Application Note

Report No:

AN101

Title:

SPI and JTAG In-System Programming (ISP) guidelines for the Atmel ATmega AVR FLASH Microcontroller Family

Author:	Date:	Version Number:
John Marriott	12th June 2007	1.07

Abstract:

This application note describes the connections required to implement In-System Programming of the Atmel ATmega AVR FLASH Microcontroller Family using either the SPI or the JTAG Programming Interfaces. The document describes the physical connections required from the programmer to the target Microcontroller and also details the different ISP Header Connector pin-outs which are currently available.

Contents

1.0 Introduction

This application note describes the connections required to implement In-System Programming of the Atmel AT90S and ATmega AVR FLASH Microcontroller Family using either the SPI or JTAG Programming Interface. The document describes the physical connections required from the programmer to the target Microcontroller and also details the different ISP Header Connector pin-out which are currently available.

Please note:

- The Atmel ATtiny AVR Family features both a 'Low Voltage' and 'High Voltage' Serial Programming Modes. Please refer to AN104 for full details.

1.1 Programmers supported

This Application Note covers any Equinox programmer which is capable of SPI programming of Atmel microcontrollers. The table below lists the Equinox programmers and also details whether each one can be upgraded for JTAG programming.

Fig. 1.1 Equinox Programmer – SPI and JTAG ISP Support

Programmer	SPI algorithms	JTAG algorithms	Upgrade Order Code
EPSILON5	YES	UPGRADE	EPSILON5-UPG3
FS2000A	YES	UPGRADE + JTAG Module with RESET Switch	FS2000A-UPG7
FS2003	YES	UPGRADE	FS2003-UPG7
PPM3 MK1	YES	N/A	N/A
PPM3 MK2	YES	UPGRADE + IO-CON-3 JTAG Connector Module	PPM3A1-UPG7
Micro-ISP Series IV	YES	N/A	N/A
Micro-ISP Series IV LV	YES	N/A	N/A
Activ8r(AVR)	YES	N/A	N/A
PRO101	YES	N/A	N/A

Key:

- YES - Enabled as standard
- UPGRADE – Chargeable license upgrade required
- TBA – To be announced
- N/A – Not available

1.2 Device Support

The table below lists the Atmel ATmega devices which are currently supported by the Equinox range of programmers. The SPI Algorithms are supplied as standard with all programmers, where as the JTAG algorithms require an upgrade license to be purchased to enable the JTAG Library of devices. Some ATmega devices such as the ATmega8(L) and ATmega161(L) do not have a JTAG port and so cannot support JTAG programming.

Device	**SPI algorithm**	**JTAG algorithm**	**Minimum firmware version**	**SPI programming pin-out**
AT90CAN32	YES	YES	3.04	Check datasheet
AT90CAN64	YES	YES	3.04	Check datasheet
AT90CAN128	YES	YES	3.04	Check datasheet
ATmega8(L)	YES	No JTAG port	3.04	MOSI / MISO / SCK
ATmega16(L)	YES	YES	3.04	MOSI / MISO / SCK
ATmega323(L)	YES	YES	3.04	MOSI / MISO / SCK
ATmega32(L)	YES	YES	3.04	MOSI / MISO / SCK
ATmega64(L)	YES	YES	3.04	RXD / TXD / SCK
ATmega103(L)	YES	No JTAG port	3.04	RXD / TXD / SCK
ATmega128(L)	YES	YES	3.04	RXD / TXD / SCK
ATmega161(L)	YES	No JTAG port	3.04	MOSI / MISO / SCK
ATmega162(L)	YES	YES	3.04	MOSI / MISO / SCK
ATmega164P-20	YES	No JTAG port	3.04	Check datasheet
ATmega164PV-10	YES	No JTAG port	3.04	Check datasheet
ATmega169(L)	YES	YES	3.04	RXD / TXD / SCK
ATmega169PV-8	**YES	**YES	3.04	RXD / TXD / SCK
ATmega324P-20	YES	**YES	3.04	Check datasheet
ATmega324PV-10	YES	**YES	3.04	Check datasheet
ATmega329V-8	YES	YES	3.04	Check datasheet
ATmega3290V-8	YES	YES	3.04	Check datasheet
ATmega3290-16	YES	YES	3.04	Check datasheet
ATmega406	YES	YES	3.04	Check datasheet
				Check datasheet
ATmega640-16	YES	YES	3.04	Check datasheet
ATmega640V-8	YES	YES	3.04	Check datasheet
ATmega644-10	YES	**YES	3.04	Check datasheet
ATmega644P	YES	**YES	3.04	Check datasheet
ATmega644PV	YES	**YES	3.04	Check datasheet
ATmega6490	YES	**YES	3.04	Check datasheet
ATmega6490V	YES	**YES	3.04	Check datasheet
ATmega1280V-8	**YES	**YES	3.04	RXD / TXD / SCK
ATmega1280-16	**YES		3.04	RXD / TXD / SCK
ATmega1281V-8	**YES	**YES	3.04	RXD / TXD / SCK
ATmega2560-16	**YES	**YES	3.04	RXD / TXD / SCK
ATmega2560V-8	**YES	**YES	3.04	RXD / TXD / SCK

ATmega2561-16	**YES	**YES	3.04	RXD / TXD / SCK
ATmega2561V-8	**YES	**YES	3.04	RXD / TXD / SCK

Please note:

- ** Devices with greater than 128kb of FLASH memory require a firmware upgrade to version 3.01 or above in order to support programming of the upper 128kb.
- Please see section 4 for instructions on updating your programmer firmware.
- As a rule of thumb, only Atmel Atmega AVR devices with 16k bytes of FLASH or greater will feature the JTAG Programming Interface.
- Please check the Device support section on the Equinox website for an up-to-date Device Support List.

1.3 SPI Algorithm Overview

The SPI algorithm is a simple 3-wire interface which can be used to program most AVR Microcontrollers. The advantages and disadvantages of this algorithm are detailed below.

Advantages:

- The SPI algorithm is supported by almost all Atmel AVR microcontrollers including AT90S, AT90CANxxx, ATtiny and ATmega devices. This means that the same Programming Interface can be used on any products containing any AVR microcontroller.
- The SPI Programming Interface uses only 3 SPI pins (MOSI, MISO, SCK) and the RESET pin.
- The SPI pins can be used to drive other circuitry such as LED's and switches on the Target Board as well as being used for ISP purposes. However, this will require careful design on the Target Board to ensure that the programming signals are not compromised.
- In SPI Mode, it is possible to reprogram a single byte of the EEPROM area without having to perform a Chip Erase first.
- The SPI algorithms are supported as standard on all Equinox ISP Programmers.

Disadvantages

- In general terms, the SPI algorithm is 3-4 times slower than the JTAG algorithm.
- When using the SPI algorithm, the clock used during programming is supplied from either the AVR Internal RC Oscillator or from an external crystal / resonator. The programming SPI speed is completely dependent on the speed of this oscillator.
- If the oscillator speed is slow, then the maximum SPI speed is seriously limited and the overall programming will be very slow.
- If the Clock Selection Fuses are incorrectly programmed in SPI mode, then the chip may no longer have a valid oscillator and so will not respond to the programmer. This can render the chip unprogrammable except by physically removing it from the Target Board and using either a JTAG or Parallel programmer to resurrect the correct Fuse Settings.

1.4 JTAG Algorithm Overview

The JTAG algorithm provides a method of performing high-speed programming of an Atmel Atmega AVR microcontroller. The same JTAG port can also be used for on-chip debugging of code using the Atmel JTAG-ICE Debugger. The advantages and disadvantages of the JTAG algorithm are detailed below.

Advantages:

- The JTAG algorithm is approximately 3-4 times faster at programming compared to the SPI algorithm.
- The programming time using JTAG for the EEPROM is significantly faster than the SPI algorithm because in JTAG mode a 'Page' of EEPROM is programmed at a time rather than a single byte. Each byte may take eg. 9ms to program in SPI mode, where as a whole page of eg. 4 bytes may take 9ms to program in JTAG mode.
- The JTAG algorithm uses the same 'JTAG Port' as the Atmel JTAG-ICE Debugger. This means that the same port can be used for both debugging during the development phase and also programming during the production phase of the product.
- With the JTAG algorithm, the programming clock is supplied by the programmer and JTAG logic inside the Target AVR device does not require any other clocking. This means that the chip is not dependent on the settings of the 'Clock Selection Fuses' in JTAG Mode.
- In JTAG mode is it possible to change the 'Clock Selection Fuses' to any value and still program the chip. (with the exception of the 'JTAGEN' Fuse)
- It is possible to use the JTAG port of the Target Microcontroller to perform in-circuit testing of the microcontroller and surrounding circuitry. This testing is performed by shifting Test Data through the JTAG port of the Target Microcontroller. A JTAG Test System is required to perform this testing. It is not supported by any Equinox Programmer or the Atmel JTAG ICE.

- It is possible to daisy-chain multiple JTAG devices on the JTAG bus and then select to program a particular device. This functionality is not currently supported by Equinox.

Disadvantages

- The JTAG Programming Interface uses 5 pins: TCK, TDI, TDO, TMS and RESET.
- The JTAG pins of the microcontroller are not designed for off-board use and should not be shared with any other circuitry on Target Board. This means that the JTAG port pins must be dedicated for programming / debugging.
- In JTAG mode the EEPROM is divided into 'Pages' rather than 'Single Bytes'. It is therefore more complicated to program a single byte in the EEPROM as the entire page (usually 4 or 8 bytes) must be read back and then the single byte overlaid on top of this data and finally the entire page is then re-programmed back into the EEPROM.
- In JTAG Mode, it is not possible to re-program any location in the EEPROM which is not 0xFF without first performing a Chip Erase operation. This means that if the EEPROM already contains any data, it is not possible to re-program this data without erasing the entire chip first.

2.0 SPI Programming Algorithm

2.1 Overview of the SPI Programming Interface

The SPI Programming Interface is a simple synchronous 3-wire communications bus which is commonly used for control / data transfer between a Master Processor and a Slave Peripheral such as an external SPI memory device – see figure 2.1.

Fig 2.1a – SPI Master / Slave example

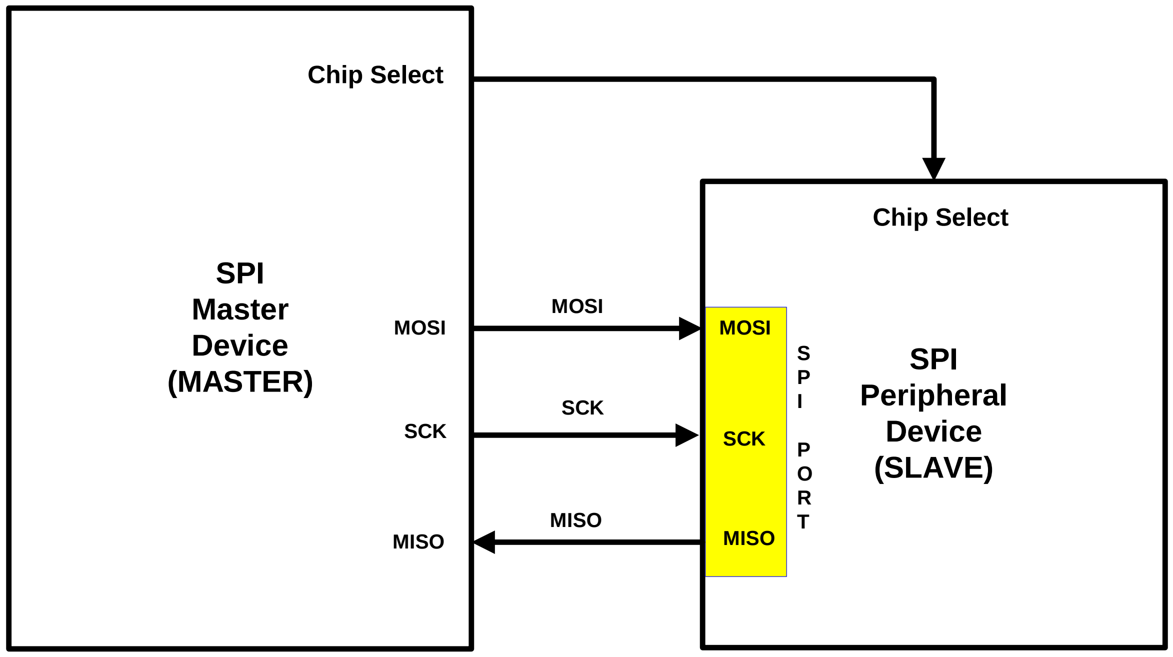


Fig 2.1b – SPI Signal names and directions

Signal Name	Signal description	Signal direction (from Master)
MOSI	Master OUT, Slave In	Output
MISO	Master IN, Slave OUT	Input
SCK	Serial Clock	Output
Chip Select (CS)	Chip Select	Output

Data is transferred from the Master to the Slave using the MOSI (Master OUT, Slave In) signal line. The Slave transfers data back to the Master using the MISO (Master IN, Slave OUT) signal line. The data transfer is clocked by the SCK signal line which is generated by the Master on the SPI bus. The Slave uses the SCK signal to know when to sample the MOSI signal for valid data and when to output valid data on the MISO signal line.

Most SPI Slave devices have a 'Chip Select' signal which the Master asserts to select a particular Slave device on the SPI bus. In the example above with only one Slave SPI device, the Master would still have to assert the Chip Select line in order to communicate with the Slave device.

2.2 Atmel AVR Microcontroller - SPI Implementation

Atmel have chosen the SPI interface to implement fast In-System Programming (ISP) of their AT90S, ATmega and ATtiny AVR Microcontroller families. This implementation allows the on-chip FLASH, EEPROM, Configuration Fuses and Security Fuses of a target AVR Microcontroller to be In-System Programmed using a suitable external ISP Programmer or an-board SPI Master Controller – see fig 2.2a.

Fig 2.2a – ISP Programming Implementation of Atmel AVR Microcontrollers

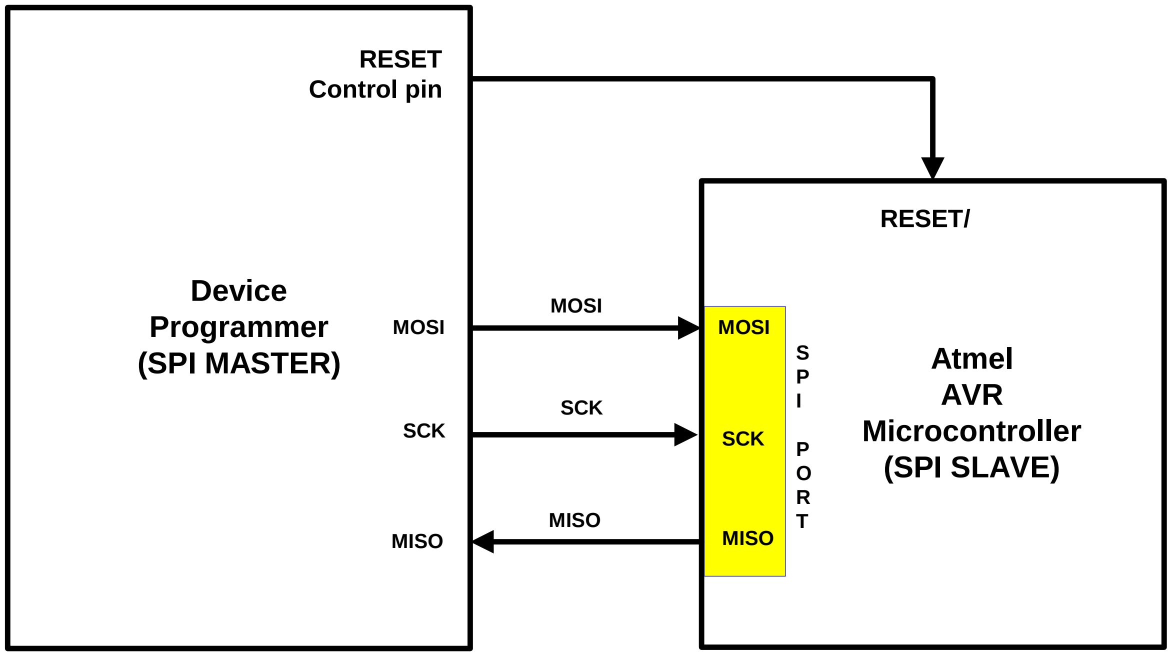


Fig 2.2b – Programmer / Microcontroller - SPI Signal names and directions

Signal Name	Signal description	Signal direction (from Programmer)	Signal direction (from Microcontroller)
MOSI	Master OUT, Slave In	Output	Input
MISO	Master IN, Slave OUT	Input	Output
SCK	Serial Clock	Output	Input
RESET	Chip Select	Output	Input

The external Device Programmer is the SPI Bus Master and the AVR Microcontroller on the Target System is the SPI Slave. The RESET control signal from the programmer is used to force the Target Microcontroller to enter the so-called AVR 'Serial Programming Mode'. For Atmel AVR Microcontrollers, the programmer must drive the RESET pin LOW and then send a command on the SPI bus to enter programming mode. This has the effect of resetting the target Microcontroller so it is no longer running firmware (i.e. the user application ceases to execute).

Once the Target Device is in 'Serial Programming Mode', the external programmer can transfer data to / from the target AVR device across the SPI bus. At the end of the programming cycle, the programmer simply creates a RESET pulse and then the device should start to run the firmware which has been programmed into it.

2.3 Atmel AT90S AVR Microcontrollers

The Atmel AT90S AVR Microcontroller Family use the standard SPI pins (MOSI, MISO, SCK) for In-System Programming (ISP) – see fig. 2.3.

Fig 2.3a AT90Sxxxx AVR – ISP Connections

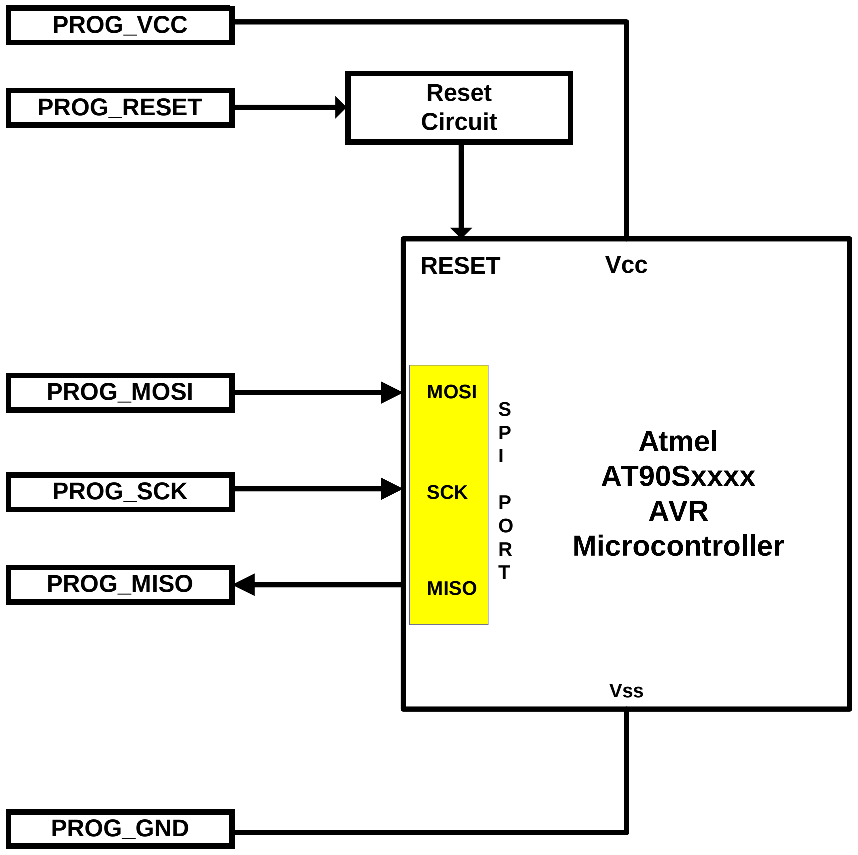


Fig 2.3.b – AT90S AVR Microcontroller - SPI Signal names and directions

Programmer Signal Name	Signal description	Signal direction (from Programmer)	Connect to AVR Microcontroller Pin	Signal direction (from Microcontroller)
PROG_MOSI	Master OUT, Slave In	Output	MOSI	Input
PROG_MISO	Master IN, Slave OUT	Input	MISO	Output
PROG_SCK	Serial Clock	Output	SCK	Input
PROG_RESET	RESET	Output	RESET	Input

2.4 Atmel ATmega AVR Microcontrollers

2.4.1 Overview of possible ATmega pin-outs

The majority of devices in the Atmel ATmega AVR family conform to the standard SPI pin-out for In-System Programming using the MOSI, MISO and SCK pins of the target device. However, there are also a few devices which use the TXD pin as MISO and the RXD pin as MOSI during In-System Programming. These derivatives are referred to as ‘UART SPI Pin-out’ devices. Special care must be taken to route the programmer MOSI / MISO pins to the correct pins of the target AVR device otherwise during In-System Programming will not function.

Please refer to the ‘Device Support’ table in section 1.2 for details of which ATmega devices feature either the ‘Standard’ or ‘UART’ SPI pin-out. Please look up the device you are trying to program in the table and then refer to relevant section for the correct ISP pin-out.

2.4.2 ATmega AVR - Standard SPI Pin-out

This pin-out is compatible with all ATmega devices which use the MOSI and MISO pins for In-System Programming.

Fig. 2.4.2a ATmega – Standard Pin-out - ISP connections

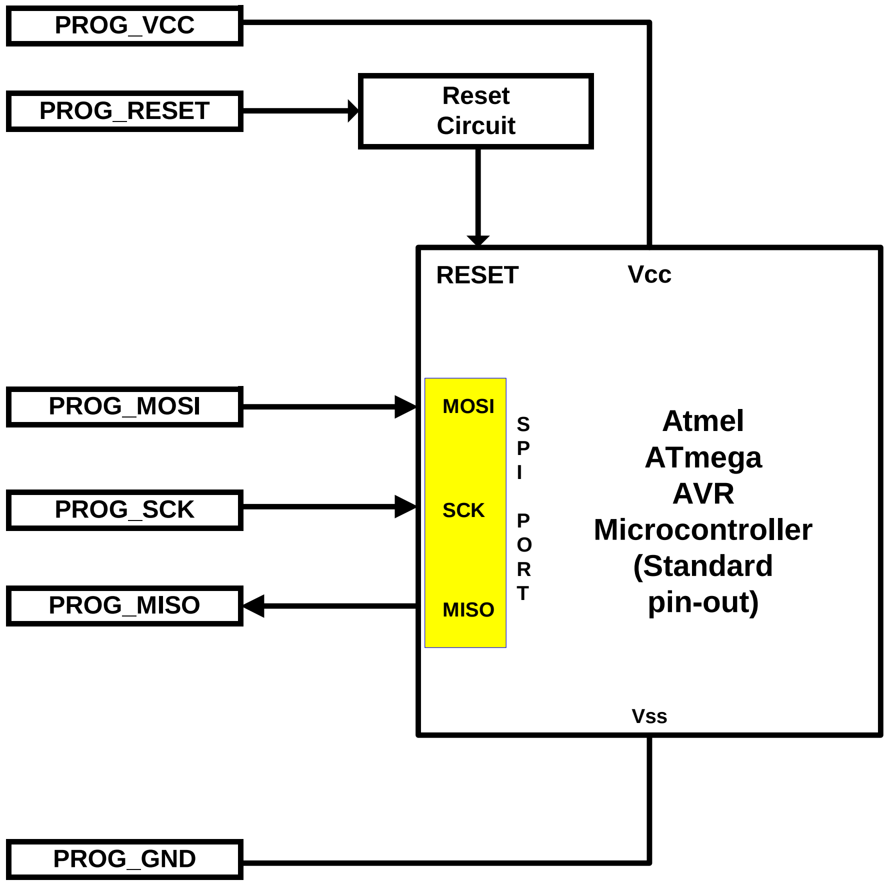


Fig 2.4.2b – ATmega AVR – Standard Pin-out - SPI Signal names and directions

Programmer Signal Name	**Signal description**	**Signal direction (from Programmer)**	**Connect to AVR Microcontroller Pin**	**Signal direction (from Microcontroller)**
PROG_MOSI	Master OUT, Slave In	Output	MOSI	Input
PROG_MISO	Master IN, Slave OUT	Input	MISO	Output
PROG_SCK	Serial Clock	Output	SCK	Input
PROG_RESET	RESET	Output	RESET	Input

2.4.3 ATmega AVR - UART SPI Pin-out

This pin-out is compatible with the Atmel ATmega 64(L), ATmega103(L), ATmega128(L) and ATmega169(L) devices which use the TXD and RXD pins for SPI during In-System Programming. The standard SPI MOSI and MISO pins are not used at all during In-System Programming and can be freely connected to other SPI devices.

2.4.3a ATmega – Standard Pin-out - ISP connections

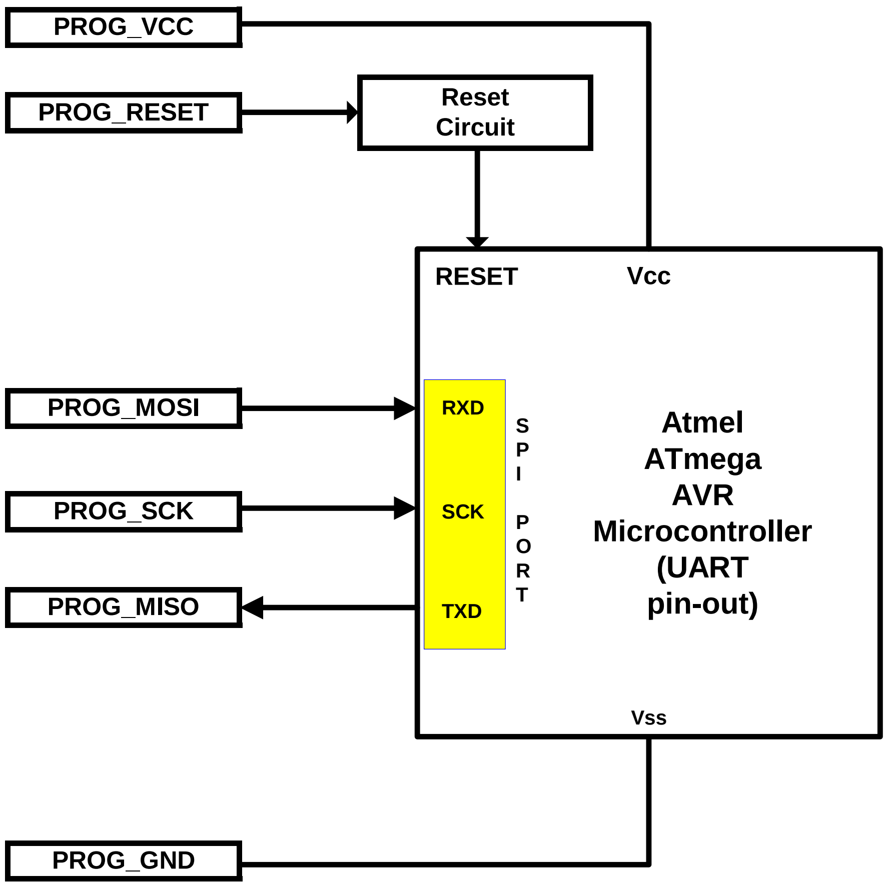


Fig 2.4.3b – ATmega AVR – Standard Pin-out - SPI Signal names and directions

Programmer Signal Name	Signal description	Signal direction (from Programmer)	Connect to AVR Microcontroller Pin	Signal direction (from Microcontroller)
PROG_MOSI	Master OUT, Slave In	Output	**RXD***	Input
PROG_MISO	Master IN, Slave OUT	Input	**TXD***	Output
PROG_SCK	Serial Clock	Output	SCK	Input
PROG_RESET	RESET	Output	RESET	Input

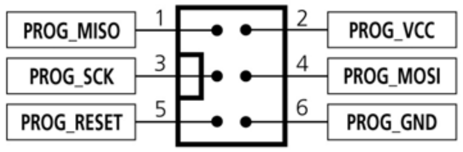
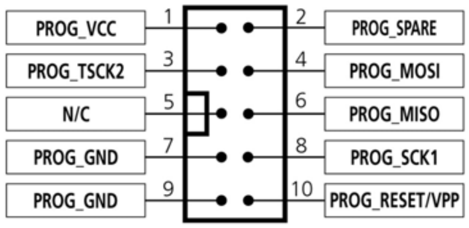
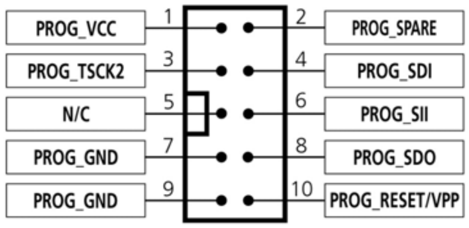
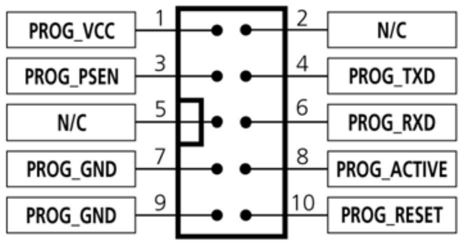
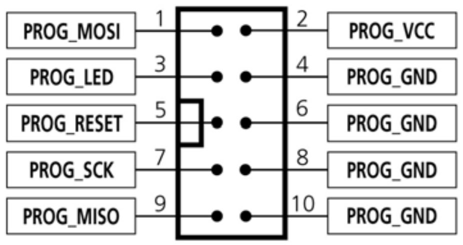
* Please note – The TXD and RXD pins must be used for ISP instead of the MISO and MOSI pins.

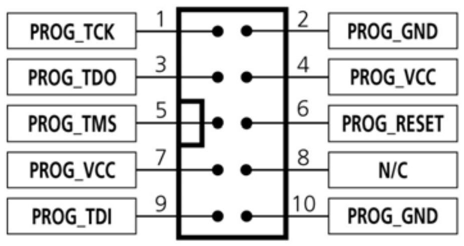
Special Considerations

1. The TXD and RXD pins must be used for ISP instead of the MISO and MOSI pins.
2. For the ATmega103(L) device only, the ATmega103 MISO pin (pin 13) is active during In-System Programming even though this pin is not actually used for programming. If this pin is used as an output make sure that whatever it is connected to can cope with the pin toggling during ISP. It may also be necessary to insert a current limiting resistor in this line.

2.5 ISP Header Selection Chart (by header)

The FOUR ISP Headers featured on the most Equinox ISP Programmers are detailed in the table below. Please refer to the section indicated in the 'refer to section' column for specific details of each header.

#	ISP Header	Description / Function	Refer to section	ISP Header Pin-out
1	J3	Atmel 6-way ISP Header		 PROG_MISO 1 2 PROG_VCC PROG_SCK 3 4 PROG_MOSI PROG_RESET 5 6 PROG_GND
Header J6 can have THREE different pin-outs depending on which Target Device is to be programmed. See (2a), (2b) and (2c).				
2a	J6(a)	**Equinox 10-way Header(a)** **Device support:** Atmel AT90S, ATmega, ATtiny, AT89S devices		 PROG_VCC 1 2 PROG_SPARE PROG_TSCK2 3 4 PROG_MOSI N/C 5 6 PROG_MISO PROG_GND 7 8 PROG_SCK1 PROG_GND 9 10 PROG_RESET/VPP
2b	J6(b)	**Equinox 10-way Header(a)** **Device support:** Atmel ATtiny11/12/15 High Voltage (+12V Vpp) Programming Mode		 PROG_VCC 1 2 PROG_SPARE PROG_TSCK2 3 4 PROG_SDI N/C 5 6 PROG_SII PROG_GND 7 8 PROG_SDO PROG_GND 9 10 PROG_RESET/VPP
2c	J6(c)	**Equinox 10-way Header(b)** **Device support:** Atmel Wireless T89C51Rx2 Philips P89C51Rx2 / 66x		 PROG_VCC 1 2 N/C PROG_PSEN 3 4 PROG_TXD N/C 5 6 PROG_RXD PROG_GND 7 8 PROG_ACTIVE PROG_GND 9 10 PROG_RESET
3	J7	**Atmel 10-way Header** **Device support:** Atmel AT90S, ATmega, ATtiny, AT89S devices		 PROG_MOSI 1 2 PROG_VCC PROG_LED 3 4 PROG_GND PROG_RESET 5 6 PROG_GND PROG_SCK 7 8 PROG_GND PROG_MISO 9 10 PROG_GND

4	J8	**Atmel 10-way JTAG Header** **Device support:** Atmel ATmega32/128 + any new devices with JTAG port		 PROG_TCK PROG_TDO PROG_TMS PROG_VCC PROG_TDI 1 2 3 4 5 6 7 8 9 10 PROG_GND PROG_VCC PROG_RESET N/C PROG_GND

3.0 JTAG Programming Algorithm

3.1 Overview

The JTAG Programming Interface provides a method for both In-System Debugging and In-System Programming of Atmel ATmega AVR Microcontrollers. The JTAG Interfaces uses an industry-standard set of signals to provide the connection between the programmer/debugger and the AVR device. However, the actual JTAG Header used by Atmel and Equinox is specific to Atmel AVR JTAG programming and will not match JTAG connectors for other manufacturer's devices.

In the development environment, the JTAG Interface can be used for In-System Debugging of the code running on the actual Target System. This method of operation requires the use of the Atmel 'JTAG-ICE MK1 or MK2' which is an In-System Debugger. Using this equipment, it is possible to download code (firmware) to a Target Chip and then execute this code under PC control. The Debugger Software allows you to set breakpoints in the code, read / write memory locations, look at register contents etc.

In a production environment, the JTAG Interface can be use for high-speed In-System Programming (ISP) of the Target AVR Microcontroller. This method of operation requires the use of any Equinox ISP Programmer which has been enabled to support the 'ATmega JTAG' algorithms.

3.2 Upgrading your Equinox Programmer to support JTAG

The JTAG algorithms are not supported as standard on any Equinox programmers. It is necessary to purchase a 'License Upgrade' for JTAG support from Equinox. Equinox will then send you a 'JTAG Upgrade License String' which will upgrade your programmer to support JTAG programming.

3.2.1 Purchasing a JTAG License

All Equinox ISP programmers require the purchase of a 'License Upgrade' to enable JTAG support. Please see the table in section 1.1 for the relevant upgrade for your programmer.

3.2.2 How do I enable the programmer for JTAG?

To enable your programmer to support JTAG ISP programming, please purchase the relevant JTAG Upgrade from Equinox or an Equinox distributor:

1. **If you purchase the upgrade directly from Equinox**
   - Equinox will email you a 'JTAG License String'.
   - This string can be entered directly into the <Enter License> screen in EQTools.

2. **If you purchase the upgrade from a distributor**
   - The distributor will send you the Upgrade Pack by courier.
   - Within the Upgrade Pack you will find an Upgrade Form with a Code String on it.
   - Email this Code String plus your programmer Serial Number to support@equinox-tech.com
   - Equinox will then send you a 'JTAG License String' which is keyed to your programmer Serial Number.
   - This string can be entered directly into the <Enter License> screen in EQTools.

3.3 Upgrading an Epsilon5 or FS2003 to support JTAG

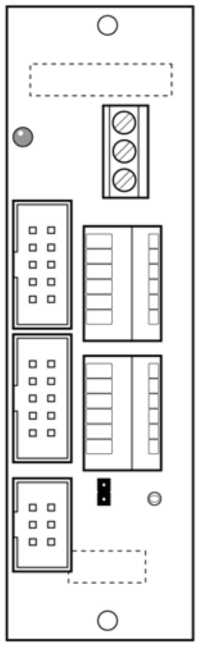
To upgrade an Epsilon5 or FS2003 programmer to support JTAG, please follow the steps below:

- Order a JTAG License from Equinox
- Enter the 'JTAG Upgrade License String' given to you by Equinox into EQTools
- Make sure you have the required version of Programmer Firmware to support the device you wish to program.
- Plug the 10-way ISP cable supplied with the programmer into the **'J8 – JTAG-10'** ISP Header on the programmer.
- Connect the other end of the 10-way ISP cable to the JTAG port on your Target Board
- You are now ready to program a Target Chip via JTAG.

3.4 Upgrading a PPM3-MK2 Programmer to support JTAG

To upgrade a PPM3-MK2 programmer to support JTAG, please follow the steps below:

- Order a 'PPM3-MK2 JTAG upgrade' from Equinox TechnologiesLicense from Equinox
- Enter the 'JTAG Upgrade License String' given to you by Equinox into EQTools
- The JTAG upgrade also includes a new 'I/O Connector Module' for the PPM3-MK2 called the **'I/O-CON-3'**. This module has a JTAG 10-way header which has the same pin-out as the JTAG-ICE.
- Make sure you have the required version of Programmer Firmware to support the device you wish to program.
- Plug the **'I/O-CON-3'** module into the PPM-MK2 programmer
- Plug the 10-way ISP cable supplied with the programmer into the **'JTAG'** ISP Header on the **'I/O-CON-3'** module.
- Connect the other end of the 10-way ISP cable to the JTAG port on your Target Board
- You are now ready to program a Target Chip via JTAG

EQ-IOCON-3 	**I/O Connector Module 3 (JTAG) – Fast Connect Version** I/O connector module for In-System Programming (ISP) of Atmel microcontrollers using JTAG protocol **Features:** • Plugs into suitable Equinox programmer e.g. PPM3 Module • Atmel 10-way JTAG IDC ISP connector (same as JTAG-ICE) • Atmel 6-way IDC ISP Header • Equinox 10-way IDC ISP header • Single-in-line header with all programmer I/O brought out for wire-wrapping to bed-of-nails probe wires • Screw terminals for power connections • Target Vcc Status LED • Link to connect / isolate the programmer Vcc from the Target Vcc **Please note** The 'Atmel AVR JTAG License' (Order code: PPM3A1-UPG7) is also required to enable the PPM3 to program Atmel AVR devices via JTAG.

3.5 Entering the License String to upgrade your programmer

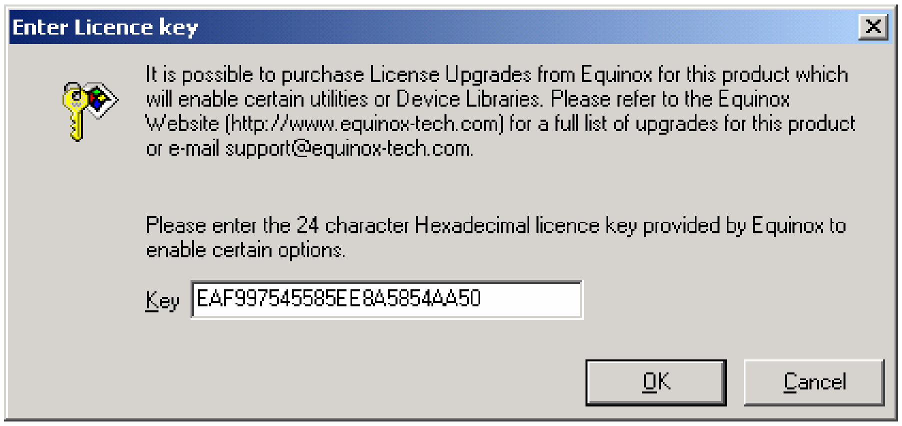
Once you have received the License String from Equinox, please follow the steps below to apply the upgrade to your programmer:

- Launch EQTools
- From the top menu bar, select <Programmer><Programmer Info>

→ the Programmer Information screen is displayed

- Click the <Enter License> button

→ The <Enter License Key> screen is displayed.



It is possible to purchase License Upgrades from Equinox for this product which will enable certain utilities or Device Libraries. Please refer to the Equinox Website (http://www.equinox-tech.com) for a full list of upgrades for this product or e-mail support@equinox-tech.com.

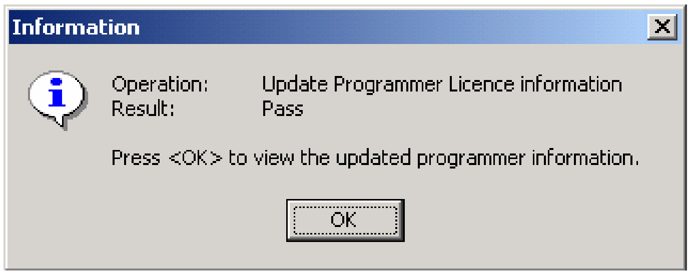
Please enter the 24 character Hexadecimal licence key provided by Equinox to enable certain options.

Key: EAF997545585EE8A5854AA50

Enter the License String you were sent by Equinox

- Click <OK>

→ EQTools should acknowledge that the attached programmer has been upgraded.



Operation: Update Programmer Licence information
Result: Pass

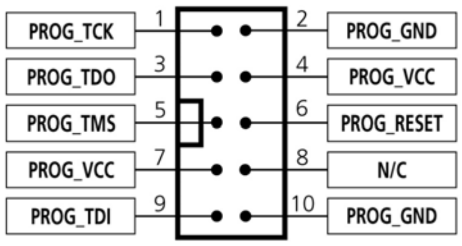
Press <OK> to view the updated programmer information.

- Click <OK>
- If you now check the Programmer Info screen, you should find that the entry for 'ATmega JTAG ISP' is now ENABLED.
- Your programmer is now upgraded to support JTAG programming of Atmel AVR Microcontrollers.

3.6 Using the JTAG Port for Debugging and Programming

The JTAG port of the ATmega AVR Microcontroller can be used for both debugging and programming purposes. The Equinox 'JTAG ISP Header' pin-out found on all Equinox ISP Programmers uses the same pins as the Atmel 'JTAG ICE MK1 / MK2' Debugger so it is possible to use the same connector / cabling for both programming and debugging.

Fig. 3.6 JTAG ISP 10-way IDC Header



The RESET pin of the AVR Microcontroller should be brought out to the ISP Header. It is not actually required for the JTAG algorithm as the control of programming initiated via a JTAG command. However, the programmer / Atmel JTAG-ICE can use the RESET pin to RESET the Target AVR microcontroller to ensure that the AVR JTAG port is not driving any I/O pins which could cause contention during programming. The JTAG-ICE also needs control of the RESET pin to force the AVR microcontroller to execute code when in debugging mode.

3.7 JTAG Programming Schematic

The diagram below details the connections between the Equinox Programmer JTAG connector and the JTAG Port on the Target AVR Microcontroller.

Fig 3.7a – ATmega AVR – JTAG Programming Interface connections

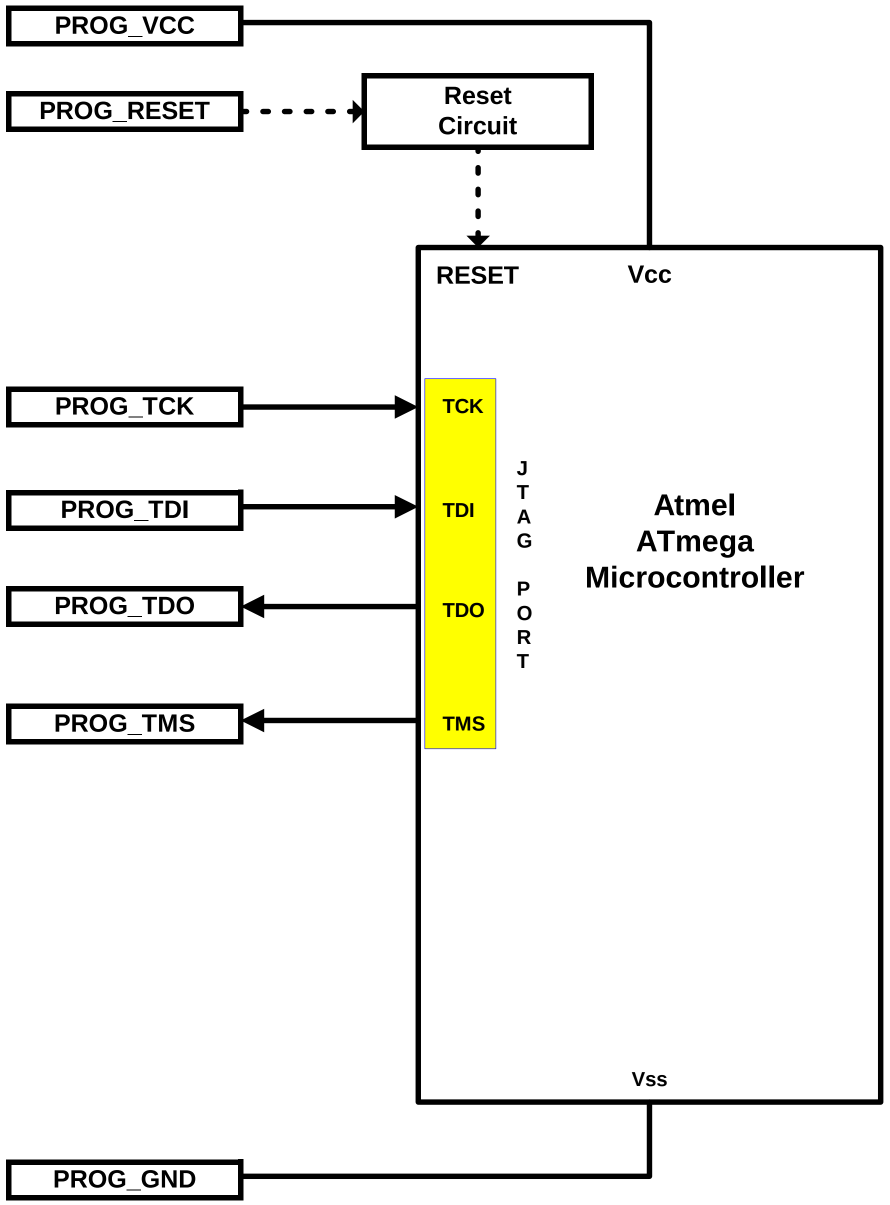


Fig 3.4b – ATmega AVR – JTAG Programming Interface - signal names and directions

Programmer Signal Name	Signal description	Signal direction (from Programmer)	Connect to AVR Microcontroller Pin	Signal direction (from Microcontroller)
PROG_TCK	Test Clock Pin	Output	TCK	Input
PROG_TDI	Test Data Input	Output	TDI	Input
PROG_TDO	Test Data Output	Input	TDO	Output
PROG_TMS	Test Mode Select	Input	TMS	Output
PROG_RESET	RESET	Output	RESET	Input

3.8 Atmel 10-way JTAG Header (JTAG Interface)

This connection method is suitable for interfacing any Equinox ISP Programmer to a Target System which features the following:

- An Atmel device which features a JTAG ISP port e.g. ATmega128 / 323 / 64
- Atmel 10-way IDC JTAG Header
- This is the same header as used on the Atmel JTAG-ICE emulator.

To implement this connection, simply plug the 10-way ISP cable into the ***JTAG ISP Header*** and plug the other end of the cable into the matching header on the Target System.

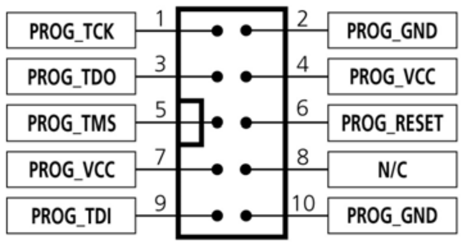


Figure 3.5 - Atmel 10-way JTAG IDC Header viewed from above

Warning!
Connecting to the wrong ISP Header may cause catastrophic damage to the Programmer & Target System

Pin No	Programmer Pin name	Programmer Input / Output	Connect to pin on Target Device	Description
1	PROG_TCK	O	TCK	**JTAG TCK – Test Clock Signal pin** Clock signal from programmer to Target Device JTAG port.
2	PROG_GND	P	GROUND	**Ground Connection** Common ground connection between Programmer and Target System.
3	PROG_TDO	I	TDO	**JTAG TDO – Test Data Output pin** Data signal from Target device JTAG port to programmer.
4	PROG_VCC	P	TARGET_VCC	**Target Vcc Connection** - Pins 4 + 7 are physically connected inside the programmer. - Connects to Vcc rail of Target System. - Pin referred to as VTref on Atmel JTAG-ICE.
5	PROG_TMS	O	TMS	**JTAG TMS – Test Mode Select pin** Mode Select Signal from programmer to Target Device JTAG port.
6	PROG_RESET	O	RESET	**Microcontroller RESET control signal** This pin connects to the main RESET pin of the Target Microcontroller. This pin is not strictly needed for JTAG programming, but it can be used to RESET the Target Device before and after programming.

7	PROG_VCC	P	TARGET_VCC	**Target Vcc Connection** - See pin 4 - Pins 4 + 7 are physically connected inside the programmer.
8	N/C	O	N/C	**Not Connected**
9	PROG_TDI	O	TDI	**JTAG TDI – Test Data Input pin** Data signal from programmer to Target Device JTAG port.
10	PROG_GND	P	GROUND	**Ground Connection** Common ground connection between PROGRAMMER and Target System.

Key
O - Output from programmer to Target Device
I - Input to programmer from Target Device
P - Passive eg. GROUND and power rails
N/C - Not connected

3.9 Creating a JTAG Programming Project

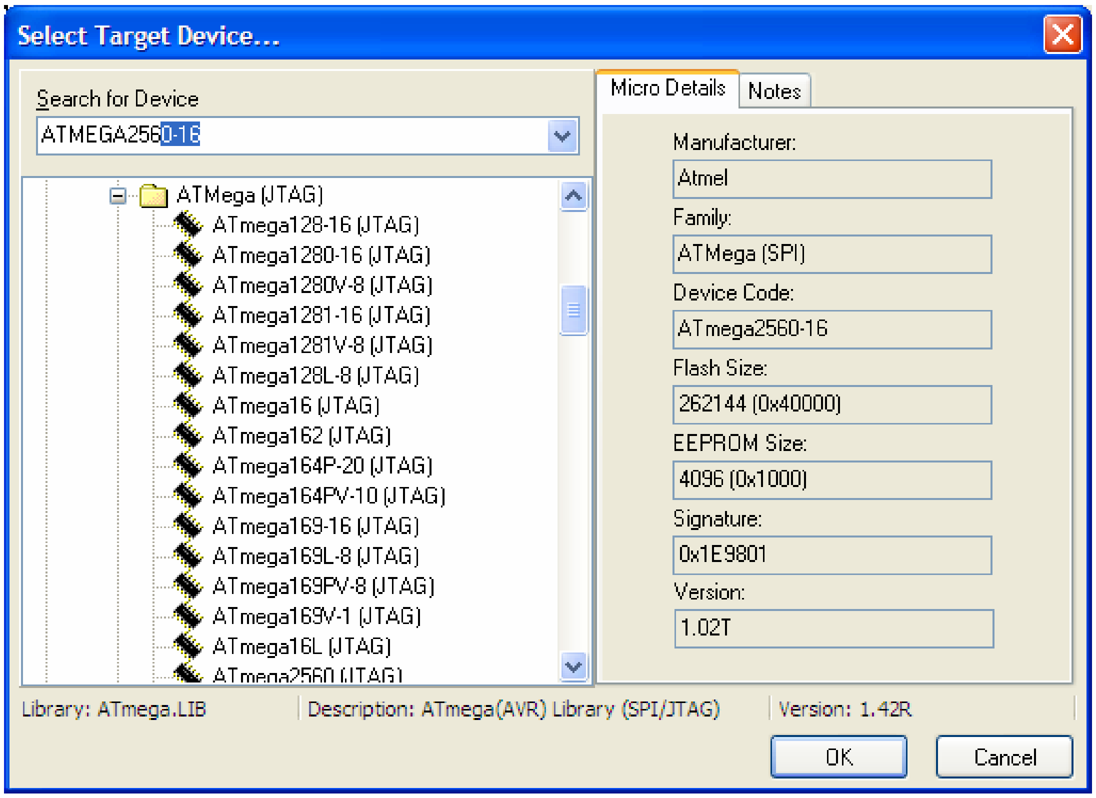
3.9.1 Overview

A Programming Project for a 'JTAG Device' can be created in exactly the same as you would for an 'SPI Device' except that the device must now be selected from the JTAG Device Library. All the settings are the same except for the <Pre-Programming State Machine> and the <JTAG Settings>.

3.9.2 Creating an EDS (Development project)

The simplest way to create a Programming Project for a JTAG device is to use the EDS (Development Mode) Wizard as follows:

- Launch EQTools
- Select <Create a new Development (EDS) Project → the EDS (Development) Wizard will launch
- Click <Next> → the <Select Device> screen will be displayed.


Select Target Device...

Search for Device: ATMEGA2560-16

ATMega (JTAG)
- ATmega128-16 (JTAG)
- ATmega1280-16 (JTAG)
- ATmega1280V-8 (JTAG)
- ATmega1281-16 (JTAG)
- ATmega1281V-8 (JTAG)
- ATmega128L-8 (JTAG)
- ATmega16 (JTAG)
- ATmega162 (JTAG)
- ATmega164P-20 (JTAG)
- ATmega164PV-10 (JTAG)
- ATmega169-16 (JTAG)
- ATmega169L-8 (JTAG)
- ATmega169PV-8 (JTAG)
- ATmega169V-1 (JTAG)
- ATmega16L (JTAG)
- ATmega2560 (JTAG)

Micro Details | Notes

Manufacturer: Atmel
Family: ATMega (SPI)
Device Code: ATmega2560-16
Flash Size: 262144 (0x40000)
EEPROM Size: 4096 (0x1000)
Signature: 0x1E9801
Version: 1.02T

Library: ATmega.LIB | Description: ATmega(AVR) Library (SPI/JTAG) | Version: 1.42R

- Select and click the '+' next to the 'ATmega(JTAG) folder → the list of JTAG devices will be displayed
- Select the required device from the list and then click <OK> → the device is now selected.
- Click <Next> and then fill in the remaining screens in the EDS Wizard
- At the end of the Wizard, click the <Test> button and save the project as eg. ATmega256.ppm.

3.9.3 Testing a JTAG Project in Development (EDS) Mode

If you have clicked the <Test> button at the end of the EDS Wizard, then an EDS (Development Mode) session will now launch.

i. Checking that the programmer can communicate with the Target Chip
To make sure that the programmer can communicate to the Target Chip, try reading back the Device signature as follows:
Select the <FLASH> tab
Locate the <Check Sig> button on the right-hand side of the screen and click it.
→ The programmer will now try to communicate with the Target Chip via the JTAG Interface
→ If the Target Chip responds correctly, then EDS will report 'Signature Read: Pass'.
→ If the Target Chip does not respond, then EDS will report 'Signature Read: Fail'. If this happens, please check that the JTAG connections are correct between the programmer and the Target System and that there is definitely power applied to the Target System.

ii. Programming the FLASH Area

- Select the <FLASH> tab
- Click the <Write> button
- Select the address range you wish to program
- EDS will automatically perform a Chip Erase by default which will erase the entire FLASH before programming any data into it.
- Click <OK> to program the FLASH of the Target Chip.

iii. Programming the EEPROM Area

- Select the <EEPROM> tab
- Click the <Write> button
- Select the address range you wish to program
- EDS will automatically perform a Chip Erase by default which will erase the entire FLASH before programming any data into it.
- The EEPROM address range which you are trying to program must contain 0xFF otherwise the programmer will be unable to program the bytes.
- Click <OK> to program the EEPROM of the Target Chip.

-

4.0 Upgrading your Programmer Firmware

4.1 Overview

Many newer device algorithms require that the firmware of the programmer is upgraded to a specified firmware version. If your programmer has a firmware version of less than 3.00, then will need to use the Configit Utility to upgrade to firmware version 3.00. All firmware upgrades after version 3.00 can be carried out using the Upload Wizard Utility.

4.2 How to check your programmer firmware version

Please follow the steps below to check what version of firmware your programmer is currently running:

- Launch EQTools
- Select <Programmer> <Programmer Info>

→ The firmware version should now be displayed. Eg. 3.00.

4.2 How to upgrade from firmware version 2.xx

If your programmer is running firmware version 2.xx, then it is necessary to upgrade it to version 3.00 first using the Configit – Firmware Update Utility.

Please follow the steps below to upgrade from firmware version 2.xx to version 3.00:

- Download the '***Configit – Firmware Update Utility'*** from the Equinox website
- Unzip the ***Configit*** utility into a temporary directory
- Consult the relevant instructional text file for your programmer. You can find these files contained within the zip file.
- Follow the instructions in the zip file
- Configit will upgrade your programmer firmware to **version 3.00** and it will also install a Boot Loader into the programmer so all future upgrades can be carried out using the ***Upload Wizard*** utility instead.

4.3 How to upgrade from firmware version 3.xx

If your programmer is already running firmware version 3.xx, then you can use the Upload Wizard utility to upgrade your programmer firmware.

Please follow the steps below to upgrade from firmware version 3.xx to a higher version:

- Connect your programmer(s) to the PC COM port
- Power up the programmer(s)
- Download the latest '***EQTools'*** from the Equinox website
- Install this EQTools software
- The installation will install the latest 'Firmware Update Projects' into the following directory: \program files\equinox\Firmware.
- Launch EQTools
- From the 'Welcome to EQTools' screen, select 'Upload a Project Collection' to a programmer

Or

- From the EQTools top menu bar, click the <Upload> icon

→ The Upload Wizard will launch

- Click <Next> twice

→ Detect Programmer(s) screen

- Click <Detect Programmer(s)> button

→ a list of detected programmers is displayed

- Click <Next>

→ 'Select a Project Collection to upload' screen

- Click the <Browse> button and then browse to the following directory: \program files\equinox\Firmware
- Select the relevant Firmware Update Collection file for your programmer and click <OK>
- Click <Next>

→ The 'Select Projects to Upload' screen

  - There should only be 1 project selected
  - Click <Next>

→ 'The attached programmed contains the following Project Collection' screen

  - Click <Backup> if you wish to save the data already resident in the programmer.
  - Click <Next>

→ 'Upload Project(s) to Programmer(s) screen

  - Click the <Upload and Verify> button

→ The firmware project will now be uploaded to the programmer

  - Once uploaded, the programmer GREEN and YELLOW LED's will light up for eg 10 seconds which indicates that the firmware is being upgraded.
  - Once the firmware upgrade is finished, a message will be displayed to tell you the firmware has been upgraded from version x..xx to y.yy.
  - Click <OK> and then <Finish>

→ The programmer firmware update is now finished.