AVRminiV3.1 Manual

1. AVRminiV3.1 Overview

The AVRminiV3.1 board is a low-cost versatile development board for Atmel AVR processors. The AVRminiV3.1 supports all AVR processors in 40-pin and 64-pin packages except the Atmega169. This means that the AVRmini board can be used to experiment with and develop hardware and code for more than 25 different AVR processors (about 65% of the total AVR product line).

The AVRmini board offers enough built-in debugging hardware and configuration jumpers to make it useful for development, yet it is simple and small enough to use as an application board within many products or devices.

The AVRmini is highly compatible with Atmel's own AVR development board, the STK500. I/O port header and ISP header pinouts are the same between the AVRmini and the STK500. The compatibility helps speed the transition from the STK500 to the AVRmini.

2. AVRminiV3.1 Features and Specifications

2.1. Standard Features:

- **Socket for analog-pinout 40-pin AVR processors** (Atmega163, Atmega16, etc)
- **Socket for digital-pinout 40-pin AVR processors** (AT90S8515, Atmega161, etc)
- **Solder pads for 64-QFP AVR processors** (Atmega128, Atmega64, Atmega103)
- **Switching 5V power supply** (5V 0.5A max output, up to 20VDC input, ~90% efficient)
- **AVR ports A,B,C,D,E,F available through 10-pin headers**
  (Port headers match pin-out used by STK500)
- **AVR ports A,B,C,D protected by 100Ω series resistors**
- **Bypass/testing headers for port A,B,C,D series resistors**
- **Header for HD44780-type character LCDs** (with on-board contrast adjust, -8V to 5V)
- **ISP header** (connects to STK500 or AVRISP for programming)
- **CPU crystal** (standard HC49U crystal socket or surface-mount crystal)
- **Two configurable DB-9 serial port connectors**
  (one DB-9 may be used as a PonyProg-style programming interface)
- **4 user pushbuttons + 4 user LEDs**
- **Reset Pushbutton**
- **Power LED and Power Switch**
- **Power jack for Wall-brick AC adapter**
- **Board Size 3"x4"**

2.2. Optional Features:

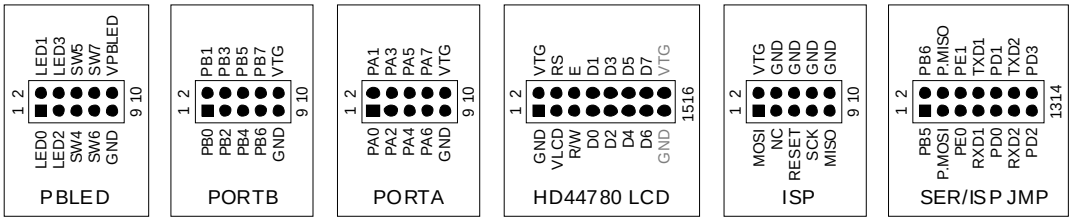
- Up to 512KByte external SRAM memory and 74HC573 address latch
- Real-time clock crystal (32.768KHz)
- Bridge rectifier on power input

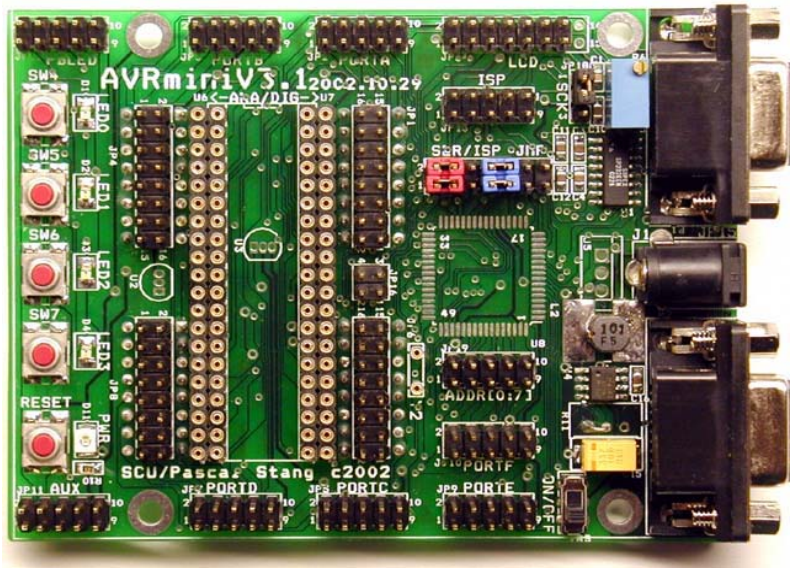
- Advantage: allows board to use any AC adaptor brick 7V-30V regardless of connector polarity
- Disadvantage: input voltage must be at least 7V instead of 5V

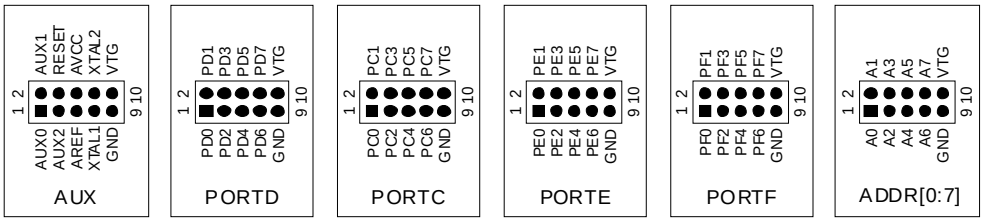
- Screw-terminal power input (instead of AC adapter style power jack)

3. Board Reference

3.1. Connectors







3.2. Power Supply and Power Input

The AVRminiV3.1 board uses a high-efficiency switching regulator to produce a steady +5V to power the AVR processor and associated circuitry. Unregulated input power may be applied to the board's coaxial power jack located between the serial ports. Input power can be of any polarity, but must be 7-20VDC. Almost any typical DC "wall block" transformer in the 7-16V range is suitable for powering the AVRmini. The regulator will convert the input voltage to 5VDC with an efficiency of 80-90%. Unlike less-efficient linear regulators (e.g. 7805), the AVRmini power supply will stay

cool under the full range of input voltages and loads. Typical power consumption of the AVRmini including AVR processor is ~30mA, leaving ample current to drive user circuits. The regulated 5V power may be tapped at pins 9 & 10 of any I/O port connector.

Regulator Input Voltage Polarity	Any (when D10 is installed)
Regulator Input Voltage	7-20VDC
Regulator Output Voltage	5VDC +/- 10%
Regulator Output Current	500mA max
Current consumption of AVRmini (all on-board components)	100mA max @ 5V 30mA typ. @5V

3.3. Serial Ports

The AVRminiV3.1 has two standard DB-9 connectors which can function as RS-232 serial ports. Both SERIAL1 and SERIAL2 are "3-wire" serial ports with TxD, RxD, and GND signals active (pins 2, 3, and 5 respectively). After passing through the level-shifter IC U1, the logic-level TxD and RxD data signals are available on the SER/ISP jumper block. From there, TxD and RxD may be connected to the AVR processor UART. See Section 4 for more information on how to route these signals.

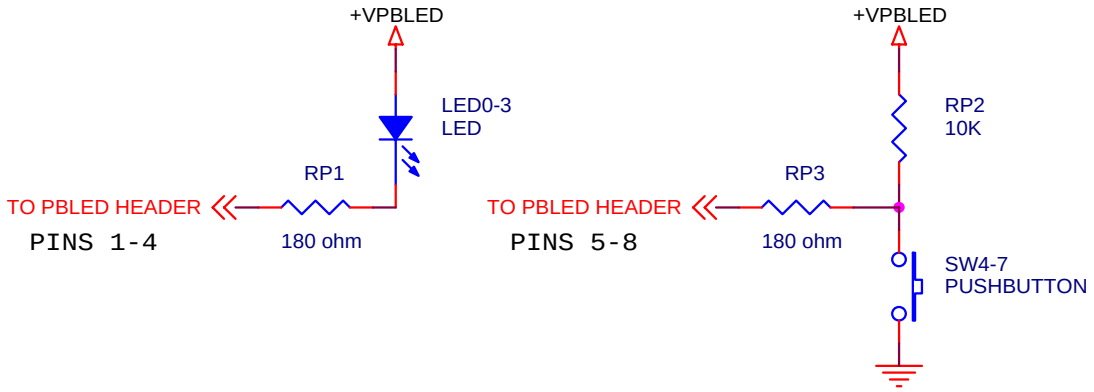
SERIAL2 is equipped with additional circuitry to allow direct in-system programming (ISP) of the processor using a special serial protocol on this port. Several publicly-available programs can generate this protocol, including *PonyProg* and *uisp*. The protocol involves non-standard use of the RS-232 serial control lines and may not work with USB-to-serial adapters*.

* For reprogramming the processor using standard serial signaling (compatible with USB-to-serial adapters), a bootloader should be used. Your AVRminiV3.1 may have come preloaded with a bootloader program. See Section 5 for more information.

3.4. LED and Pushbutton Circuits

The figure below shows how the four LEDs and pushbuttons on the AVRmini are connected. Both the LEDs and pushbuttons are active-low. Sending a logic low (0V) output to pins 1-4 of the PBLED header will cause LED0-3 to light, respectively. Pressing pushbuttons 4-7 will cause a logic low (0V) to be generated on pins 5-8 of the PBLED header, respectively.

Note that the power and ground for the LED and pushbutton circuit is separate from the power and ground used by the rest of the AVRmini board. The LEDs and pushbuttons will not work unless power is applied to pins 9 & 10 of the PBLED header.



4. Configuring the AVRminiV3.1

There are a number of configurable settings on the AVRminiV3.1 board. This section describes how these settings are made and shows some examples.

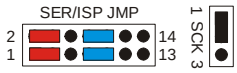
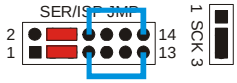
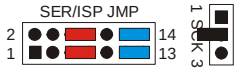
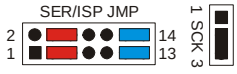
4.1. Setting the Serial/ISP and SCK Jumpers (SER/ISP JMP & SCK)

The Serial/ISP and SCK headers allow the user to configure how the AVR processor is connected to the following connectors:

- ISP (In-system Programming) Connector
- P1 – the Programming/Serial Port #2 Connector
- P2 – the Serial Port #1 Connector

Since different AVR processors use different pins for serial port I/O and programming, the jumpers on the SER/ISP and SCK headers must be set correctly for the type of processor being used, and for the way the user intends to use it.

NOTE: Incorrect jumper placement should not cause damage to the AVRmini board or the AVR processor, but may make programming of the processor or use of the serial ports impossible.

Example SER/ISP and SCK Jumper Settings	
	**40-pin AVR Processors** (incl. Atmega163, mega323, mega16, mega32, AT90S8515, AT90S8535, etc) This setting makes the following connections: • ISP Connector enabled for programming via STK500/AVRISP • SERIAL1 connected to AVR processor's UART (serial port) • SERIAL2 connected for programming via PonyProg, UISP, etc.
	**64-pin AVR processors** (incl. Atmega128, mega64, mega103) This setting makes the following connections: • ISP Connector enabled for programming via STK500/AVRISP • SERIAL1 connected to AVR processor's UART1 (2nd serial port) • SERIAL2 connected for programming via PonyProg, UISP, etc. • **AVR processor's UART0 not accessible Note: This configuration requires the use of a 2-pin jumper cable like those provided with the STK500 kit.
	**64-pin AVR processors** (incl. Atmega128, mega64, mega103) This setting makes the following connections: • ISP Connector unusable • SERIAL1 connected to AVR processor's UART0 • SERIAL2 connected to AVR processor's UART1 • **Programming port not accessible Note: Because this configuration offers two serial ports at the expense of no programming port, it is recommended that a bootloader program be used on the processor to allow programming through one of the serial ports.
	**64-pin AVR processors** (incl. Atmega128, mega64, mega103) This setting makes the following connections: • ISP Connector enabled for programming via STK500/AVRISP • SERIAL2 connected for programming via PonyProg, UISP, etc. • SERIAL2 also connected to AVR processor's UART1 • SERIAL1 unused • **AVR processor's UART0 not accessible

5. Programming the AVR processor on the AVRminiV3.1

This section discusses methods to load your code into the AVR processor on the AVRmini board. Note that in this case, *programming* refers to the act of loading the code into the processors FLASH memory, and not the writing of the code in the first place (such as in C or assembly). The term *programming* also can mean changing of certain AVR processor settings such as the Fuse Bits, Lock Bits, or reading and writing the on-chip EEPROM memory. Here we will focus on the loading of code.

Once compiled or assembled, programs that are ready to be loaded into the AVR processor memory are usually in the form of a **.hex* file. This **.hex* file is equivalent to a PC **.exe* file; it is a program that is ready to run. There are three distinct ways to load your **.hex* file into the processor.

5.1. Programming via the 10-pin ISP Port

This is the most direct method of programming. To use this method, you will need one of an Atmel STK500 board, an Atmel AVRISP dongle, or a third-party ISP tool. In general, you will need to connect the programming tool/dongle to the AVRmini's 10-pin ISP port. Since this connector is easily reversed, pay special attention to the location of pin 1 on the programming cable and match it to pin 1 on the AVRmini board. Once connected, follow the instructions that came with the programming tool/dongle.

For example, if using an STK500 board, you should follow this procedure:

1. Remove any processor from the STK500 board.
2. Use a 10-pin ribbon cable to connect the STK500 **ISP10PIN** port to the AVRmini ISP port. Take care to match pin1 to pin1 at both ends.
3. Check that the AVRmini SER/ISP jumpers are set to allow ISP programming.
4. Connect a serial cable from your computer to the STK500 RS232 CTRL port.
5. Connect power to the STK500 and turn it on. The AVRmini will automatically draw power from the STK500.
6. Use Atmel's free AVR Studio software to operate the STK500 board. Use the software as though you were programming a processor located on the STK500 board.

5.2. Programming via SERIAL2

This method is the same as regular ISP described above, except that the special ISP signals are generated by a non-standard use of an RS-232 serial port. The advantage is that no additional hardware, beyond the AVRmini and a serial cable, is required to accomplish programming. However, due to the specialized signaling involved, this method seldom works with USB-to-serial converter dongles; a built-in serial port is required. Follow this procedure:

1. Connect a serial cable from your computer to SERIAL2.
2. Check that the AVRmini SER/ISP jumpers are set to allow SERIAL2 programming.
3. Connect power to the AVRmini and turn it on.
4. If using PonyProg software
    a. Select the serial port to use for programming (Setup menu).
    b. Do a "Probe" which should succeed.
    c. Perform a calibration if you haven't done so before (Setup menu).
    d. Select the appropriate AVR processor from the drop-down lists.
    e. Open your **.hex* file.

f. Select Write Program (Flash) from the Command menu.

5. If using UISP software
    a. Consult the option help for USIP to determine the current format for arguments and commands.
    b. You will want to use the DASA2 programming protocol via a serial port.

5.3. Programming via SERIAL1 and a Bootloader

This method is possibly the fastest and easiest, but requires a bootloader (a small piece of software) to be installed in your AVR processor. If a bootloader is not already loaded, you will need to use one of the other methods to install it initially. A bootloader is much like the BIOS of a PC, allowing the computer to boot from any available disk. In this case, the bootloader will allow you to supply your program via the serial port.

The initial loading of a bootloader is beyond the scope of this document. From here on, it is assumed that your processor is already loaded with the STKLOAD bootloader. If you purchased your AVRmini after June 1, 2004, your board was shipped with the bootloader preprogrammed.
The STKLOAD bootloader emulates the serial programming protocol of an STK500 board. Using this bootloader, you can program your processor with any software meant for the STK500 (such as the freely-available easy-to-use Atmel AVR Studio).

Follow this procedure for programming:

1. Connect a serial cable from your computer to the AVRmini SERIAL1 port.
2. Connect power to the AVRmini and turn it on.
3. Open the AVR Studio program
4. Press the RESET button on the AVRmini to activate the bootloader
5. Within 3 seconds of pressing reset, click the programmer icon in AVR Studio (small black "AVR" chip).
6. AVR Studio will attempt to detect the bootloader
    - If the programming window opens and the Log shows "Detect failed", then return to step 4.
    - If the Log shows "Detected STK500…" then proceed.
7. Select your processor from the Device drop-down list.
8. Select your hex file in the FLASH window
9. Click Program
10. Watch the Log, and when programming has completed, you may close the programmer window if you wish.
11. Press RESET again. This time, do not click anything in AVR Studio for 3+ seconds. The bootloader will timeout and your program will begin running.
12. To load a new hex file, repeat from step 4. If you did not close the programmer window, you can skip steps 5-8.