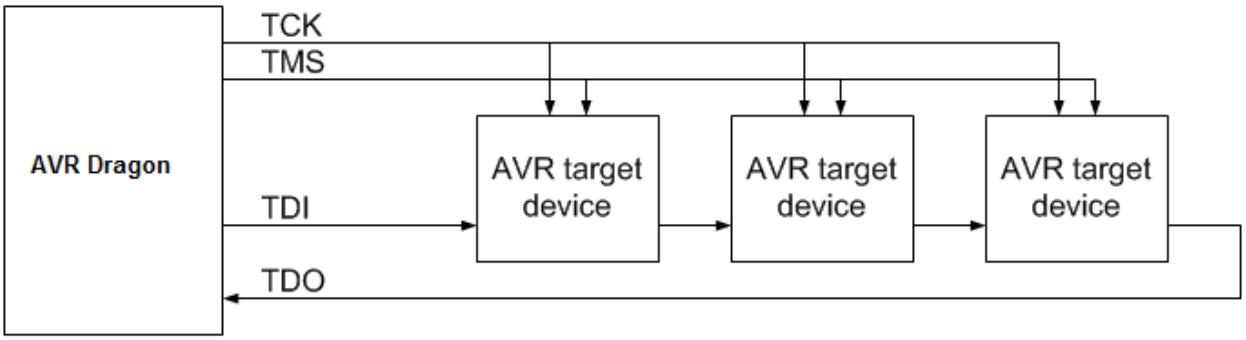
Name	Pin	Description
TCK	1	Test Clock (clock signal from the AVR Dragon into the target device)
TMS	5	Test Mode Select (control signal from the AVR Dragon into the target device)
TDI	9	Test Data In (data transmitted from the AVR Dragon into the target device)
TDO	3	Test Data Out (data transmitted from the target device into the AVR Dragon)
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 AVR Dragon to hold the target device in a reset state, which can be essential to debugging in certain scenarios.

Name	Pin	Description
VTref	4	Target voltage reference. The AVR Dragon samples the target voltage on this pin in order to power the level converters correctly. The AVR Dragon draws less than 1mA from this pin.
GND	2, 10	Ground. Both must be connected to ensure that the AVR Dragon and the target device share the same ground reference.

4.1.2. Connecting Atmel AVR Dragon to Several Devices Placed in a JTAG Chain

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.

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

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 8-bit microcontrollers have 4 IR bits)

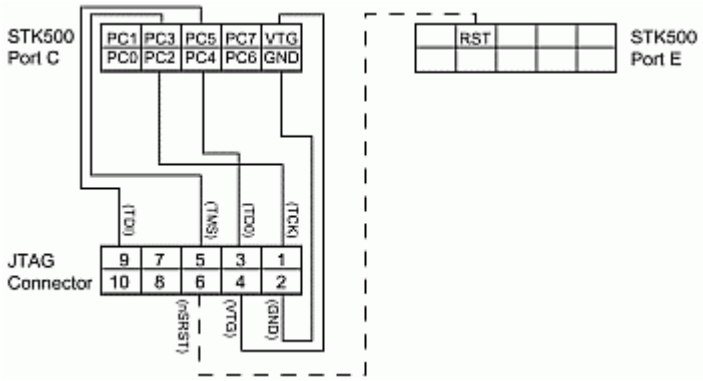
Instruction bits before: 5 (AVR 32-bit microcontrollers have 5 IR bits)

4.1.3. Connecting Atmel AVR Dragon to Atmel STK500

Atmel STK500 does not have a dedicated JTAG interface connector. To connect the Atmel AVR Dragon to the STK500 board, the JTAG Probe must be strapped to the appropriate JTAG Port Pins of the target device using a squid cable. Alternatively an STK500 JTAG adapter can be used, see section The STK500 JTAG adapter. Check the target device datasheet for the location of the JTAG pins on the appropriate device. The figure below shows an example on how the pins should be connected for an ATmega32 on the STK500. Remember to remove the reset jumper on the STK500 if the reset pin is going to be controlled from the AVR Dragon.

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

4.1.3.1. Example: Connecting Atmel AVR Dragon to Atmel STK500 with ATmega32



4.1.3.2. Atmel STK500 JTAG Adapter

The Atmel STK500 JTAG Adapter, that is shipped with the STK500 (and previously with the JTAGICE mkII), can be used to simplify the connection to the STK500 for Atmel AVR devices with JTAG that mates with socket SCKT3100A3 and SCKT3000D3 on the STK500.

4.1.4. Enabling the JTAG Enable Fuse

If the JTAGEN fuse (JTAG Enable) in the target device is un-programmed, the JTAG Interface will be disabled. This fuse cannot be programmed through the JTAG Interface and must therefore be programmed through the SPI Interface or High Voltage Serial or Parallel Interface.

4.2. Connecting to Target through the debugWIRE Interface

A minimum of three wires are required for communication between Atmel AVR Dragon and the target board with the debugWIRE interface. These signals are RESET, VTref, and GND.

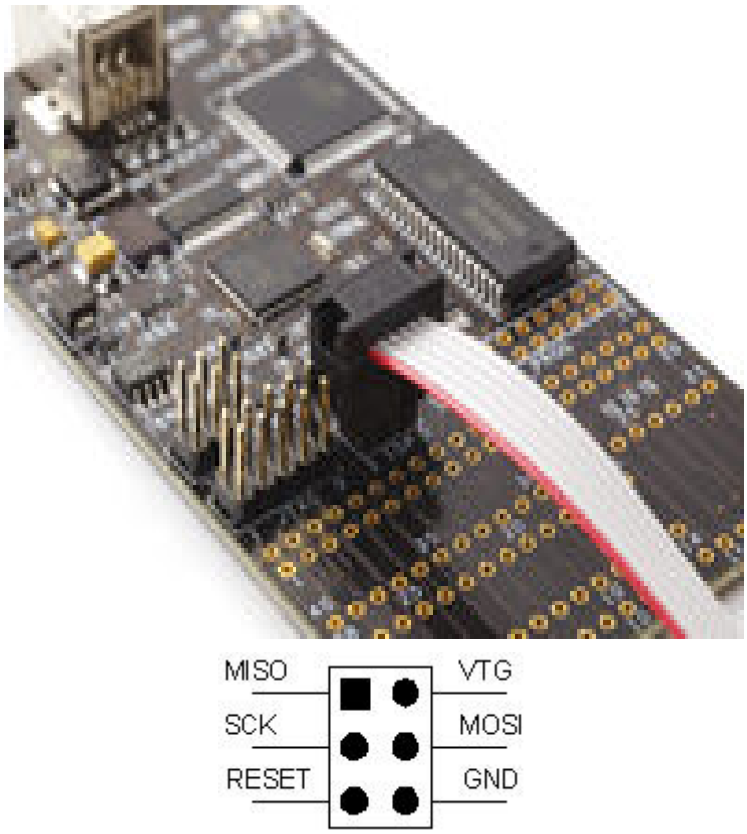
Important!

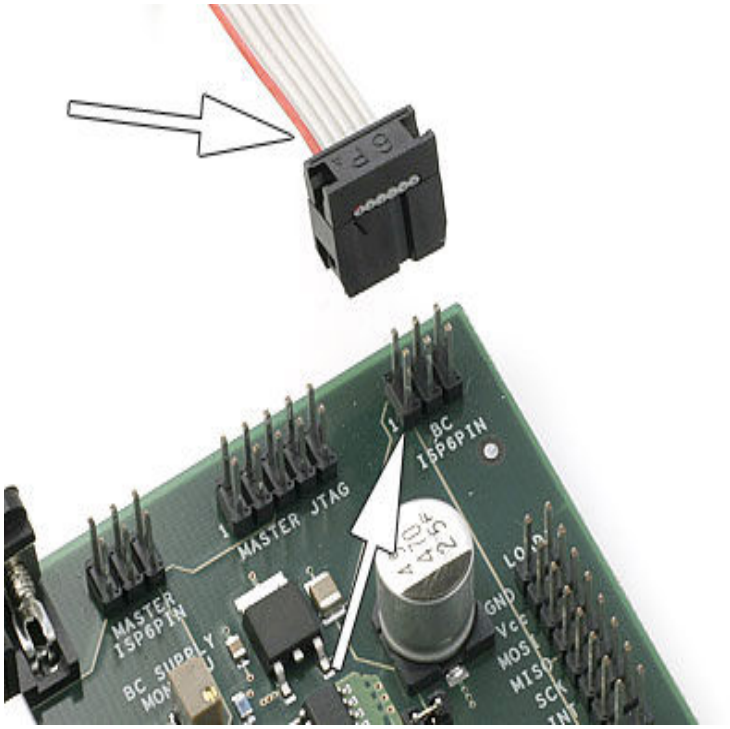
This interface uses only 1 pin, (RESET pin) for communication with the target. To enable the debugWIRE interface on an AVR Device, the debugWIRE Enable fuse (DWEN) must be programmed, (DWEN=0). AVR devices with debugWIRE interface are shipped with the DWEN fuse un-programmed from the factory. The debugWIRE interface itself cannot enable this fuse. The DWEN fuse can be programmed through SPI programming mode, which requires connection to a 6-pin header. For this reason it is recommended to place the full 6-pin SPI connector on your target board to simplify debugging and programming.

NOTE: 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!

If using this connection from AVR Dragon on a Atmel STK500, be sure to detach the RESET jumper on the STK500. And connect to the correct ISP header for the actual AVR device, guided by the color code in the STK500 silk-print.

4.2.1. Atmel AVR Dragon debugWIRE Connector





4.2.2. Connecting Atmel AVR Dragon Probe to 6-pins SPI Header using a 6-pin Cable

When the DWEN fuse is programmed, there is only need for the GND, VTref, and RESET lines to be able to use the debugWIRE interface. However, to ease the task of changing between SPI programming mode and debugWIRE mode, it is recommended to use debugWIRE with all six lines connected. The SPI pins will not be driven by the Dragon when running debugWIRE, but pull-up resistors will still be active.

4.2.3. Re-enabling the SPI Interface

By following the description below, the SPI Interface is re-enabled.

1. Connect the Atmel AVR Dragon to the target with SPI (6-pin connection) as described above.
2. Load a project and Start a debug session using the "Start Debugging" command (found under the Debug pull-down menu in Atmel Studio).
3. In the debug menu, you should now be able to choose "Disable debugWIRE and close ".

Note:
Some precautions regarding the RESET line must be taken to ensure proper communication over the debugWIRE interface. If there is a pull-up on the RESET line, this resistor must be larger than 10kΩ, and there should be no capacitive load. The pull-up resistor is not required for debugWIRE functionality. Other logic connected to the RESET line should be removed.

It's 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, hence clear the lockbits. When the lockbits are cleared the debugWIRE interface will be re-enabled.

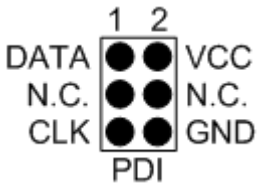
The SPI Interface is only capable of reading fuses, signature, and do a chip erase when the DWEN fuse is unprogrammed.

4.3. PDI Programming

In System Programming using PDI is well suited for programming devices soldered onto external target boards. This section explains how to connect the Atmel AVR Dragon to PDI program an external target. The PDI lines are equipped with level converters that automatically will level shift the AVR Dragon signals to the target board voltage.

It is recommended that a 6-pin header connector with 2.54mm (100 MIL) spacing is placed on the target board to allow easy access to the PDI programming interface. The following pinout should be used.

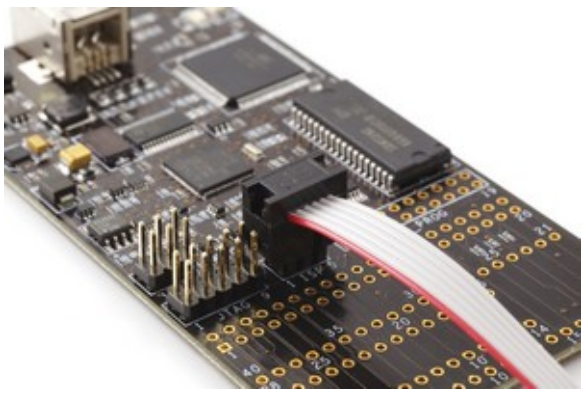
Figure 4-5. 6pin Header Connector with 2.54mm (100 MIL) Spacing

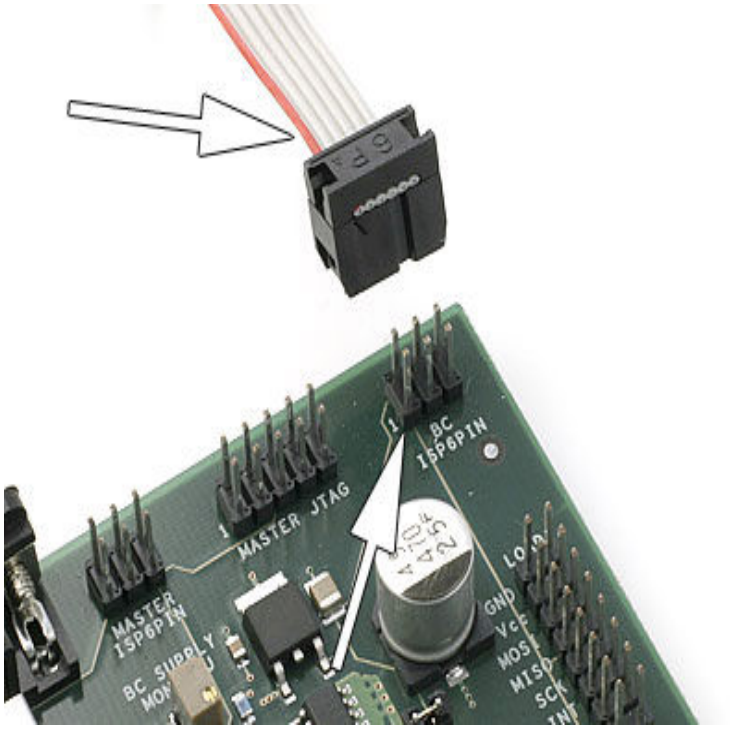


Note:
When connecting the AVR Dragon to the target, connect DATA to DATA pin on the target device, CLK to CLK, and so on.

Note:
AVR Dragon must sense the target voltage on pin 2 on the PDI header in order to set up the level-converters. When using off-board targets there should be no connection between the VCC header and pin 2 of the PDI header.

Connect the 6pin cable from the AVR Dragon to the external target as shown in these pictures:



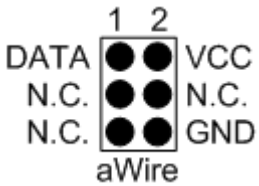


4.4. aWire Programming

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

When designing an application PCB, which includes an AVR with the aWire interface, it is recommended to use the pinout as shown in Figure 4-6 aWire Header Pinout.

Figure 4-6. aWire Header Pinout



Tip:
Since aWire is a half-duplex interface, a pull-up resistor on the RESET line in the order of 47kΩ 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.

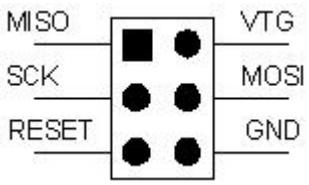
4.5. SPI Programming

In System Programming using SPI is well suited for programming devices soldered onto external target boards. This section explains how to connect the Atmel AVR Dragon to SPI program an external target.

The SPI lines are equipped with level converters that automatically will level shift the AVR Dragon signals to the target board voltage.

It is recommended that a 6-pin header connector with 2.54mm (100 MIL) spacing is placed on the target board to allow easy access to the SPI programming interface. The following pinout should be used.

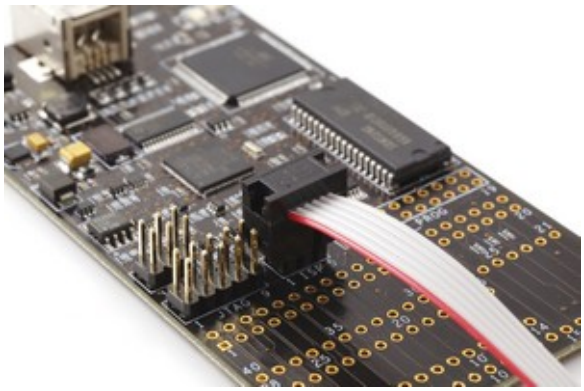
Figure 4-7. 6pin Header Connector with 2.54mm (100 MIL) Spacing

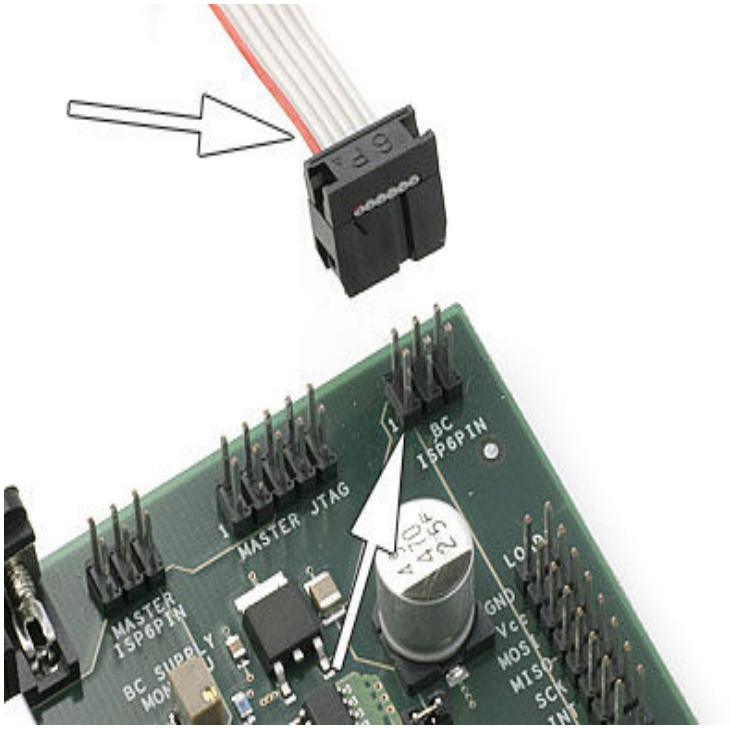


Note:
When connecting the AVR Dragon to the target, connect MISO to MISO pin on the target device, MOSI to MOSI, and so on.

Note:
AVR Dragon must sense the target voltage on pin 2 on the SPI header in order to set up the level-converters. For on-board targets, the voltage must be supplied from pin 2, 4, 6 on the VCC header (5V) into pin 2 (VTG) on the SPI header. When using off-board targets there should be no connection between the VCC header and pin 2 of the SPI header.

Connect the 6pin cable from the AVR Dragon to the external target as shown in these pictures:





debugWIRE OCD interface is also accessed through this SPI header.

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!

4.6. Parallel Programming Description

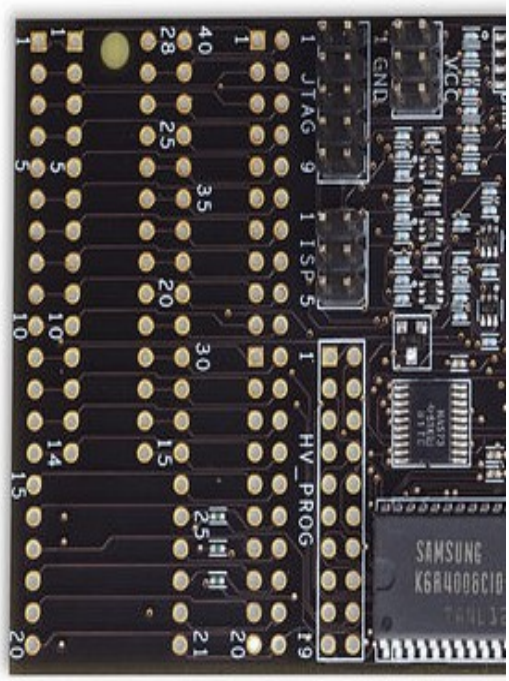
High pin count Atmel AVR devices support the full Parallel Programming (PP) interface. This interface offers high speed programming, and also supports programming all fuse and lock bits in the AVR Device.

Attention:
Extreme care should be taken if using PP mode to program an AVR device on an external target. The PP lines do not have level converters, so it is important that the target board is powered by the AVR Dragon VCC header, and not using its own power supply. In addition the AVR Dragon will apply 12V to the reset pin, so it is important that the target board is designed to handle 12V on this line.

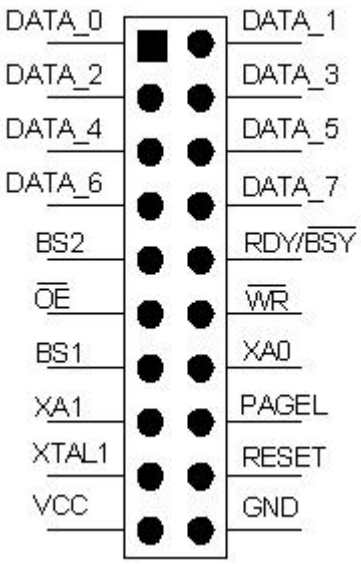
Note:
The target voltage, i.e. the 5V from the VCC header must be applied to either pin 2 on the SPI header or pin 4 on the JTAG header. This is because the AVR Dragon must read the target voltage.

To avoid damaging the Target Board, the AVR Dragon or both, it is recommended to **only** use PP mode on devices placed in the 28/40 pin DIP socket on the AVR Prototype area on the AVR Dragon.

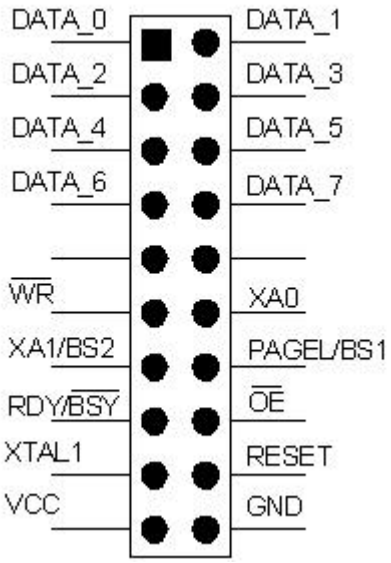
Figure 4-8. Prototype Area



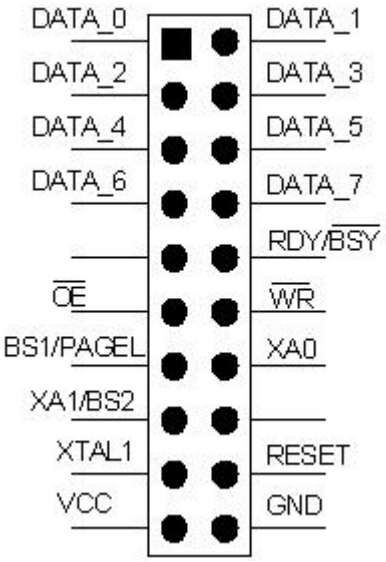
The HV_PROG header pinout is listed below. This is the standard pinout for about all Atmel AVR parts. However, the pinout on the HV_PROG header is slightly different for some parts. These exceptions are listed further down in this page.



For ATtiny26/261/461/861 the HV_PROG header will have this pinout:



For ATtiny2313 the HV_PROG header will have this pinout:



See the Device Connection Sheet section for information on how to connect AVR Dragon for PP programming.

4.7. High Voltage Serial Programming Description

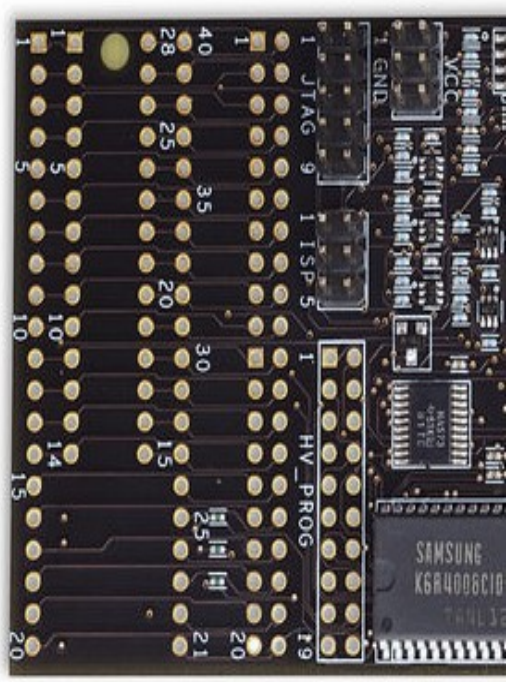
Low pin count Atmel AVR devices do not have enough I/O pins to support the full Parallel Programming interface. These devices use HVSP programming instead, which is a serial version of the Parallel Programming interface.

Attention:
Extreme care should be taken if using HVSP mode to program an AVR device on an external target. The HVSP lines do not have level converters, so it is important that the target board is powered by the AVR Dragon VCC header, and **not** using another power supply. In addition the AVR Dragon will apply 12V to the reset pin, so it is important that the target board is designed to handle 12V on this line.

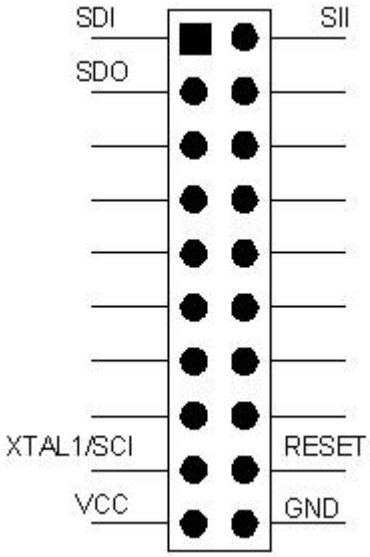
Note:
The target voltage, i.e. the 5V from the VCC header, must be applied to either pin 2 on the SPI header or pin 4 on the JTAG header. This is because the AVR Dragon must read the target voltage.

To avoid damaging the Target Board, the AVR Dragon or both, it is recommended to **only** use HVSP mode on devices placed in the 28/40 pin DIP socket on the AVR Prototype area on the AVR Dragon.

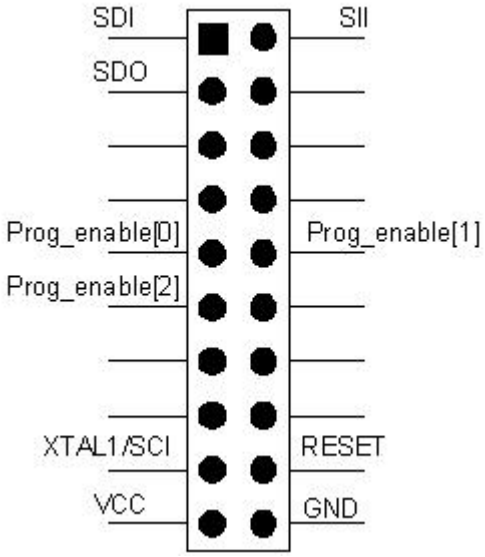
Figure 4-9. Prototype Area



The High Voltage Serial Programming (HVSP) interface is found on the HV_PROG header. The figure below shows the pinout of the HV_PROG header when used for HVSP for all parts except ATtiny24/44/84:

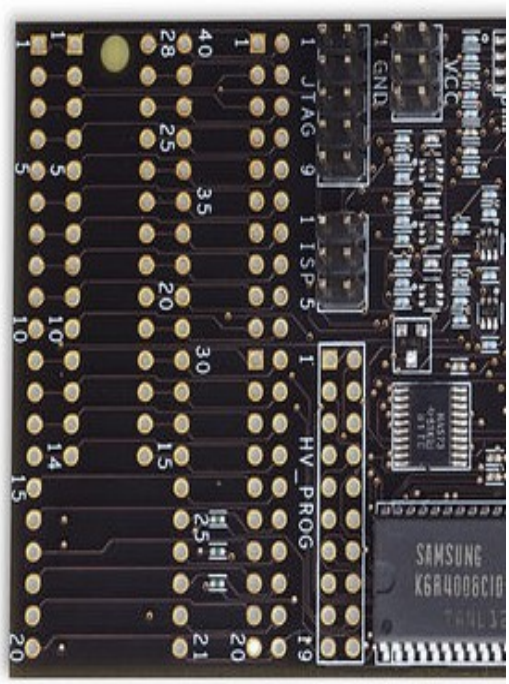


The ATtiny24/44/84 uses separate pins for entering programming mode. See the pinout of the HV_PROG header when used for these parts below:



See the Device Connection Sheet section for information on how to connect AVR Dragon for HVSP programming of the different supported devices.

5. Using the Onboard Prototype Area

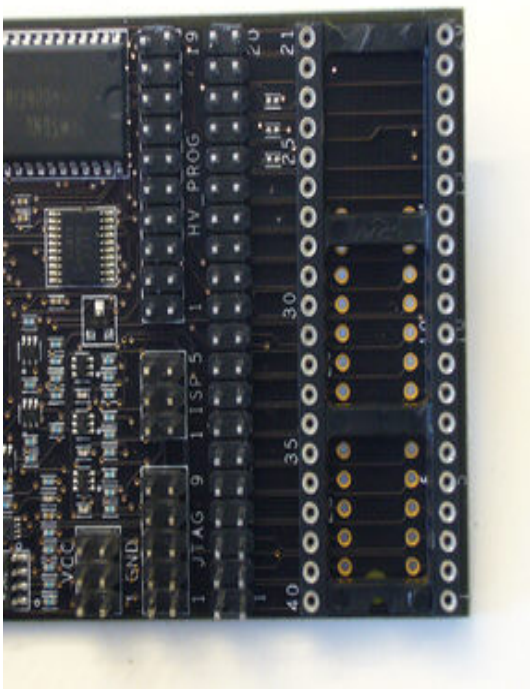


The Atmel AVR Dragon has layout for a 40-pin and a 28-pin PDIP socket. The DIP socket pins are connected directly to the 40-pin Header connector. By strapping the SPI, JTAG, HV_PROG, and VCC header signals to the 40-pin header connector, programming and debugging can be preformed without the need of an external target board.

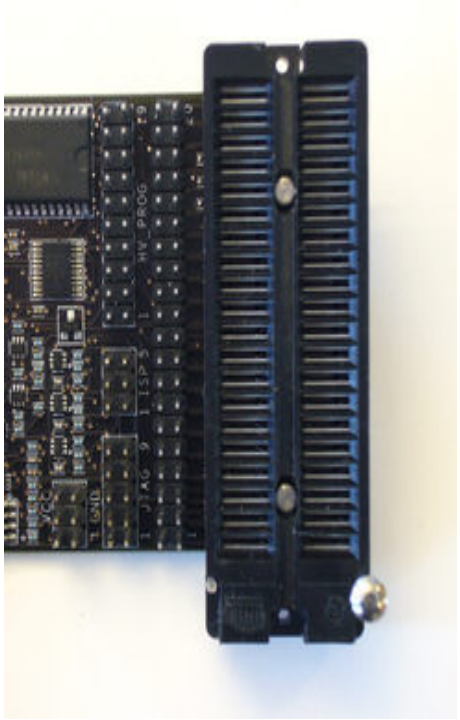
This section shows how to strap the AVR Dragon for different operation modes. Each supported AVR device has its own Device Connection Sheet containing all information required.

There is a number of ways to utilize the prototype area. If only one device type/programming mode is to be used, the easiest and cheapest way is to just solder wires directly from the HV_PROG, SPI, JTAG, and VCC headers to the EXPAND header. However, to make the board more flexible, header connectors can be soldered in to allow connecting cables without soldering.

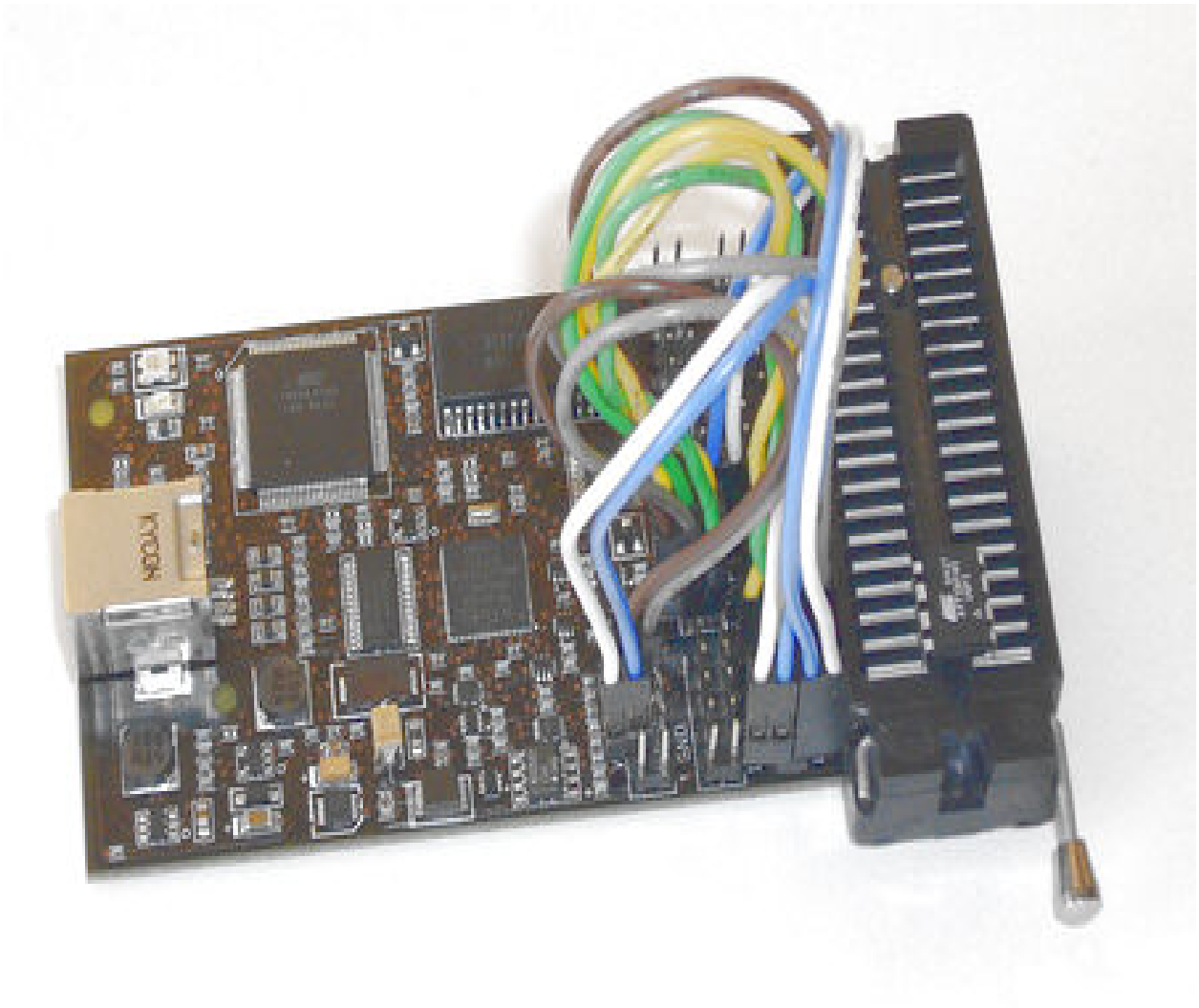
Here is a suggestion on how to modify the AVR Dragon board to make it flexible and able to use all DIP socket AVR devices.



In this picture one 20-pin header connector, a 40-pin header connector and a 40-pin DIP socket has been soldered onto the AVR Dragon.



To make it even more flexible and allow for narrow DIP packages, a ZIF (Zero Insertion Force) DIP socket has been added in the picture above. Additional sockets can be bought from third party vendors to support MLF/QFN, TQFP, SOIC, etc. packages. www.atmel.com/products/AVR/thirdparty.asp#adapters.



And finally the complete setup for debugWIRE and SPI programming of the ATtiny45. For details on how this is connected, have a look at the ATtiny45 Device connection sheet.

6. Device Connection Sheets

6.1. Devicesheet: SCKT3100A3

- Supported Programming Modes: SPI, Parallel programming, and JTAG
- Supported Debugging Modes: JTAG
- applicable to the following devices:
  - **ATmega16, ATmega32 (and 'A' variants)**
  - **ATmega164, ATmega324, ATmega644, and ATmega1284 family (and 'P', 'A' and 'PA' variants)**

The following pictures shows how to connect these devices to the Atmel AVR Dragon:

Figure 6-1. SPI Programming

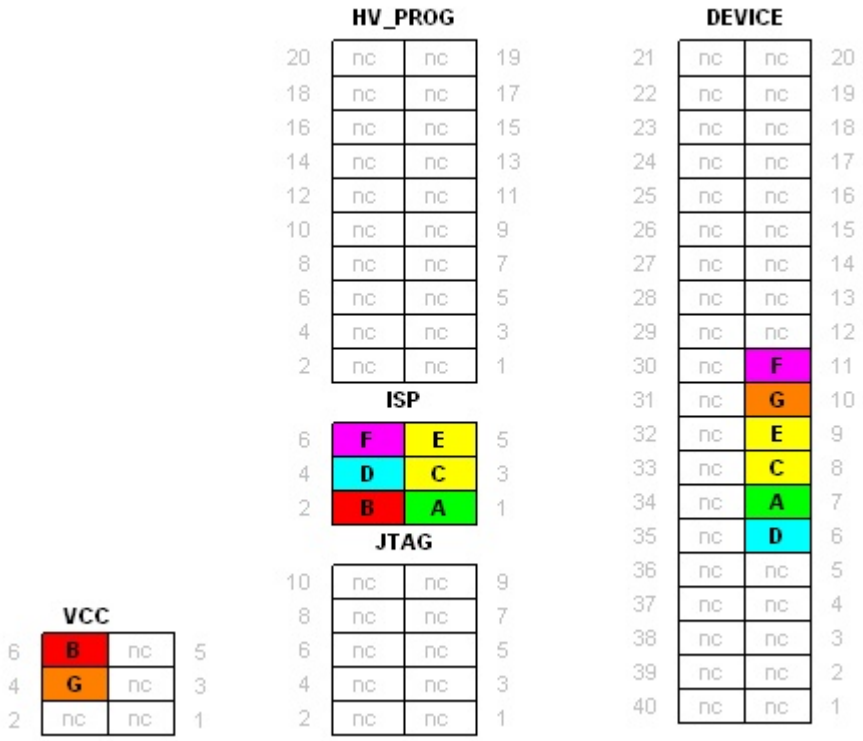


Figure 6-2. JTAG Programming and Debugging

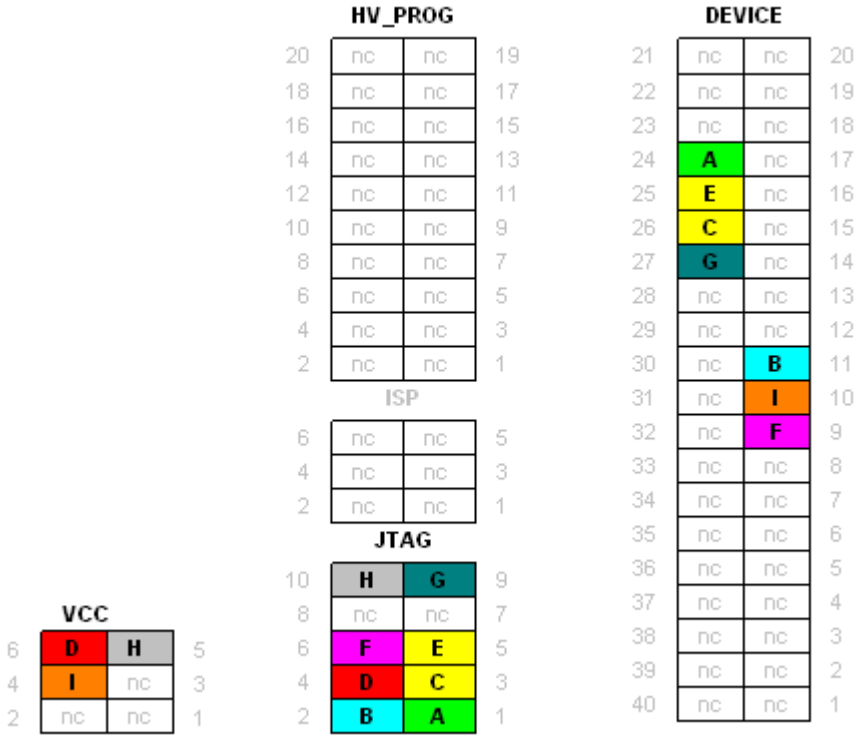
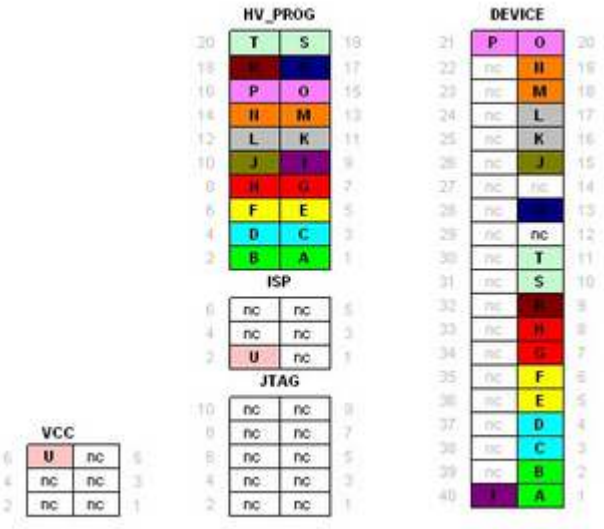


Figure 6-3. Parallel Programming



6.2. Devicesheet: SCKT3200D2

- Supported Programming Modes: SPI, and Parallel programming
- Supported Debugging Modes: debugWIRE
- applicable to the following devices:
  - **ATmega48, ATmega88, and ATmega168 family (and 'P', 'A' and 'PA' variants)**
  - **ATmega328 (and 'P' variant)**

- **ATtiny48, ATtiny88 family**

The following pictures shows how to connect these devices to the Atmel AVR Dragon:

Figure 6-4. SPI Programming and debugWIRE Debugging

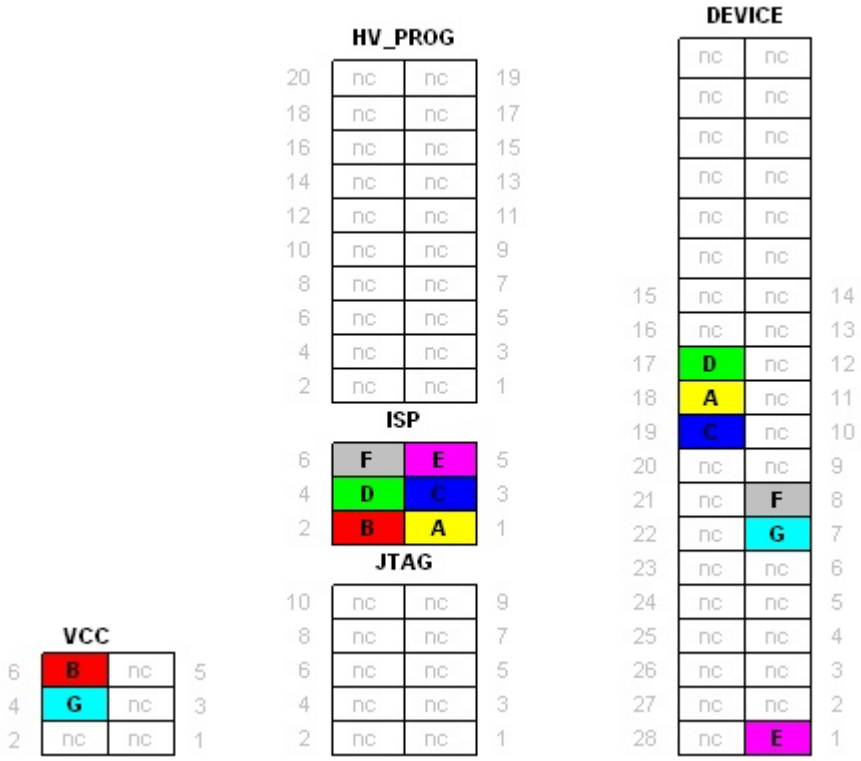
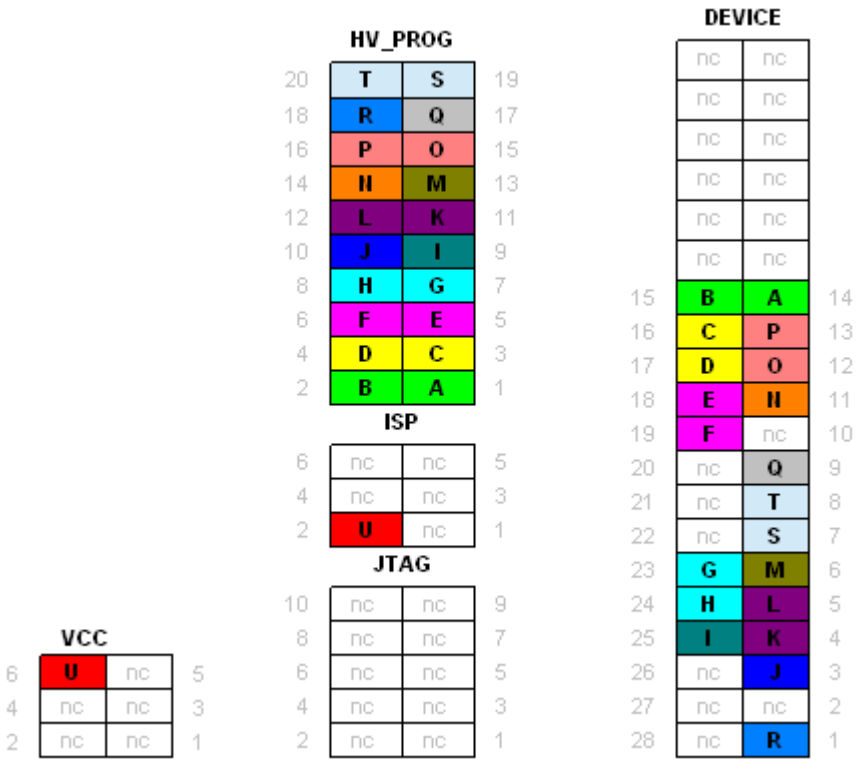


Figure 6-5. Parallel Programming



6.3. Devicesheet: SCKT3300D3

- Supported Programming Modes: SPI, and Parallel programming
- Supported Debugging Modes: debugWIRE
- applicable to the following devices:
  - **ATtiny2313 and ATtiny4313 family (and 'A' variants)**

The following pictures shows how to connect these devices to the Atmel AVR Dragon:

Figure 6-6. SPI Programming and debugWIRE Debugging

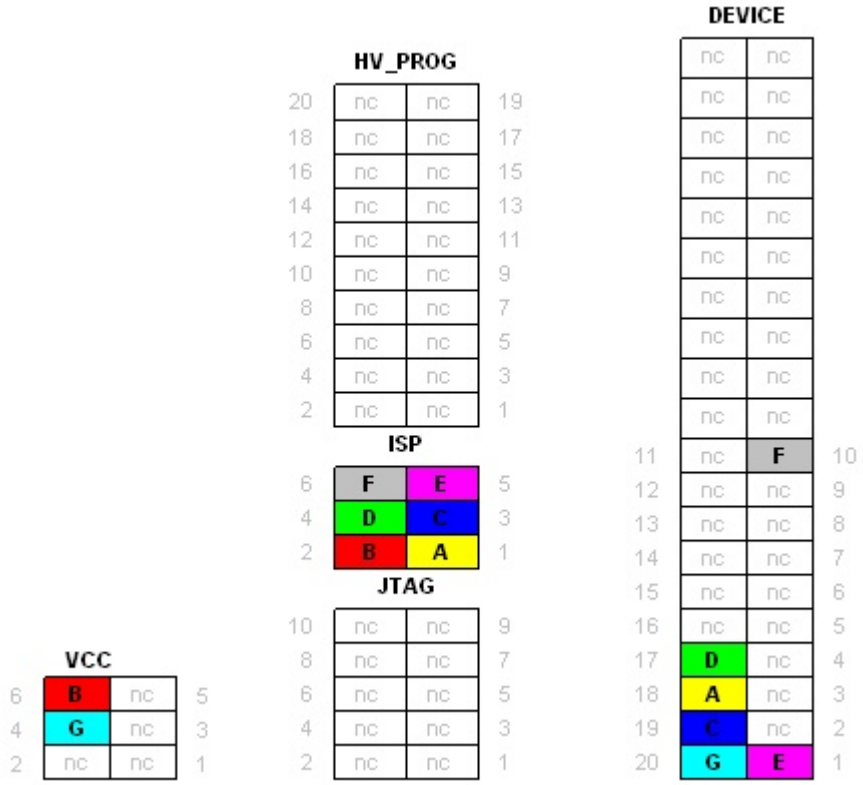
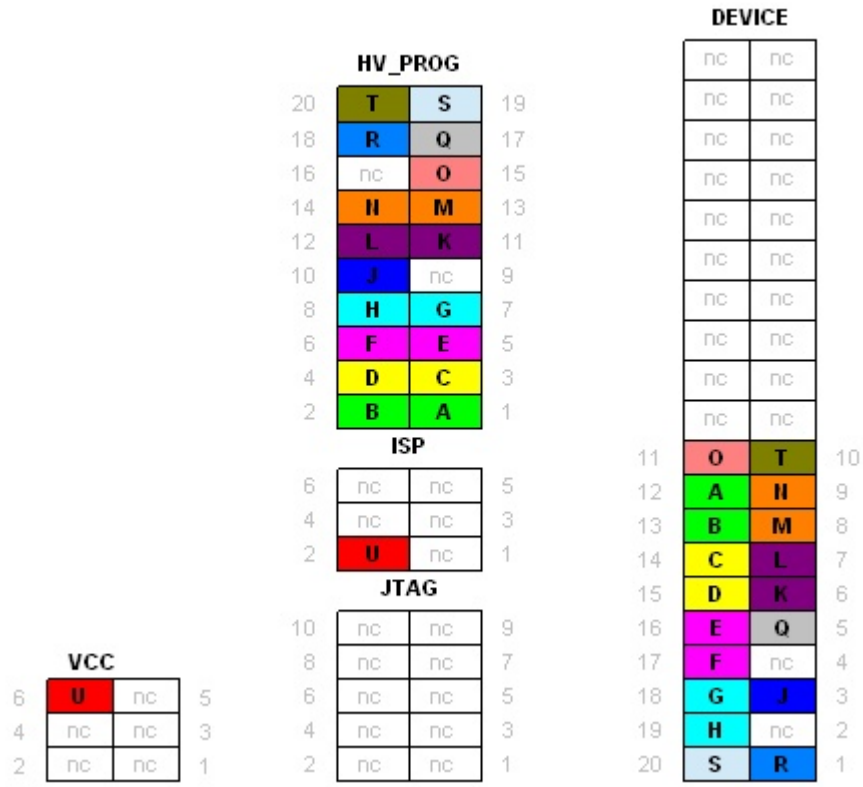


Figure 6-7. Parallel Programming



6.4. Devicesheet: SCKT3400D1

- Supported Programming Modes: SPI, and High Voltage Serial Programming
- Supported Debugging Modes: debugWIRE
- applicable to the following devices:
  - **ATtiny13 (and 'A' variant)**
  - **ATtiny25, ATtiny45, and ATtiny85 family**

The following pictures shows how to connect these devices to the Atmel AVR Dragon:

Figure 6-8. SPI Programming and debugWIRE Debugging

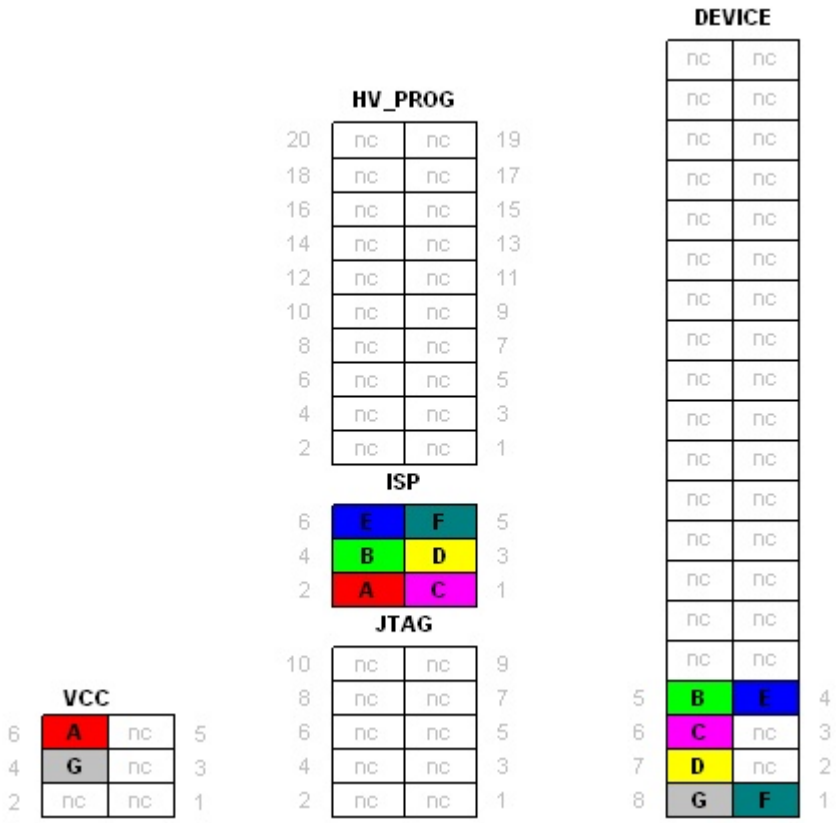
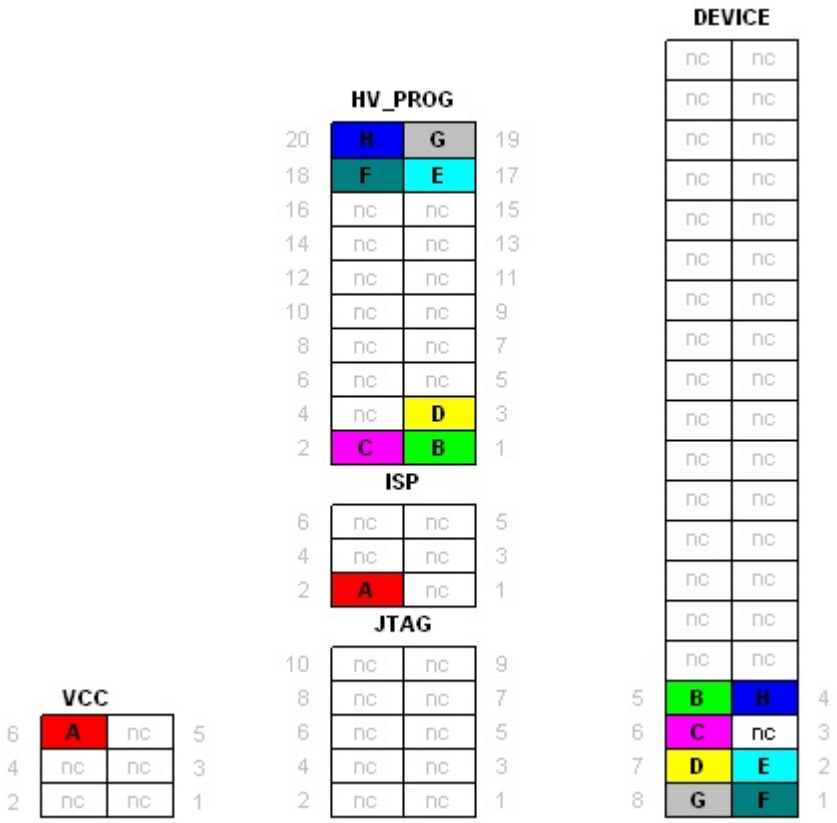


Figure 6-9. High Voltage Serial Programming

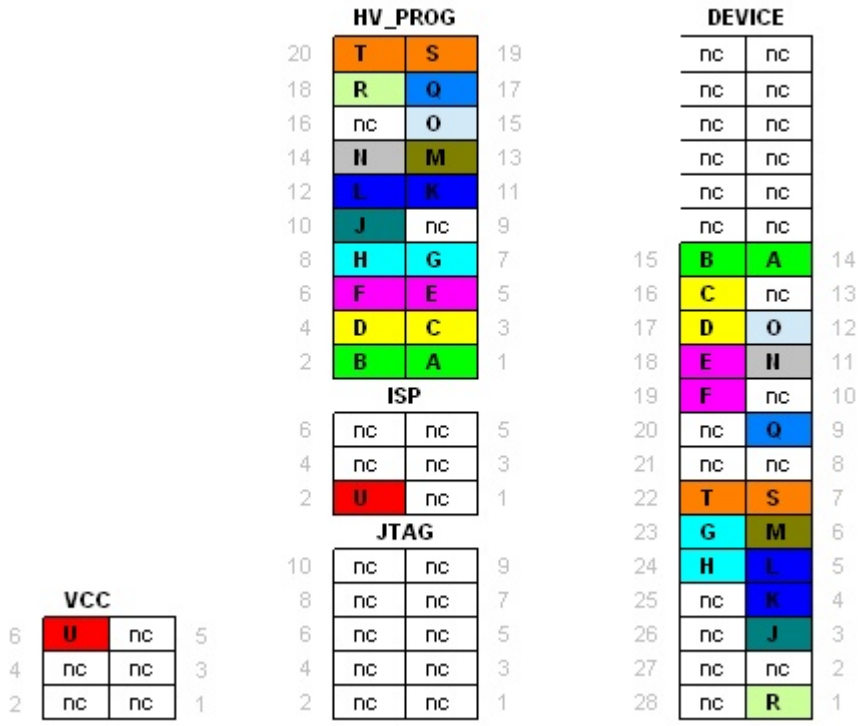


6.5. Devicesheet: SCKT3500D-

- Supported Programming Modes: Parallel programming
- Supported Debugging Modes: **none**
- applicable to the following devices:
  - **ATtiny28** - Note: Only High Voltage Programming available for tiny28

The following pictures shows how to connect these devices to the Atmel AVR Dragon:

Figure 6-10. Parallel Programming



6.6. Devicesheet: SCKT3700A1

- Supported Programming Modes: SPI and Parallel programming
- Supported Debugging Modes: debugWIRE
- applicable to the following devices:
  - **ATtiny26** - Note: debugWIRE not supported by tiny26, PP - prog. issue, see the Known issues section.
  - **ATtiny261, ATtiny461, and ATtiny861 family (and 'A' variants)**

The following pictures shows how to connect these devices to the Atmel AVR Dragon:

Figure 6-11. SPI Programming and debugWIRE Debugging

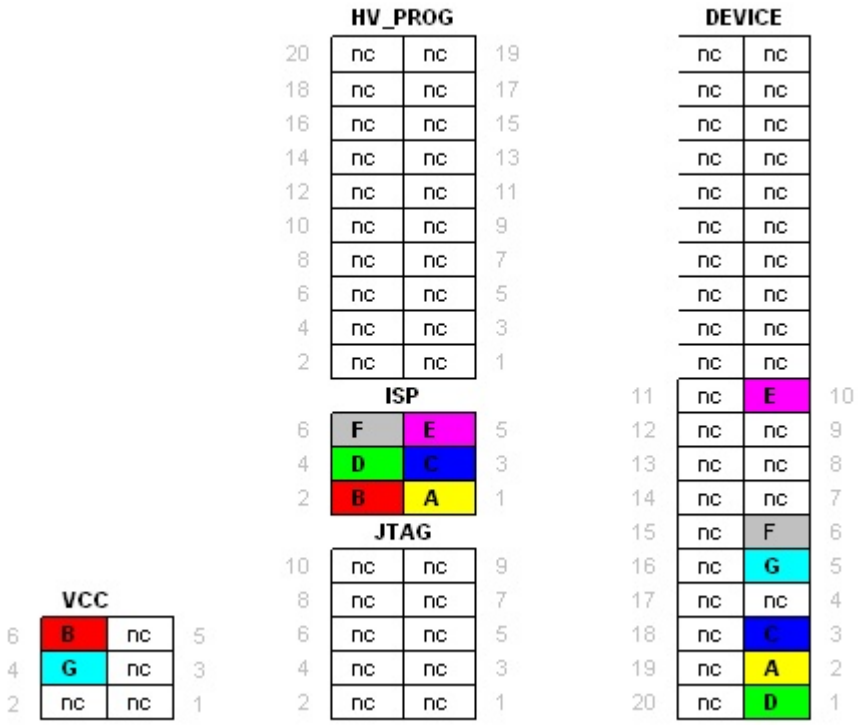
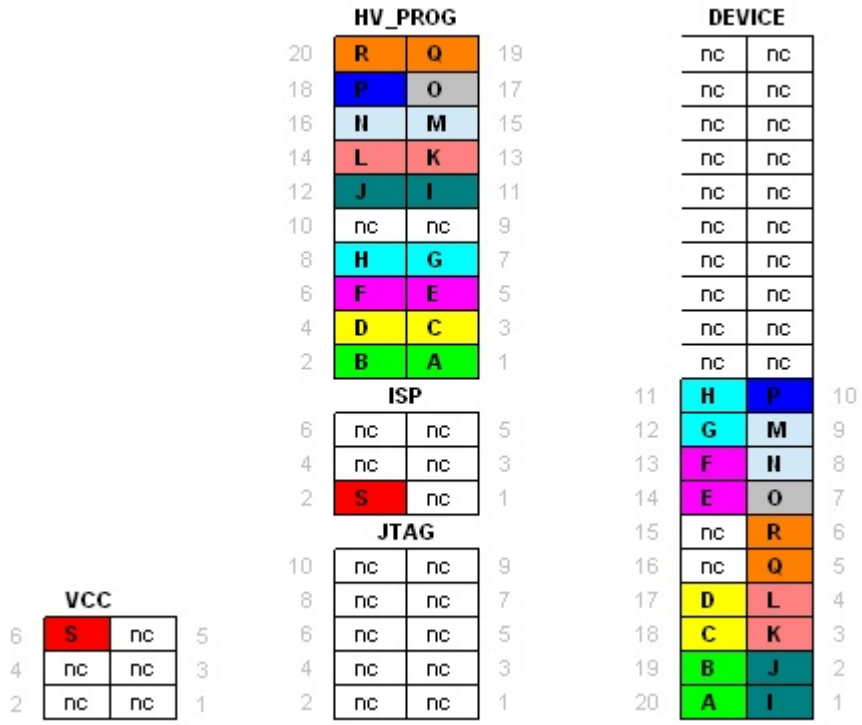


Figure 6-12. Parallel Programming



6.7. Devicesheet: SCKT244484

- Supported Programming Modes: SPI and High Voltage Serial Programming
- Supported Debugging Modes: debugWIRE
- applicable to the following devices:
  - **ATtiny24, ATtiny44, and ATtiny84 family (and 'A' variants)**

The following pictures shows how to connect these devices to the Atmel AVR Dragon:

Figure 6-13. SPI Programming and debugWIRE Debugging

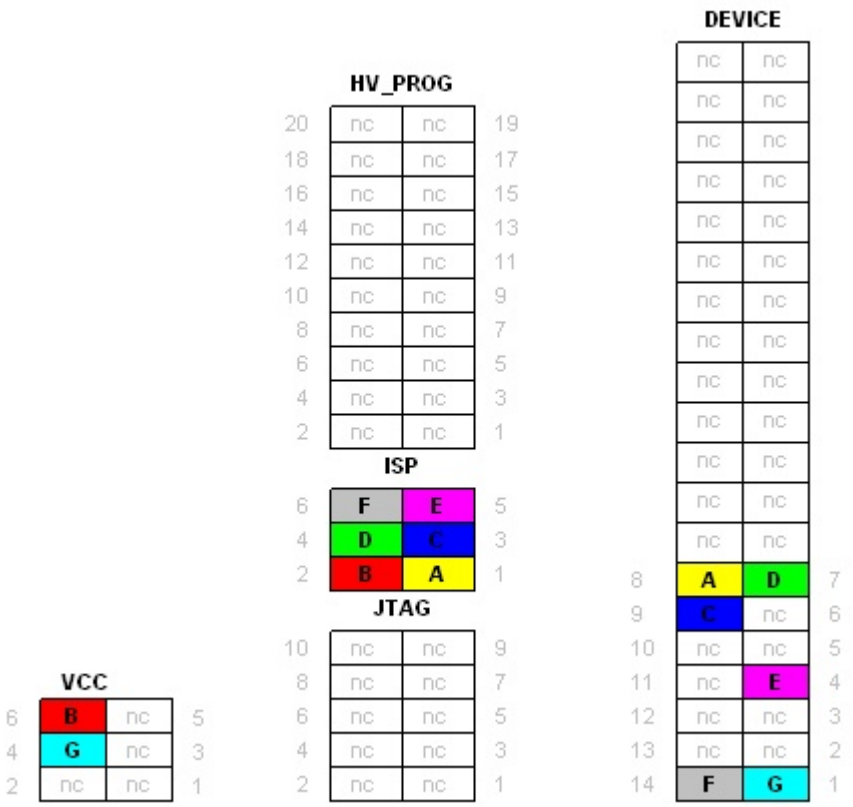
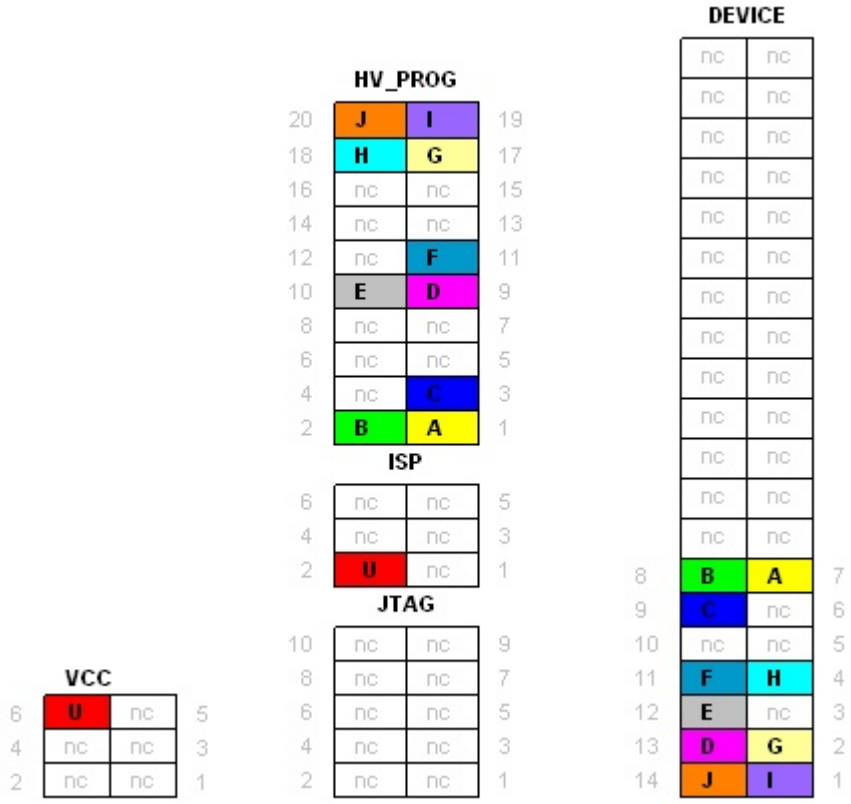


Figure 6-14. High Voltage Serial Programming



6.8. Devicesheet: SCKT000162

- Supported Programming Modes: SPI, Parallel programming, and JTAG
- Supported Debugging Modes: JTAG
- applicable to the following devices:
  - **ATmega162**

The following pictures shows how to connect these devices to the Atmel AVR Dragon:

Figure 6-15. SPI Programming

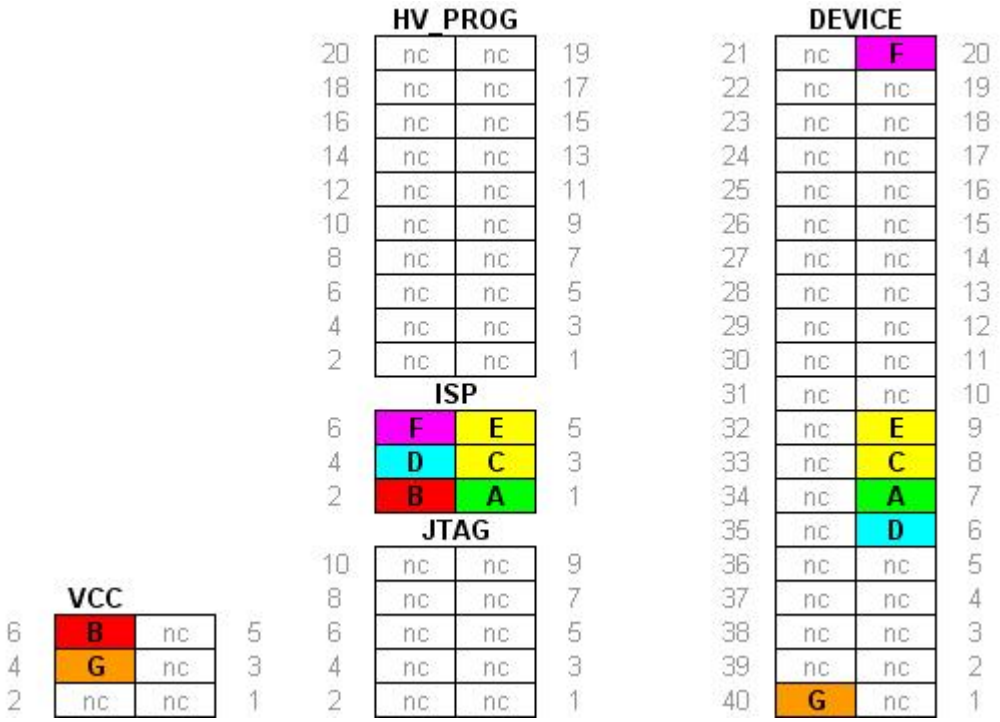


Figure 6-16. Parallel Programming

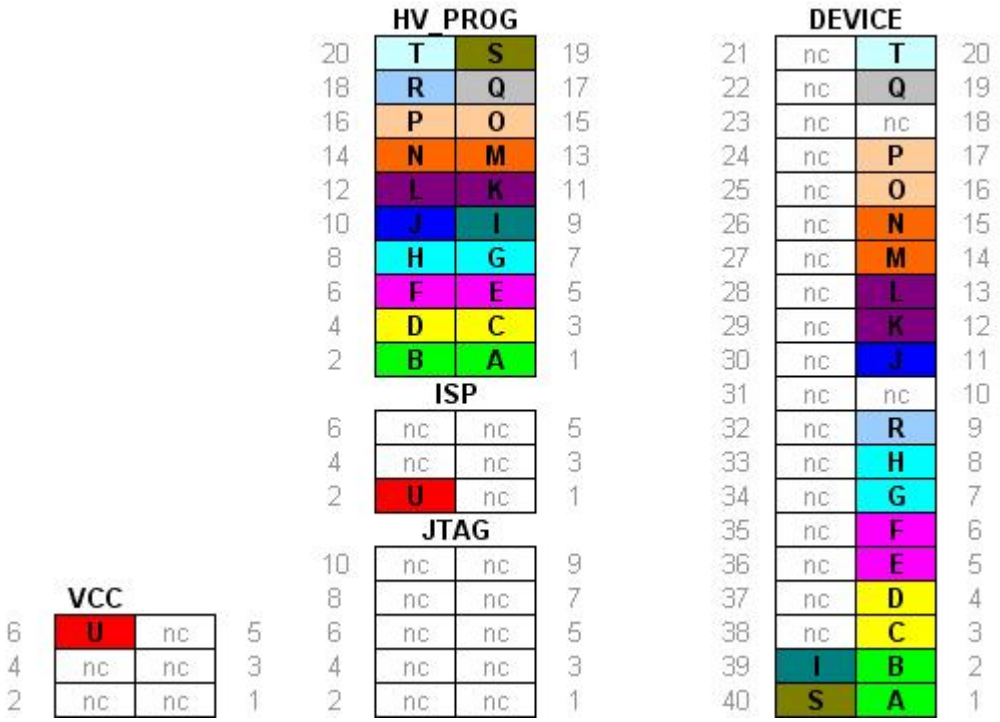
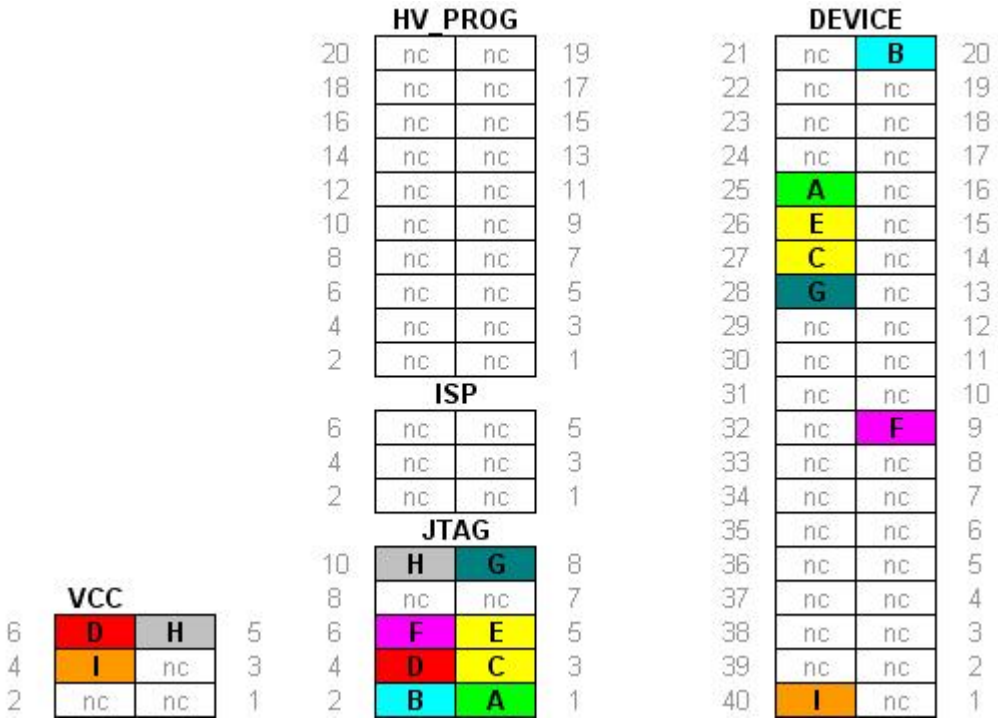


Figure 6-17. JTAG Programming and Debugging



6.9. Devicesheet: Off board Targets

- Supported Programming Modes: SPI, High Voltage Serial Programming, Parallel programming, JTAG, PDI, and aWire
- Supported Debugging Modes: debugWIRE, JTAG, PDI, and aWire

When the device is not available in PDIP package it can not directly be connected to the Atmel AVR Dragon Prototype area. See the applicable datasheet for information on how to connect for SPI, PP, JTAG, debugWIRE, PDI, and aWire.

7. On-Chip Debugging

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

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

Run Mode

When in Run mode, the execution of code is completely independent of the AVR Dragon. The AVR Dragon 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/Os 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.

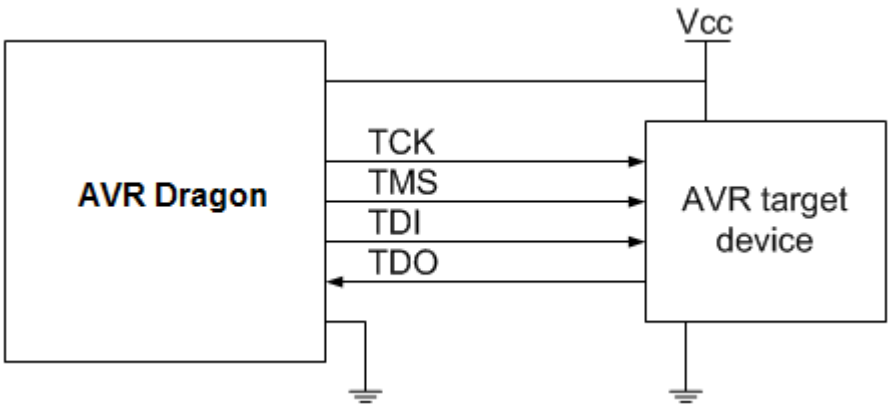
7.2. Physical Interfaces

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

7.2.1. 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 7-1. 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 7-2 JTAG Header Pinout. The AVR Dragon 100-mil probe connectors support this pinout.

Figure 7-2. JTAG Header Pinout

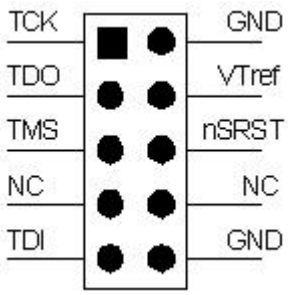


Table 7-1. JTAG Pin Description

Name	Pin	Description
TCK	1	Test Clock (clock signal from the AVR Dragon into the target device)
TMS	5	Test Mode Select (control signal from the AVR Dragon into the target device)
TDI	9	Test Data In (data transmitted from the AVR Dragon into the target device)
TDO	3	Test Data Out (data transmitted from the target device into the AVR Dragon)
nTRST	8	Test Reset (optional, only on some AVR devices). Used to reset the JTAG TAP controller.
nSRST	6	Source Reset (optional). Used to reset the target device. Connecting this pin is recommended since it allows the AVR Dragon 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 AVR Dragon.

Name	Pin	Description
VTref	4	Target voltage reference. The AVR Dragon samples the target voltage on this pin in order to power the level converters correctly. The AVR Dragon draws less than 1mA from this pin.
GND	2, 10	Ground. Both must be connected to ensure that the AVR Dragon and the target device share the same ground reference.

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

Note: The AVR Dragon 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 7-1 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 4-4 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 AVR Dragon probe
- All devices must be operating on the same target voltage level. VTref on the AVR Dragon 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 AVR Dragon probe must be connected to RESET on the devices if any one of the devices in the chain disables its JTAG port
- "Devices before" refers to the number of JTAG devices that the TDI signal has to pass through in the daisy chain before reaching the target device. Similarly "devices after" is the number of devices that the signal has to pass through after the target device before reaching the AVR Dragon TDO pin.
- "Instruction bits before" and "after" refers to the total sum of all JTAG devices' instruction register lengths which are connected before and after the target device in the daisy chain
- The total IR length (instruction bits before + 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 AVR XMEGA device, the daisy chain settings are:

Devices before: 1

Devices after: 1

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

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

7.2.2. aWire Physical

aWire is a single-pin interface for programming and debugging of low-pin-count Atmel AVR UC3 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 4-6 aWire Header Pinout should be used.

7.2.3. 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.

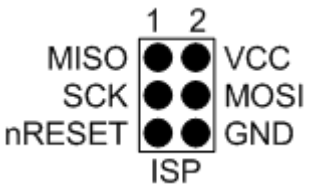
When designing an application PCB which includes an AVR with the PDI interface, the pinout shown in Figure 4-5 6pin Header Connector with 2.54mm (100 MIL) Spacing should be used.

7.2.4. 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 7-3 debugWIRE (SPI) Header Pinout should be used.

Figure 7-3. debugWIRE (SPI) Header Pinout



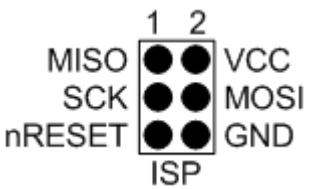
Note:
The debugWIRE interface can not be used as a programming interface. This means that the SPI interface must also be available (as shown in Figure 7-4 SPI Header Pinout) in order to program the target.

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.

7.2.5. SPI

In-System Programming uses the target 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 Atmel AVR with the SPI interface, the pinout shown in Figure 7-4 SPI Header Pinout should be used.

Figure 7-4. SPI Header Pinout



7.3. Atmel AVR OCD Implementations

7.3.1. 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.

7.3.2. 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
- 1 dedicated program address comparator or symbolic breakpoint (reserved)
- 4 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.

7.3.3. 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) (except ATmega128[A])

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

7.3.4. Atmel megaAVR/tinyAVR OCD (debugWIRE)

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

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

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

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 Dragon does not offer a variable clock rate for 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 AVR XMEGA the I/O modules are stopped in stop mode. This means that USART transmissions will be interrupted, timers (and PWM) will be stopped.

Hardware breakpoints

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

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

Here are the different combinations that can be set:

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

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

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

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.

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 AVR Dragon will assert the RESET line while connecting. If this line is wired correctly, it will force the target AVR device into reset, thereby allowing a JTAG connection.

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

IDR events

When the application program writes a byte of data to the OCDR register of the AVR device being debugged, the AVR Dragon 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 AVR Dragon 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 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 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
- 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 devices the JTAG port is not enabled by default. When using these devices it is essential to connect the RESET line so that the AVR Dragon 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 AVR Dragon 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 AVR Dragon 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. What's New

- **Atmel Studio 6.2 - Sw: 7.26 7.26**

  Fixed oscillator calibration.
- **Atmel Studio 6.0 - Sw: 7.22 7.22**

  Only minor internal bug fixes.
- **AVR Studio 5.1 - Sw: 7.21 7.21**
  - Improved debugWIRE single-stepping performance
  - aWire auto-baud calculation improvements
  - Fixed Atmel AVR XMEGA flash page programming error (seen at low voltages)
  - Support for high SUT (start-up time) values on AVR XMEGA devices
- **May 2011, AVR Studio 5 public release - Sw: 7.14 7.14**
  - Fixed ATmega64M1 programming
- **April 2011, AVR Studio 5 public beta 2 - Sw: 7.11 7.11**
  - Improved aWire speed
  - Fixed JTAG enable on UC3 devices with aWire
- **February 2011 - Sw: 7.02 7.02**
  - Support for AVR Studio 5 (beta)

10. Command Line Utility

Atmel Studio comes with a command line utility called atprogram that can be used to program targets using the AVR Dragon. 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
```

11. Troubleshooting

Table 11-1. Common Problems Resolutions

Problem	Reason	Solution
Errors when programming or debugging Atmel AVR UC3 devices over JTAG	Ringing on the TCK line might lead to incorrect detection of rising edges on TCK. This is depending on the target board, and has shown to be a problem when connected to an AVR UC3 target on some Atmel STK600 configurations.	Add a series resistor on TCK (typically 68Ω) or add a capacitor of at least 1nF between TCK and ground. The series resistor solution is the preferred one since it does not stress the Dragon level converters as much as the capacitor solution, but both solutions should work.
Signature Bytes read as `0x00 0x00 0x00`	SPI Frequency is to high	Lower SPI frequency in the Debugging properties (Property Pages item on View menu)
Not able to communicate with device through debugWIRE	RESET pullup resistor too small	Remove or increase the pull-up value to 10kΩ or more
Not able to communicate with device through debugWIRE	Decoupling capacitor destroys communication on RESET line	Remove decoupling capacitor on reset line during debugWIRE debugging
debugWIRE communication fails when using Atmel STK500	RESET line strongly tied to VCC	Remove RESET jumper on STK500 to allow AVR Dragon to control the line
After successfully enabling the DWEN fuse, AVR Dragon is not able to enter debug mode	RESET line strongly tied to VCC	Remove RESET jumper on STK500 to allow AVR Dragon to control the line
Target voltage is read as 0V for on-board targets	AVR Dragon gets no reference voltage to the target voltage sensing. Target voltage is sensed from either pin 2 on the SPI header or pin 4 on the JTAG header.	In order to get reference voltage to the level converters of AVR Dragon, connect Pin 2, 4, or 6 on the VCC header to pin 2 on SPI header or pin 4 on the JTAG header[1]
Target voltage is read as 0V for off-board targets	AVR Dragon gets no reference voltage to the target voltage sensing. Target voltage is sensed from either pin 2 on SPI header or pin 4 on the JTAG header.	Connect the target voltage from the target board to pin 2 on SPI header or pin 4 on the JTAG header
Not able to set SPI frequency	AVR Dragon is not reading any target voltage	See above

[1] Note that VCC pins on AVR Dragon are set to 5V and this procedure should therefore not be used for off-board targets.

Problem	Reason	Solution
SPI/PP/HVSP programming fails	AVR Dragon is not reading any target voltage	See above
Not able to connect to AVR Dragon, and status led is yellow	AVR Dragon firmware upgrade failed, or firmware is corrupt	For information on how to upgrade the firmware, see the Atmel Studio user guide
Programming or debug sessions fails, resetting the AVR Dragon	The target circuitry draws too much power from the AVR Dragon	Try to power the circuitry from an external source in stead of the AVR Dragon

12. Technical Information

12.1. Atmel AVR Dragon Requirements

12.1.1. System Unit

Physical Dimensions	(H x W x D) 53 x 105 x 15mm
Power Voltage Requirements	5.0V USB powered
Atmel AVR Dragon Current Consumption	150mA
Maximum Current Source Capability (to target)	300mA
Ambient Temperature	0-70ºC

12.1.2. Operation

Target Voltage Range	1.8 - 5.5V

12.1.3. I/O Pins

Maximum Pull-up on SPI/JTAG header	1kΩ
Maximum Pull-down on SPI/JTAG header	10kΩ
Maximum Source Current VCC header	Up to total 300mA

Note:
The Atmel AVR Dragon requires a USB port that can deliver up to 500mA. (Self-powered USB hub.)

12.2. Technical Support

Before requesting technical support make sure you have the latest available Atmel Studio version and tool firmware installed. The latest Atmel Studio version can be downloaded from http://www.atmel.com/atmelstudio, and contains the latest firmware version for all Atmel AVR tools. When connecting your tool, Atmel Studio will automatically check the firmware version and request an update if needed.

For technical support, contact avr@atmel.com. When requesting technical support for AVR Dragon include the following information:

- Version number of Atmel Studio. This can be found in Atmel Studio menu **Help** → **About**.
- PC processor type and speed
- PC operating system and version
- What target AVR device is used (complete part number)
- Fuse settings on the AVR target device
- Target clock frequency
- If CLKPR (Clock Prescaler Register) is used (for AVRs with this feature)
- Target voltage
- Programming/debugging speed
- A detailed description of the problem, and how to recreate it

- Any error or warning information generated by Atmel Studio when the error occurred

13. Evaluation Board/Kit Important Notice

This evaluation board/kit is intended for use for **FURTHER ENGINEERING, DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY.** It is not a finished product and may not (yet) comply with some or any technical or legal requirements that are applicable to finished products, including, without limitation, directives regarding electromagnetic compatibility, recycling (WEEE), FCC, CE or UL (except as may be otherwise noted on the board/kit). Atmel supplied this board/kit "AS IS", without any warranties, with all faults, at the buyer's and further users' sole risk. The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies Atmel from all claims arising from the handling or use of the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge and any other technical or legal concerns.

EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER USER NOR ATMEL SHALL BE LIABLE TO EACH OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

No license is granted under any patent right or other intellectual property right of Atmel covering or relating to any machine, process, or combination in which such Atmel products or services might be or are used.

Mailing Address:

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA

14. Revision History

Doc. Rev.	Date	Comments
42723A	06/2016	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**