MT-DB-U4 User Guide

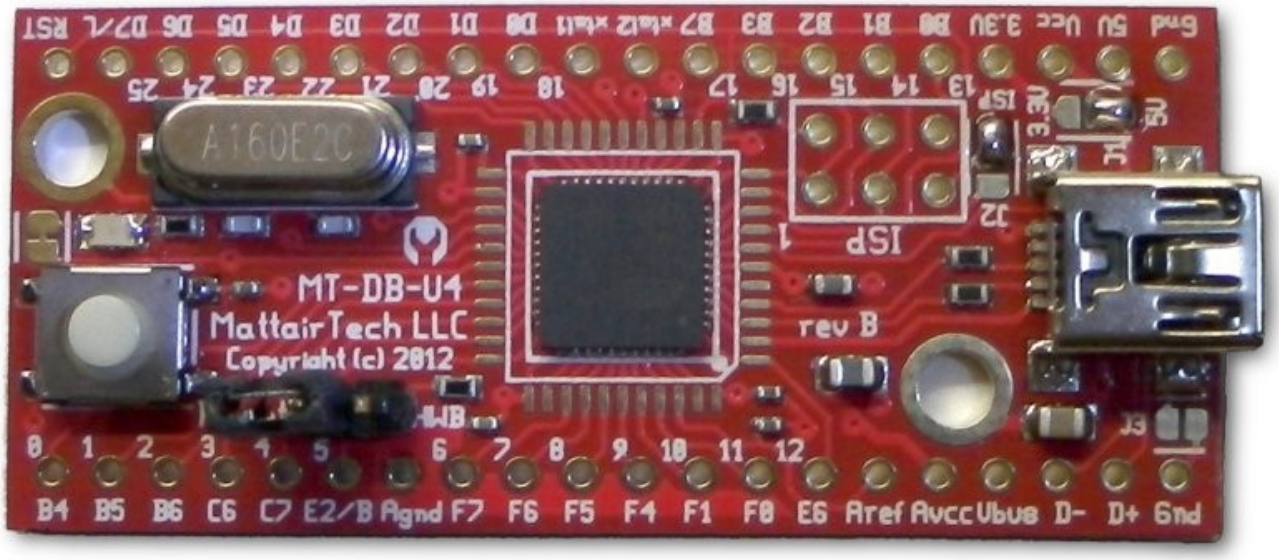


Table of Contents

Overview

Introduction

The MT-DB-U4 is a development board for the Atmel ATmega32U4 USB microcontroller. The board has 40 pins in a dual inline configuration with 100 mil pin spacing and 700 mil row spacing which allows for easy mounting on a breadboard. It includes a mini USB connector, status LED, 16MHz crystal, reset button, HWB boot jumper, and ISP header pads. A bootloader comes preinstalled which allows programming of the chip over USB without an external programmer. An ISP header is available which can be used with an external programmer. This header can be reconfigured to allow the MT-DB-U4 itself to be used as an ISP programmer (hex file available on website), or to be used as a SPI master or slave. The board can be powered via USB at 5V or it can be externally powered (3V - 5.5V). All pins are routed to headers, including those used by on-board hardware. The chip can be clocked externally, and the board is compatible with HV programming. The USB connections are also routed to header pins, which allows for panel-mount USB connectors. The PCB is high-quality with ENIG (gold-plated) finish, red soldermask, and white screenprinting showing the pinout and Arduino pin numbering. There are two 3mm mounting holes (~5mm pad). It measures approximately 2.1” x 0.9” (52mm x 23mm) and is 0.062” (1.6mm) thick.

MT-DB-U4 Features

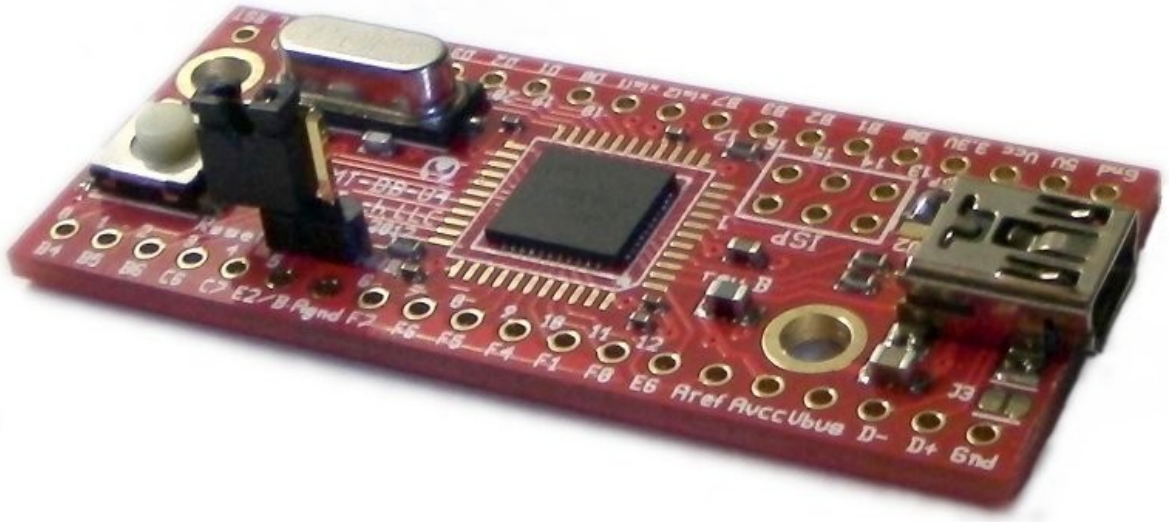
- ATmega32U4 USB microcontroller
    - 32KB FLASH, 2.5KB SRAM, 1KB EEPROM
    - 12 10-bit ADC channels (1 used by LED which can be disconnected)
    - Serial USART, SPI, and TWI (I2C) communications
    - 4 timers with 14 PWM channels (up to 7 simultaneous)
- Arduino compatible (Arduino pin numbering printed on pcb)
- CDC (Arduino/AVRDUDE) or DFU (FLIP) bootloader preinstalled
- Bitlash preinstalled (Arduino command shell)
- ISP header (program chip using external programmer)
- 16MHz crystal
- Green Status LED
- Reset button
- Bootloader jumper
- Mini USB connector
- Powered by USB or external power supply
    - 5V (USB) or 3V - 5.5V (external)
- All pins routed to headers (including those used by on-board hardware)
- Can be mounted on a breadboard
- USB pins routed to header pins (for panel-mount USB connector)
- Inductor on analog supply with separate ground pin
- High-quality PCB with gold-plated finish
- Two 3mm mounting holes (~5mm pad)
- Measures approx. 2.1” x 0.9” (52mm x 23mm) and 0.062” (1.6mm) thick.

ATmega32U4 Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  – 135 Powerful Instructions – Most Single Clock Cycle Execution
  – 32 x 8 General Purpose Working Registers
  – Fully Static Operation
  – Up to 16 MIPS Throughput at 16 MHz
  – On-Chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  – 32K Bytes of In-System Self-Programmable Flash
  – 2.5K Bytes Internal SRAM
  – 1K Bytes Internal EEPROM
  – Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  – Data retention: 20 years at 85°C/ 100 years at 25°C
  – Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
    All supplied parts are preprogrammed with a default USB bootloader
  – Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 compliant) Interface
  – Boundary-scan Capabilities According to the JTAG Standard
  – Extensive On-chip Debug Support
  – Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- USB 2.0 Full-speed/Low Speed Device Module with Interrupt on Transfer Completion
  – Complies fully with Universal Serial Bus Specification Rev 2.0
  – Supports data transfer rates up to 12 Mbit/s and 1.5 Mbit/s
  – Endpoint 0 for Control Transfers: up to 64-bytes
  – 6 Programmable Endpoints with IN or Out Directions and with Bulk, Interrupt or Isochronous Transfers
  – Configurable Endpoints size up to 256 bytes in double bank mode
  – Fully independent 832 bytes USB DPRAM for endpoint memory allocation
  – Suspend/Resume Interrupts
  – CPU Reset possible on USB Bus Reset detection
  – 48 MHz from PLL for Full-speed Bus Operation
  – USB Bus Connection/Disconnection on Microcontroller Request
  – Crystal-less operation for Low Speed mode
- Peripheral Features
  – On-chip PLL for USB and High Speed Timer: 32 up to 96 MHz operation
  – One 8-bit Timer/Counter with Separate Prescaler and Compare Mode
  – Two 16-bit Timer/Counter with Separate Prescaler, Compare- and Capture Mode
  – One 10-bit High-Speed Timer/Counter with PLL (64 MHz) and Compare Mode
  – Four 8-bit PWM Channels
  – Four PWM Channels with Programmable Resolution from 2 to 16 Bits
  – Six PWM Channels for High Speed Operation, with Programmable Resolution from 2 to 11 Bits
  – Output Compare Modulator
  – 12-channels, 10-bit ADC (features Differential Channels with Programmable Gain)
  – Programmable Serial USART with Hardware Flow Control
  – Master/Slave SPI Serial Interface
  – Byte Oriented 2-wire Serial Interface
  – Programmable Watchdog Timer with Separate On-chip Oscillator
  – On-chip Analog Comparator
  – Interrupt and Wake-up on Pin Change
  – On-chip Temperature Sensor
- Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection
– Internal 8 MHz Calibrated Oscillator
– Internal clock prescaler & On-the-fly Clock Switching (Int RC / Ext Osc)
– External and Internal Interrupt Sources
– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby

- I/O and Packages
  – All I/O combine CMOS outputs and LVTTL inputs
  – 26 Programmable I/O Lines
  – 44-lead TQFP Package, 10x10mm
  – 44-lead QFN Package, 7x7mm
- Operating Voltages
  – 2.7 - 5.5V
- Operating temperature
  – Industrial (-40°C to +85°C)
- Maximum Frequency
  – 8 MHz at 2.7V - Industrial range
  – 16 MHz at 4.5V - Industrial range

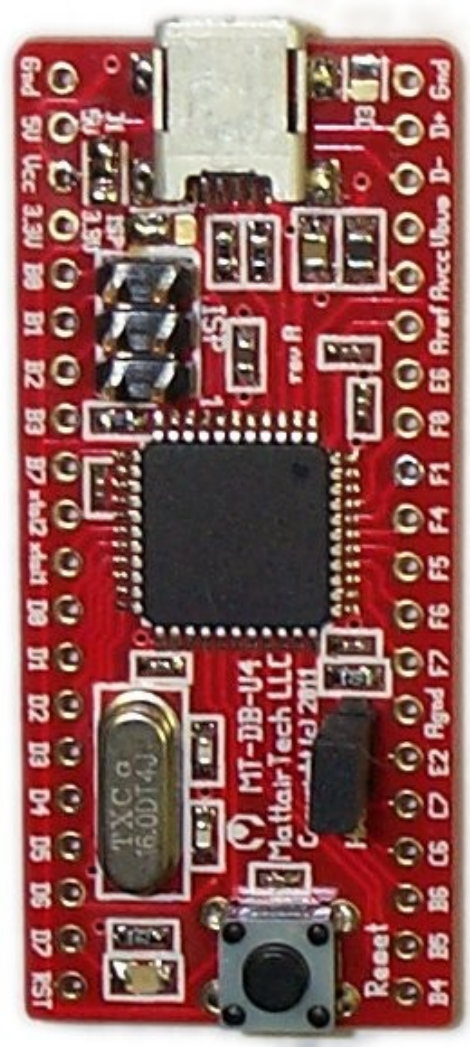


MT-DB-U4 Hardware

Board Revisions

There are two board revisions, A and B. Boards shipped after November 22, 2012 are revision B. Revision B adds two 3mm (~5mm pad) mounting holes, adds a solder jumper to allow disconnection of the LED from pin D7, adds solder jumpers to allow connection of the xtal traces to two main header pins, adds Arduino pin numbering to the screen printing, and makes other various changes to improve the manufacturing process.

Revision A



Revision B (current)

Pin Descriptions

Pin	Description
Gnd	Digital ground
5V, Vbus	5V output from USB Vbus. Vbus pin is tied to 5V pin. These pins are connected to the Vbus and UVcc pins of the microcontroller.
Vcc	Voltage input pin. Use solder jumper J1 to connect this pin to 5V (default setting) when using USB power. In this case, Vcc is an output. Leave J1 unconnected to supply power from an external source to the Vcc pin. This pin is connected to the Vcc and AVcc pins on the microcontroller, as well as the ISP header and reset pullup. See Power Configuration Section.
3.3V	3.3V output from the microcontrollers internal 3.3V regulator. This pin is connected to Ucap on the microcontroller.
Avcc	Voltage input to the analog section of the microcontroller. This pin is connected to Vcc through a 10uH inductor and 100nF capacitor and provides power to the microcontroller analog section (Avcc pin).
Aref	Voltage input. This is the reference voltage used by the ADC in the microcontroller. **DO NOT** connect if using an internal reference.
Agnd	Analog ground
RST	Connects to reset pin of microcontroller as well as the reset button. A 10K pullup resistor and 100nF capacitor are connected to this pin.
E2 / B (Boot)	This pin is connected to the HWB jumper. The jumper is connected to ground through a 240 ohm resistor. The pin is sampled after all reset sources, including power-up. If the pin is low (HWB jumper installed), then the bootloader is run. If the pin is high (HWB jumper removed), then the user application is run. This pin functions as a normal GPIO pin at all other times. The 240 ohm resistor provides short-circuit protection in case the pin is used as an output and the jumper is installed.
D7 / L (LED)	The green status LED is connected to this pin. The LED is connected to ground through a 240 ohm resistor. Drive the pin high to turn on the LED. It can be disconnected by cutting the solder jumper trace (Rev B only).
xtal1	This pin is connected to the on-board 16MHz crystal. If using an external clock, connect it to this pin, connect the solder jumper (Rev B only), and configure the microcontroller fuses to use an external clock. This is also useful for HV programming or recovery from incorrectly set fuses.
xtal2	This pin is connected to the on-board 16MHz crystal. This pin is useful along with xtal1 to connect an external crystal. Board revision B has a solder jumper to connect this to the external header pin.
E6, F0, F1, F4, F5, F6, F7	These are generally used as analog pins. Analog ground plane runs under these pins. Consult datasheet for functionality.
All other pins	Consult datasheet for functionality.

Boot Jumper / RESET button / LED

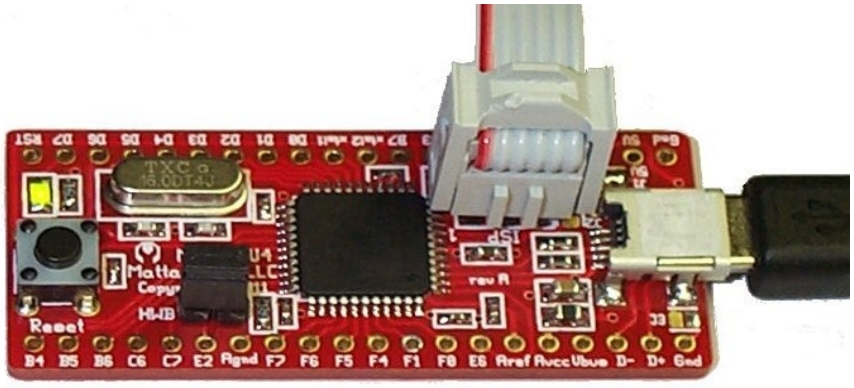
This section does not apply to the Atmel DFU bootloader option, in which case, the fuses will be left at their factory default settings and the factory installed Atmel DFU bootloader will not be overwritten.

The boot jumper (labeled HWB) selects between the bootloader and user application. The pin is sampled after reset or power-up. Note that the hardware HWB function of the ATmega32U4 is disabled (HWBE fuse is disabled) and the bootloader startup code is always run after reset or power-up (BOOTRST fuse is set). This startup code then samples the state of the HWB pin. If it is high, the user application runs. Otherwise, the bootloader continues to run, waiting for programming instructions while pulsing the LED. The LED remains on when jumping to the user application. The bootloader always runs at 8 MHz, which is compatible with lower voltages. It remains at 8MHz when jumping to the user application. The user may then set the cpu speed to 16MHz in software, if running at 5V.

It is not necessary to remove and replace the jumper when switching between the bootloader and the user application. The jumper can be left on. After FLASH programming, the CDC bootloader will automatically jump to the application. If using the DFU bootloader, then you can command FLIP or dfu-programmer to jump to the application. Then, when running the application, the reset button can be pressed to re-enter the bootloader. This is useful when writing and debugging firmware. When the firmware is complete, the jumper can be removed so that future resets will always run the application.

ISP Header

The ISP header is configured by default to allow ISP programming using an external programmer. That is, RESET is routed to pin 5. Pin 1 is marked on the board (it is the pin closest to the chip). The ISP header can be reconfigured so that pin PB0 (SS) is connected to pin 5 rather than RESET. This can be done by switching solder jumper J2, next to the ISP header, to the position opposite to the “ISP” label. This allows the MT-DB-U4 to be used as an AVRISPmkII programmer itself, using Dean Camera's AVRISPmkII software available at http://www.fourwalledcubicle.com/. A precompiled hex file will be made available at http://www.mattairtech.com/ on the MT-DB-U2 product page. Note that when using the ISP header in this way, Vcc and ground are output to the target board. Therefore, the target board should not be powered itself. You should also verify that it is safe to power the target board through the ISP connector. Another use for the ISP header configured with SS on pin 5 is to make use of SPI, either as a master or slave. SPI can also be used on the normal DIL headers.



JTAG

JTAG can be used for programming and debugging. While there is no JTAG header, all JTAG signals are available on the DIL header pins. Four JTAG signals are shared with ADC pins (F4-F7). JTAG is enabled while running the bootloader only. It is disabled when the user application is run to allow access to the ADC pins. It can be re-enabled in software. When using the Atmel DFU bootloader option, JTAG will always be enabled. It will then have to be disabled in software or by fuse setting to enable access to the four ADC channels.

Mounting Holes

There are two grounded mounting holes. They have a hole diameter of 3mm and the pad is 4.8mm. Each one is located 6.2mm from the closest long edge of the board. The hole nearest the USB connector is 11.4mm from the closest short edge. The hole furthest from the USB connector is 2.7mm from the closest short edge.

Power Configuration

Bus Powered - 5V

By default, the MT-DB-U4 is configured for 5V from the USB connector (Vbus). In this configuration, solder jumper J1 is set to the 5V position. This shorts 5V (Vbus) to Vcc. Thus, the 5V and Vcc pins are both outputs. The 3.3V pin is also an output from the AVR internal regulator, which must be enabled. This pin can supply about 55mA.

Externally Powered – 3.4V to 5.5V

In this configuration, disconnect solder jumper J1 (not set to 5V or 3.3V). Then supply 3.4V to 5.5V to the Vcc pin, which is now an input. The 5V pin still outputs 5V when the USB cable is plugged in. The 3.3V pin is also an output from the AVR internal regulator, which must be enabled. This pin can supply about 55mA. Note that when using a voltage less than 4.5V, the AVR should be set to run at 8MHz or less. This can be done in software using the prescaler (no need to change the crystal).

Externally Powered – 3.0V to 3.6V

In this configuration, change your code to disable the internal 3.3V regulator. Disconnect solder jumper J1 (not set to 5V or 3.3V). Then supply 3.0V to 3.6V to both the Vcc pin and the 3.3V pin, which are now both inputs. Alternatively, the solder jumper J1 can be set to the 3.3V position so that only the Vcc pin need be connected. Note that unlike the AT90USBXX2 or ATmegaXXU2, the internal 3.3V regulator cannot be used to power Vcc because the regulator is disabled on reset or power-up. Also note that if the bootloader is set to run (HWB jumper installed) the regulator will be enabled. The regulator is then disabled before jumping to the user application. The regulator is never enabled if the application is configured to run (HWB jumper not installed). Therefore, if the bootloader is to be used in this configuration, only 3.3V should be connected to Vcc. The 5V pin still outputs 5V when the USB cable is plugged in. In this configuration, the AVR should be set to run at 8MHz or less. This can be done in software using the prescaler (no need to change the crystal).

USB Shield

Jumper J3 can be soldered to connect the USB shield to ground. The USB specification calls for the USB shield to be connected to ground on the host side only. However, some prefer to have it grounded on the device side as well, though a ground loop would be formed. An 0603 SMT component may be soldered on the solder jumper pads as well.

Arduino Compatibility

Features

- Arduino core and libraries ported to MattairTech USB boards
- 26 digital, 11 analog, 7 PWM, 5 INT, 8 PCINT, TWI, SPI
- 32KB FLASH, 2.5KB SRAM, 1KB EEPROM
- USB Serial interface replaces USART0 (USART1 also available; can be used at the same time)
  - Hardware maximum speed of 8Mbps
  - Can use terminal emulator or serial monitor
  - LED blinks when data is transferred (can be disabled)
  - Uses the same methods as the original HardwareSerial.cpp (ie: Serial.println("Hello!"))
- Based on LUFA USB library by Dean Camera (www.fourwalledcubicle.com)
- Arduino/AVRDUDE compatible CDC bootloader preinstalled (DFU bootloader also available)
- Bitlash Arduino command interpreter (bitlash.net) preinstalled
- All libraries included with Arduino download are now supported
- Bootloader automatically jumps to sketch after upload, reset button returns to bootloader
- Arduino pin numbering printed on PCB (0-25)

Pin Mapping

MattairTech MT-U4 (ATmega32U4)

INT/Other	PWM	Digital	Analog*	AVR Pin		AVR Pin	Analog*	Digital	PWM	INT/other	Comm
PCINT4		D0	A0 (ADC11)	B4		RST					
PCINT5	TIMER1A	D1	A1 (ADC12)	B5		D7	A25 (ADC10)	D25	TIMER4D	LED	
PCINT6	TIMER1B	D2	A2 (ADC13)	B6		D6	A24 (ADC9)	D24			
	TIMER3A	D3	A3 (GND)	C6		D5	A23 (REF)	D23			
	TIMER4A	D4	A4 (GND)	C7		D4	A22 (ADC8)	D22			
JUMPER		D5	A5 (GND)	E2		D3	A21 (GND)	D21		INT3	USART1 TX
				Agnd		D2	A20 (GND)	D20		INT2	USART1 RX
		D6	A6 (ADC7)	F7		D1	A19 (GND)	D19		INT1	TWI SDA
		D7	A7 (ADC6)	F6		D0	A18 (GND)	D18	TIMER0B	INT0	TWI SCL
		D8	A8 (ADC5)	F5		xtal1					
		D9	A9 (ADC4)	F4		xtal2					
		D10	A10(ADC1)	F1		B7	A17 (GND)	D17	TIMER1C	PCINT7	
		D11	A11(ADC0)	F0		B3	A16 (GND)	D16		PCINT3	SPI MISO
INT6		D12	A12(TEMP)	E6		B2	A15 (GND)	D15		PCINT2	SPI MOSI
				Aref		B1	A14 (GND)	D14		PCINT1	SPI SCLK
				Avcc		B0	A13 (GND)	D13		PCINT0	SPI SS
				Vbus		3.3V					
				D-		Vcc					
				D+		5V					
				Gnd		Gnd					

* Because of the unusual layout of the ATmega32U4, all pins can be used with analogRead(). 12 of these pins are actual analog inputs (1 used by LED), the rest connect to the internal reference, internal temperature sensor, or ground.

Installation

1. Download and install Arduino version 1.0.1 from http://arduino.cc/en/Main/Software
2. Download the MattairTech_Arduino_1.0.1.zip file from http://www.mattairtech.com/ (see product page). Unzip this file into your arduino user directory (ie: My Documents/Arduino). You may need to create this folder. Do not unzip into the arduino system directory from step 1. If you installed a previous version of the MattairTech Arduino port, move or remove it. It is OK if there are other cores, libraries, or sketches already present.
3. Now, plug in the board with the jumper installed so that the bootloader runs. Point the driver installer to the directory <arduino user directory>/hardware/MattairTech/install to install MattairTech_CDC.inf. This same driver is also used for the USB serial interface (if used).

Using Arduino

Within the Arduino IDE, select the appropriate MattairTech board and COM port. There are 2 configurations for each board, 16MHz(5V) and 8MHz(3.3V). You may select 8MHz even if using 5V. When operating at 3.3V, you should select 8MHz. Operating at 16MHz at 3.3V is out of spec, but should work fine at room temperatures. When using the USB serial interface, it is no longer necessary to include the LUFA USB library header file. Then, with the bootloader running, compile and upload your sketch. Note that when using a terminal emulator to communicate with the board, be sure to disconnect before switching to the bootloader. If the jumper is left on, the reset button can be used to switch from the sketch to the bootloader. When the sketch has finished uploading, it will run automatically. If you installed the bootloader yourself, be sure that the BOOTRST fuse is set.

Using Libraries

There are several libraries included with Arduino. Some of these needed to be ported to work with MattairTech boards. If a library was ported, it is included in the MattairTech download and installed in the Arduino user directory with "_MattairTech_Port" appended to the name of the original directory name. This can be seen in the Arduino IDE in File->Sketchbook->libraries and File->Examples. If you see the Files->Examples version of a particular library then you must use it instead of the original library which will still be shown lower on the menu. If there is no Files->Examples version, then you must use the original, which did not require porting. If there is a library you would like to use that is not included with Arduino, email support and I should be able to quickly support it. Often, only pin mappings need to be changed. The I2cMaster library contains a software I2C library that can be used with the MT-DB-U1 and MT-DB-U2, which do not contain I2C hardware.

USB Serial interface

The LUFA directory contains a reorganized subset of the LUFA USB library by Dean Camera (fourwalledcubicle.com). It implements a CDC class device, which appears as a COM port on the host computer. A terminal emulator or the Arduino serial monitor can be used to communicate with the board. Use this interface the same way you would on a standard Arduino (ie: Serial.println()). The interface is nearly the same as the one in HardwareSerial.cpp. For example:

```
void setup() {
  Serial.begin(9600); // The default settings for USB options are used (all enabled)
  pinMode(2, INPUT);
}

void loop() {
  int sensorValue = digitalRead(2);
  Serial.println(sensorValue, DEC);
  //Serial.flush();       // needed if autoflush is not used
  delay(1000);
}
```

Serial.begin() sets up the USB serial interface with a single 32-bit argument. This value is setup by ORing three USB options together along with the optional baud rate. This works because the three options are stored in the upper bits of the 32-bit value. The baud rate is ignored because the fastest speed supported is always used (2Mbps for the AT90USB162 and ATmega32U2, 8Mbps for the ATmega32U4). For example:

```
Serial.begin(9600 | USB_LED_ENABLED | USB_WAITFORCONNECT_ENABLED | USB_AUTOFLUSH_ENABLED);
```

USB_LED_ENABLED, USB_LED_DISABLED

If USB_LED_ENABLED is set, then the LED will display the state of the USB connection (on whenconnected) as well as blink when data is transferred. Otherwise, the LED will be left on and you can manually control it. The default setting is USB_LED_ENABLED.

USB_AUTOFLUSH_ENABLED, USB_AUTOFLUSH_DISABLED

If USB_AUTOFLUSH_ENABLED is set, the upstream buffer (to the PC) will be flushed at periodic intervals. The hardware USB DPRAM is used for the RX and TX buffers. There are actually two buffers per direction in a ping-pong configuration. As one buffer fills up, it is swapped with the other, allowing the USB hardware to read from the filled one, and the user to write to the empty one. Any number of characters can be sent to the upstream buffer without any need to manage it, but it must be flushed at the end of the transmission if USB_AUTOFLUSH_ENABLED is not set. In this case, use Serial.flush(). The default setting is USB_AUTOFLUSH_ENABLED.

USB_WAITFORCONNECT_ENABLED, USB_WAITFORCONNECT_DISABLED

If USB_WAITFORCONNECT_ENABLED is set, Serial.begin() will wait for the host to open a connection before returning. That is, a program like a terminal emulator or serial monitor must connect to the COM port before continuing. This is different than the USB CDC connection. This is useful to prevent the board from sending data before the host is ready, and is required in many cases for microcontrollers with onboard USB (ie: Leonardo). The wait is performed using: while(!Serial);

Updated Tone.cpp

Tone.cpp now supports multiple simultaneous tone generation (one tone per timer). The MT-DB-U4 currently supports up to 3 simultaneous tones using timers 3, 1, and 0. A future release may support a fourth tone from timer 4. The MT-DB-U2 and MT-DB-U1 support 2 simultaneous tones using timers 1 and 0. Note that timer 0 has a lower accuracy for tone generation because it is 8-bit (timers 3 and 1 are 16-bit). Note also that use of timer 0 temporarily disables the use of delay(), USB autoflushing, and proper USB LED handling, all of which will return to normal operation once the tone stops playing. Thus, timer 0 is set with the lowest priority. For example, if generating DTMF tones on the MT-DB-U4, timers 3 and 1 will be used. However, the MT-DB-U2 and MT-DB-U1 will both use timer 0 for the second tone.

If timer 0 is used, delay() should not be called while timer 0 is generating a tone. Instead, use _delay_ms(), which is included with avr-libc. If sending data to the USB host (ie: using Serial.print()) before or during timer 0 tone generation, then it must be manually flushed with Serial.flush() prior to calling tone() and after any subsequent printing during tone generation. Otherwise, some data may not be sent until the tone stops and autoflushing returns to normal operation. The USB LED handling (if enabled) will also be disrupted during timer 0 tone generation. During this time, the LED will not be able to change state. If USB traffic occurs, the blink will be delayed until tone generation stops.

The DTMF_Demo sketch demonstrates usage of Tone.cpp for DTMF generation on the MT-DB-U4.

CDC Bootloader (Arduino/AVRDUDE)

CDC Serial Driver

The CDC Serial driver allows the board to appear as a COM port. The driver itself is included with Windows, but an .inf file is needed to configure it. Download MattairTech_CDC.inf from the MT-DB-U4 page at http://www.mattairtech.com/. You may need to rename the file so that it has the inf extension. Next, plug in the board with the jumper removed. Windows will then prompt you for the MattairTech CDC Serial driver. Point the installer to the directory where you downloaded the driver and install, ignoring any warnings. Once the driver is loaded, the device will appear as the MattairTech CDC Serial device using a COM port in the device manager.

If you wish, double-click on the CDC Serial device entry in the device manager to configure the driver. Nothing on the port settings tab needs to be changed. We are using a virtual COM port so the settings are ignored. The baud rate will always be as fast as possible. On the advanced tab, you can adjust the FIFO buffer sizes. If you experience any buffering problems (ie: a delayed response to user input), then change both buffer sizes to 1.

CDC Bootloader

The CDC bootloader uses the AVR109 protocol, and can be used withing the Arduino environment, or directly with AVRDUDE. If you did not install Arduino for MattairTech USB boards, then you must replace the default avrdude.conf file with the one provided in the arduino_MT-Ux.zip file from http://www.mattairtech.com/ (see Arduino installation). The bootloader will jump to the user application at the end of FLASH programming. Other operations with AVRDUDE, like writing the EEPROM, will not trigger this. Just press reset to get back to the bootloader (as long as the jumper is installed). If using a terminal emulator, you must first disconnect before running the bootloader. Note that if the user application enables the watchdog timer, then the bootloader will not run when reset is pressed (the user application will). In this case, only a power-on reset will enter the bootloader.

Example for Windows:

```
avrdude -p m32u4 -c avr109 -P COM5 -U flash:w:"bitlashdemo_MT-DB-U4.hex"
```

Example for Linux:

```
avrdude -p m32u4 -c avr109 -P /dev/ttyACM0 -U flash:w:"bitlashdemo_MT-DB-U4.hex"
```

Arduino environment:

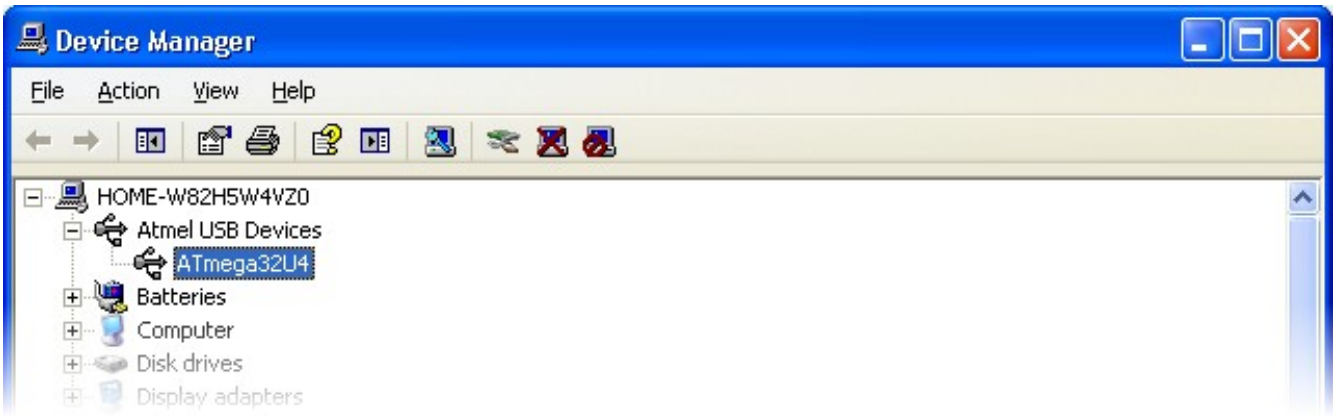
Be sure to select the COM port. Then upload your sketch with the Upload button.

DFU Bootloader (FLIP/dfu-programmer)

Installation

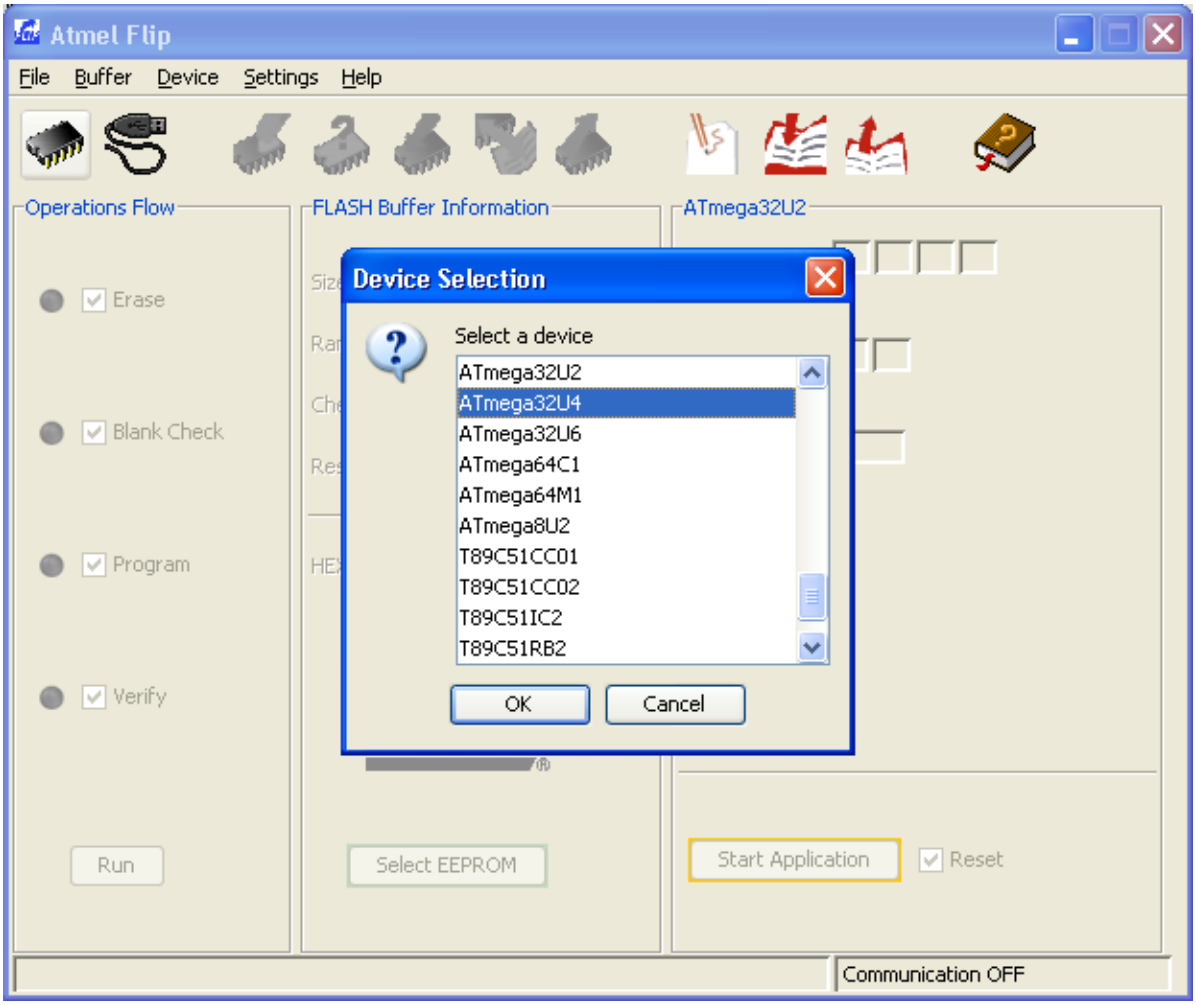
FLIP is a graphical utility used to load firmware into the ATmega32U2. FLIP includes the DFU bootloader driver. It supports Windows XP through Windows 7 (32 or 64 bit). Download FLIP 3.4.2 or higher from http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3886 and install.

Once FLIP is installed, the DFU bootloader drivers can be loaded. Install the HWB jumper and power-up the board (or press reset). This will enter the DFU bootloader. The LED should be pulsing. Windows will then prompt you for the ATmega32U2 driver. By default, this is located in the Program Files/Atmel/Flip 3.4.2/usb directory. Point the installer to that directory and install. If required to install signed drivers and the Atmel drivers do not work, download the signed drivers at http://www.avrfreaks.net/index.php?module=Freaks%20Academy&func=viewItem&item_type=project&item_id=2196 and install. Once the driver is loaded, the device will appear as the ATmega32U2 device under Atmel USB Devices in the device manager.

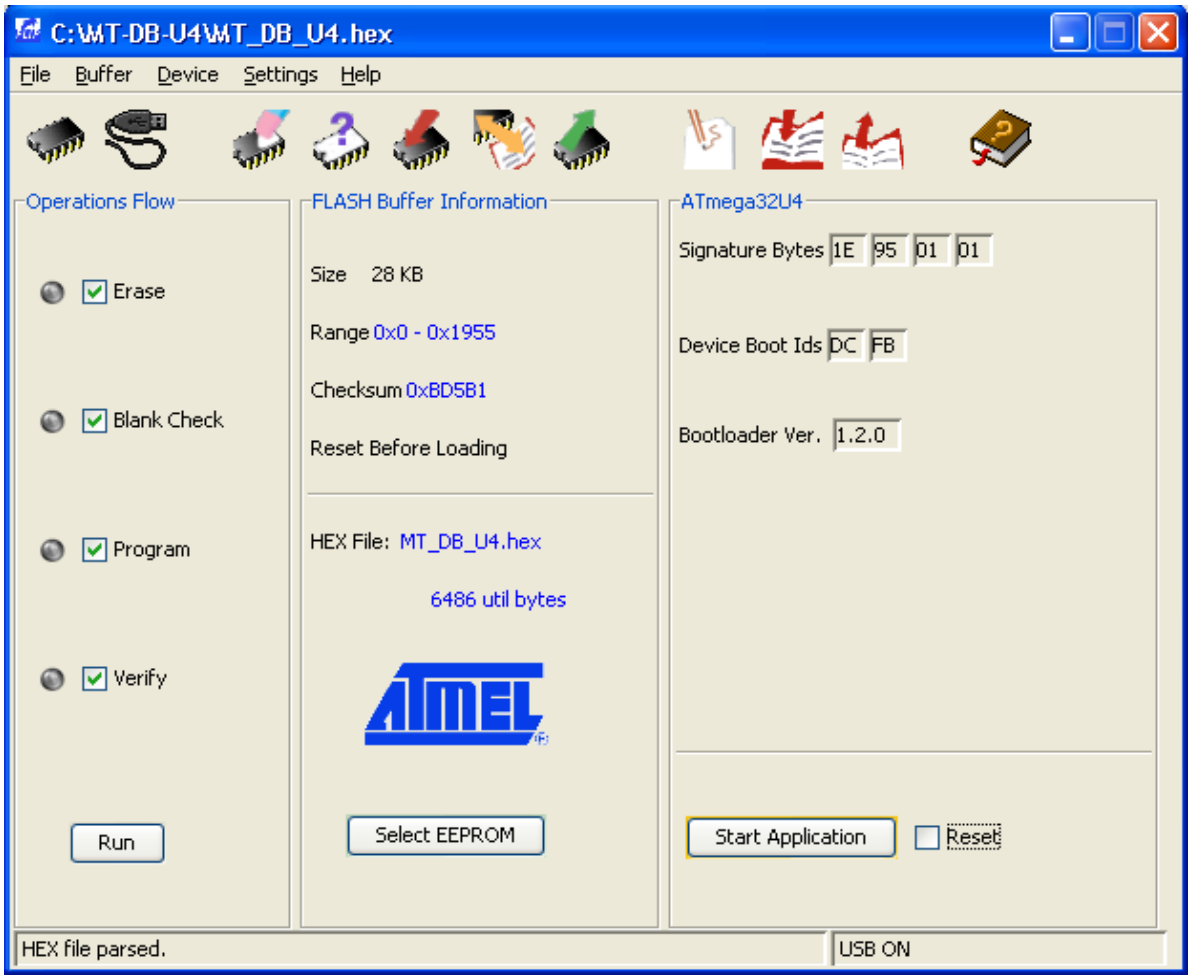


FLIP

Install the HWB jumper and power-up the board (or press reset). This will enter the DFU bootloader. The LED should be pulsing (unless using the Atmel bootloader). Now launch the FLIP utility. When it has loaded, click on the chip icon and select the Atmega32U4.



Next, click on the USB icon, select USB, then connect. The screen should now show information about the ATmega32U4. Click on the File menu, and open the appropriate hex file. More information will appear about the program. Be sure that erase is checked. The firmware cannot be loaded unless the flash is erased first. Program must be checked. Verify should also be checked. Now click on the Run button in the lower-left of the screen, and the firmware will be quickly loaded onto the ATmega32U4.



You may also program the EEPROM. If so, click on Select EEPROM at the bottom. Then, click on the File menu and open the appropriate eep file. You will have to change the file filter to allow you to see the eep file. Note that eep files are just hex files but with the eep extension instead of hex. More information will appear about the file when selected. Both Program and Verify should be checked. Click run to program the EEPROM.

You can run your application without removing the jumper or pressing reset by unchecking the reset box and pressing the “Start Application” button (lower right).

dfu-programmer

dfu-programmer is a command line utility used to program the ATmega32U4 that runs under Linux. A DFU driver installation is not required. Download version 0.5.4 or higher from http://dfu-programmer.sourceforge.net/ . The following commands can be used:

```
dfu-programmer atmega32u4 erase
dfu-programmer atmega32u4 flash-eeprom YourHex.eep (if applicable)
dfu-programmer atmega32u4 flash YourHex.hex
dfu-programmer atmega32u4 start (to jump to application section without reset)
```

Running Bitlash Demo

Bitlash is an open source interpreted language shell and embedded programming environment. The preinstalled Bitlash demo was compiled in the Arduino environment and supports Arduino functions (ie: dw() for digitalWrite()). A terminal emulator (recommended) or the Arduino serial monitor may be used. See the CDC Bootloader section for details on installing the CDC Serial driver. The following example saves three functions to EEPROM. It is then run in the background, pulsing the LED using analog write (PWM):

```
bitlash here! v2.0RC4 (c)2011 Bill Roy, bitlash.net -type HELP- 1706 bytes free
> print free, " bytes free"
1702  bytes free
> pinMode(25,1)
> d25=1
> x=255;d=0;
> function brighter {if (x==255) {d=0;} else { a25=++x; snooze(2);}}
saved
> function dimmer {if (x==0) {d=1;} else {a25=--x; snooze(2);}}
saved
> function pulseLED {if (d==0) {dimmer();} else {brighter();}
saved
> ls
function brighter {if (x==255) {d=0;} else { a25=++x; snooze(2);}};
function dimmer {if (x==0) {d=1;} else {a25=--x; snooze(2);}};
function pulseled {if (d==0) {dimmer();} else {brighter();};
> run pulseled
> ps
0: pulseled
> stop 0
>
```

Documentation for Bitlash is available at http://bitlash.net/wiki/docindex

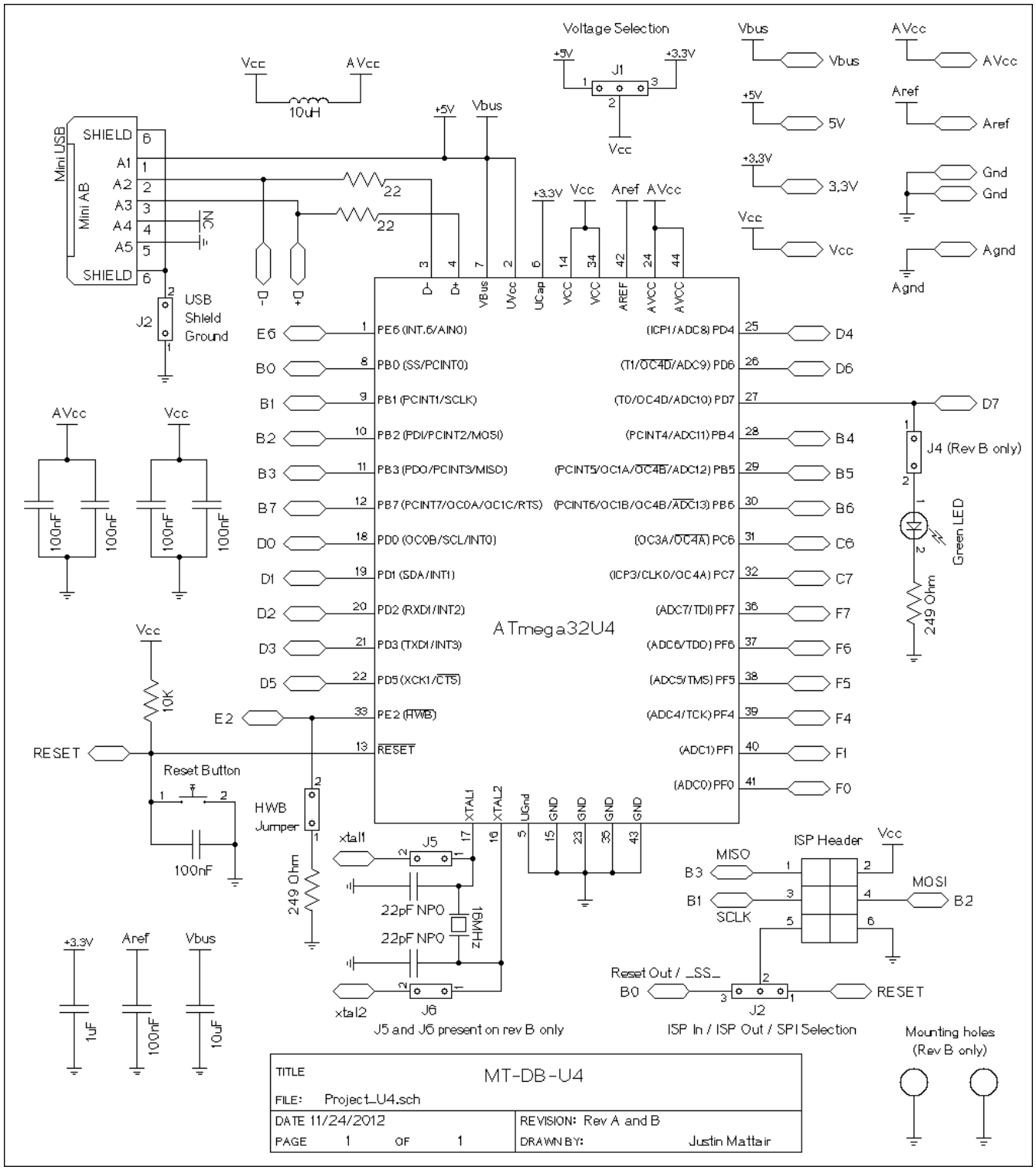
Commands	arg else function help if ls peep print ps return rm run stop switch while
Functions	abs ar aw bc beep br bs bw constrain delay dr dw er ew free inb max millis min outb pinmode printf pulsein random shiftout sign snooze

Old Demo Program

The old demo program makes use the use of the MT-DB-U4 as a CDC device (virtual COM port). This is one of the most common ways to connect to a PC over USB. It uses Dean Camera's open-source LUFA USB library available at http://www.fourwalledcubicle.com/. The LUFA download includes many examples that can be easily compiled for the ATmega32U4.

See the CDC Bootloader section for details on installing the CDC Serial driver. The old demo requires an ANSI terminal to allow drawing of the menu system. If you see garbage on the terminal screen, click on the configuration icon and change the emulation to ANSI (or ANSIW). After connecting, a message that reads “Press any Key” is printed periodically. If you do not see this message, just press any key to continue.

Schematic


Voltage Selection
Vbus
AVcc
Aref
Gnd
Agnd
5V
3.3V
Vcc
10uH
Mini USB
Mini AB
SHIELD
USB Shield Ground
22
NC
ATmega32U4
PE6 (INT.6/AIN0)
PB0 (SS/PCINT0)
PB1 (PCINT1/SCLK)
PB2 (PDI/PCINT2/MOSI)
PB3 (PDO/PCINT3/MISO)
PB7 (PCINT7/OC0A/OC1C/RTS)
PD0 (OC0B/SCL/INT0)
PD1 (SDA/INT1)
PD2 (RXD1/INT2)
PD3 (TXD1/INT3)
PD5 (XCK1/CTS)
PE2 (HWB)
RESET
(ICP1/ADC8) PD4
(T1/OC4D/ADC9) PD6
(T0/OC4D/ADC10) PD7
(PCINT4/ADC11) PB4
(PCINT5/OC1A/OC4B/ADC12) PB5
(PCINT6/OC1B/OC4B/ADC13) PB6
(OC3A/OC4A) PC6
(ICP3/CLK0/OC4A) PC7
(ADC7/TDI) PF7
(ADC6/TDO) PF6
(ADC5/TMS) PF5
(ADC4/TCK) PF4
(ADC1) PF1
(ADC0) PF0
D4
D6
D7
B4
B5
B6
C6
C7
F7
F6
F5
F4
F1
F0
J4 (Rev B only)
Green LED
249 Ohm
100nF
10K
Reset Button
HWB Jumper
xtal1
xtal2
J5
J6
22pF NPO
16MHz
J5 and J6 present on rev B only
ISP Header
MISO
MOSI
SCLK
B3
B1
B2
B0
Reset Out / _SS_
J2
ISP In / ISP Out / SPI Selection
Mounting holes (Rev B only)
1uF
10uF
TITLE
MT-DB-U4
FILE: Project_U4.sch
DATE 11/24/2012
REVISION: Rev A and B
PAGE 1 OF 1
DRAWN BY: Justin Mattair

Troubleshooting / FAQ

Nothing yet

Support Information

Please check the MattairTech website (http://www.MattairTech.com/) for firmware and software updates. Email me if you have any feature requests, suggestions, or if you have found a bug. If you need support, please contact me (email is best). You can also find support information at the MattairTech website. A support forum is planned. Support for AVRs in general can be found at AVRfreaks (http://www.avrfreaks.net/). There, I monitor the forums section as the user physicist.

Justin Mattair
MattairTech LLC
PO Box 1079
Heppner, OR 97836 USA
541-626-1531
justin@mattair.net
http://www.mattairtech.com/

Acknowledgments

Thanks to Dean Camera (http://www.fourwalledcubicle.com/) for his excellent LUFA library and bootloaders. Thanks to the members of AVRfreaks (http://www.avrfreaks.net/) for their support. Finally, thanks to Atmel for creating a great product, the AVR microcontroller.

Legal

Copyright Notices

Software Warranty Disclaimer

Hardware Disclaimer

Trademarks

Appendix A: Precautions

CAUTION
Do not change power configuration while unit is powered. Do not short 5V, Vbus, 3.3V, Avcc, or ground to each other. When connecting Aref externally, connect to a voltage source <= Vcc and be sure that the internal reference is disabled.

CAUTION
The MT-DB-U4 contains static sensitive components. Use the usual ESD procedures when handling.

CAUTION
Improper fuse settings may result in an unusable AVR. Be certain that you know the effects of changing the fuses, that you understand the convention used for describing the state of the fuses (programmed = 0), and that you are using an appropriate programming speed before attempting to change fuse settings.