MT-X1 Manual

MattairTech

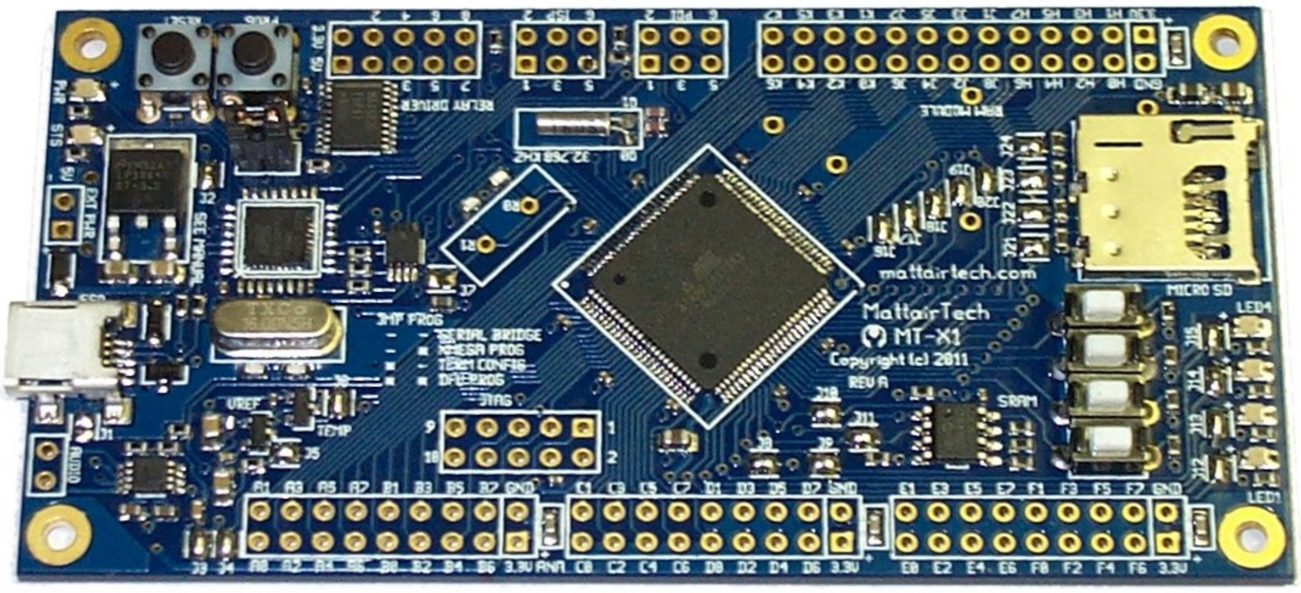


Table of Contents

Overview

Introduction

The MT-X1 is a flexible USB development board for the Atmel ATxmega128a1 microcontroller. Included is a MicroSD card slot, 32KB SPI SRAM, audio amplifier, lowside / relay driver, temperature sensor, 4 buttons, 4 LEDs, and an onboard 1.25V precision reference for the ADC. This reference is setup to work with the XMEGA errata. The XMEGA can be programmed over USB using the onboard AVRISP mkII compatible PDI programmer. The XMEGA can communicate with a computer using the onboard USB to serial bridge. Speeds up to 2Mbps are supported in synchronous mode (1Mbps in asynchronous mode). The Atmel AT90USB162 USB AVR, which provides these features, will automatically sleep when USB is disconnected. The board can be powered via USB or an external header. Voltage is regulated by a 3.3V, 1A LDO regulator. There are several clock options available onboard, including a 32.768KHz crystal, an external clock, an external HC49 crystal landing, and several internal clock options. Most XMEGA pins are routed to headers. The included peripheral devices are connected to the XMEGA via solder jumpers, which allows use of the pins if the device is not used. A demo program is preinstalled on the XMEGA demonstrating use of each peripheral device, as well as demonstrating sleep mode. Additionally, an Atmel Studio 6 ASF template is provided. All software used on both the XMEGA and USB AVR is open-source.

MT-X1 Features

- Atmel XMEGA 128A1 (chip rev. H), 128KB flash, 8KB RAM
- Onboard USB PDI programmer (no external programmer needed)
    - AVRISPmkII compatible
    - Program flash, EEPROM, fuses, lock bits, and more
    - Works with AVR Studio 4 and 5, Atmel Studio 6, AVRDUDE, Codevision, and BASCOM
- USB - Serial Bridge
    - Up to 2MHz baud rate (1MHz async)
    - Synchronous or asynchronous operation
    - Optional USB ready signal
- 3.3V, 1A LDO regulator
- Powered via USB or external header
- 32.768KHz crystal connected to TOSC (RTC) pins
- 8MHz external clock available from USB AVR
- HC49 crystal landing connected to XTAL pins
- Most pins routed to headers (Port A through Port K)
- Solder jumpers can be used to disconnect devices when not used (frees up header pin)
- MicroSD card slot with push-push spring action
- 32KB SPI SRAM chip
- 8 channel lowside / relay driver with kickback protection
    - Up to 70mA per channel
    - 5V or 3.3V devices (relays, LCD backlights, etc.)

    - Can be used as general-purpose lowside driver
- Audio amplifier connected to XMEGA DAC
- Temperature sensor with low-power operation
- 1.25V precision voltage reference
    - Works with XMEGA ADC errata
    - Use for signed differential conversions from 0V to ~2.5V at the pin
    - Routed to both reference inputs via solder jumpers
- 4 buttons
- 4 LEDs
- Available 1MB low power external SRAM (see http://www.mattairtech.com/)
- JTAG (XMEGA), PDI (XMEGA)*, and ISP (USB AVR) headers
- 4 boot modes selectable via jumper and button
    - Serial bridge (default)
    - AVRISP mkII compatible PDI programmer
    - Configuration (uses terminal emulator)
    - DFU bootloader (to update firmware on USB AVR via USB)
- Boot button can be used to toggle between the PDI programmer and the serial bridge
- Entire board can consume down to 75uA or less in sleep mode
- Atmel Studio 6 ASF (Atmel Software Framework) template to get started quickly
- Preloaded demo program demonstrates onboard peripheral devices as well as sleep mode
- PCB measures 10cm x 5cm
- Compatible with Windows XP/Vista/7/8 and Linux
- All firmware is open-source (MIT license)
- Uses LUFA USB library and AVRISP mkII clone by Dean Camera (`http://www.fourwalledcubicle.com/`)

ATxmega128a1 Features

- High-performance, Low-power 8/16-bit AVR® XMEGA Microcontroller
- Non-Volatile Program and Data Memories
  128K Bytes of In-System Self-Programmable Flash
  8K Bytes Boot Section with Independent Lock Bits
  2 KB EEPROM
  8 KB Internal SRAM
  External Bus Interface for up to 16M bytes SRAM
  External Bus Interface for up to 128M bit SDRAM
- Peripheral Features
  Four-channel DMA Controller with support for external requests
  Eight-channel Event System
  Eight 16-bit Timer/Counters
      4 Timer/Counters with 4 Output Compare or Input Capture
      4 Timer/Counters with 2 Output Compare or Input Capture
      High-Resolution Extension on all Timer/Counters

  - Advanced Waveform Extension on two Timer/Counters
- Eight USARTs
  - IrDA modulation/demodulation for one USART
- Four 2-Wire Interfaces w/ dual address match (I2C and SMBus)
- Four SPI (Serial Peripheral Interface) peripherals
- AES and DES Crypto Engine
- 16-bit Real Time Counter with separate Oscillator
- Two Eight-channel, 12-bit, 2 Msps Analog to Digital Converters
- Two Two-channel, 12-bit, 1 Msps Digital to Analog Converters
- Four Analog Comparators with Window compare function
- External Interrupts on all General Purpose I/O pins
- Watchdog Timer with Separate On-chip Ultra Low Power Oscillator

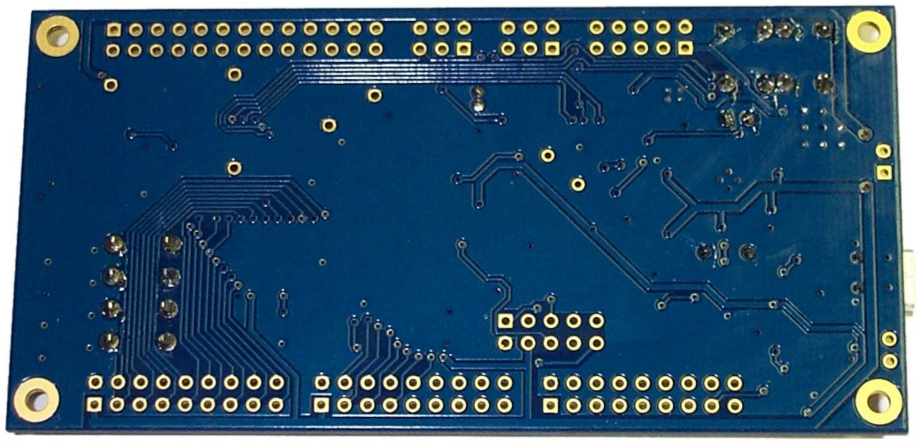
● Special Microcontroller Features
- Power-on Reset and Programmable Brown-out Detection
- Internal and External Clock Options with PLL and Prescaler
- Programmable Multi-level Interrupt Controller
- Sleep Modes: Idle, Power-down, Standby, Power-save, Ext. Stby.
- Advanced Programming, Test and Debugging Interfaces
  - JTAG (IEEE 1149.1 Compliant)
  - PDI (Program and Debug Interface)

● I/O and Packages
- 78 Programmable I/O Lines
- 100 - lead TQFP

● Operating Voltage (MT-X1 operates at 3.3V)
- 1.6 - 3.6V

● Speed performance (MT-X1 can operate at 0-32MHz)
- 0 - 12 MHz @ 1.6 - 3.6V
- 0 - 32 MHz @ 2.7 – 3.6V

Bottom View



MT-X1 Hardware

Solder Jumpers

Jumper	*Description*
J1	USB Shield to gnd (not connected by default)
J2	~5V to relay driver (5V header pin and kickback diodes common cathode)
J3	DACA0 (pin A2) to amplifier audio input
J4	AREF B (pin B0) to 1.25V reference
J5	AREF A (pin A0) to 1.25V reference
J6	ADCA1 (pin A1) to temperature sensor output
J7	External clock input (pin R1) to CLKO from AT90USB162 (8 MHz)
J8	SPI D SS (pin D4) to SRAM chip select (external pullup)
J9	SPI D MOSI (pin D5) to SRAM SI (external pullup)
J10	SPI D SCK (pin D7) to SRAM clock input (external pullup)
J11	SPI D MISO (pin D6) to SRAM SO (external pulldown)
J12	Pin D0 to LED_1
J13	Pin D1 to LED_2
J14	Pin D2 to LED_3
J15	Pin D3 to LED_4
J16	SPI F SCK (pin F7) to relay driver clock input
J17	SPI F MOSI (pin F5) to relay driver SI input
J18	USART F0 RXD (pin F2) to AT90USB162 USART TX
J19	USART F0 TXD (pin F3) to AT90USB162 USART RX (shared with PDI_DATA)
J20	USART F0 XCK (pin F1) to AT90USB162 USART XCK (also USB ready signal)
J21	SPI E MISO (pin E6) to SD card SO (must enable XMEGA pullup)
J22	SPI E SCK (pin E7) to SD card clock input (external pullup)
J23	SPI E MOSI (pin E5) to SD card SI (external pullup)
J24	SPI E SS (pin E4) to SD card chip select (external pullup)

Headers / Pin Descriptions

Pin	Description
External Power Header	Under the default configuration, 5V should be supplied to this pin. Lower voltages may be used down to around 4V (or lower if using less current). **Voltages greater than 5.5V require J2 to be disconnected**. Disconnecting J2 will disable the 5V output pin and inductive kickback protection of the relay driver. But it will allow voltages up to ~7.5V. This header is reverse polarity protected using a schottky diode.
3.3V output headers (x4)	There are four 2-pin power output headers next to each port header group. The header next to the analog ports (ports A and B) comes from the analog 3.3V rail. Note that if these headers are installed, there will not be enough room to plug in IDC connectors next to each other.
Relay Header 3.3V	This can be used for the positive 3.3V side of a relay or other device.
Relay Header 5V	This can be used for the positive 5V side of a relay or other device. This is also the common cathode of the kickback diodes in the relay driver. Both of these are disabled when J2 is diconnected.
Relay Header 1-8	These are the 8 relay driver outputs. They are open-drain active low. When enabled, the output is connected to ground. When disabled, the pin is in a high impedance state. When driving inductive loads, like relays, free-wheeling diodes provide kickback protection (when J2 connected). Non-inductive loads can also be connected (ie: LCD backlight). Each output is capable of sinking 70mA. Outputs can be combined.
Audio Header	This is the singe channel output from the audio amplifier. It can drive 8 ohm loads. 4 ohm loads may also be connected, but under some conditions, distortion or automatic thermal shutdown may occur.
JTAG Header	JTAG header for the XMEGA which can be used for programming, debugging, and JTAG boundary scans. Disable JTAG to gain access to the four underlying analog/GPIO pins.
PDI Header	PDI header for the XMEGA which can be used for programming or debugging. Note that an onboard programmer is already provided. When using this header, **J19 MUST be disconnected.** This is due to the fact that RX and PDI_DATA are shared. This means that the XMEGA serial TX won't be connected to the USB AVR RX. This doesn't affect programming, but may present a problem in certain situations when debugging. If serial TX is required when debugging, the JTAG header can be used. Alternatively, an external USB-serial bridge can be connected.
ISP Header	ISP header for the USB AVR which can be used for programming or debugging. The USB AVR can be programmed over USB using the DFU bootloader.
Port A	All pins are routed to headers. The 1.25V precision reference can be connected to pin A0 (Vref input) through a solder jumper. The temperature sensor output and audio amplifier input are also connected to this port.

Port B	All pins are routed to headers. The 1.25V precision reference can be connected to pin B0 (Vref input) through a solder jumper. Note that JTAG is connected to pins B4 – B7. JTAG must be disabled to use these pins for other purposes like the ADC.
Port C	All pins are routed to headers. No peripheral devices are connected to this port.
Port D	All pins are routed to headers. This port also connects to the LEDs and 32KB SPI SRAM memory through solder jumpers.
Port E	All pins are routed to headers. This port also connects to the buttons and the MicroSD card slot through solder jumpers.
Port F	All pins are routed to headers. This port also connects to the USART of the USB AVR (RX, TX, and optionally, XCK) as well as the SPI inputs of the relay driver (MOSI and SCK) through solder jumpers. To minimize power consumption. TX should be tristated before entering sleep.
Ports H, J, and K	All pins are routed to headers. No peripheral devices are connected to these ports. They can be used for GPIO or for use with external memory.
Pins Q0 and Q1	The 32.768KHz crystal is connected to these pins, which serve as the TOSC input pins of the RTC.
Pins Q2 and Q3	Pin Q2 is routed to the chip select pin of the relay driver. Pin Q3 is routed to the audio amplifier power-down pin. Neither pin is routed to a header.
Pins R0 and R1	Both of these pins are routed to an HC49 crystal footprint. A 22pF capacitor is also connected to each line. Additionally, R1 can be connected to the USB AVR 8MHz clock output (CLKO) through solder jumper J7.

Buttons / Jumper

There are four modes of operation which are selected using the PROG button and JMP jumper. The button and jumper are sampled when powering up or pressing reset. Additionally, the MT-X1 can be switched between the AVRISP mkII programmer and the serial bridge during runtime by pressing the PROG button. This is useful, for example, to program the XMEGA, then switch to the serial bridge for printf() debugging. The reset button resets the USB AVR, which will in turn reset the XMEGA when it boots. The following table lists the mode selection during power-up and reset.

Mode Selection During Power-up and Reset

PROG Button	*JMP Jumper*	*Mode*
Pressed	Installed	DFU Bootloader
Not Pressed	Installed	Configuration Mode
Pressed	Not Installed	AVRISP mkII PDI Programmer
Not Pressed	Not Installed	USB Serial Bridge

Power / Status LEDs

There are two green LEDs that are used to indicate USB status, the mode of operation, communication activity, programmer status, and more. The following table lists LED functionality in each mode. Both LEDs are turned off in sleep mode.

LED Functionality

Mode	*STS LED*	*PWR LED*
AVRISP mkII Programmer	Programmer Activity	PWM pulsing
Configuration Mode	On	On
USB Serial Bridge	RX Activity	TX Activity
DFU Bootloader	On	Off

Power Supply

The MT-X1 can be powered via USB or via an external header. Both sources are connected to the input of a 1A, 3.3V LDO linear regulator through Schottky diodes rated at 2A each (2A was chosen also to keep the dropout voltage low throughout the range of current). The diodes provide reverse-polarity protection as well as ensuring that current will not flow from one source to the other. For example, if the external header has a greater voltage than the USB VBUS voltage, the diode prevents VBUS from rising to the level of the external voltage. **Note that there is a minimum load of 100uA for this regulator.** The MT-X1 can consume less than 75uA in the deepest sleep modes. Use an external load resistor between 3.3V and Gnd to ensure that this requirement is met. A MicroSD card inserted may consume enough to meet the specification without the resistor.

The 3.3V regulator has thermal protection and foldback current limiting. There is a 10uF capacitor on both the input and output. Note that 10uF is the maximum allowed by the USB specification. When using the external header, additional capacitance may be needed with higher impedance voltage sources (ie: batteries, long cable runs). The regulator input can also be routed through J2 to the header pin labeled 5V (near the relay driver). **Voltages greater than 5.5V on the external power input header require J2 to be disconnected, which will disable the relay driver kickback protection.**

Clock Sources / RTC

By default, a 32.768KHz crystal is installed and connected to the TOSC pins of the XMEGA (R0 and R1). An HC49 crystal landing is available as well, with 22pF load capacitors preinstalled. An external clock can also be connected to pin R1. There are several internal clock options as well. The demo program makes use of the 32MHz internal RC oscillator. This oscillator is configured to be auto-calibrated by a DFLL, which uses the 32.768KHz crystal as input. The crystal is also the source for the RTC. A 2MHz RC oscillator and two different 32KHz oscillators are also available. A PLL and prescalers can be used to obtain the various clocks. Be aware that the ATxmega128a1 requires both the 2MHz and 32MHz oscillators to be running and both DFLLs to be enabled for either DFLL to operate due to errata. Atmel ASF (Atmel Software Framework) does not support this arrangement, but the example code shows how to set this up. Also note that the DFLL calibrated oscillators will still not be as accurate as an external high speed crystal. If using an external crystal, it must be 0.4MHz to 16MHz. The PLL can be used to obtain higher clock speeds.

Programming Headers

The PDI header has the standard 6-pin layout. Because an onboard programmer is provided, an external programmer is not necessary. However, debugging requires use of an external debugger connected to the PDI header or the JTAG header. Because the XMEGA TX pin (USB AVR RX/D pin) is connected to the XMEGA PDI_DATA pin, the debugger cannot be used when using TX as this would cause contention. However, solder jumper J19 can be disconnected to avoid contention, but TX will then be disconnected from the USB module. TX will still available on the header pin, so you can connect to an external serial bridge if debugging requires it. Alternatively, the JTAG header can be used for debugging. When using an external debugger or programmer on the PDI header, the USB AVR should be in any mode other than the PDI programmer. An ISP header is available for programming the USB AVR. It can also be programmed over USB (see Firmware Updates).

Solder Jumpers / USB Shield

There are many solder jumpers on the PCB connecting XMEGA pins to the onboard peripheral devices. This allows unused devices to be disconnected, freeing up the XMEGA pin, which is also routed to a header, to be used for other purposes. External pull resistors are installed to keep the peripheral pins at a defined state during boot or when the peripheral is disconnected. They pull chip select lines to the deselected state to minimize power consumption. Most solder jumpers are connected by default. To disconnect for the first time, a small trace connecting the two jumper pads must be cut. To reconnect, create a solder bridge across the pads. Jumper J1 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, it may be desired to ground this on the device side. An 0603 SMT component may be soldered on the solder jumper pads as well.

USB Serial Bridge

The USB Serial bridge allows the XMEGA to communicate with a computer over USB by simply using a USART. There is no need to learn the USB protocol or utilize a USB library. All USB functionality is handled by the USB AVR (AT90USB162). It simply relays bytes between the XMEGA and the host. The MT-X1 uses two pins on USART F (RX and TX) in asynchronous mode and three pins (adding XCK) in synchronous mode. Optionally, a USB ready signal is available on the XCK pin. To minimize power consumption. TX should be tristated before entering sleep. This is due to the sharing of PDI_DATA and TX. PDI_DATA has a pulldown active, which will consume current when TX is set to output high. All three pins can be disconnected from the USB AVR using the solder jumpers.

MicroSD Card

The MicroSD card slot has a spring-loaded mechanism that locks the card in place when inserted (push-in, push-out). The contacts are gold-plated. It is connected to SPI E using four pins. All pins have external 47Kohm pullups installed, except for MISO, which **requires the internal pullup to be enabled**. All four pins can be disconnected from the MicroSD card slot using the solder jumpers. Note that when in the deepest sleep modes and a card is installed, it will likely consume the most current. Since the minimum load required by the regulator is 100uA, and the rest of the onboard components may consume less than 75uA, having a card installed may allow disconnection of the minimum load resistor (solder jumper), which itself consumes 100uA.

32KB SPI SRAM

The 32KB SPI SRAM is the 23K256-I/SN from Microchip. It has a very simple protocol, and can be quite fast operating at 16MHz with sequential access (ie: data capture). It is less suitable for storage that requires random access. It is connected to SPI D using four pins, all of which have 47Kohm external pull resistors. All four pins can be disconnected using the solder jumpers. A simple driver is provided in the ASF template. More more information, consult the datasheet.

Audio Amplifier

The audio amplifier is the LM4889MM/NOPB from National Semiconductor. It is a single channel, class AB, 400mW @ 3.3V amplifier with depop and thermal protection. It is connected to the XMEGA DAC A0 on pin A2, which can be disconnected using a solder jumper. The shutdown pin is routed to pin Q3 and has a 47 Kohm pull resistor to keep the IC in shutdown when Q3 is not driven. The differential gain is set to 2, so the internal 1V reference or the external 1.25V reference can be used. An 8ohm or 16ohm speaker can be connected to the output which is routed to a two pin header. A 4ohm load can also be connected, but the amplifier may enter thermal shutdown if using a higher voltage reference and the signal magnitude remains large for a long enough period of time. For more information on this IC, please consult the datasheet.

1.25V Precision Reference

The 1.25V precision reference is the ISL60002DIH312Z-TK from Intersil. It is a low-power FGA reference with a low 20ppm/C temperature coefficient. The initial accuracy is +/-5mV. Because each MT-X1 board is intended to be calibrated individually, the initial accuracy was deemed less important than the temperature coefficient. The reference is connected to both reference inputs, pins A0 and B0. It can be disconnected using the solder jumpers. The reference voltage of 1.25V was chosen as a workaround to the ADC errata of the ATxmega128a1. It is intended to be used with the ADC in differential mode and with signed conversions. The voltage to be measured is connected to the positive input, and the reference to the negative. This results in conversions in the ~0 to 2.5V range. See the source code for example setup and usage. For more information on this IC, please consult the datasheet.

Temperature Sensor

The temperature sensor is the MCP9701AT-E/TT from Microchip. It is connected to pin A1, and can be disconnected by using a solder jumper. It can sense from -10C to 125C, though the high temperature is limited to the maximum PCB temperature. It has an accuracy of +/- 2C (max.) and it outputs 19.5mV/C. It consumes only 6uA (typ.). For more information on this IC, please consult the datasheet.

8-channel Lowside / Relay Driver

The 8-channel lowside / relay driver is the MAX4820 from Maxim. The outputs are open-drain. Each channel can drive low at 70mA each. Channels can be connected together to increase the current capability. All channels have kickback protection diodes, allowing them to driver relays. Note that solder jumper J2 must be connected for kickback protection to be available. All eight outputs, along with 3.3V (regulated Vcc) and ~5V (external voltage), are routed to a 10-pin header. Thus, devices that use either 3.3V or 5V (ie: 5V relay, 5V LCD backlight) are supported. **Voltages greater than 5.5V on the external power input header require J2 to be disconnected, which will disable the relay driver kickback protection.** The IC is connected to SPI F and can be disconnected using the solder jumpers. Note that MISO is not connected, so the XMEGA cannot read from the IC. Also note that the SPI F SS line is not used as the chip select, but instead, Q2 is used. Thus, it is necessary to configure the SS pin as an output, or enable the pullup and leave it as an input so that SPI will operate as a master. The maximum operating speed is 2MHz. The protocol is very simple; essentially just a shift register. A simple driver is provided in the ASF template. For more information on this IC, please consult the datasheet.

LEDs

There are four LEDs connected to pins D0-D3, and can be disconnected by using the solder jumpers. The LEDs are on when the outputs are high. The LEDs are connected to ground through 249 ohm series resistors.

External 1MB low-power SRAM (optional)

An external 1 MB low-power SRAM module is available separately.

* 8-Mbit (1024 x 8) static RAM
* Cypress CY62158EV30 IC
* 45ns
* 2.2V-3.6V
* 18mA (25mA max) @ max speed
* 2uA (8uA max) when not selected
* -40C to +85C
* 2 latches for address lines (20-bit)
* dedicated 8-bit data lines
* Measures 34.6mm x 25.0mm (1.36" x 0.98")
* Fits EBI header on MT-X1 development board

Windows Installation

Before plugging in the MT-X1 for the first time, the latest software and drivers must be downloaded. The MT-X1 is supported under Windows XP, Vista (32 and 64 bit), and Windows 7 (32 and 64 bit). There is limited support for Windows 2000. The MT-X1 appears as three different devices to the PC depending on which mode is selected by the button and jumper. These devices are the AVRISP mkII compatible programmer, the DFU bootloader for firmware updates of the USB AVR, and the USB CDC device (Virtual COM port) which is used for configuration mode and the USB Serial bridge. Therefore, three drivers are required. The AVRISP mkII and DFU drivers are included with software available on the Atmel website. The CDC driver is included with Windows, but requires an .inf file available on the MattairTech website. The following table lists the minimum versions of the required software. If the software provides a driver, is is listed as well. See the Firmware Updates section for installation of the DFU bootloader driver.

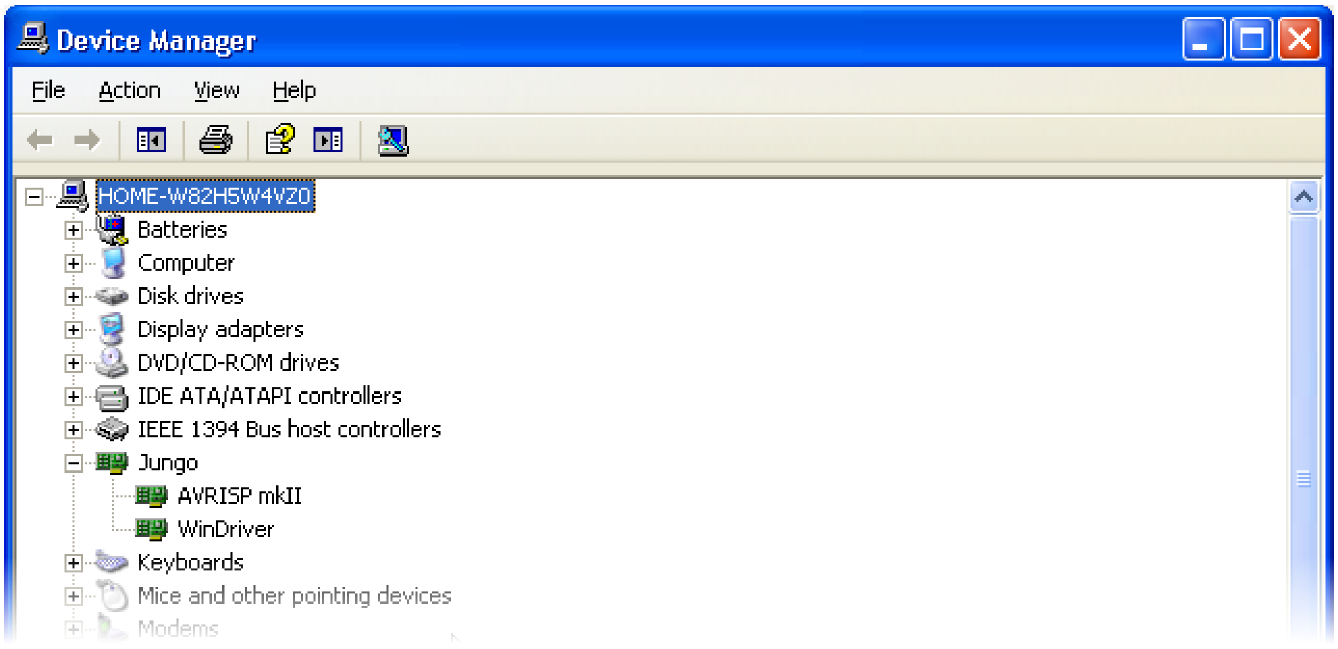
Required Downloads

Software	*Version*	*Driver*	*URL*
AVR Studio / Atmel Studio	4.19, 5.x, or 6.x	AVRISP mkII	http://www.atmel.com/tools/AVRSTUDIO4.aspx OR http://www.atmel.com/tools/ATMELAVRSTUDIO.aspx
MT-X1 Driver	latest	CDC driver	https://www.mattairtech.com/software/MattairTech_CDC_Driver_Signed.zip

Atmel Studio (AVR Studio) / AVRISP mkII driver

Atmel Studio (formerly AVR Studio) is a free IDE provided by Atmel that runs on Windows operating systems. It includes an assembler, debugger, simulator, and an AVR chip programming utility. As of June 2012, there are three main versions supported, the 4.x, 5.x, and 6.x series. The 4.x series is mature and stable, and can run on older hardware, however, it requires the use of the WinAVR gcc toolchain, which is out of date. It also lacks proper support for newer devices, like the XMEGA microcontrollers, but is a good option for older devices. AVR Studio 4.x is also smaller and less demanding on PC resources. If you choose to use the 4.x series, download version 4.19. You will also need to download and install WinAVR 20100110 prior to installation of AVR Studio. Both AVR Studio 5.x and Atmel Studio 6.x are supported by the MT-X1. They includes the compiler toolchain, as well as the AVR Software Framework (ASF). Use of the 5.x or 6.x series is recommended for the best support of the ATxmega128a1. Whichever version you choose, be sure to install the Jungo drivers when asked, which include the AVRISP mkII driver needed by the MT-X1 AVR programmer.

Once Atmel Studio is installed, remove jumper JMP and plug in or reset the MT-X1 while holding down the PROG button. This will run the AVRISP mkII compatible PDI programmer. LED_STS should be lit and LED_PWR should be PWM flashing on and off. You will then be prompted for the AVRISP mkII driver. By default, this is located in the Program Files/Atmel/AVR Jungo USB directory. Point the installer to the appropriate subdirectory for your PC architecture (usb32 or usb64) and install the driver. Do not use the driver in the AVR Tools/usb directory. Once the driver is loaded, the device will appear as the AVRISP mkII device under Jungo in the device manager.



WinAVR / AVRDUDE

WinAVR contains the GNU GCC compiler for C and C++, compiler tools, and libraries (including AVR Libc). It also includes AVRDUDE for Windows, which is a command line tool for transferring firmware to AVR microcontrollers. A graphical tool is included with AVR Studio. Download WinAVR from http://sourceforge.net/projects/winavr/files/WinAVR/20100110/ and install it first. To use AVRDUDE, you will need to download and install an update to libusb-win32 available at http://sourceforge.net/projects/libusb-win32/files/libusb-win32-releases/. Choose the libusb-win32-devel-filter-x.x.x.x.exe file. Do this only after installing AVR Studio. You will also need to change the MT-X1 AVRISP mkII Programmer host configuration to AVRDUDE. Note that WinAVR is outdated. It is not recommended for newer devices like the XMEGA series. AVRDUDE can also be installed separately.

MT-X1 Driver / Serial Configuration

Next, the MT-X1 CDC driver can be installed, which is used by the serial bridge and configuration mode. This 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. Now, plug in or reset the MT-X1 with jumper JMP removed. This will run the USB-serial bridge. Both LEDs should be lit. Windows will then prompt you for the MT-X1 CDC driver. Point the installer to the directory where you downloaded the driver and install. Note that you may need to rename the driver in order for it to show up in the installer. Windows may add the .txt extension to the file after downloading. Rename it so that it ends with .inf. Ignore any warnings given by the installer (ie: unsigned driver). Once the driver is loaded, the device will appear as the MT-X1 CDC device using a COM port in the device manager. There is no need to configure serial port parameters. The buad rate, for example, is ignored. The MT-X1 will always communicate with the computer at full speed (up to 2Mbps). If you experience any buffering problems, for example, a delayed response to user input, then change both buffer sizes to 1.

Terminal Emulator

Finally, the terminal emulator can be configured. Windows XP includes HyperTerminal, which has been tested with the MT-X1 and will be documented here. There are several other terminal emulators available freely on the Internet. If you wish to use any of them, it should be no trouble to adapt the instructions presented here.

Next, start HyperTerminal. Create a new connection. You will refer to this connection again, so give it an appropriate name (after it is configured, you can copy it to your desktop). Select the MT-X1 COM port (ie: COM4) and continue. It is not necessary to configure the baud rate or any other serial parameters. Now, click on the connect icon.

After connecting, you may see garbage on the terminal screen. If this is the case, click on the configuration icon and change the emulation to ANSI (or ANSIW). The configuration mode requires an ANSI terminal to allow drawing of the menu system. Normally, when first entering a mode that uses the CDC driver, a message that reads “Press any Key” is printed periodically. If you do not see this message, just press any key to continue. Note that it may not be possible to switch between modes using the button until a key is pressed.

It is important to always click the disconnect icon before switching to the PDI programmer. Then click the connect icon a couple seconds after returning. This is required because changing to the AVRISP mkII driver unloads the CDC driver, then loads the AVRISP mkII driver. In order for the terminal to use the same COM port as before, it must be disconnected when returning to the CDC driver so that it does not assign a new COM port.

Linux Installation

Linux is supported as well. You must download and build the toolchain from the latest script available at AVR Freaks on the AVR GCC Forum (Script for building AVR GCC sticky at http://www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&t=42631). All firmware written for the MT-X1S is developed under Linux using this toolchain.

Drivers

TODO (drivers should already be installed)

GCC Toolchain

TODO (see opening paragraph)

AVRDUDE

TODO (ie: avrdude -p x128a1 -c avrisp2 -P usb -U flash:w:"myfirmware.hex")

dfu-programmer

TODO (must use version 0.5.2 or higher)

Terminal Emulator

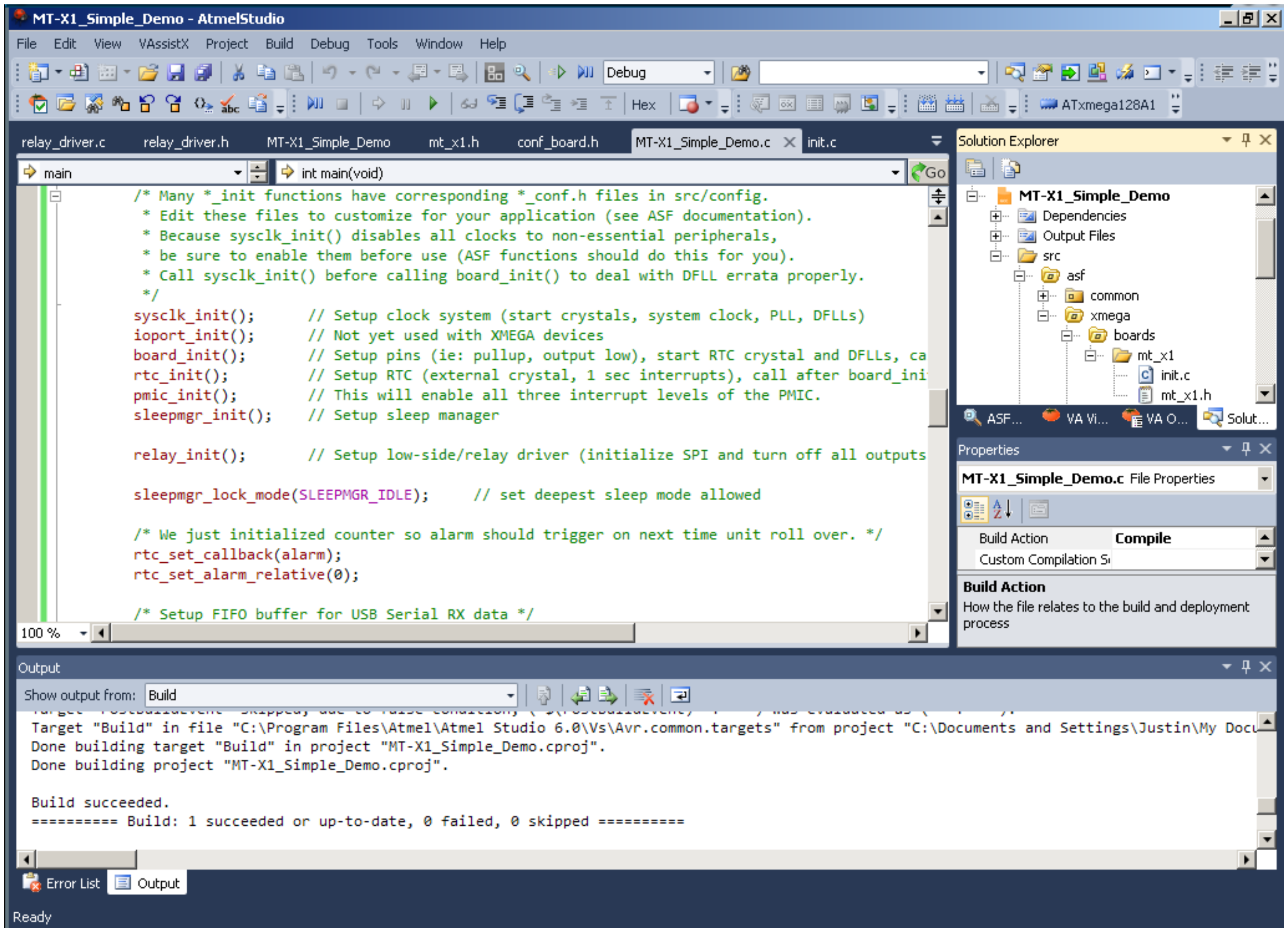
TODO (can use minicom, config port (ie: /dev/tty/ACM0), save config, run with minicom -o)

AVRISP mkII Compatible PDI Programmer

The MT-X1 onboard PDI Programmer is based on the AVRISP mkII compatible programmer written by Dean Camera (http://www.fourwalledcubicle.com/). AVR Studio 4.19, 5.x, Atmel Studio 6.x, and AVRDUDE are supported.

Using Atmel Studio (AVR Studio)

Start Atmel Studio and open or create a new project. An example project, which can be used as a template, is available for the MT-X1 at http://www.mattairtech.com/software/MT-X1/MT-X1_Simple_Demo.zip. To install, click File->Import->Project Template. Once installed, open the template by clicking File->New->Project and selecting the MT-X1_Simple_Demo. Once loaded, you can read the main source file. Also have a look at the src/config and src/asf/xmega/boards/mt-x1 directories using the solution explorer pane. You may also wish to view the toolchain options with Project->Properties.


MT-X1_Simple_Demo - AtmelStudio
File Edit View VAssistX Project Build Debug Tools Window Help
Debug
Hex
ATxmega128A1
relay_driver.c relay_driver.h MT-X1_Simple_Demo mt_x1.h conf_board.h MT-X1_Simple_Demo.c init.c
main
int main(void)
Go
Solution Explorer
MT-X1_Simple_Demo
Dependencies
Output Files
src
asf
common
xmega
boards
mt_x1
init.c
mt_x1.h
ASF... VA Vi... VA O... Solut...
Properties
MT-X1_Simple_Demo.c File Properties
100 %
Output
Show output from: Build
Error List Output
Ready

```
/* Many *_init functions have corresponding *_conf.h files in src/config.
 * Edit these files to customize for your application (see ASF documentation).
 * Because sysclk_init() disables all clocks to non-essential peripherals,
 * be sure to enable them before use (ASF functions should do this for you).
 * Call sysclk_init() before calling board_init() to deal with DFLL errata properly.
 */
sysclk_init();      // Setup clock system (start crystals, system clock, PLL, DFLLs)
ioport_init();      // Not yet used with XMEGA devices
board_init();       // Setup pins (ie: pullup, output low), start RTC crystal and DFLLs, ca
rtc_init();         // Setup RTC (external crystal, 1 sec interrupts), call after board_ini
pmic_init();        // This will enable all three interrupt levels of the PMIC.
sleepmgr_init();    // Setup sleep manager

relay_init();       // Setup low-side/relay driver (initialize SPI and turn off all outputs

sleepmgr_lock_mode(SLEEPMGR_IDLE);     // set deepest sleep mode allowed

/* We just initialized counter so alarm should trigger on next time unit roll over. */
rtc_set_callback(alarm);
rtc_set_alarm_relative(0);

/* Setup FIFO buffer for USB Serial RX data */
```

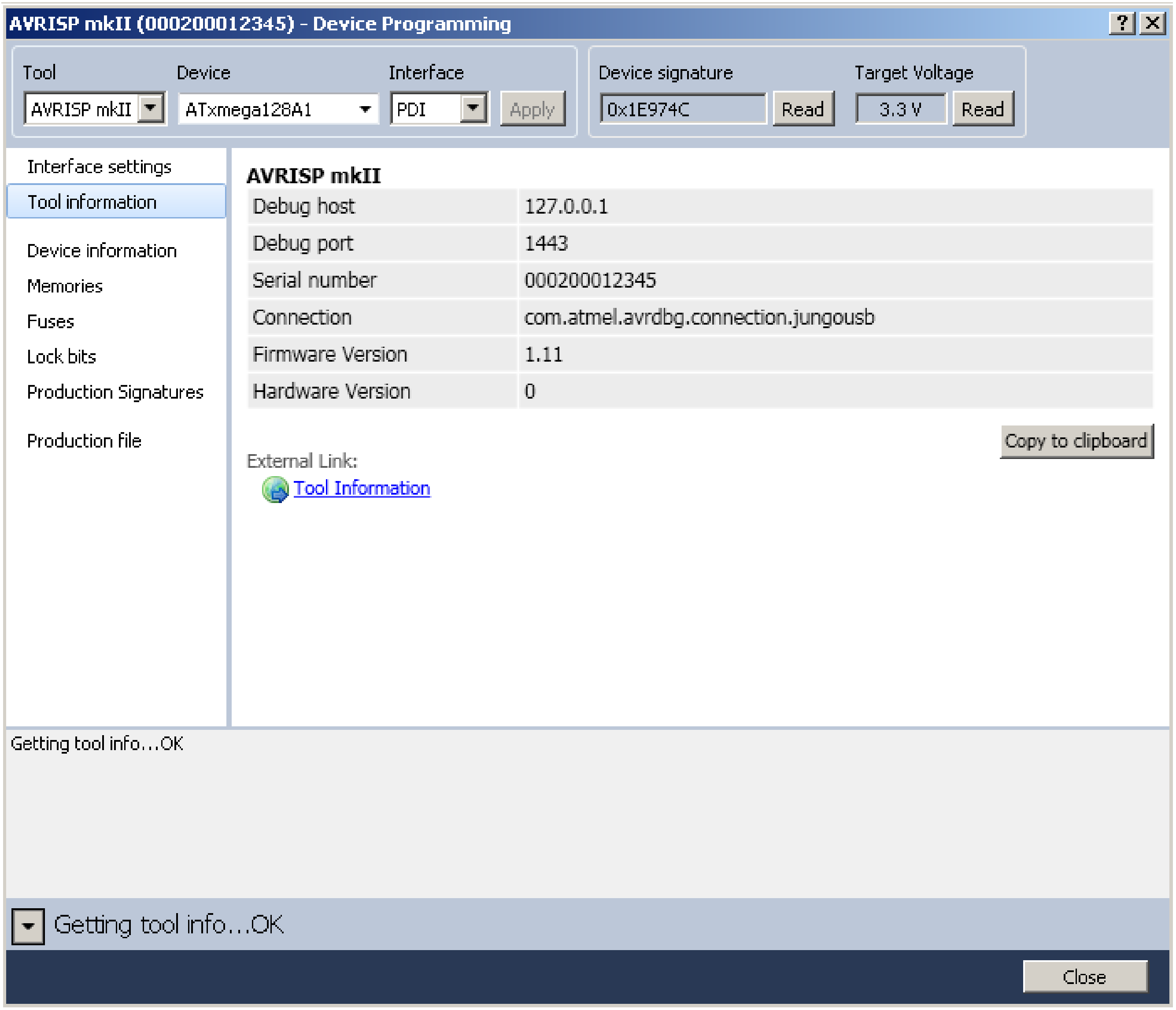
Build Action	Compile
Custom Compilation S	

Build Action
How the file relates to the build and deployment process

```
Target "Build" in file "C:\Program Files\Atmel\Atmel Studio 6.0\Vs\Avr.common.targets" from project "C:\Documents and Settings\Justin\My Docu
Done building target "Build" in project "MT-X1_Simple_Demo.cproj".
Done building project "MT-X1_Simple_Demo.cproj".

Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Next, build the project. Then, click on the Device Programming button. In the Device Programming window, select the AVRISP mkII as the tool. If no tool appears, be sure that the MT-X1 is plugged in and in programming mode (STS LED will be pulsing). Select the ATxmega128A1 as the device and PDI as the interface and click Apply. You should now be connected to the AVRISP mkII compatible programmer with serial number 000200012345. Now click Read next to Device signature. It should match the device if all is well. It is recommended to always perform this step first to verify the connection. The target voltage will always read 3.3V.



Tool: AVRISP mkII
Device: ATxmega128A1
Interface: PDI
Device signature: 0x1E974C
Target Voltage: 3.3 V

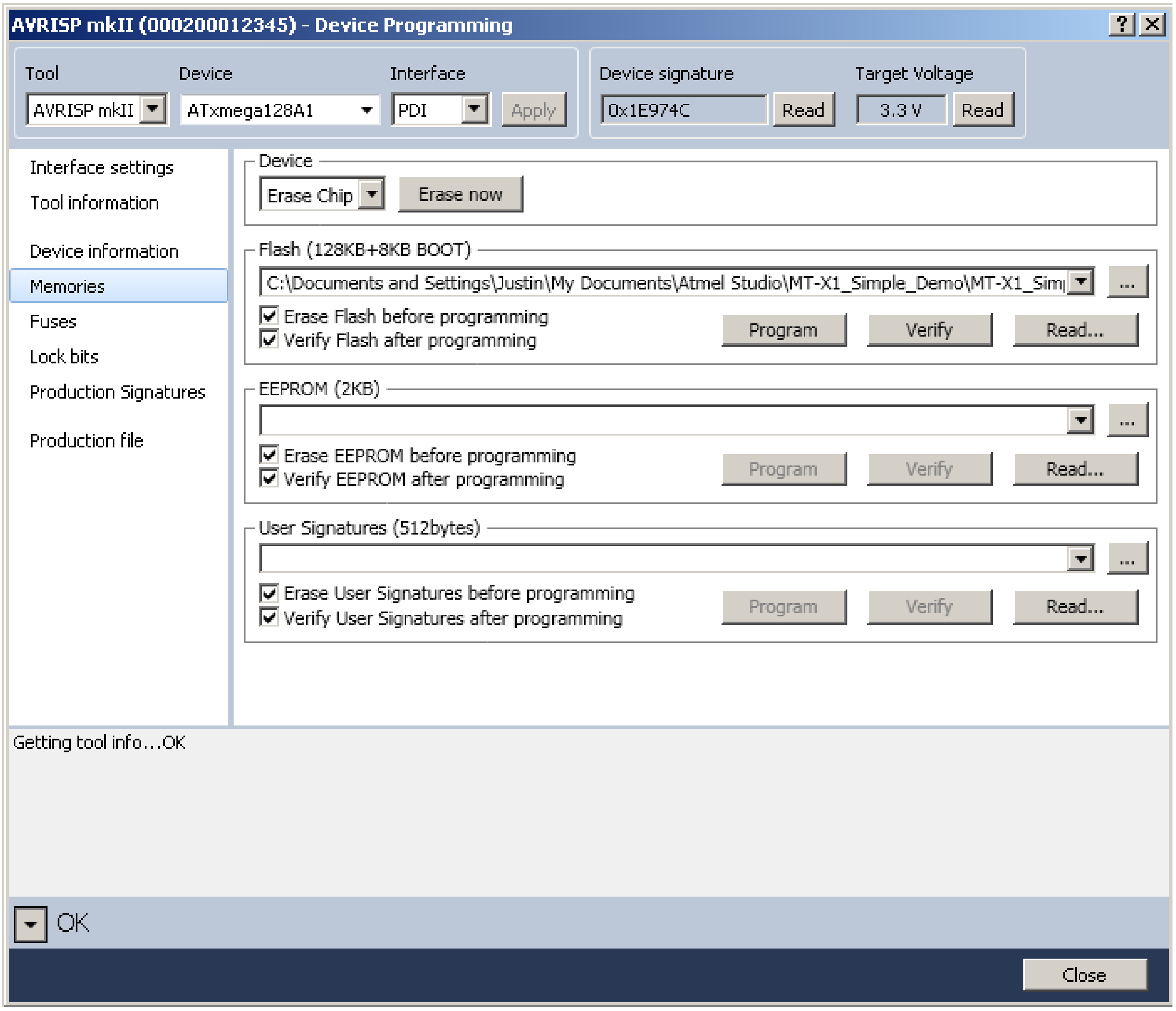
AVRISP mkII

Debug host	127.0.0.1
Debug port	1443
Serial number	000200012345
Connection	com.atmel.avrdbg.connection.jungousb
Firmware Version	1.11
Hardware Version	0

Getting tool info...OK

Getting tool info...OK

Next, select the Memories page. In the Flash section, a hex file can programmed into the targets flash memory. Load your hex file, then click Program. The hex file for the MT-X1_Simple_Demo is located in the Debug folder. You will need to erase the target first if you do not have “Erase Flash before programming” checked. You should also verify the flash as well.


AVRISP mkII (000200012345) - Device Programming

Tool: AVRISP mkII
Device: ATxmega128A1
Interface: PDI
Device signature: 0x1E974C
Target Voltage: 3.3 V

Flash (128KB+8KB BOOT)
C:\Documents and Settings\Justin\My Documents\Atmel Studio\MT-X1_Simple_Demo\MT-X1_Sim
☑ Erase Flash before programming
☑ Verify Flash after programming

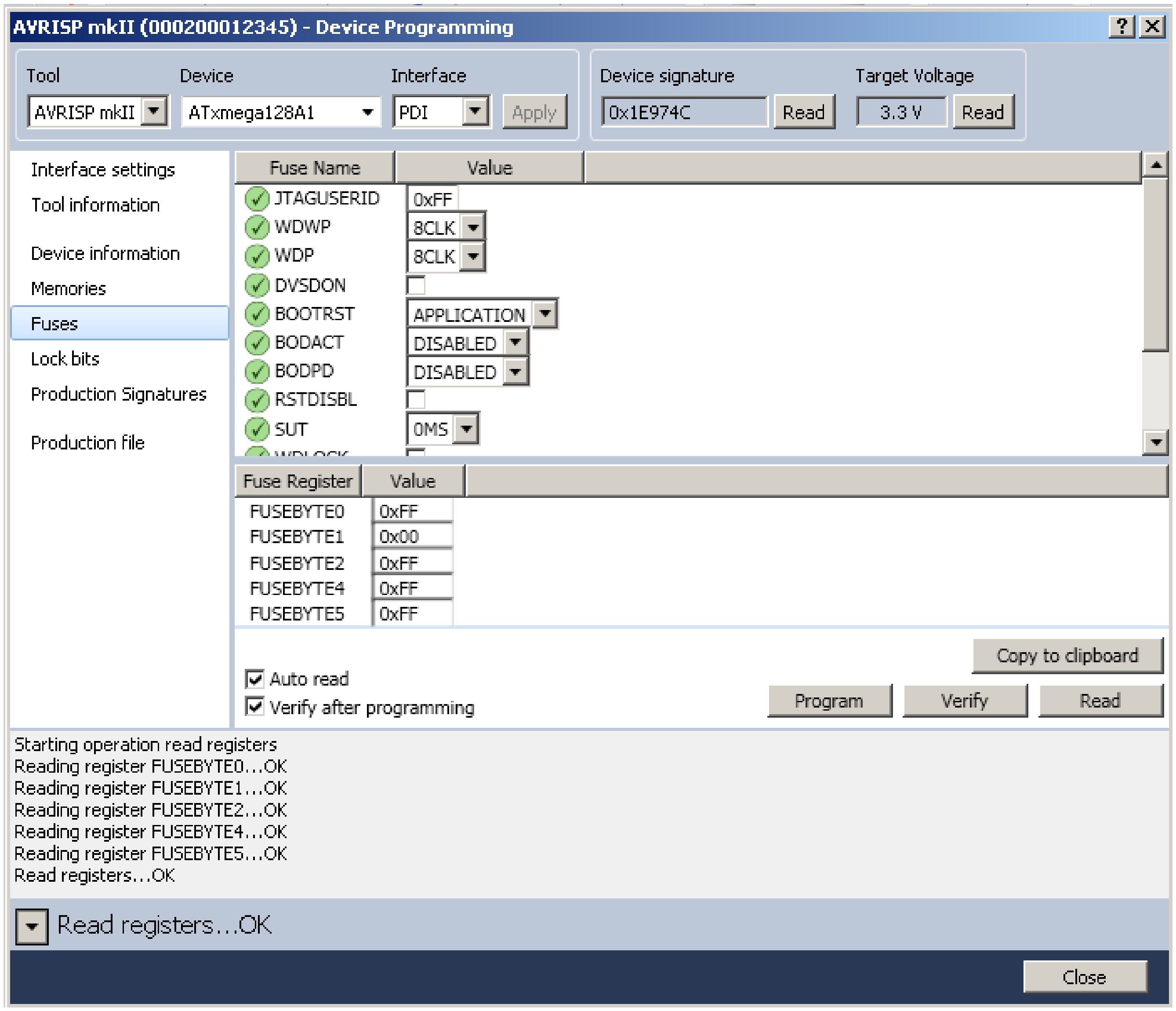
EEPROM (2KB)
☑ Erase EEPROM before programming
☑ Verify EEPROM after programming

User Signatures (512bytes)
☑ Erase User Signatures before programming
☑ Verify User Signatures after programming

```
Getting tool info...OK
```

OK

Next, select the Fuses page. It is best to leave the fuse settings alone until you understand what they do. In particular, do not set the BOD (Brown-out detection) voltage too close to 3.3V, as this could cause the target to be held perpetually in reset. Due to errata, the BOD should not be enabled in sampled mode when active or idle (BODACT). Sampled mode is OK for other sleep modes (BODPD).



Fuse Name	Value
JTAGUSERID	0xFF
WDWP	8CLK
WDP	8CLK
DVSDON	☐
BOOTRST	APPLICATION
BODACT	DISABLED
BODPD	DISABLED
RSTDISBL	☐
SUT	0MS

Fuse Register	Value
FUSEBYTE0	0xFF
FUSEBYTE1	0x00
FUSEBYTE2	0xFF
FUSEBYTE4	0xFF
FUSEBYTE5	0xFF

```
Starting operation read registers
Reading register FUSEBYTE0...OK
Reading register FUSEBYTE1...OK
Reading register FUSEBYTE2...OK
Reading register FUSEBYTE4...OK
Reading register FUSEBYTE5...OK
Read registers...OK
```

Read registers...OK

Now you may wish to look at the other pages. Note that any firmware upgrade feature should not be used. The MT-X1 PDI Programmer is not an actual AVRISP mkII, it just emulates one, so you should not attempt to update the MT-X1 firmware using Atmel Studio. Any firmware updates will be posted to the website and loaded using FLIP or dfu-programmer.

Using AVRDUDE

TODO (ie: avrdude -p x128a1 -c avrisp2 -P usb -U flash:w:"myfirmware.hex")

Serial Bridge

The serial bridge can connect the XMEGA to a host application (ie: terminal emulator) over USB. The XMEGA simply uses one USART as it would with, for example, RS-232. There is no need to learn the USB protocol or use a USB library. On the host side, the MT-X1 will appear as a virtual COM port. Speeds of up to 2Mbps are supported.

Configuration

Before using the serial bridge, it must be configured to be compatible with the target. This configuration is stored in EEPROM. There is no need to duplicate the settings on the host side, as communication between the host and MT- X1 will always be the maximum supported USB speed, and the other parameters are ignored by the host. Only the connection between the USB AVR and the XMEGA use these settings. Note that when configuring the speed to be manual, it is possible to set the speed higher than 2MHz, but the maximum speed supported by the USB link is 2MHz. The serial bridge is configured in configuration mode (jumper on, button not pressed).

Serial Bridge Configuration Options

Configuration Option	*Possible Values*
Speed	2M, 1M, 500K, 250K, 125K, 76.8K, 57.6K, 38.4K, 19.2K, 9600, 2400, manual
Baud Rate Register	0x0000 - 0x0FFF (if manual selected as speed)
Clock 2X	1X, 2X
Clock Mode	async, sync
Data Bits	5, 6, 7, 8, 9

When in synchronous mode, the USB AVR is the master, so the XCK pin is enabled as an output. The XMEGA must enable its clock pin as an input and be configured as a slave. When using 9-bit data frames, two bytes are sent or received for every frame. The first byte simply contains the 9th bit, thus the first byte will always be 0 or 1. The second byte contains the rest of the 8 bits.

Baud Rate Register Value (Manual Speed)

Async 1X	Async 2X	Synchronous
$UBRR=\frac{f_{osc}}{16*BAUD}-1$	$UBRR=\frac{f_{osc}}{8*BAUD}-1$	$UBRR=\frac{f_{osc}}{2*BAUD}-1$
$BAUD=\frac{f_{osc}}{16*(UBRR+1)}$	$BAUD=\frac{f_{osc}}{8*(UBRR+1)}$	$BAUD=\frac{f_{osc}}{2*(UBRR+1)}$

where $f_{osc}=8000000$

Configuration

The MT-X1 PDI programmer, serial bridge, and other features can be configured by entering configuration mode. This configuration is stored in non-volatile EEPROM memory. Configuration mode requires an ANSI terminal emulator. Configuration options are highlighted by using the up and down arrow keys, and selected using the enter key. Some dialogs are for entering numbers in hexadecimal. Here, the left and right arrow keys and backspace are used. The menu system is structured as follows:

- **Serial Speed** (Serial bridge speed selection)
  - List of selectable speeds: 2400, 9600, 19.2K, 38.4K, 50.0K, 76.8K, 125K, 250K, 500K, 1M, 2M
  - Manual (when selected, configure using Manual Settings below)
- **Manual Settings**
  - Baud Rate Register (enter value in hex)
  - Clock 2X (async mode only)
- **Serial Mode**
  - Asynchronous or synchronous
- **Sleep Mode** (Which sleep mode is used when USB is disconnected or suspended)
  - Power Down or Standby
- **Ready Signal** (USB ready signal is open-drain active low on XCK pin from USB AVR)
  - Disabled or Enabled
- **AVRISPmkII** (select which software will be interfacing with the MT-X1 PDI programmer)
  - AVR Studio or AVRDUDE
- **Credits** (displays list of firmware authors)

The USB AVR automatically enters sleep mode when the USB cable is disconnected or the USB bus is suspended. Sleep mode is by default set to Power Down, which provides for the lowest current consumption.

The USB ready signal is useful when the XMEGA needs to know when the USB cable is disconnected or the USB bus suspended. The signal is open-drain active-low from the USB AVR XCK line, which may also be used for synchronous serial operation. The XMEGA must enable the pullup on this line before reading it. If it reads low, USB is enumerated and ready. Otherwise, it will read high. If synchronous operation in used, the XCK clock signal will override this. However, when USB is disconnected or suspended, the clock will stop and the the line driven low.

When changing serial speeds, be sure to also change the speed in the XMEGA.

XMEGA Demo Program

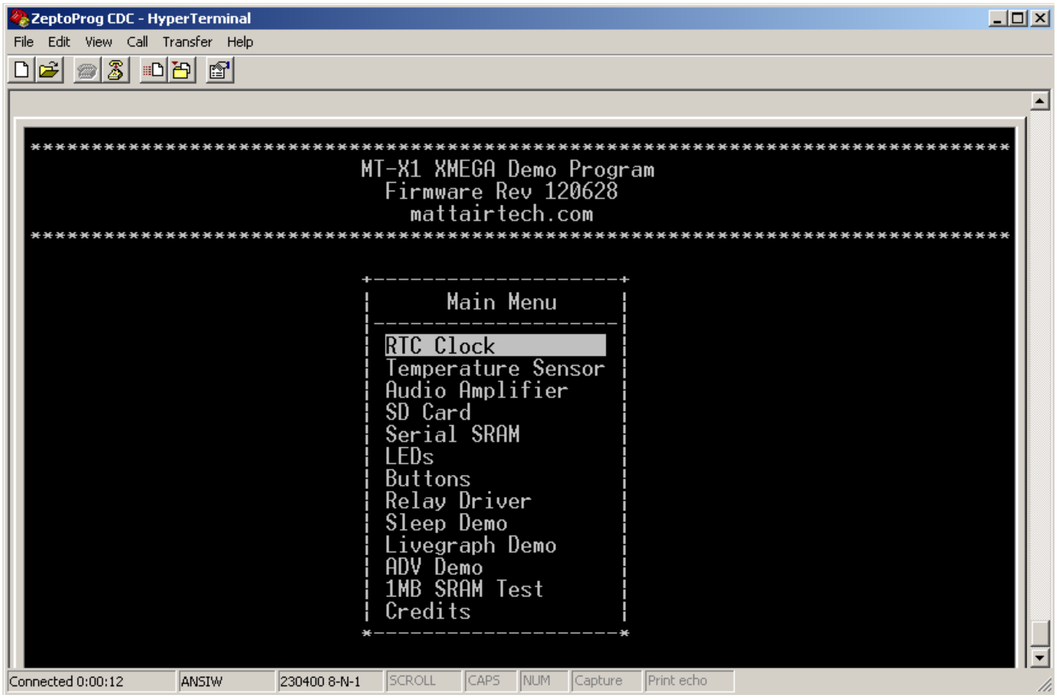
Note: This page describes the preinstalled demo program. This program is primarily meant for testing. While the source code is available, it is not intended to be reused. The code is a collection of many code fragments. It is disorganized and complicated. It is recommended to use the Atmel Studio 6 ASF (Atmel Software Framework) template to get started (see Using Atmel Studio (AVR Studio)). The following relates to the preinstalled demo.

XMEGA uses serial bridge to display menu system on ANSI compatible terminal emulator screen. Remove the jumper and boot the board without pressing the button (remember that the terminal must be disconnected while resetting). Press any key, and the main demo menu will appear. Now, all interaction is with the XMEGA via the USB serial bridge.

MicroSD Card Demo: The SD card demo makes use of the FatFS module from ChaN. FAT12, FAT16, and FAT32 are supported. Press 'h' for a help menu.

Audio Demo: You can load wav files from the SD card and play them over an ~8 ohm speaker connected to the audio amplifier. You will need to use an audio program (like sox), to encode music or sound to 8-bit or 16-bit (recommended), 44.1KHz or less, mono uncompressed PCM. Then, in the audio demo, select the file chooser and pick a file. You can use the slider to adjust volume.

When using the sleep demo, current measurements can be made using the external power header. Disconnect J2 to allow voltages above 5.5V. Supply 6.0V – 7.5V to this header so that current is drawn from this connector rather than from USB. Running the demo will put the XMEGA to sleep with the RTC running and waking the cpu every second to update the time. Most current consumption will then be from the USB AVR. Now unplug USB. The current consumption will drop further. Currently, ~75uA is consumed by the board in this state (without MicroSD card installed). Not all low power features are used, so this number can be lowered further. But note that a minimum load of 100uA should be present on the regulator output. The measured current at the external connector includes the regulator ground current, which will add to the 100uA minimum load.


ZeptoProg CDC - HyperTerminal
File Edit View Call Transfer Help

```
**********************************************************************************
                         MT-X1 XMEGA Demo Program
                          Firmware Rev 120628
                            mattairtech.com
**********************************************************************************

                         +---------------------+
                         |      Main Menu      |
                         |---------------------|
                         | RTC Clock           |
                         | Temperature Sensor  |
                         | Audio Amplifier     |
                         | SD Card             |
                         | Serial SRAM         |
                         | LEDs                |
                         | Buttons             |
                         | Relay Driver        |
                         | Sleep Demo          |
                         | Livegraph Demo      |
                         | ADV Demo            |
                         | 1MB SRAM Test       |
                         | Credits             |
                         *---------------------*
```

Connected 0:00:12 | ANSIW | 230400 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

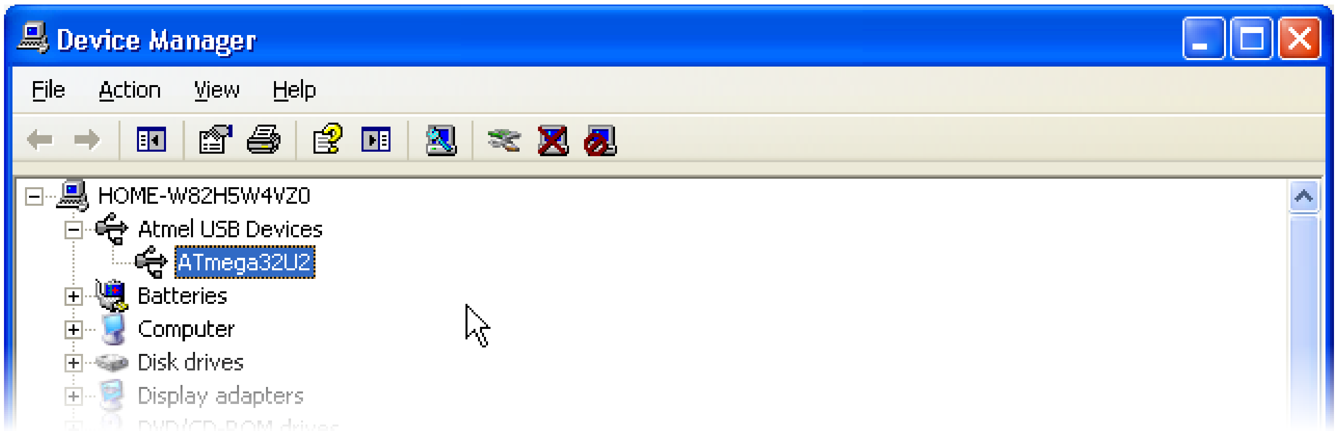
Firmware Updates

The MT-X1 firmware will be updated periodically to add new features and fix bugs. These updates will be available on the MattairTech website. The updates may include just a hex file (for programming flash), or both a hex file and eep file (for programming both flash and EEPROM). FLIP is a graphical utility for Windows used to load firmware updates onto the MT-X1. FLIP includes the DFU bootloader driver. Download FLIP 3.4.2 or higher from http://www.atmel.com/tools/FLIP.aspx and install. If required to install a signed driver, then consult the table below for the download link.

Downloads required for Firmware Updates

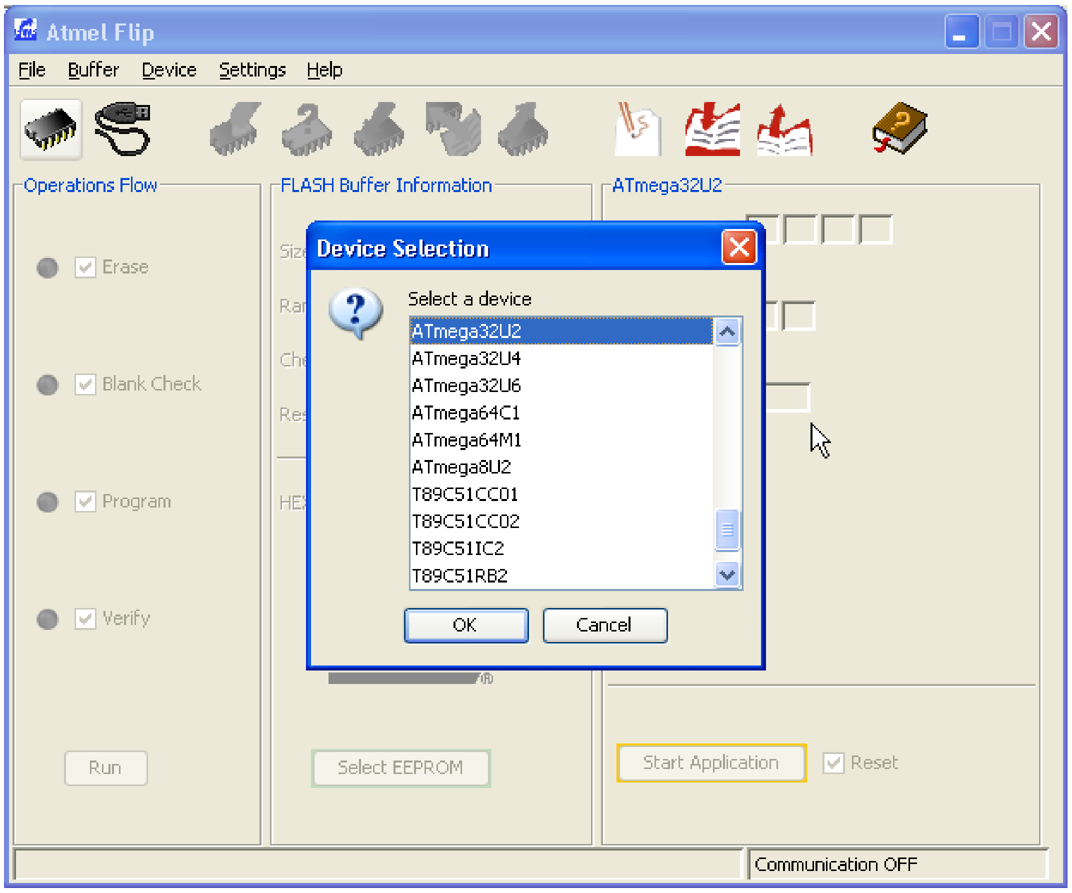
Software	*Version*	*Driver*	*URL*
MT-X1 Firmware	latest (At90USB162)	N/A	http://www.mattairtech.com/software/MT-X1/MT_X1.hex
FLIP	3.4.2 +	DFU driver	http://www.atmel.com/tools/FLIP.aspx
Signed DFU Driver	latest	DFU driver	http://www.avrfreaks.net/index.php?module=Freaks%20Academy&func=viewItem&item_type=project&item_id=2196

Once FLIP is installed, the DFU bootloader driver can be loaded. Plug in the MT-X1 with jumper JMP installed and while holding down the PROG button. This will enter the DFU bootloader. LED_STS should be on and LED_PWR should be off. Windows will then prompt you for the AT90USB162 driver. By default, this is located in the Program Files/Atmel/Flip 3.4.2/usb directory. Once the driver is loaded, the device will appear as the AT90USB162 device under Atmel USB Devices in the device manager (note that ATmega32U2 is shown in screenshots).

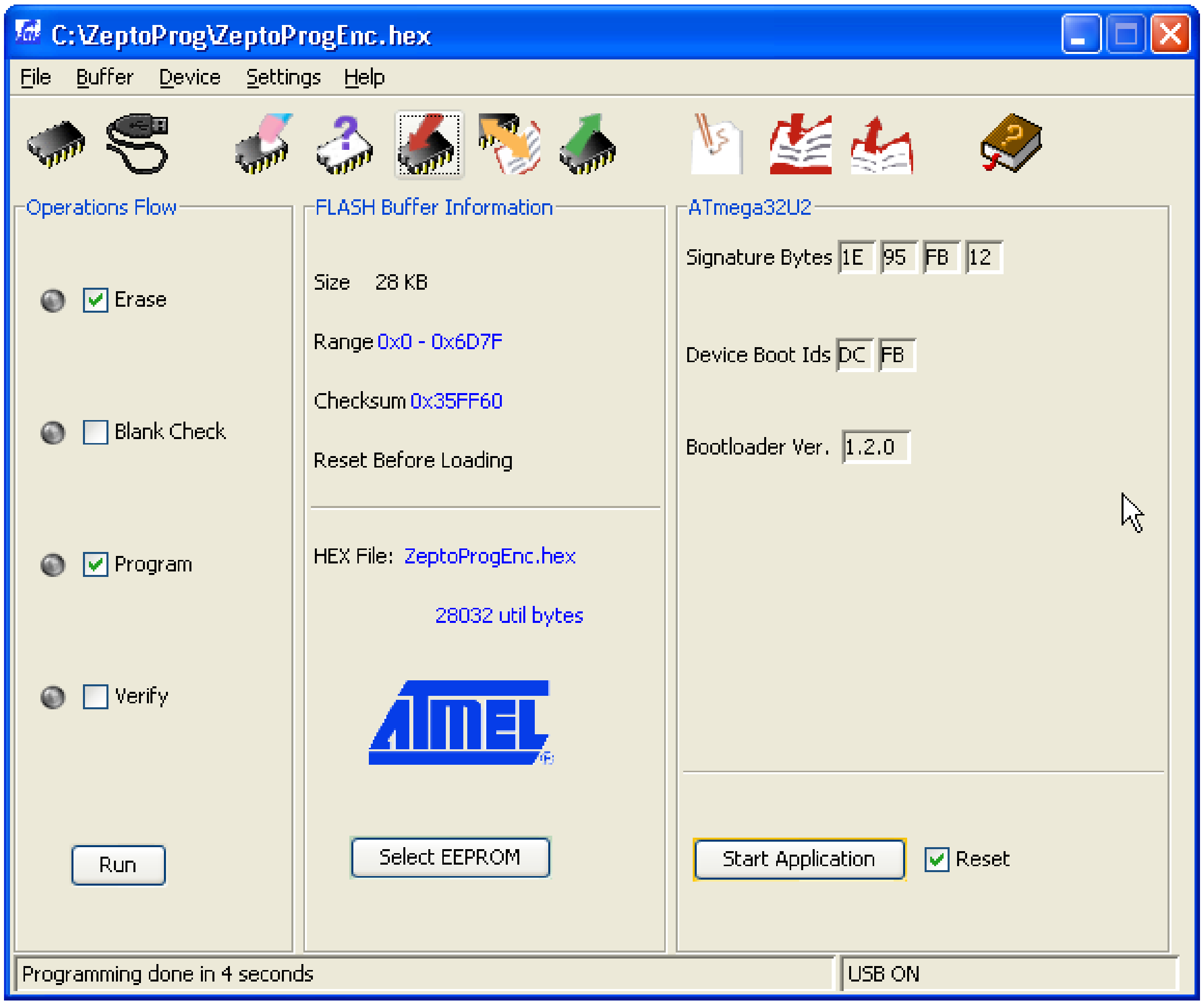


FLIP

Plug in the MT-X1 with jumper JMP installed and while holding down the PROG button. This will enter the DFU bootloader. LED_STS should be on and LED_PWR should be off. Now launch the FLIP utility. When it has loaded, click on the chip icon and select the AT90USB162.



Next, click on the USB icon, select USB, then connect. The screen should now show information about the AT90USB162. 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 MT-X1 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 MT-X1. If you encounter problems, you will need to unplug the MT-X1, disconnect FLIP, and start over making certain that the above settings are observed.



Size 28 KB

Range 0x0 - 0x6D7F

Checksum 0x35FF60

Reset Before Loading

HEX File: ZeptoProgEnc.hex

28032 util bytes

Signature Bytes 1E 95 FB 12

Device Boot Ids DC FB

Bootloader Ver. 1.2.0

Programming done in 4 seconds

USB ON

You may also need to 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.

dfu-programmer

TODO

Must erase chip first. Cannot read flash.

```
dfu-programmer at90usb162 erase
dfu-programmer at90usb162 flash-eeprom MT_X1.eep (if applicable)
dfu-programmer at90usb162 flash MT_X1.hex
```

Troubleshooting / FAQ

- AVRDUDE 6.x does not yet support the MT-X1S. A working patched version can be found at http://www.mattairtech.com/software/avrdude_6.0.1_patched_windows.zip. Thanks to Larry Viesse. For support on Linux 64-bit, download http://www.mattairtech.com/software/avrdude_6.0.1_patched_Linux_64.zip.
  For support on other Linux (especially with xhci (USB 3.0)), replace the usb_libusb.c file from 6.0.1 with http://www.mattairtech.com/software/usb_libusb.c.

- If you are having problems communicating with the programmer using Atmel Studio 6.x, download the Zadig USB driver manager at http://zadig.akeo.ie/.
  Under options, List All Devices. The AVRISP mkII should show up in the list. Replace the current driver with libusb-win32 (v1.2.6.0), which comes embedded with Zadig. Alternatively, please use the procedure at https://www.olimex.com/forum/index.php?topic=4188.0

Support Information

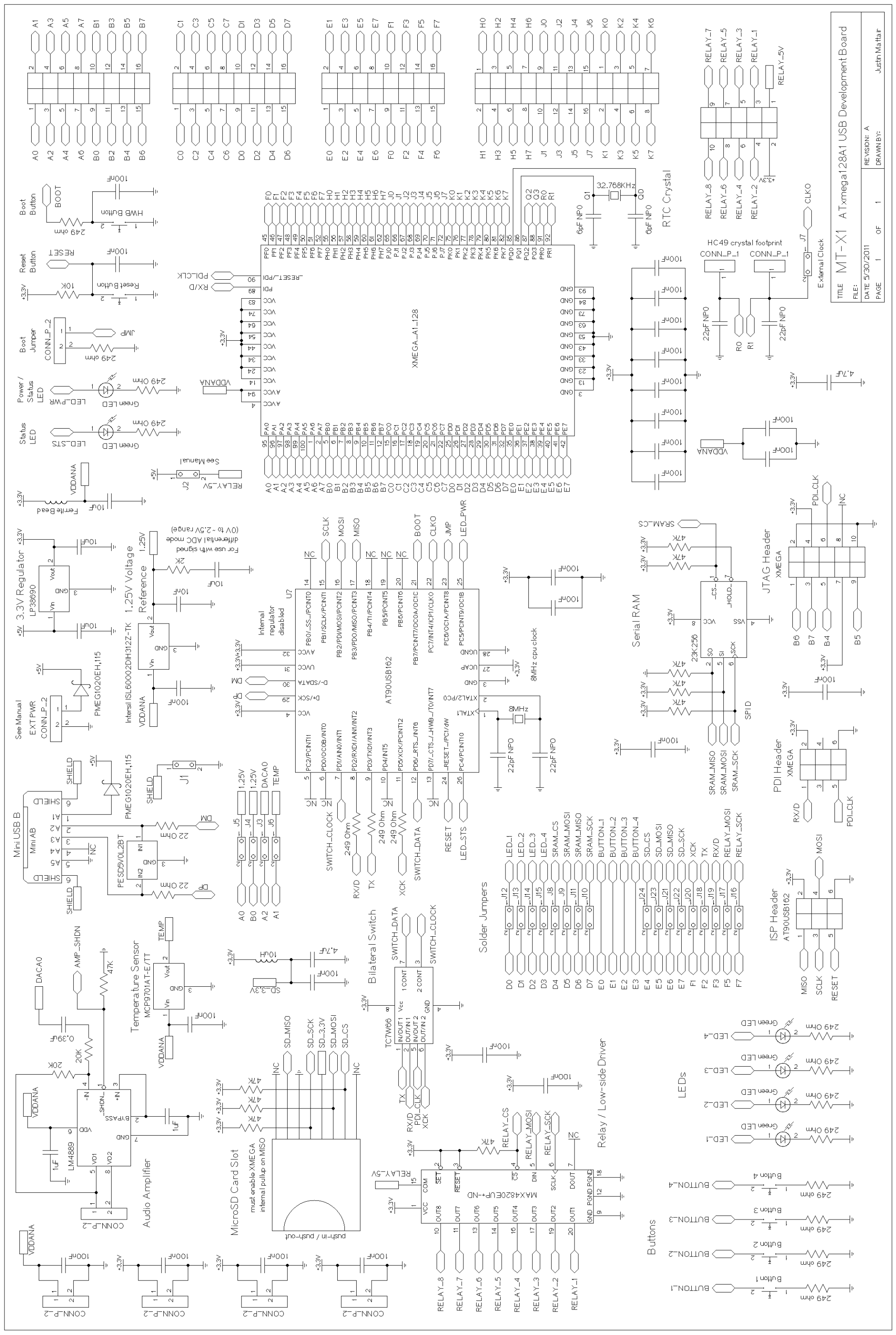
Please check the MattairTech website (http://www.MattairTech.com/) for firmware and software updates. Email me if you have any feature requests, suggestions, or if you have found a bug. If you need support, please contact me (email is best). You can also find support information at the MattairTech website. 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, AVRISP mkII clone, and DFU bootloader, all of which are used in the MT-X1 firmware. 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.

Schematic



MT-X1 ATxmega128A1 USB Development Board
Justin Mattair
XMEGA_A1_128
AT90USB162
RTC Crystal
32.768KHz
3.3V Regulator
1.25V Voltage Reference
Serial RAM
JTAG Header
PDI Header
ISP Header
Solder Jumpers
Bilateral Switch
Temperature Sensor
Audio Amplifier
MicroSD Card Slot
Relay / Low-side Driver
LEDs
Buttons
Mini USB B
Boot Button
Reset Button
Boot Jumper
Power / Status LED
Status LED
8MHz cpu clock
External Clock

Legal Notices

Copyright Notices

Software Disclaimer

Hardware Disclaimer

Trademarks

Licenses

LUFA USB Library

Appendix A: Precautions

CAUTION
The MT-X1 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.