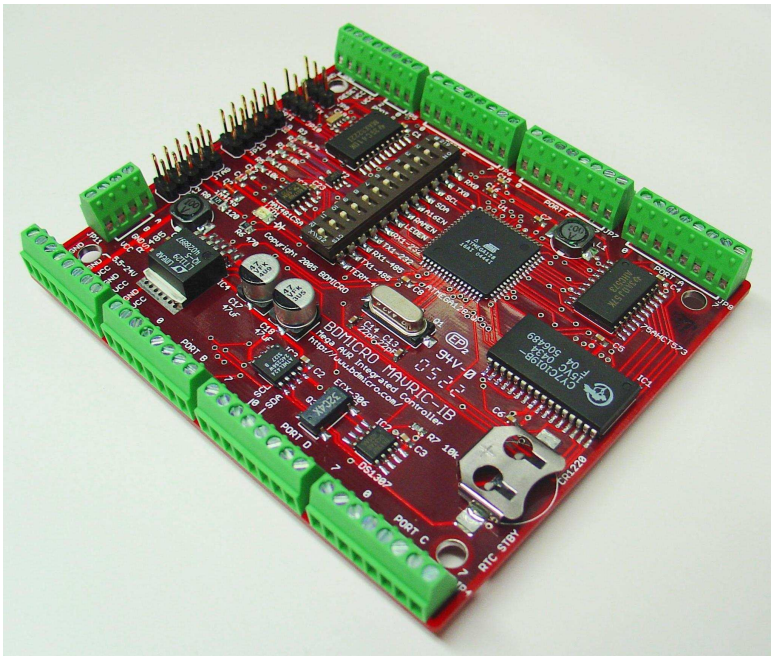


MAVRIC-IB

Mega AVR Integrated Controller I
Revision B
Technical Manual

BDMICRO
http://www.bdmicro.com/

May 30, 2006

DISCLAIMER

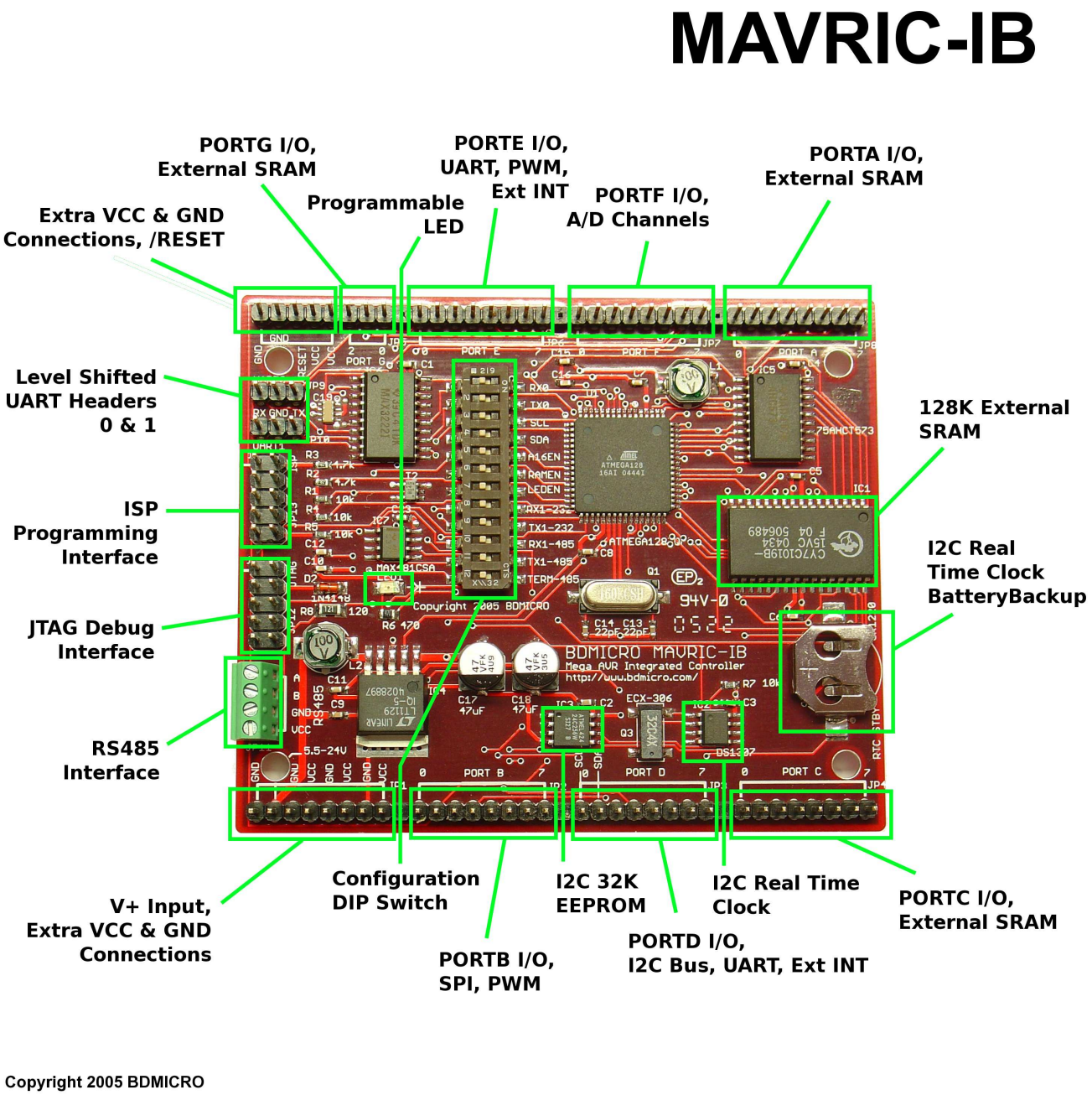
BDMICRO products are designed and assembled with care and all assembled products must pass functional testing to ensure quality before they are shipped. However, our products are not rated for use in medical, life support, or systems where failure could result in the loss of life or have serious life threatening or economic impact.

BDMICRO does not warrant merchantability for any purpose and shall not be liable for any direct, indirect, incidental, special, exemplary, or consequential damages of any kind, however incurred, through the use of our products.

Contents

1 MAVRIC-IB Layout

The BDMICRO MAVRIC-IB is a powerful microcontroller board based on the ATmega128 MCU. It is fully programmable using familiar languages such as C and BASIC. The GNU GCC C compiler is very popular as it comprises the core of the de-facto standard tool chain of the open source community. Additionally, MCS Electronic's BASCOM-AVR BASIC compiler is also very popular, economical, and highly regarded by AVR enthusiasts.



The MAVRIC-IB is the perfect microcontroller for many applications such as robotics where a fusion of many sensors and digital I/O are needed. Its copious program and data space allows one to implement sophisticated algorithms and logic without having to compromise features. MAVRIC-IB's 8 channel, 10-bit A/D converter makes short work of reading analog sensors like the SHARP GP2D12 IR distance sensor. It has 2 on-board UARTs both of which are level-shifted to provide true RS232 levels, or they may be used using the unshifted TTL levels if that is more convenient. One of the UARTs can optionally be run as an RS485 interface which incorporates on-board termination and independent transmitter/receiver control. MAVRIC-IB is I2C ready with on-board pull-ups so one can hook up existing I2C devices such as SRF08 Ultrasonic Rangers, CMPS01 compass modules, plus many other I2C peripherals. In addition to all that, 6 high resolution PWM outputs are available for driving servos and PWM motors with high precision. And with up to 51 digital I/O pins, it can handle even the most complex and demanding control tasks.

The on-board standard programming headers make development easy — 10-pin serial ISP header for connecting to the AVRISP or STK500, and also the standard 10-pin JTAG header for using a JTAGICE for programming and single-step, source level debugging.

The MAVRIC-IB incorporates an advanced, on-board, low-dropout voltage regulator and can be powered with as little as 5.5 Volts. Four mounting holes are provided for securely mounting the MAVRIC-IB board, and with it's small size of 2.2 x 3.6 inches, it can be mounted just about anywhere.

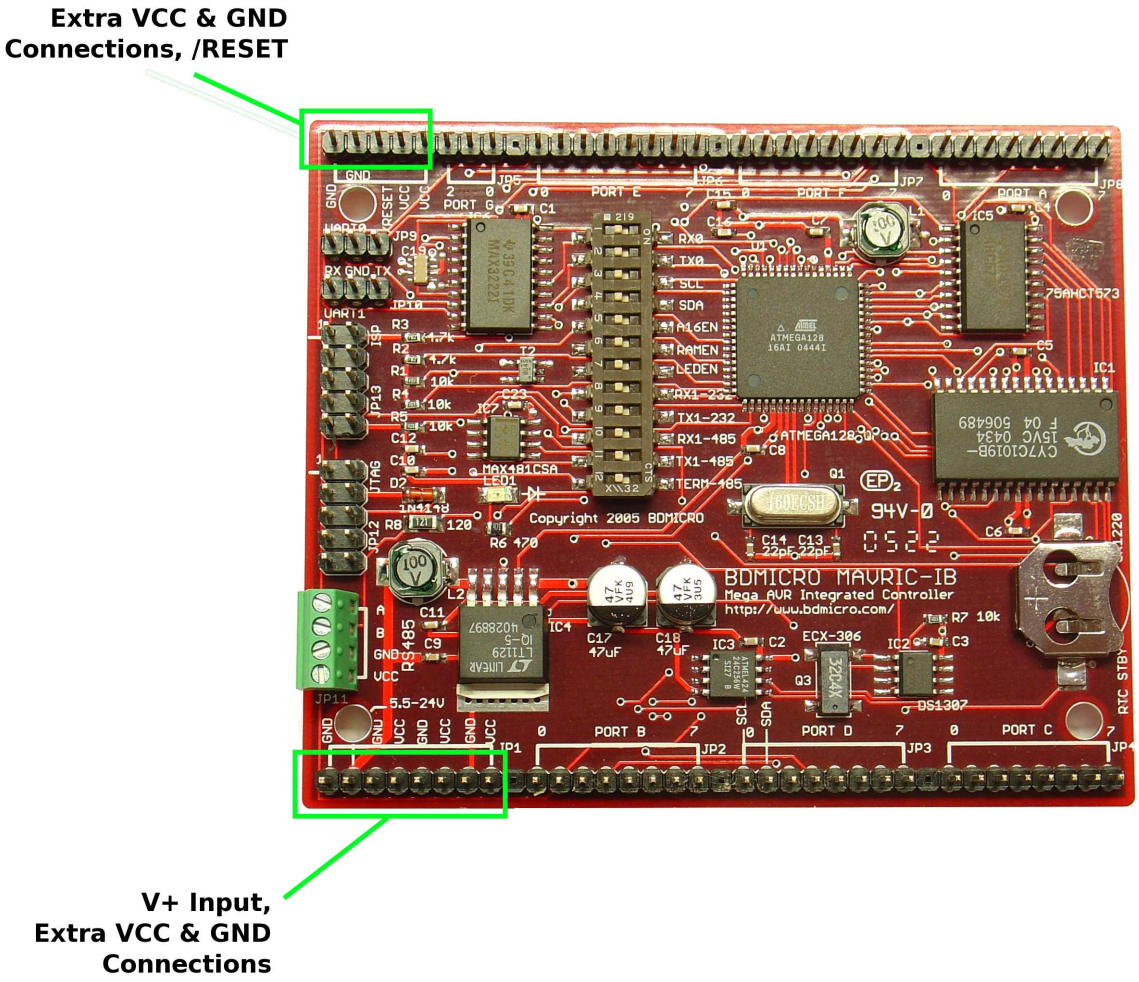
The capabilities of the ATmega128 MCU are numerous. Here are some of the highlights of the chip:

- 128K Flash memory (program space)
- 4K internal static RAM
- 4K EEPROM
- optional 64K external static RAM
- 8 channel analog to digital converter
- dual UARTs
- I^2C interface
- up to 53 digital I/O pins
- JTAG interface for programming and debugging
- 2 8-bit timers, 1 16-bit timer
- 6 PWM channels
- Watchdog timer

For a more detailed look at the ATmega128, the best reference is the datasheet provided by the chip manufacturer, Atmel. The datasheet is available from Atmel in the form of a PDF file here:

`http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf`

2 Power Supply Connections

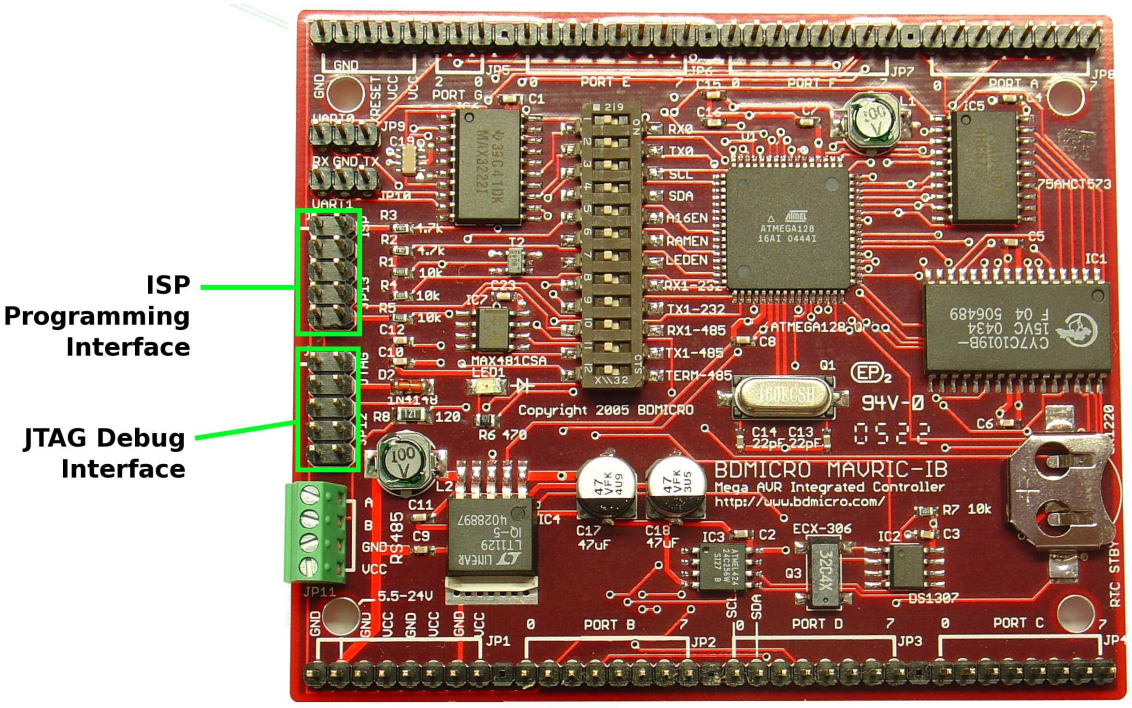


Note that all **VCC** connections are regulated 5.0 Volts, input or output. Connections labeled `V+` are unregulated 5.5 - 24V input and feed into the on-board regulator where it is regulated down to 5 Volts for the board electronics.

One can either supply `Vcc` directly from an off-board 5V regulated power supply in which case power into the board should connect directly to one of the `Vcc` connection points. Or one can supply the MAVRIC-IB board with unregulated power input, in which case, it should be applied to the `V+` power input. In this case, a small amount of regulated 5V is available for LEDs or low-current sensors from the `Vcc` connection points. The amount of current available depends on the voltage level on the V+ pin - the higher the voltage on V+, the less current available for auxiliary devices. One should never supply power to both the `Vcc` power connection and the `V+` power connection at the same time.

3 In-System Programming and JTAG Debug Headers

The standard 10-pin ISP programming header and the JTAG programming / debug headers are shown below.

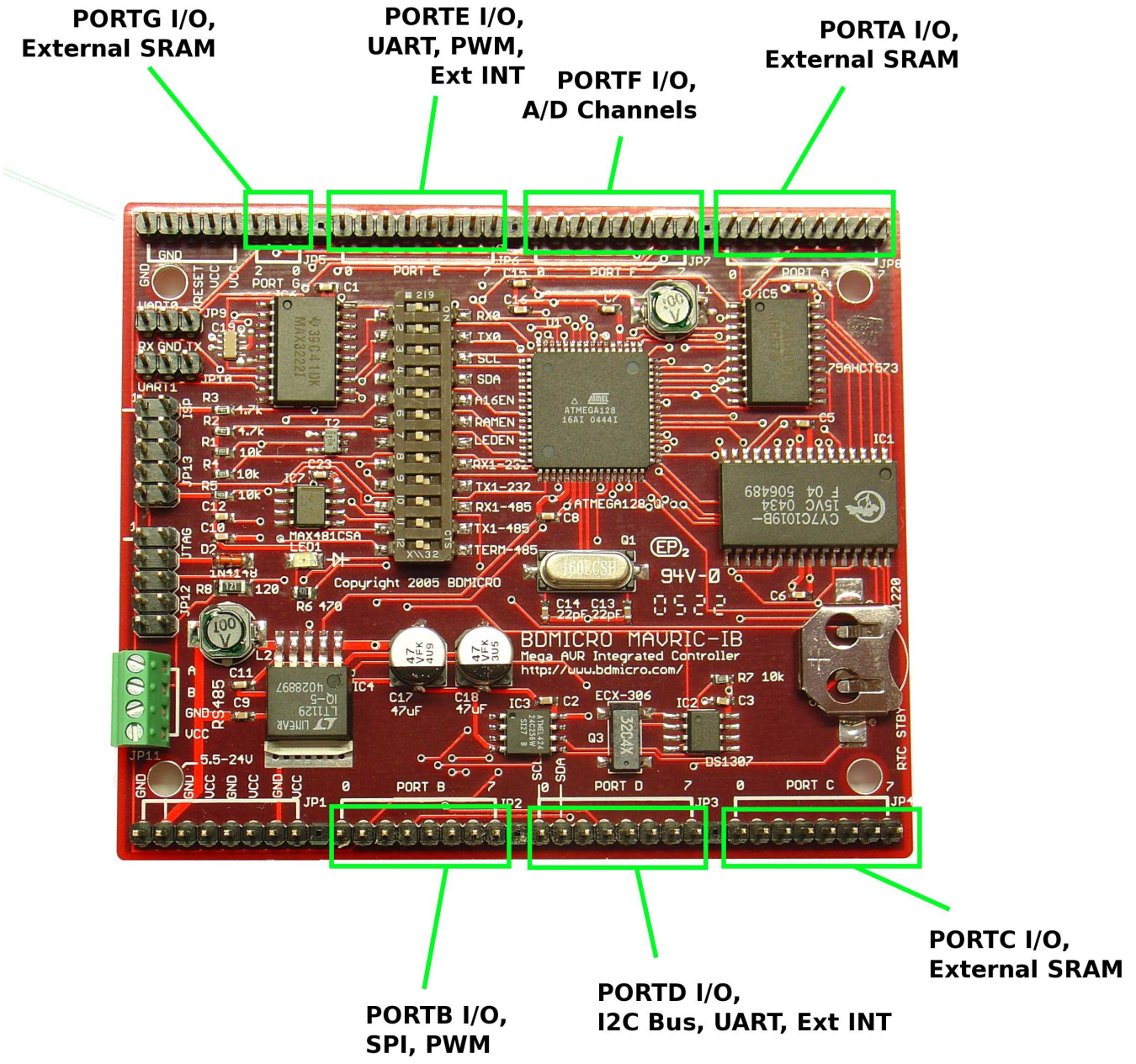


Both headers can be used to download the firmware to the MAVRIC-IB board. The JTAG header, in addition, can be used to provide source level single-step debugging of the program as it runs natively on the chip itself. The JTAGICE or equivalent programmer is required to utilize the JTAG header.

Many programmers are available for utilizing the ISP programming header. Most are very economical and range from do-it-yourself versions that can be built with just a few resistors and utilize the parallel port of a standard PC, to advanced development boards like Atmel's STK500. One of the best and most economical ISP programmers is the Atmel AVRISP available from several sources including Digikey for around $30.

4 Port Header Connections

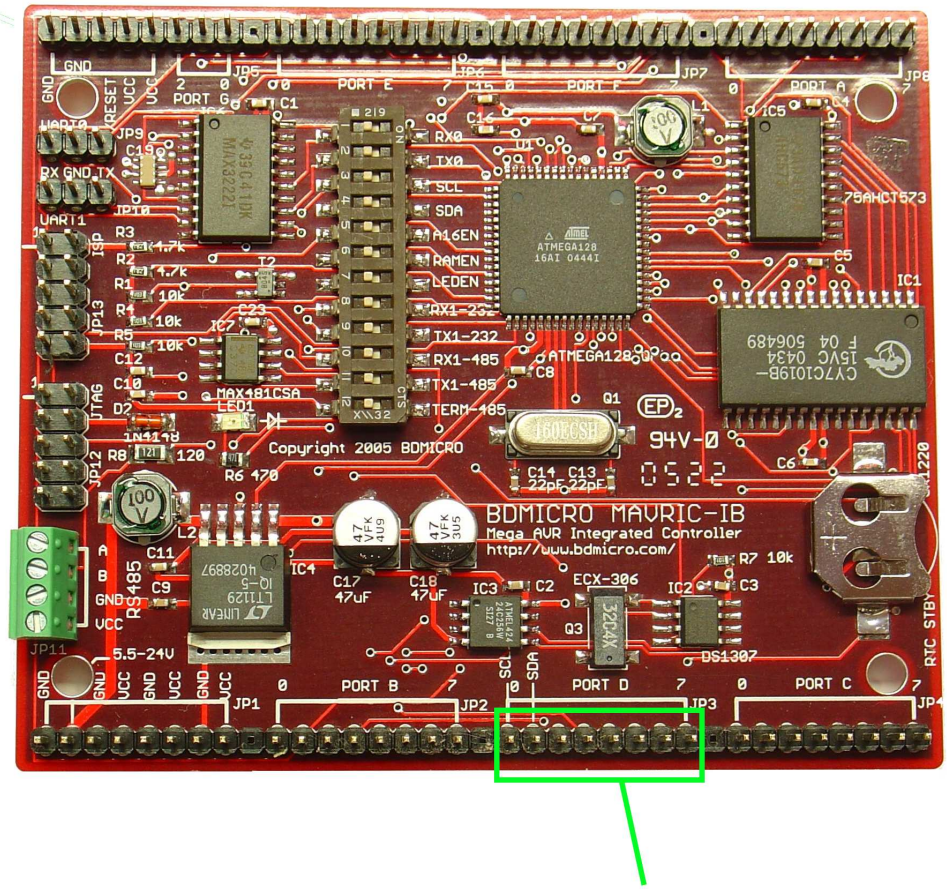
Nearly all ATmega128 port pins are brought out to headers around the perimeter of the MAVRIC-IB board. The exceptions are PORTG3 and PORTG4, AREF, the crystal connections, and the PEN pin. All other pins are brought out. All headers are labeled along the perimeter of the board, and connect directly to the MCU pin of the same name, see the diagram below.



Note that while most pins connect only to the MCU pin, some pins also connect to other chips on the board. These include PORTE0, PORTE1, PORTD2, and PORTD3 which connect to the MAX3222 RS232 level shifter and RS485 tranceiver, PORTD0 and PORTD1 which connect to the I^2C bus pull-ups, PORTB0 (on-board LED), and /RESET which connect to the ISP programming header, and ports PORTF4-7 which connect to the JTAG programming header. Also, PORTA, PORTC, and PORTG0-2 connect to the external SRAM chip and latch for the external memory interface. When the external SRAM chip is disabled through the DIP switch, the SRAM and latch pins are in a high impedance state allowing the PORTA, PORTC, and PORTG pins to be used for other purposes. All of these connections can be disabled through the use of the configuration DIP switch. See the schematic diagram for additional information.

4.1 I^2C Connections

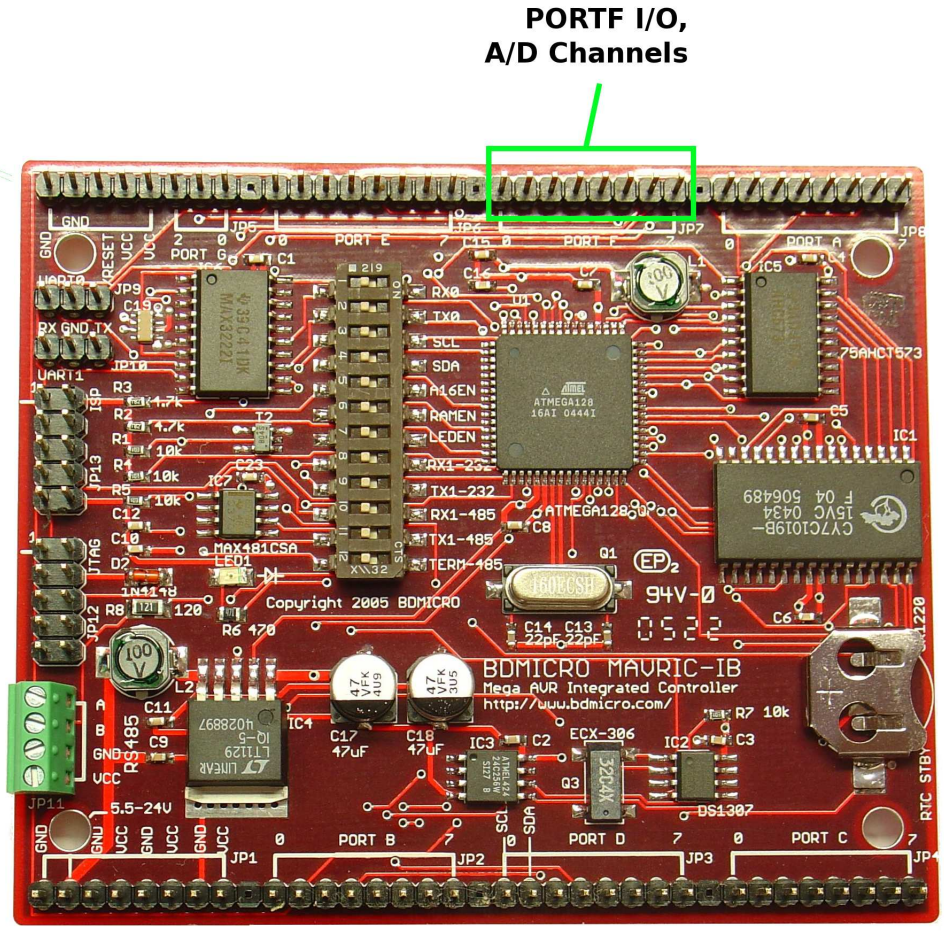
The I^2C bus SCL and SDA connection points are shared with PORTD0 and PORTD1 are are shown in the diagram below. The first two pins on the left, PORTD0 and PORTD1 are I^2C SCL and I^2C SDA respectively.



Note that in order to utilize these pins for I^2C, the I^2C pull-ups must be enabled (DIP switch positions 3 & 4 ON), and the I^2C interface must be enabled by the firmware program.

4.2 A/D Inputs

The A/D Converter inputs are shared with the 8 pins of PORTF as shown below. The AREF reference connection is not brought out to the board edge, however, the ATmega128 MCU incorporates a good on-chip band-gap reference or one may use VCC. These are selectable using the REF0 and REF1 bits of the MCU's ADMUX register.



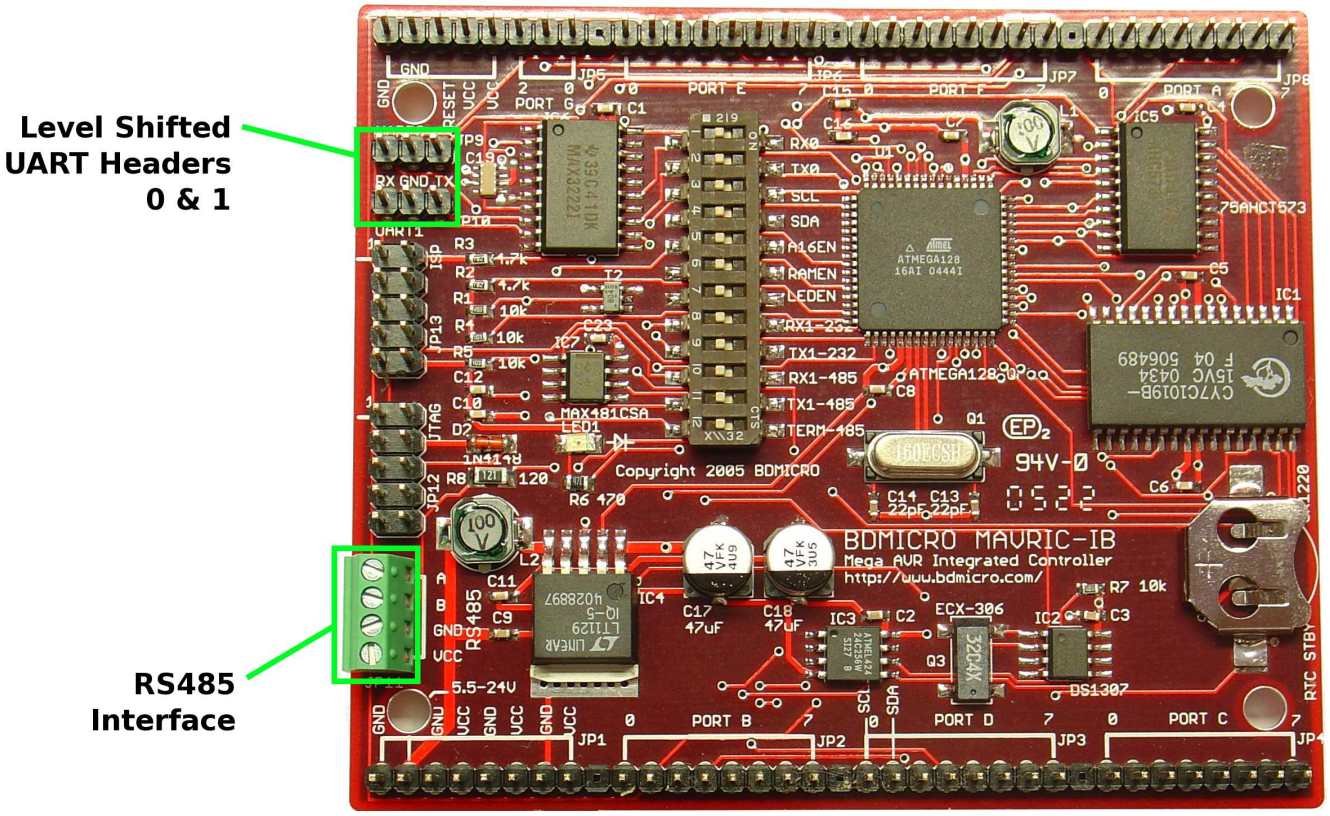
The on-board voltage reference of the ATmega128 remains available and can be configured to match either Vcc (5.0V) or set to 2.56V by setting the appropriate bits of the ATmega128's ADMUX register. Note that while the external A/D reference pin is provided, the on-board reference is sufficient for many applications.

4.3 PORTG3 and PORTG4 Connections

PORTG3 and PORTG4 are hard-wired to the RS485 tranceiver's /RE (NOT receiver enable) and DE (driver enable) lines respectively. These pins are thus dedicated to controlling the state of the RS485 tranceiver's state. The special function of these pins as an alternate clock source for TIMER0 is not available.

5 Serial Communications Connections

The level shifted RS232 and RS485 connection points use convenient screw terminals. These are found on JP6 on the left hand side of the board. Also included on that screw terminal are a `Vcc` and `Gnd` connection point for convenience. See the digram below.



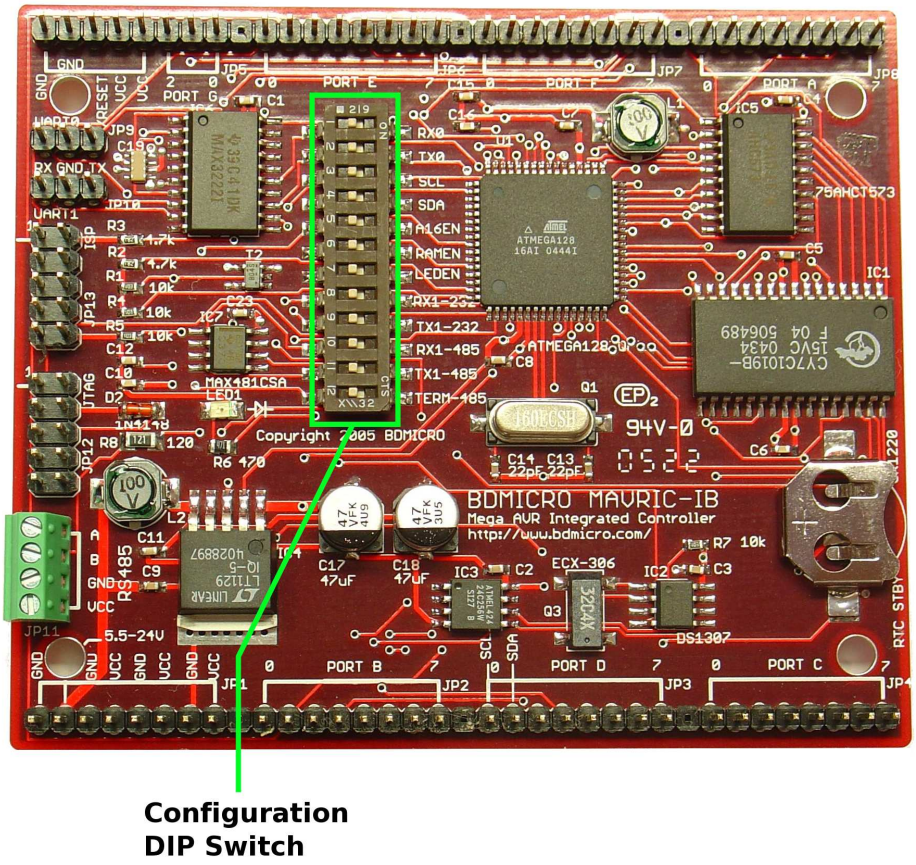
5.1 UART0 Startup Delay

Note that the RS232 level shifter is in a shutdown state when the MCU's /RESET line is active. The purpose is so that the RS232 level shifter does not interfere with ISP programming since they both use the PORTE0 and PORTE1 pins. When the level shifter is in shutdown, its I/O lines are in a high impedance state, and thus does not interfere with ISP programming.

Because the level shifter's charge pump takes a small amount of time to charge, one should not immediately begin sending RS232 communications from the MAVRIC-IIB after program reset. Ensure that at least a few milliseconds elapse after program reset either through normal initialization or through a hard coded delay before initiating any RS232 communications from the MAVRIC-IIB to give the level shifter's charge pump time to reach capacity.

6 DIP Switch Settings

MAVRIC-IB configuration options are selected using the on-board 12 position DIP switch. The DIP switch is used to enable the on-board LED, I^2C pull-up resistors, external static RAM chip enable, level shifted RS232 ports, RS485 transceiver, and the RS485 terminating resistor. Refer to the diagram below for the location of the DIP switches.



Note that the UART1 RS485 and RS232 selection DIP switches should not be in the “ON” position at the same time. Enabling both the UART1-RS485 and UART1-RS232 DIP switches at the same time connects the RS485 and RS232 drivers directly together. Thus, when changing the configuration of the UART1 selection, turn off UART1-RS485 and UART1-RS232 first, then turn on either UART1-RS485 to enable the RS485 interface, or turn on positions UART1-RS232 to enable the RS232 interface.

For a complete list of all DIP switch positions and their function, see the table below:

Position	Name	Function
1	RX0	**On =** Enable UART0 receiver; **Off =** disconnect RX0 (PORTE0) from the MAX3222 chip
2	TX0	**On =** Enable UART0 transmitter; **Off =** disconnect TX0 (PORTE1) from the MAX3222 chip
3	SCL	**On =** Enable I^2C SCL; **Off =** disconnect SCL (PORTD0) from the I^2C bus
4	SDA	**On =** Enable I^2C SDA; **Off =** disconnect SDA (PORTD1) from the I^2C bus
5	A16-EN	**On =** Connect PORTD5 to the external RAM A16 address line; **Off =** pull the external RAM A16 address line to GROUND
6	RAM-EN	**On =** Enable external RAM and Latch; **Off =** Disable external RAM and Latch (all external RAM inputs and outputs are set to a high impedance state)
7	LED-EN	**On =** Connect on-board LED to PORTB0; **Off =** on-board LED not connected
8	RX1-RS232	**On =** Connect UART1 RX (PORTD2) to RS232 level shifter; **Off =** disconnect RX1 (PORTD2) from RS232 level shifter
9	TX1-RS232	**On =** Connect UART1 TX (PORTD3) to RS232 level shifter; **Off =** disconnect TX1 (PORTD3) from RS232 level shifter
10	RX1-RS485	**On =** Connect UART1 RX (PORTD2) to RS485 transceiver; **Off =** disconnect RX1 (PORTD2) from RS485 transceiver
11	TX1-RS485	**On =** Connect UART1 TX (PORTD3) to RS485 transceiver; **Off =** disconnect TX1 (PORTD3) from RS485 transceiver
12	TERM-485	**On =** RS485 terminator enabled; **Off =** RS485 terminator disabled

7 I^2C Addresses

Up to 128 devices can coexist on the I^2C bus. The MAVRIC-IB board incorporates 2 such devices, the Dallas DS1307 Real Time Clock and the 24C256 Serial EEPROM. The DS1307 resides at the fixed I^2C address 0x68 and the AT24C256 serial EEPROM resides at I^2C address 0x50.

8 Utilizing the full 128K of RAM

The ATmega128 can directly access up to 64K of external memory. However, the MAVRIC-IB incorporates a 128K RAM chip, the upper 64K of which can be accessed via bank switching. On the MAVRIC-IB, the A16 address line of the RAM chip is optionally tied to PORTD5 of the ATmega128 via dip switch position 5. When switch 5 is in the `ON` position, the A16 address line follows the state of PORTD5, and thus the extra 64K can be accessed under program control. When switch 5 is in the `OFF` position, the A16 address line is tied to ground and the PORTD5 pin is free for other purposes.

Only one 64K bank of RAM can be seen by the CPU at a time. Thus one needs to be careful when switching to the other memory bank so that variables and data that are being referenced in the previous bank aren't accidentally referenced while that memory bank is switched out.

9 Special Considerations for PORT A, PORT C, and PORT G

The design of the ATmega128 MCU precludes the use of ports A and C while at the same time utilizing external memory. Thus, if one intends to use port A or C for general purpose I/O, one must ensure that the RAM chip is disabled by switching dip switch position 6 to the `OFF` position. If the RAM chip is left enabled while ports A or C are used for general purpose I/O, it is possible to damage the RAM chip or the ATmega128 chip, or both.

Likewise, PORTG0, PORTG1, and PORTG2 are used by the MCU when external RAM is enabled for the read, write, and address latch selection logic. Thus, these I/O ports are not available for general purpose I/O while external RAM is enabled.

10 Real Time Clock Battery Backup

The MAVRIC-IB board incorporates an on-board battery holder for providing backup power to the Dallas DS1307 real time clock when power is not connected. The holder is designed to accomodate a 12 mm battery such as the Sony CR1220. When installed, the battery will provide power to the real time clock which will accurately keep time even though external power to the board is not connected.

11 Fuse Bit Settings

Atmel AVR processors incorporate *fuse bits* which control various functions of the chip and persist even across a chip erase. By default, Atmel ships the ATmega128 with several fuse bits already programmed by default. Notably, the M103C fuse bit (ATmega103 compatibility mode) is enabled, as well as JTAGEN which enables the JTAG debugging lines of PORT F. Also, by default, the internal clock source is selected to run the processor at 1 MHz.

If a kit was ordered, the included ATmega128 processor's fuse bits are all at their default values and will need to be changed to run on the MAVRIC-IB board at 16 MHz. For 16 MHz operation, one will need to program the CKOPT fuse bit (set to '0'). The clock selection lines will need to be modified as well: unprogram CKSEL3, CKSEL2, CKSEL1, and CKSEL0 (set to '1'). Also recommended when using the external crystal is to set SUT1 and SUT0 to '1' (unprogrammed). This setting is for slow rising power which may be necessary depending on the power supply. Additionally, one will need to unprogram the M103C fuse bit (set to '1') if compiling code to run on an ATmega128 (as opposed to an ATmega103).

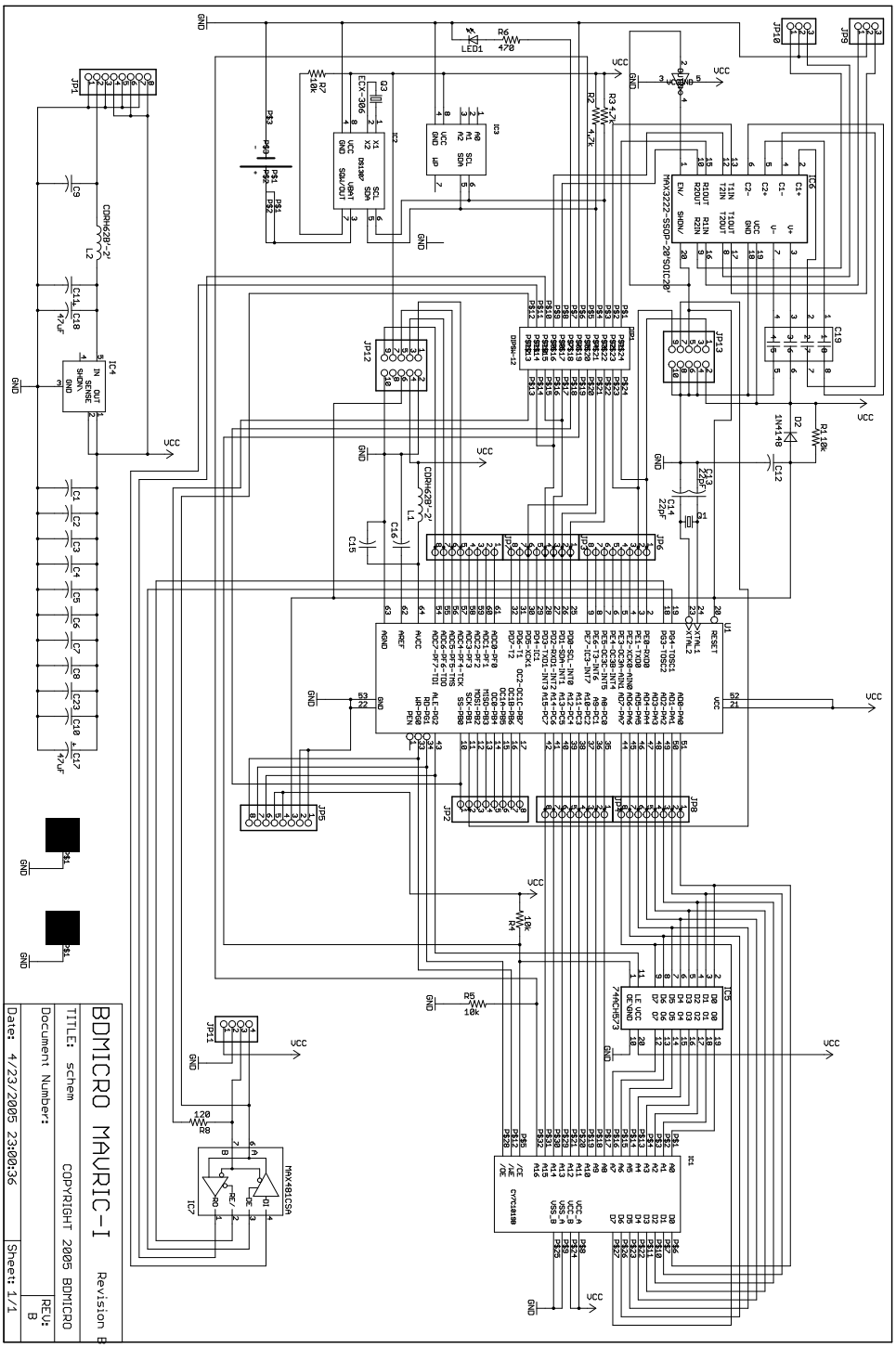
If an assembled and tested board was ordered, all these fuse bits are already set appropriately to run at 16 MHz, and the M103C fuse bit has been unprogrammed.

One other note with regard to fuse bits — the JTAG interface uses the upper nibble of PORT F, and the JTAGEN fuse bit is programmed by default. Since the JTAG interface supersedes all other functions of the PORT F lines, their other functions are not available while the JTAGEN fuse bit is programmed. Thus, if one are not using a JTAG programmer/debugger, you may wish to unprogram the JTAGEN fuse bit. We leave the JTAGEN fuse bit in its default programmed state since the recipient of the board may only have a JTAG programmer/debugger. If JTAGEN was

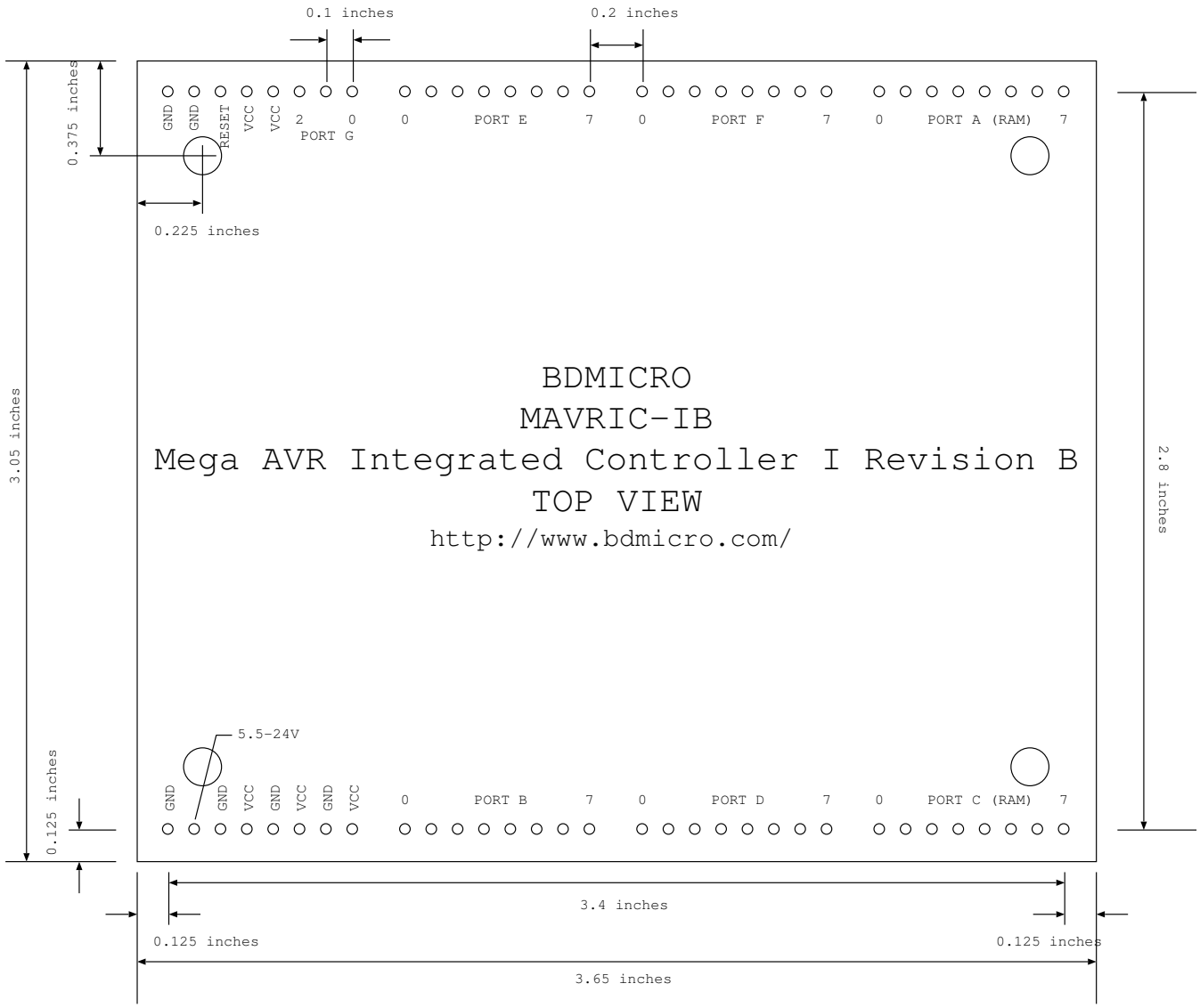
unprogrammed, those with JTAG programmers would have to find another type of programmer (serial or parallel) in order to change the state of the JTAGEN bit, just so they could begin working on the board.

For a description of all the fuse bits, see pages 286 and 287 of the ATmega128 data sheet available from Atmel's web site (`http://www.atmel.com/`).

12 Schematic Diagram



13 Mechanical Drawing



14 Soldering the MAVRIC-IB Board

If a bare board or a kit with a bare board was purchased, it will be necessary to solder the components on the board. A small diameter solder such as 0.015 inches works well for the small surface mount parts. A temperature controlled iron also makes the job easier as well as a 3 to 5x magnifying visor. A short tutorial with some photos is available on the web site at `http://www.bdmicro.com/smt/`. The main points to remember are:

- Use solder sparingly, it takes only a tiny amount to make a good connection.
- Don't use the soldering iron directly to melt the solder. Doing so causes the solder to bead up due to surface tension on the iron and then release in a big drop across the pins. This results in too much solder and causes bridging, shorting adjacent pins together.

  Instead, use the iron to heat the pin, which then melts the pad underneath the pin. At this point, touch the 0.015 inch solder to the base of the pin. It should melt very quickly, consuming only one or two millimeters of solder, and flow onto the pin and around the pad. Remove the solder quickly.
- Make sure the iron is hot — about 630 degrees F seems to be about right. It needs to be hot enough to melt the solder indirectly, but not so hot that it damages the part.
- While not absolutely required, a good temperature-controlled soldering iron can make the process go much easier than with the standard soldering pencil type irons.

The order in which the parts are soldered can make the soldering easier. In general, solder the parts based on size with the smaller and lower profile parts first followed by successively larger parts. Following this guideline, install the surface mount capacitors, resistors, and diodes first, followed by the ICs, the larger surface mount caps and inductor, crystals, and finally the pin headers.

A good technique that for soldering on even the tiniest of surface mount parts is this simple procedure, especially for tiny resistors and capacitors:

- First, prepare one pad by heating it with the tip of the soldering iron and flowing a bit of solder onto the pad. This will make a raised solder surface. Only do one pad at this point, not both.
- Use a pair of fine tweezers to move the part into position.
- Use the tip of ones fingernail to hold the part in place, then re-apply heat to the raised solder area. The solder will liquify and the part will sink into position and be held firmly in place by the solder.
- Now move to the unsoldered side and apply a bit of solder to complete the connection.
- Finally, if necessary, return to the first side of the part, and re-apply a tiny bit of solder to ensure a good connection there.

This technique is described on the web site with photos for clarification. See `http://www.bdmicro.com/smt/`.

One can save time by first preparing one pad of each part by applying solder, then attach one side of all the parts, then solder the other side of all the parts, etc, instead of performing all the steps for a part before moving to the next one.