MT-DB-U4 User Guide

MattairTech

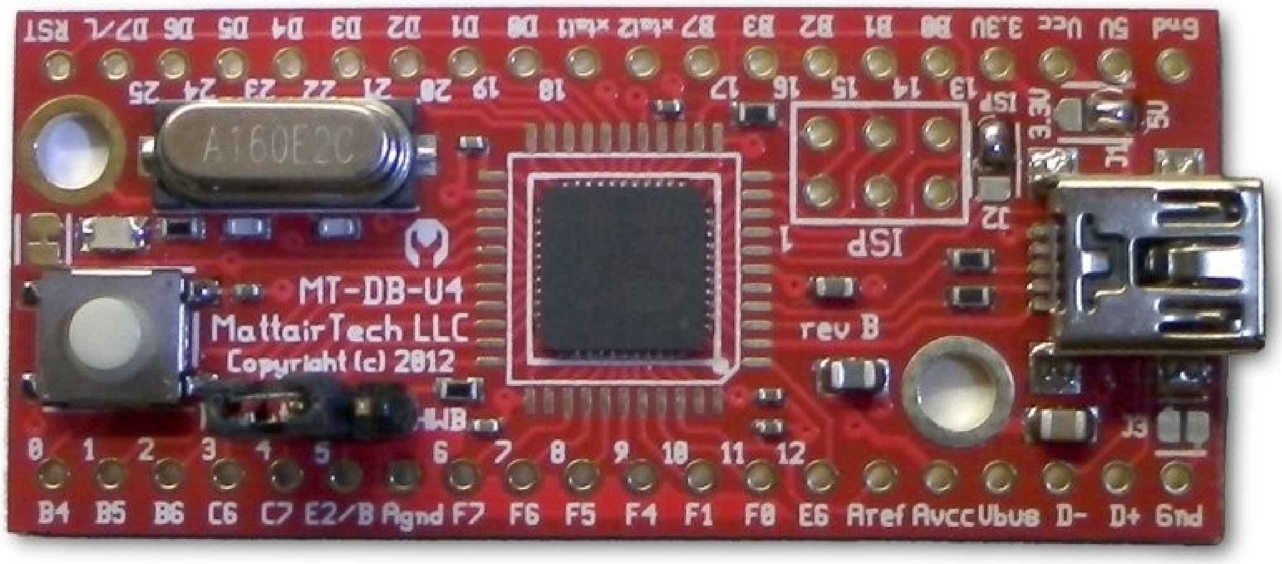

MT-DB-U4
MattairTech LLC
Copyright (c) 2012
rev B
ISP
A160E2C
RST D7/L D6 D5 D4 D3 D2 D1 D0 xtal1 xtal2 B7 B3 B2 B1 B0 3.3V Vcc 5V Gnd
25 24 23 22 21 20 19 18 17 16 15 14 13
0 1 2 3 4 5 6 7 8 9 10 11 12
B4 B5 B6 C6 C7 E2/B Agnd F7 F6 F5 F4 F1 F0 E6 Aref Avcc Vbus D- D+ Gnd
J1 J2 J3
3.3V 5V
HWB

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

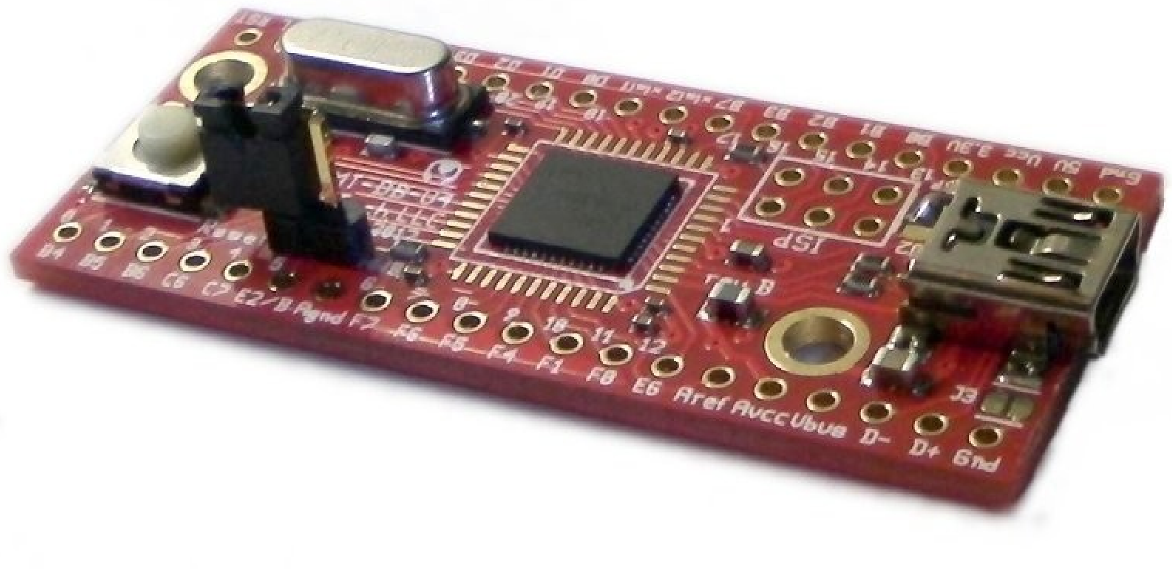
- ATmega32U4 USB microcontroller
  - 32KB FLASH, 2.5KB SRAM, 1KB EEPROM
  - 12 10-bit ADC channels (1 used by LED which can be disconnected)
  - Serial USART, SPI, and TWI (I2C) communications
  - 4 timers with 14 PWM channels (up to 7 simultaneous)
- Arduino compatible (now supports IDE 1.6.7 and boards manager)
- 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

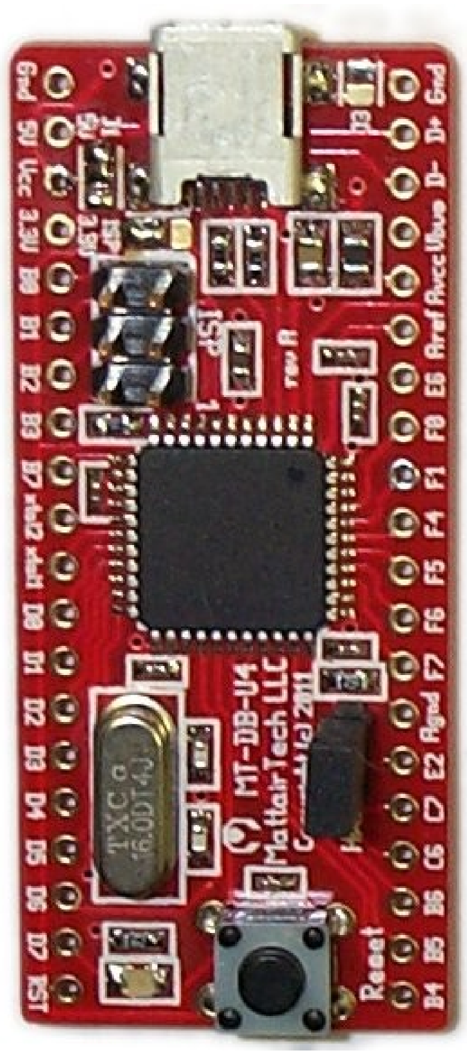

B4
B5
B6
C6
C7
E2/B
Agnd
F7
F6
F5
F4
F1
F0
E6
Aref
Avcc
Vbus
D-
D+
Gnd
ISP
5V
3.3V
Vcc
B0
B1
B2
B3
B7
D0
D1
D2
D3
J3
MT-DB-U4

MT-DB-U4 Hardware

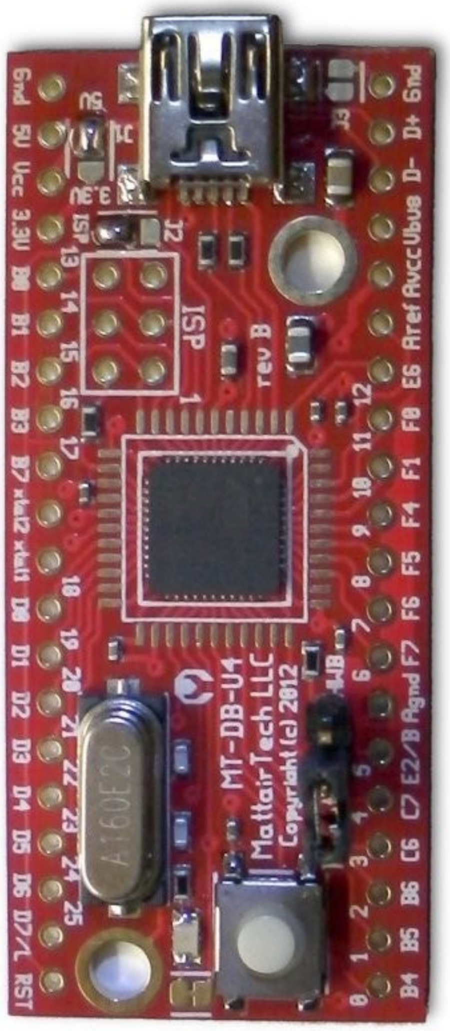
Board Revisions

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

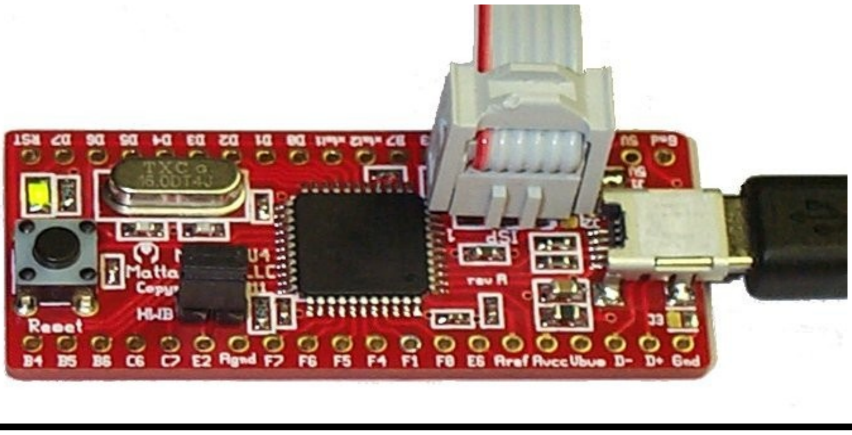
This section does not apply to the Atmel DFU bootloader option, in which case, the fuses will be left at their factory default settings and the factory installed Atmel DFU bootloader will not be overwritten.

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 (IDE 1.6.7)

This is a fork of the Arduino AVR core from arduino/Arduino (hardware/arduino/avr/ directory) on GitHub. This will be used to maintain Arduino support for AVR boards including the MattairTech MT-DB-U1, MT-DB-U2, MT-DB-U4, and the MT-DB-U6 (see https://www.mattairtech.com/).

This core is intended to be installed using Boards Manager (see below). To update from a previous version, click on MattairTech AVR Boards in Boards Manager, then click Update.

What's New

- Initial release of the 1.6.x compatible AVR core.
- Any combination of CDC, HID, or UART can be used (or no combination), by using the Tools->Communication menu.
- Note that switching between CDC and CDC+HID will require re-selecting the COM port.
- More detailed memory usage at end of compilation (see below).
- Merged in upstream updates.

Summary

Feature	MT-DB-U6	MT-DB-U4	MT-DB-U2	MT-DB-U1
Microcontroller	AT90USB64/AT90USB128, 8-Bit AVR	ATmega32U4, 8-Bit AVR	ATmega32U2, 8-Bit AVR	AT90USB162, 8-Bit AVR
Clock Speed	16 MHz	16 MHz	16 MHz	16 MHz
Flash Memory	128 KB (AT90USB128) / 64 KB (AT90USB64)	32 KB	32 KB	16 KB
SRAM	8 KB (AT90USB128) / 4 KB (AT90USB64)	2.5 KB	1 KB	512 B
EEPROM	4 KB (AT90USB128) / 2 KB (AT90USB64)	1 KB	1 KB	512 B
Digital Pins	46*	26	21	21
Analog Input Pins	8 (10-bit)	11* (10-bit)	No analog	No analog
PWM Output Pins	7*	7	4	4
External Interrupts	6* (8 PCINT)*	5 (8 PCINT)*	8 (13 PCINT)*	8 (13 PCINT)*
USB	CDC and HID	CDC and HID	CDC and HID	CDC and HID
UART (Serial)	1	1	1	1
SPI	1	1	1	1
I2C (TWI)	1	1	No I2C	No I2C
Operating Voltage	5V/3.3V	5V/3.3V	5V/3.3V	5V/3.3V

Feature	MT-DB-U6	MT-DB-U4	MT-DB-U2	MT-DB-U1
DC Current per I/O Pin	20 mA	20 mA	20 mA	20 mA

- Only INT pins are supported in this core (PCINT pins are not supported).
- MT-DB-U4: 1 additional analog pin is available by disconnecting the LED (solder jumper on rev B and higher boards)
- MT-DB-U6-64/128: 2 additional digital, 2 additional PWM, or 2 additional INT pins available with RTC crystal removed. Note however, that the RTC crystal holes are smaller and closer together than the header pin holes.

Special Notes

- **Tools->Communications menu**
  Currently, the Tools->Communications menu must be used to select the communications configuration. This configuration must match the included libraries. For example, when including the HID and Keyboard libraries, you must select an option that includes HID (ie: CDC_HID_UART). This menu is currently needed to select the USB PID that matches the USB device configuration (needed for Windows). This may become automatic in a future release.
- **Include platform specific libraries**
  You may need to manually include platform specific libraries such as SPI.h, Wire.h, and HID.h.
- **EXCEPTION_FOR_57600**
  The MattairTech ArduinoCore-avr uses a more accurate baud rate for 57600 than the stock arduino. When using the USART to communicate with another Arduino, define EXCEPTION_FOR_57600.
- **New interrupt mapping**
  The MattairTech ArduinoCore-avr has changed interrupt pin mapping from the previous 1.0.5 release. The arduino pin number is now used with attachInterrupt() instead of the interrupt number. See 'Pin Configurations' below.

Pin Configurations

To determine the Arduino pin number, start at the upper-left corner of the board opposite of the USB connector. This is pin 0 (most boards have a 0 printed nearby). The numbering increases in a counter-clockwise direction around the board. Many pins have multiple configurations available. For example, arduino pin 29 (AVR pin D0) on the MT-DB-U6 can be a PWM output (analogWrite), an external interrupt input, digital I/O, or the SCL pin of I2C.

MT-DB-U6 (AT90USB64/AT90USB128)

```
================== MattairTech MT-DB-U6 (AT90USB64/AT90USB128) ==========================
INT/Other  PWM  Analog  Digital                              Digital   PWM      INT/Other    Comm
=========================================================================================
                               ---------------------
LED                        0  | E0/L      O O   RST |
                           1  | E1        HWB   E2/B|  37                JUMPER
                           2  | C0              D7  |  36
                           3  | C1   O O        D6  |  35
                           4  | C2   O O        D5  |  34                            XCK
                           5  | C3   O O        D4  |  33
        6 (TC3C)           6  | C4   O O *      D3  |  32                32 (INT3)   TX
        7 (TC3B)           7  | C5  PORT A      D2  |  31                31 (INT2)   RX
        8 (TC3A)           8  | C6              D1  |  30   30 (TC2B)    30 (INT1)   SDA
                           9  | C7              D0  |  29   29 (TC0B)    29 (INT0)   SCL
                              |                     |
JTAG TDI       10 (ADC7)  10  | F7              E3  |  28
JTAG TDO       11 (ADC6)  11  | F6      PWR SW  E7  |  27                27 (INT7)
JTAG TMS       12 (ADC5)  12  | F5        O     B7  |  26   26 (TC1C)
JTAG TCK       13 (ADC4)  13  | F4  - +   O     B6  |  25   25 (TC1B)
               14 (ADC3)  14  | F3  O O         B5  |  24   24 (TC1A)
               15 (ADC2)  15  | F2  PWR IN      B4  |  23   23 (TC2A)
               16 (ADC1)  16  | F1              B3  |  22                            MISO
               17 (ADC0)  17  | F0      * O O   B2  |  21                            MOSI
18 (INT6)                 18  | E6        O O   B1  |  20                            SCLK
                              | Aref      O O   B0  |  19                            SS
                              | Vbus      ISP   3.3V|
                              | D-     _____    Vcc |
                              | D+    |     |   5V  |
                              | Gnd   | USB |   Gnd |
                               ---------------------
```

* Pins 38-45 are on the PORT A header. Pins 46 and 47 are the RTC crystal pins E4 and E5 (in use by the RTC by default). With RTC crystal removed, there are 2 additional digital pins (46 and 47), 2 additional PWM pins (TIMER2A on pin 23 and TIMER2B on pin 30), and 2 additional INT pins (INT4 on pin 46 and INT5 on pin 47). All pins can be used with analogRead(). 8 of these pins are actual analog inputs, the rest connect to the internal reference (pin 47) or ground.

MT-DB-U4 (ATmega32U4)

```
======================== MattairTech MT-DB-U4 (ATmega32U4) =============================
INT/Other  PWM    Analog   Digital                 Digital  Analog   PWM  INT/other  Comm
=========================================================================================
                                 -------------------
                  0 (ADC11)  0  | B4            RST |
        1 (TC1A)  1 (ADC12)  1  | B5            D7/L| 25  25 (ADC10)  25 (TC4D)  LED
        2 (TC1B)  2 (ADC13)  2  | B6            D6  | 24  24 (ADC9)
        3 (TC3A)             3  | C6            D5  | 23  23 (REF)
        4 (TC4A)             4  | C7            D4  | 22  22 (ADC8)
JUMPER                       5  | E2/B          D3  | 21                   21 (INT3)  TX
                                | Agnd          D2  | 20                   20 (INT2)  RX
                  6 (ADC7)   6  | F7            D1  | 19                   19 (INT1)  SDA
                  7 (ADC6)   7  | F6            D0  | 18        18 (TC0B)  18 (INT0)  SCL
                  8 (ADC5)   8  | F5           xtal1|
                  9 (ADC4)   9  | F4           xtal2|
                  10 (ADC1)  10 | F1            B7  | 17        17 (TC1C)
                  11 (ADC0)  11 | F0            B3  | 16                             MISO
12 (INT6)         12 (TEMP)  12 | E6            B2  | 15                             MOSI
                                | Aref          B1  | 14                             SCLK
                                | Avcc          B0  | 13                             SS
                                | Vbus          3.3V|
                                | D-    _____   Vcc |
                                | D+   |     |  5V  |
                                | Gnd  | USB |  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.

MT-DB-U1/MT-DB-U2 (AT90USB162/ATmega32U2)

```
============== MattairTech MT-DB-U1/MT-DB-U2 (AT90USB162/ATmega32U2) ===================
 Comm    Interrupt   PWM   Digital                  Digital  Interrupt    PWM  Comm/other
=========================================================================================
                                 -------------------
SPI SS                       0  | B0            RST |
SPI SCLK                     1  | B1            D7  |  20   20 (INT7)              JUMPER
SPI MOSI                     2  | B2            D6  |  19   19 (INT6)
SPI MISO                     3  | B3            D5  |  18
                             4  | B4            D4  |  17   17 (INT5)
                             5  | B5            D3  |  16   16 (INT3)           USART1 TX
                             6  | B6            D2  |  15   15 (INT2)           USART1 RX
                   7 (TC1C)  7  | B7            D1  |  14   14 (INT1)
         8 (INT4)            8  | C7            D0  |  13   13 (INT0)   13 (TC0B)   LED
                   9 (TC1A)  9  | C6            C2  |  12
                   10 (TC1B) 10 | C5            X1  |
                             11 | C4            X2  |
                                | Vbus          3.3V|
                                | D-    _____   Vcc |
                                | D+   |     |  5V  |
                                | Gnd  | USB |  Gnd |
                                 -------------------
```

Pin Capabilities

- **Digital: All pins can be used for general purpose I/O**
  - Supports INPUT, OUTPUT, and INPUT_PULLUP.
  - Each pin can source or sink a maximum of 20 mA.
  - Internal pull-up resistors of 20-50 Kohms (disconnected by default).
  - Use the pinMode(), digitalWrite(), and digitalRead() functions.
- **Analog Inputs: 8 pins (MT-DB-U6) or 11 pins (MT-DB-U4) can be configured as ADC analog inputs.**
  - These are available using the analogRead() function.
  - All pins can be used for GPIO and some pins can be used for other digital functions (ie. pwm or serial).
  - Each pin provides 10 bits of resolution (1024 values).
  - Each pin measures from ground to 5.0 volts.
  - The upper end of the measurement range can be changed using the AREF pin and the analogReference() function.
- **PWM: 7 pins (MT-DB-U6, MT-DB-U4) or 4 pins (MT-DB-U2, MT-DB-U1) can be configured as PWM outputs.**
  - Available using the analogWrite() function.
  - Each pin provides 8 bits of resolution (256 values) by default.
- **External Interrupts: Up to 8 pins can be configured with external interrupts.**
  - 6 pins (MT-DB-U6), 5 pins (MT-DB-U4), or 8 pins (MT-DB-U2, MT-DB-U1).
  - Available using the attachInterrupt() function.
- **Serial: 1 pair of pins can be configured for TTL serial I/O.**
  - MT-DB-U6: Serial1: pin 31 (RX) and pin 32 (TX).
  - MT-DB-U4: Serial1: pin 20 (RX) and pin 21 (TX).
  - MT-DB-U2, MT-DB-U1: Serial1: pin 15 (RX) and pin 16 (TX).
- **SPI: 3 or 4 pins can be configured for SPI I/O (SPI).**
  - MT-DB-U6: Pin 21 (MOSI), pin 20 (SCK), pin 22 (MISO), and optionally pin 19 (SS, not currently used).
  - MT-DB-U4: Pin 15 (MOSI), pin 14 (SCK), pin 16 (MISO), and optionally pin 13 (SS, not currently used).
  - MT-DB-U2, MT-DB-U1: Pin 2 (MOSI), pin 1 (SCK), pin 3 (MISO), and optionally pin 0 (SS, not currently used).
  - SPI communication using the SPI library.
- **TWI (I2C): 2 pins can be configured for TWI I/O (Wire).**
  - MT-DB-U6: Pin 30 (SDA) and pin 29 (SCL).
  - MT-DB-U4: Pin 19 (SDA) and pin 18 (SCL).
  - MT-DB-U2, MT-DB-U1: TWI not present
  - TWI communication using the Wire library.
- **LED: One pin can be configured to light the onboard LED (LED_BUILTIN).**
  - Pin 0 (MT-DB-U6), pin 25 (MT-DB-U4), or pin 13 (MT-DB-U2, MT-DB-U1).
  - Bring the pin HIGH to turn the LED on.
- **AREF: One pin can be configured as an AREF analog input.**

    - The upper end of the analog measurement range can be changed using the analogReference() function.
- **Reset: Bring this line LOW to reset the microcontroller.**

Using Arduino with MattairTech USB boards

Because of the similarities with the Arduino Leonardo, please read http://arduino.cc/en/Guide/ArduinoLeonardo first.

Within the Arduino IDE Tools menu, select the appropriate MattairTech board, Frequency/Voltage, Processor, Communications setting, and COM port. There are 2 Frequency/Voltage 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. Be sure to select the Communications setting that matches your sketch (by default, this is CDC_ONLY). This is important.

Note that some example sketches indicate the use of pins using the naming convention D2, D3, etc. These are Arduino digital pins, not to be confused with port D pins. Most MattairTech USB AVR boards are printed with both port pin names as well as sequential numbers indicating the Arduino pin number. You may use the 'A' or 'D' prefixes, but they are simply aliased to the arduino pin number (ie: A2 = D2 = 2).

There are several libraries included with Arduino. Some of these need simple changes to work with MattairTech boards. Usually, only pin mappings need to be changed.

Serial Monitor

To print to the Serial Monitor over USB, use 'Serial'. Serial points to SerialUSB (Serial1 is a UART). Unlike most Arduino boards (ie. Uno), USB AVR based boards do not automatically reset when the serial monitor is opened. To see what your sketch outputs to the serial monitor from the beginning, the sketch must wait for the SerialUSB port to open first. Add the following to setup():

```
while (!Serial) ;
```

Remember that if the sketch needs to run without SerialUSB connected, another approach must be used. You can also reset the board manually with the Reset button if you wish to restart your sketch. However, pressing the Reset button will reset the AVR chip, which in turn will reset USB communication. This interruption means that if the serial monitor is open, it will be necessary to close and re-open it to restart communication.

Updated Tone.cpp

Tone.cpp now supports multiple simultaneous tone generation (one tone per timer). The MT-DB-U6 currently supports up to 4 simultaneous tones using timers 3, 1, 2, and 0 if not using the RTC, otherwise, timers 3, 1, and 0 are used for 3 tones. 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(), 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.

The DTMF_Demo sketch demonstrates usage of Tone.cpp for DTMF generation.

Detailed Memory Usage Output After Compilation

In this release, two programs are run at the end of compilation to provide more detailed memory usage. This is enabled only when verbose messages for compilation is enabled in the IDE Preferences. Just above the normal flash usage message, is the output from the size utility. Above the size utility output is the output from the nm utility. The values on the left are in bytes. The letters stand for: T(t)=.text, D(d)=.data, B(b)=.bss, and everything else (ie: W) resides in flash (in most cases).

USB Technical Notes

- **Note that USB CDC is required for auto-reset into the bootloader to work (otherwise, manually press reset with jumper installed).**

ATmegaxxU4: 832 bytes DPRAM, 1 (control, 64 byte max) + 1 (two banks, 256 byte max) + 5 (two banks, 64 byte max) endpoints AT90USBxxx6/7: 832 bytes DPRAM, 1 (control, 64 byte max) + 1 (two banks, 256 byte max) + 5 (two banks, 64 byte max) endpoints

```
// These are used by the core
#define USB_CONTROL_EP_SIZE     16
#define USB_CONTROL_EP_BANKS        1
#define USB_DEFAULT_EP_SIZE     64
#define USB_DEFAULT_EP_BANKS        2
#define USB_CDC_NOTIFICATION_EP_SIZE    16
#define USB_CDC_NOTIFICATION_EP_BANKS   1
#define USB_CDC_DATA_EP_SIZE        64
#define USB_CDC_DATA_EP_BANKS       2

// These can optionally be used by PluggableUSB libraries
#define USB_HID_EP_SIZE         16
#define USB_HID_EP_BANKS        1
#define USB_MIDI_EP_SIZE        64
#define USB_MIDI_EP_BANKS       2
#define USB_MSD_EP_SIZE         64
#define USB_MSD_EP_BANKS        2
```

AT90USBxx2: 176 bytes DPRAM, 8 - 64 byte endpoints, 1 (control) + 2 (one bank) + 2 (two banks) endpoints ATmegaxxU2: 176 bytes DPRAM, 8 - 64 byte endpoints, 1 (control) + 2 (one bank) + 2 (two banks) endpoints

```
// These are used by the core
```

```
#define USB_CONTROL_EP_SIZE     16
#define USB_CONTROL_EP_BANKS        1
#define USB_DEFAULT_EP_SIZE     32
#define USB_DEFAULT_EP_BANKS        2
#define USB_CDC_NOTIFICATION_EP_SIZE    16
#define USB_CDC_NOTIFICATION_EP_BANKS   1
#define USB_CDC_DATA_EP_SIZE        32
#define USB_CDC_DATA_EP_BANKS       2

// These can optionally be used by PluggableUSB libraries
#define USB_HID_EP_SIZE         16
#define USB_HID_EP_BANKS        1
#define USB_MIDI_EP_SIZE        32
#define USB_MIDI_EP_BANKS       2
#define USB_MSD_EP_SIZE         32
#define USB_MSD_EP_BANKS        2
```

Installation

Driver Installation

Windows

Prior to core version 1.6.9-mt1, sketches compiled with both CDC and HID USB code by default, thus requiring a CDC driver for the bootloader and a CDC-HID driver for sketches. Now that PluggableUSB is supported, sketches compile with only CDC code by default. Thus, only one driver is needed. Since HID and MIDI are currently supported (and MSD potentially in the future), driver installation will be required for each different combination of USB devices. There are currently four USB composite device combinations that include CDC as well as a CDC only device. Each supported combination has a unique USB VID:PID pair, and these are listed in the .inf file. Once the first device is installed (the CDC only device), future installations *might* be automatic, otherwise, you may direct the installer to the same .inf file. The drivers are signed and support both 32 and 64 bit versions of Windows XP(SP3), Vista, 7, 8, and 10.

1. If you do not already have the CDC bootloader installed, see below.
2. Download https://www.mattairtech.com/software/MattairTech_CDC_Driver_Signed.zip and unzip into any folder.
3. Plug in the board with the jumper installed. The LED should light.
4. Windows will detect the board. Point the installer to the folder from above to install the bootloader driver.
5. If you don't intend on using Arduino, you can skip the rest of this list. See Using AVRDUDE Standalone below.
6. If you do not already have the test firmware installed, see Using AVRDUDE Standalone below.
7. Press the reset button to run the test firmware (blink sketch).
8. Windows will detect the board. Point the installer to the above folder to install the sketch driver (if needed).

9. Continue with AVR Core Installation below.

Linux

1. No driver installation is needed.
2. On some distros, you may need to add your user to the same group as the port (ie: dialout) and/or set udev rules.
3. You MAY have to install and use Arduino as the root user in order to get reliable access to the serial port.
    - This is true even when group permissions are set correctly, and it may fail after previously working.
    - You can also create/modify a udev rule to set permissions on the port so *everyone* can read / write.
4. Continue with AVR Core Installation below.

OS X

UNTESTED

1. No driver installation is needed.
2. Plug in the board. You may get a dialog box asking if you wish to open the “Network Preferences”:
    - Click the "Network Preferences..." button, then click "Apply".
    - The board will show up as “Not Configured”, but it will work fine.
3. Continue with AVR Core Installation below.

AVR Core Installation

- To update from a previous version, click on MattairTech AVR Boards in Boards Manager, then click Update.

1. The MattairTech AVR Core requires Arduino 1.6.7+.
2. In the Arduino IDE 1.6.7+, click File->Preferences.
3. Click the button next to Additional Boards Manager URLs.
4. Add https://www.mattairtech.com/software/arduino/package_MattairTech_index.json.
5. Save preferences, then open the Boards Manager.
6. Install the MattairTech AVR Boards package.
7. Close Boards Manager, then select your board from Tools->Board.
8. Select the Frequency/Voltage with the now visible Tools->Frequency/Voltage menu.
9. Select the processor with the now visible Tools->Processor menu.
10. Select the communications option with the now visible Tools->Communications menu (must match sketch).
11. If you do not already have the bootloader or blink sketch installed, see USB CDC Bootloader

below.

12. Plug in the board. The blink sketch should be running.
13. Click Tools->Port and choose the COM port.
14. You can now upload your own sketch.

Uploading the First Sketch

1. In the Arduino IDE 1.6.7 (or above), open File->Examples->01.Basics->Blink.
2. Change the three instances of '13' to 'LED_BUILTIN'.
3. Be sure the correct options are selected in the Tools menu (see AVR Core Installation above).
4. With the board plugged in, select the correct port from Tools->Port.
5. Click the Upload button. After compiling, the sketch should be transferred to the board.
6. Once the bootloader exits, the blink sketch should be running.

USB CDC Bootloader (Arduino compatible)

Each board has several bootloaders available. The CDC bootloader can be used with Arduino. Version 130410 or above is required to support the auto-reset feature. Note that several boards that were shipped after 130410 but before 130626 still have the old bootloader.

The bootloader enters programming mode only if the jumper is installed, except when using Arduino auto-reset or when the FLASH is empty. Even with the jumper installed, programming mode will NOT be entered if the reset was from the watchdog timer, unless the boot key is enabled and the key matches, as is the case with Arduino auto-reset (the Arduino core uses a watchdog reset to enter the bootloader).

The default CDC bootloader has the following compile-time options defined:

```
#define ENABLE_LED_BOOT
#define ENABLE_LED_APPLICATION
#define DISABLE_JTAG_APPLICATION
#define ENABLE_CLKDIV_1_APPLICATION
#define ENABLE_BOOT_KEY
#define ENABLE_RESET_AFTER_PROGRAMMING
#define NO_LOCK_BYTE_WRITE_SUPPORT
```

An alternate version with the above options undefined is available on the website named Bootloader_no_options.hex. Use it if the default options interfere with your application. For example, you may disconnect the LED and use the pin as an analog input.

Bootloader Firmware Installation Using the Arduino IDE

1. If you do not already have the MattairTech AVR core installed, see AVR Core Installation above.
2. Plug a compatible programmer into a USB port, then connect it to the powered AVR board.
3. Select your programmer from Tools->Programmer.
4. Select your board from Tools->Board.

5. Click Tools->Burn Bootloader. Ignore any messages about not supporting shutdown or reset.
6. Continue with driver installation above.

Using AVRDUDE Standalone

AVRDUDE can be used standalone. You can use the version included with Arduino (in arduino-1.6.7/hardware/tools/avr/bin) or download a separate version from http://download.savannah.gnu.org/releases/avrdude/.

As an example, AVRDUDE will be used to upload the test firmware (blink sketch):

1. Download firmware from https://www.mattairtech.com/software/CDC-bootloader-test-firmware.zip and unzip.
2. If you have not already installed the bootloader driver, see Driver Installation above.
3. Be sure there is a hex file that matches your chip. On the command line (change the hex file to match yours):

```
avrdude -p m32u4 -c avr109 -P usb -U flash:w:"blink.hex"
```

1. On linux, the -P option should be something like /dev/ttyACM0.
2. See http://www.nongnu.org/avrdude/user-manual/avrdude_4.html for details.
3. Press the reset button with the jumper off to load the sketch.
4. When using AVRDUDE standalone, the jumper must be installed before pressing reset to run the bootloader.

Possible Future Additions

- Features for lower power consumption
- MSC (Mass Storage) USB Device Class
- Host mode CDC
- Better OS X support
- PCINT support

ChangeLog

- 1.6.9-mt1:
  - See 'What's New' above.

- 1.0.5.1 fixes the sketch not running when not connected to a USB host (ie: USB charger). Version 1.0.5 fixes several bugs (including BSoD's on Win7-64) and updates the Arduino core files and libraries to 1.0.5. Merged in changes to Arduino 1.0.5 core and examples. Changed a few //'s to #'s in boards.txt. Fixed blank spaces in board selection list. Eliminate descriptor serial numbers. Use new PID. Fixed Win7-64 BSoD's and code 10's. New inf file to support new PID. Initialize USB (HID and CDC) without needing Serial.begin(). Made USB_WAITFORCONNECT_DISABLED default instead of

USB_WAITFORCONNECT_ENABLED. Added two nop()'s to USBSerial::readRXEndpoint() so switching USB endpoint banks does not result in returning -1 (empty). USBSerial::peek() fixed. Wait for USB_DeviceState_Connected state before continuing. change keyboardmouse demo to use different pins.

- 1.0.4 adds HID keyboard and mouse support, adds auto-reset support, updates LUFA to 130303, updates the Arduino core files and libraries to 1.0.4, updates the bootloaders, and adds support for the new MT-DB-U6.

License and credits

This core has been developed by Arduino LLC. This fork developed by Justin Mattair of MattairTech LLC.

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 the .inf file from https://www.mattairtech.com/software/MattairTech_CDC_Driver_Signed.zip. Note that Windows Vista 64-bit, Windows 7 64-bit and Windows 8 require the signed driver. 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. Version 130410 or above is required to support the auto-reset feature (note that several boards that were shipped before 130626 still have the old bootloader). If using a terminal emulator, you must first disconnect before running the bootloader. The bootloader enters programming mode only if the jumper is installed, even when using Arduino auto-reset. The one exception is when the FLASH is empty. Even with the jumper installed, programming mode will NOT be entered if the reset was from the watchdog timer. The one exception to this is when the boot key is enabled and the key matches. The key will match when the Arduino IDE auto-resets the board to enter bootloader programming mode. The key is needed because the Arduino core part of the firmware, which listens for the IDE auto-reset signal, uses a watchdog reset to enter the bootloader. This way, the user application can make use of the watchdog timer. 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).

May 2, 2014 UPDATE:

Version 140502:

- Added #define ARDUINO_MODE to AppConfig.h. **This eliminates the requirement for the jumper to be installed.** Arduino users should now always leave the jumper off. If you cannot enter the bootloader (ie: sketch compiled with USB_AUTORESET_DISABLED), you can force the bootloader by installing the jumper (be sure to re-select the COM port).

- Removed #define ENABLE_CLKDIV_1_APPLICATION from AppConfig.h. Now, the bootloader always runs at the crystal speed (16MHz). See next entry.
- Fixed problem on Linux systems where the LED would sometimes freeze and the USB connection would fail. This was due to the bootloader running at 8MHz. Now it always runs at 16MHz. Note that when operating at 3.3V, the cpu will be overclocked, but it should work fine.
- Fixed problem where AVRDUDE would sometimes freeze at the end of programming. This was due to the bootloader exiting before the last ACK was sent to AVRDUDE. This may have affected other host software as well.
- Increased the time between disconnecting from the USB host and switching to the application.
- Updated LUFA library to version 140302.

Each board has several bootloaders available. The CDC bootloader can be used with Arduino. Version 130410 or above is required to support the auto-reset feature. Note that several boards that were shipped after 130410 but before 130626 still have the old bootloader. It is strongly recommended to use version 140502 or higher when using with Arduino.

The default CDC bootloader has the following compile-time options defined:

```
#define NO_LOCK_BYTE_WRITE_SUPPORT
#define ENABLE_LED_BOOT
#define ENABLE_LED_APPLICATION
#define DISABLE_JTAG_APPLICATION
#define ENABLE_BOOT_KEY
#define ENABLE_RESET_AFTER_PROGRAMMING
#define ARDUINO_MODE
```

An alternate version with the above options undefined is available on the website named Bootloader_no_options.hex. Use it if the default options interfere with your application. For example, you may disconnect the LED and use the pin as an analog input.

When using the auto-reset feature of Arduino, the boards.txt file must currently list the bootloader directory as caterina (the bootloader used on the Leonardo). The actual bootloader is a modified version of the LUFA CDC bootloader by Dean Camera (lufa-lib.org). It resides in the mtdbxx folder (where xx corresponds to the board you have). So, if you wish to use the Arduino IDE to burn the bootloader, you must temporarily change the appropriate entry in the boards.txt file to point toward the actual bootloader directory. Change it back to caterina when finished to re-enable auto-reset.

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:

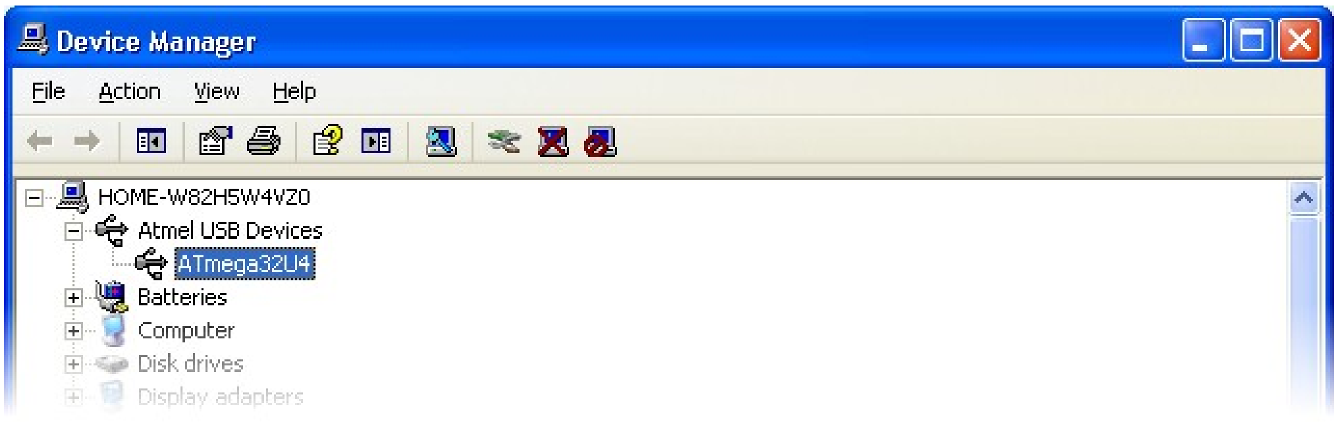
Be sure to select the COM port. Then upload your sketch with the Upload button.

DFU Bootloader (FLIP/dfu-programmer)

Installation

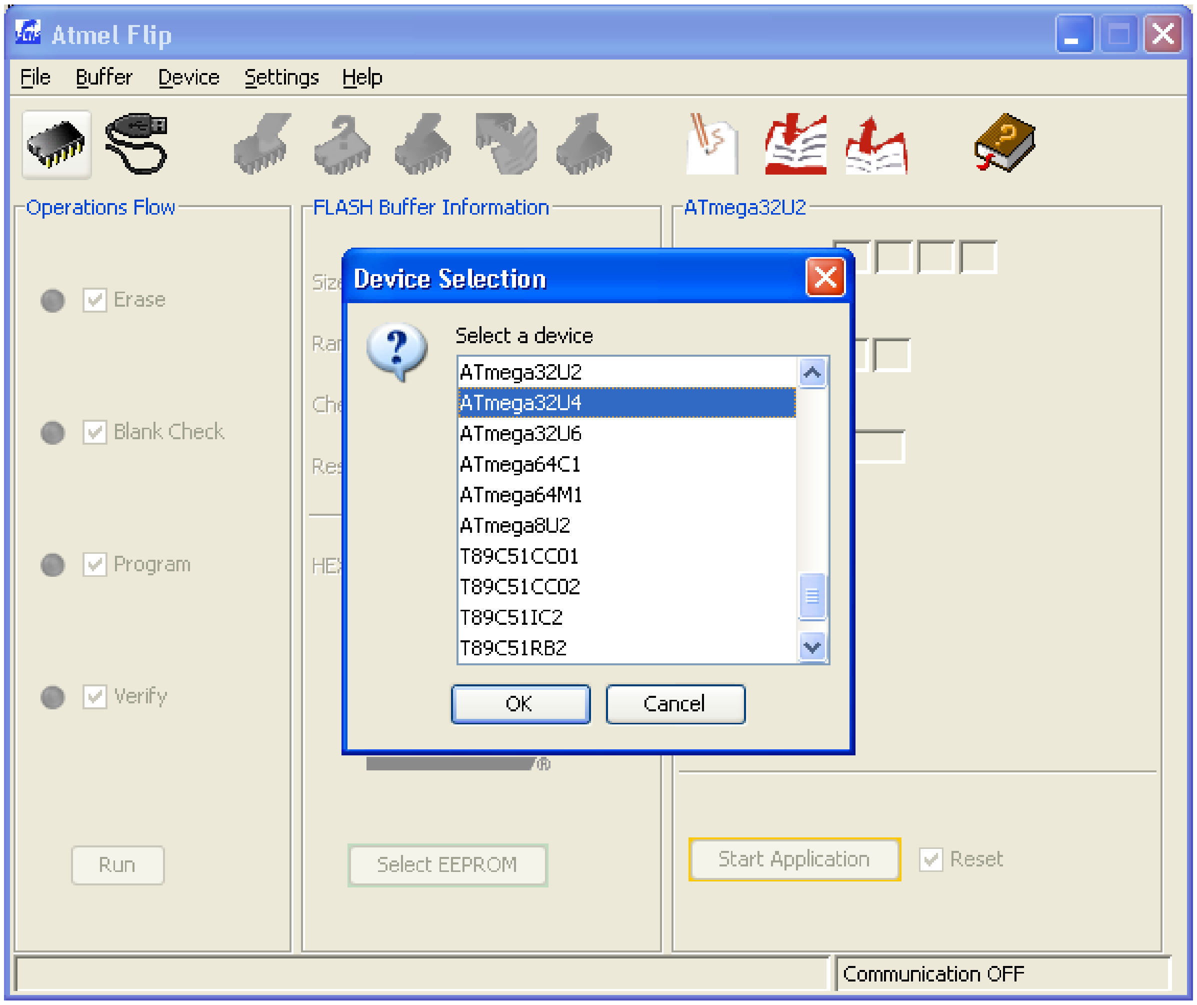
FLIP is a graphical utility used to load firmware into the ATmega32U4. FLIP also includes the driver for the bootloader. Download FLIP 3.4.2 or higher from http://www.atmel.com/tools/FLIP.aspx. Be sure to choose the version that bundles a Java JRE if you do not already have one installed. Run the downloaded executable to begin the installation.

Next, install the HWB jumper by placing the jumper cap onto the two pins marked HWB (so they are shorted). Note that HWB as printed on the board refers to the jumper function, not to the actual AVR pin. Also note that if you selected the Atmel DFU bootloader instead of the standard LUFA DFU bootloader, you will need to add an external pullup resistor to pin E2 (ie: a 10Kohm resistor between E2 and Vcc; and it must remain installed at all times). Then, connect the board to your computer via USB. Because the HWB jumper is installed, the bootloader will run. The LED should be pulsing. Windows will then prompt you for the ATmega32U4 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. Once the driver is loaded, the device will appear as the ATmega32U4 device under Atmel USB Devices in the device manager.



FLIP

Install the HWB jumper by placing the jumper cap onto the two pins marked HWB. Then, connect the board to your computer via USB. The LED should be pulsing (unless using the Atmel bootloader). Now launch the FLIP utility. Click on the chip icon (far left) and select the Atmega32U4.



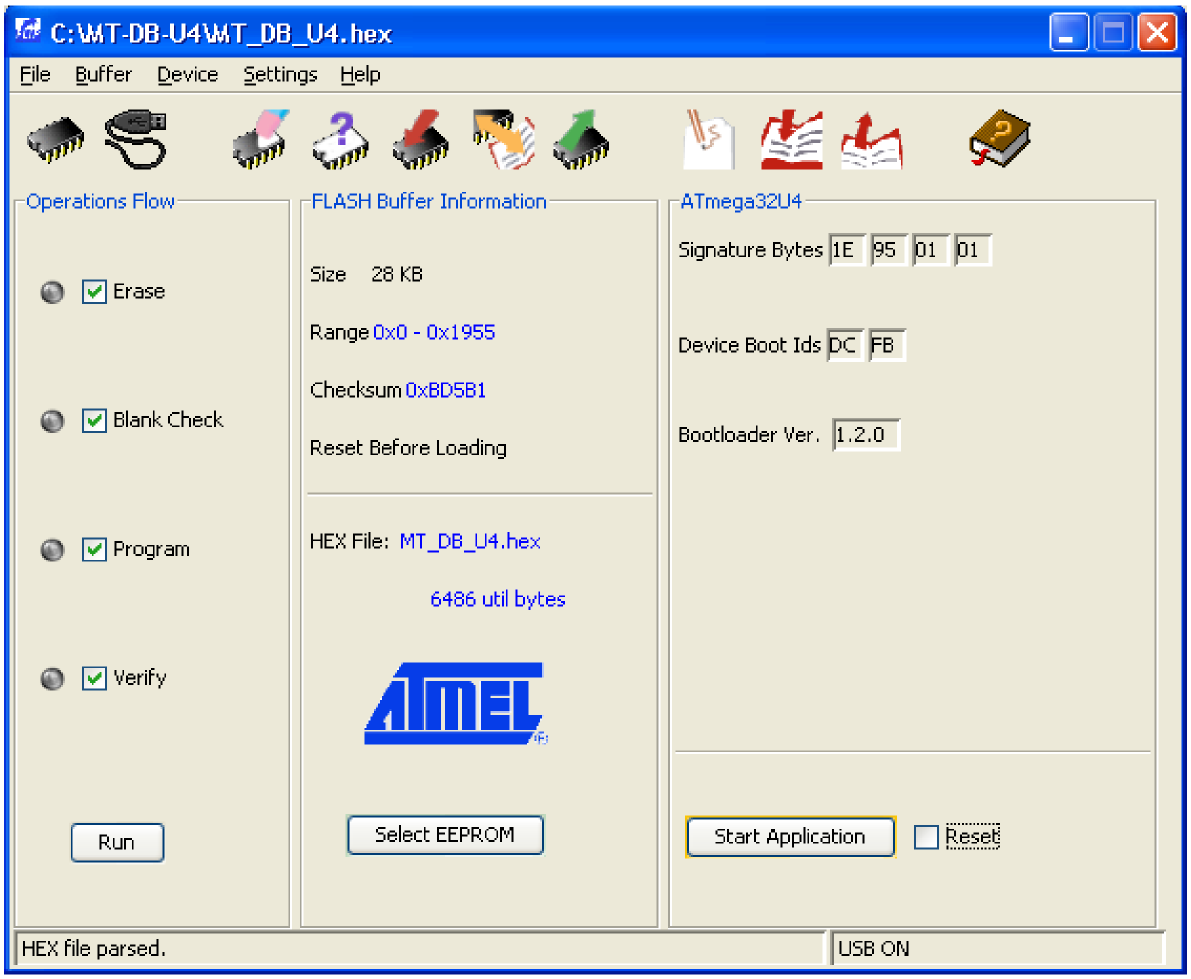
Select a device

ATmega32U2
ATmega32U4
ATmega32U6
ATmega64C1
ATmega64M1
ATmega8U2
T89C51CC01
T89C51CC02
T89C51IC2
T89C51RB2

Next, click on the USB cable icon (second from left), select USB, then connect. The screen should now show information about the ATmega32U4.

*** NOTE: If you get a "AtLibUsbDfu.dll not found message" error message from Flip when you click on the USB icon, the problem is a a wrong/missing board driver.*
- Close Flip.
- Disconnect/reconnect/reset board as needed to get Windows to reload the driver.
- Point the driver install wizard to the "..\Program Files\Atmel\Flip 3.4.7\usb\" folder.
- Once the driver is installed, go back to step 1.

Click on the File menu, and open the appropriate hex file. More information will appear about the program. All four check boxes should be checked. Be sure that erase is checked. The firmware cannot be loaded unless the flash is erased first. Now click on the Run button in the lower-left of the screen, and the firmware will be quickly loaded onto the ATmega32U4. If verification is successful, “Verify PASS” will appear in the status message area (bottom left).


C:\MT-DB-U4\MT_DB_U4.hex

Operations Flow
- [✓] Erase
- [✓] Blank Check
- [✓] Program
- [✓] Verify

FLASH Buffer Information
Size 28 KB
Range 0x0 - 0x1955
Checksum 0xBD5B1
Reset Before Loading
HEX File: MT_DB_U4.hex
6486 util bytes

ATmega32U4
Signature Bytes 1E 95 01 01
Device Boot Ids DC FB
Bootloader Ver. 1.2.0

[ ] Reset

HEX file parsed.
USB ON

If the EEPROM must also be programmed, 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.

Finally, remove the jumper so that the bootloader does not run after resetting / power-cycling. Initiate reset by unchecking the reset box and pressing the “Start Application” button (lower right). Or, simply press the reset button. The firmware that was just loaded will now run.

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

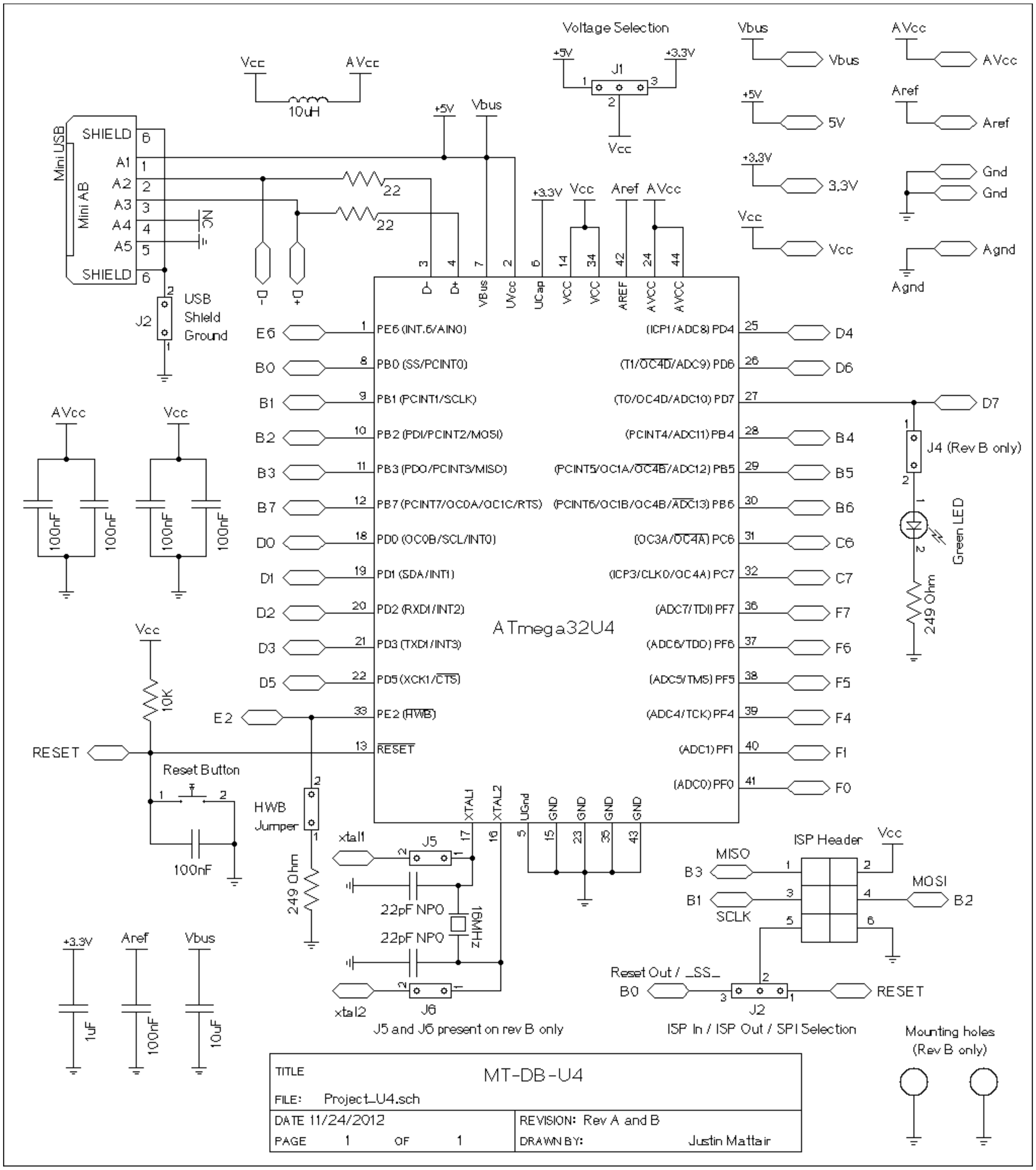
Commands	arg else function help if ls peep print ps return rm run stop switch while
Functions	abs ar aw bc beep br bs bw constrain delay dr dw er ew free inb max millis min outb pinmode printf pulsein random shiftout sign snooze

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
+5V
+3.3V
J1
Vcc
AVcc
10uH
Vbus
5V
3.3V
Aref
Gnd
Agnd
Mini USB
Mini AB
SHIELD
A1
A2
A3
A4
A5
NC
22
D-
D+
USB Shield Ground
J2
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
D-
D+
VBus
UVcc
UCap
VCC
AREF
AVCC
XTAL1
XTAL2
UGnd
GND
E6
B0
B1
B2
B3
B7
D0
D1
D2
D3
D5
E2
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
Reset Out / _SS_
ISP In / ISP Out / SPI Selection
1uF
10uF
Mounting holes (Rev B only)
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

Fuse and Lock Settings

The bootloaders were pre-installed with the following commands (ATmega32u4 CDC bootloader shown):

```
avrdude -p m32u4 -c avrisp2 -P usb -e
avrdude -p m32u4 -c avrisp2 -P usb -U lfuse:w:0x7f:m -U hfuse:w:0x98:m -U efuse:w:0xcb:m
avrdude -p m32u4 -c avrisp2 -P usb -B 4 -U flash:w:"Bootloader.hex"
```

The Bitlash program was pre-installed with the following command (ATmega32u4 CDC bootloader shown):

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

The lockbits are not set with the CDC bootloader. They ARE set with the Atmel DFU bootloader.

Troubleshooting / FAQ

- The board may have either the TQFP or the QFN chip package installed (usually the TQFP). Additionally, on rare occasions, the RC variant will be installed. If so, the fuses will be changed such that the chip is exactly the same as the non-RC variant.
- The solder jumper J1 can become unsoldered when attaching headers.

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, CDC bootloader, DFU bootloader, and AVRISP mkII clone programmer. 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
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.

CAUTION
Normally, power is supplied from Vbus. However, it is possible to supply an externally regulated voltage on the 3.3V, 5V, and/or Vcc pins. When doing this, care must be taken to limit inrush current on these pins due to the low ESR of the ceramic capacitors. Failure to do so may cause damaging inductive voltage spikes due to any wire inductance (ie: benchtop power supply leads). Inrush current is normally controlled by the PTC fuse, which has a small series resistance.

CAUTION
At higher input voltages to the regulator, the larger voltage drop will mean higher thermal dissipation for a given amount of current. Be sure to limit current consumption to prevent excessive heat. The regulator will enter thermal shutdown if it gets too hot.

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

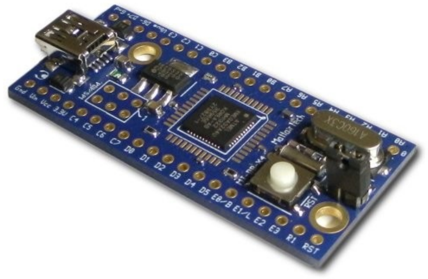
Appendix B: Other MattairTech Products

ZeptoProg II AVRISP mkII Programmer

- AVRISPmkII compatible AVR Programmer
- Supports all AVRs with ISP, PDI, or TPI
- Optional 5V output via headers to target board, with standard jumper and PTC fuse
- 4-channel Logic Analyzer
- Serial bridge / pattern generator / SPI interface
- GPIO / PWM / frequency input & output
- Atmel Studio / AVRDUDE support
- Target board voltage of 2V to 5.5V via level-shifted pins on two main headers

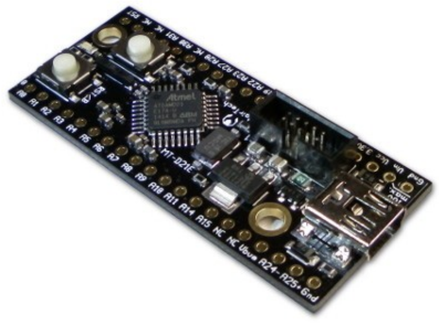
MT-DB-U6 USB AVR development board

- AT90USB646 / AT90USB1286 USB AVR
- 64KB/128KB FLASH, 4KB/8KB SRAM
- 5V, 500mA LDO regulator (3V-30V input)
- Auto power source selection IC (USB/External)
- 16MHz and 32.768KHz crystals
- Arduino compatible
- CDC or DFU bootloader



MT-DB-X4 USB AVR XMEGA board

- ATxmega128A4U USB XMEGA AVR
- 128KB FLASH, 8KB SRAM, 2KB EEPROM
- 3.3V LDO regulator (low quiescent current)
- 16MHz and 32.768KHz crystals
- LED, boot jumper, PDI header
- Reset button, mounting holes
- USB DFU bootloader preinstalled



MT-D21E USB ARM Cortex M0+ board

- ATSAMD21E17A or ATSAMD21E18A (32-pin)
- 128KB/256KB FLASH, 16KB/32KB SRAM
- Onboard 3.3V, 250mA LDO regulator (2uA quiescent)
- 16MHz and 32.768KHz crystals
- USB connector (power by USB or external up to 15V)
- Blue LED, 10-pin Cortex header, 2 buttons, I2C pullups
- USB Mass Storage Bootloader (no programmer required)