

MXMEXB Document

Overview

Mini XMega Experiment Board (MXMEXB) is a very simple starter kit for developing code and making small prototypes with the Atmel ATXMega microcontroller. It is very easy to try out different kinds of sensor, components and extensions by utilizing the integrated breadboard.

Features

- Atmel ATXMega32A4U microcontroller
- USB interface for downloading user programs and for serial interface
- Powered from USB bus protected with a fuse and a diode
- PDI-programming and debugging interface
- 4 push-buttons
- 4 LEDs
- 170 connection point breadboard for prototyping
- I/O ports accessible through pin header connectors (excluding those connected to switches and LEDs)
- Pre-programmed boot loader and a default test software
- Easily reprogrammed with USB cable and the pre-programmed boot loader
- Debugging and programming via PDI-interface with Atmel JTAGICE mkII, AVR Dragon, JTAGICE3, and AVR ONE! debugging tools.
- Can be optionally programmed with Atmel ISP mkII In-System Programmer

Quick Start

You will need a mini-USB cable connected between the MXMEXB and a PC. Power is applied by the USB connector and the PWR LED on the board will light up.

What is needed to test functioning of the MXMEXB?

- When you connect USB cable the pre-programmed test program will start automatically. Try the push buttons on the card and watch the test programs response from LEDs.

What is needed to develop programs to the MXMEXB?

To create programs to MXMEXB board Tietomyrsky recommends Atmel's free tools: The **Atmel Studio 6** integrated development environment (IDE). This IDE contains everything you need to create, compile and debug code. It also includes a programming utility for programming Atmel devices with a variety of tools like **AVR ISP mkII** (The most common tool), AVR Dragon, JTAGICE3, and AVR ONE!. Tietomyrsky recommends Atmel's latest programming and debugging tool **Atmel-ICE**.

Download the Atmel Studio IDE from Atmel web site: <http://www.atmel.com/>.

Recommended reading

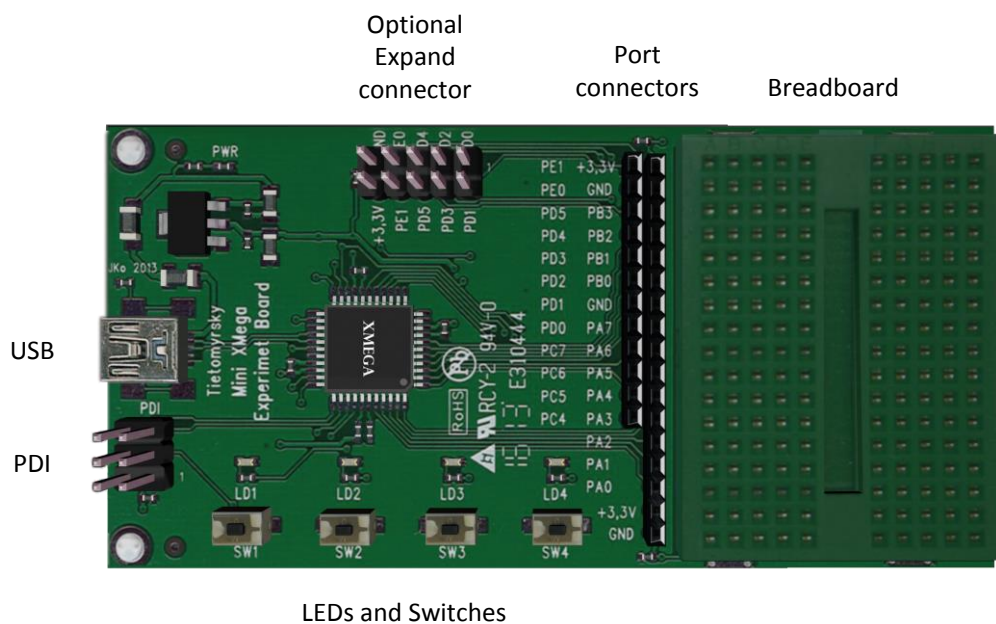
- XMEGA AU manual
- XMEGA A3U datasheet
- XMEGA application notes

Board layout

MXMEXB board is powered by the PC via the USB interface and the USB cable should always be connected to a PC or a 5V USB power supply.

Warning: *Do not try to power the board via any connector from external power supply.*

Figure 1. Board layout



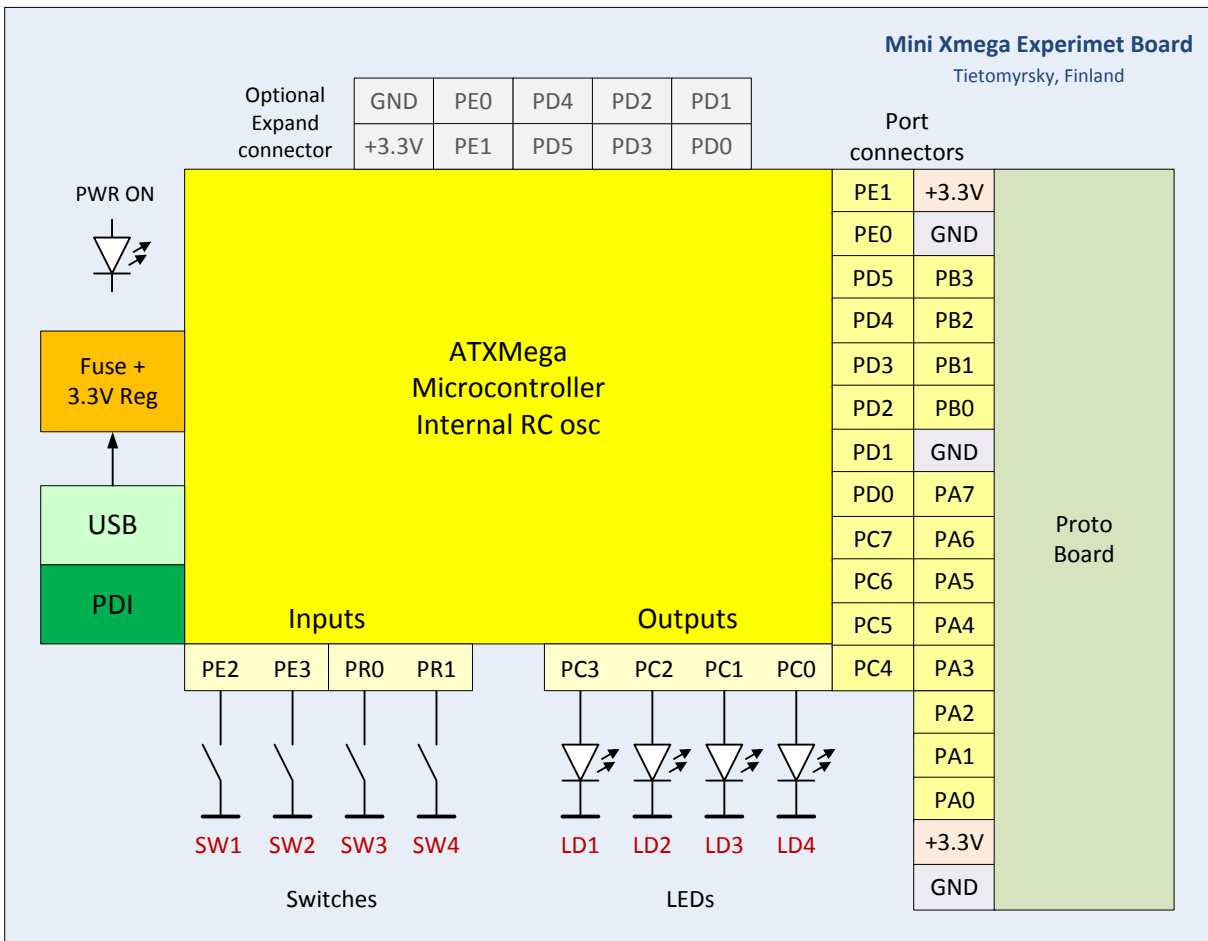
MXMEXB board can be programmed via a 6-pin **PDI** connector.

Breadboard

The breadboard of the MXMEXB board can be used to develop and test small prototype circuits. You can get the needed power (3.3 V) and GND signals from port connector besides the breadboard. The 3.3 V power is regulated and protected against short circuits and reverse current with a resettable PTC fuse and a diode. In overcurrent situation PTC fuse heats up. When the fault is removed, the PTC fuse will cool. PTC fuse starts to limit current when it reaches about 200 mA. All I/O port pin signals of XMega microcontroller, excluding those connected to switches and LEDs, are routed to port connector. The pinout of the port connector is shown in Figure 2.

Only use components that can be powered from a 3.3 V power supply.

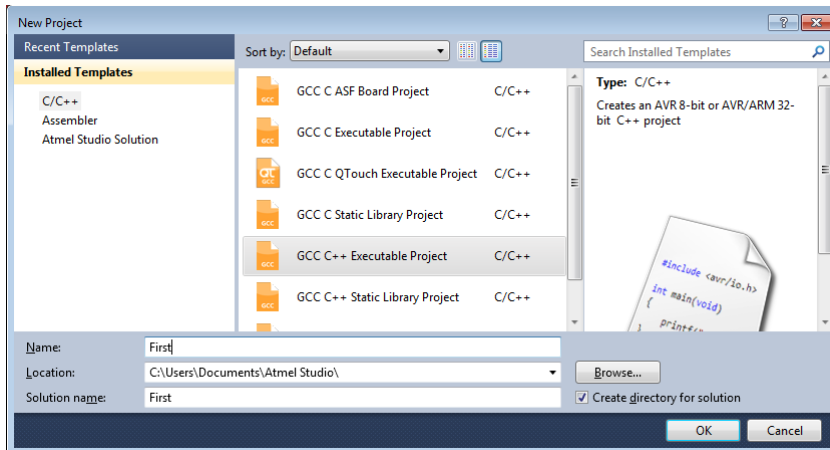
Figure 2. Block diagram of the MXMEXB evaluation board.



Creating the first project to MXMEXB board with Atmel Studio 6

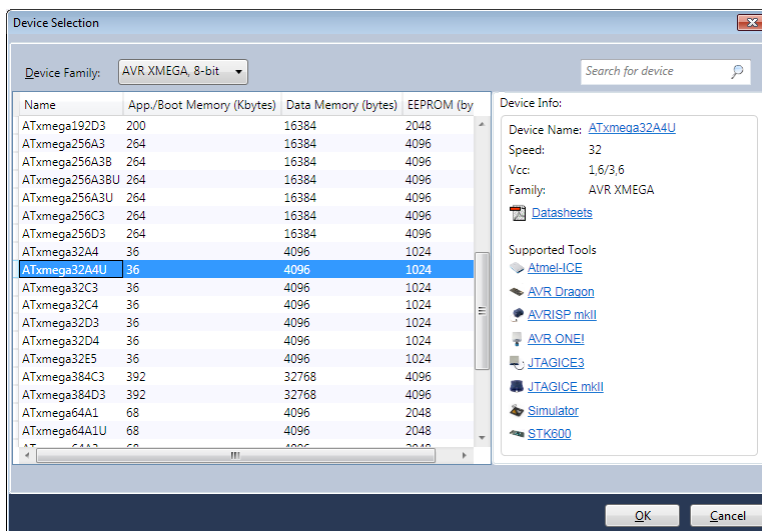
Create the project:

1. File → New Project



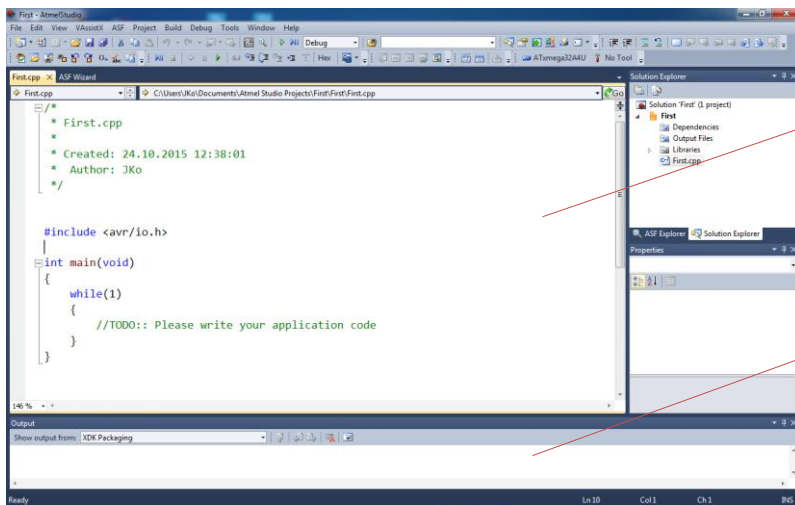
2. Select Installed Templates: C/C++

3. Choose GCC C Executable Project, and give the project a name and folder location. Click OK



4. Select the device. You can search, or filter by family. Chose ATxmega32A4U, and click OK.

Write the program code:



Editor window

Edit program code

Output window

Errors and Warnings

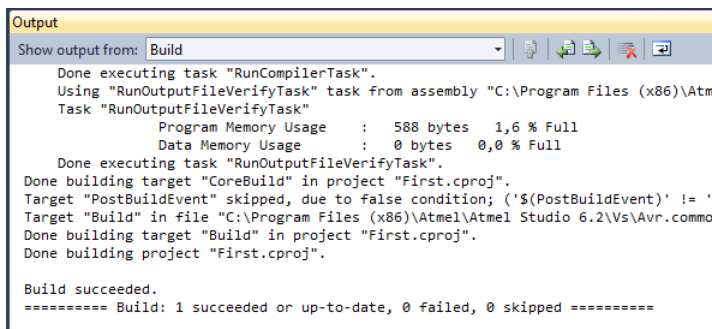
5. Edit the following into the code of editor window:

```
#include <avr/io.h>
#define F_CPU          2.0E6          // default CPU clock frequency is 2 MHz
#include <util/delay.h>

int main(void)
{
    PORTC.DIR = 0x0F;                // set four lower bits of PORTC to output

    while(1)
    {
        PORTC.OUT = 0x00;            // all leds off
        _delay_ms(200);              // 200 ms delay
        PORTC.OUT = 0x0F;            // all leds on
        _delay_ms(200);              // 200 ms delay
    }
}
```

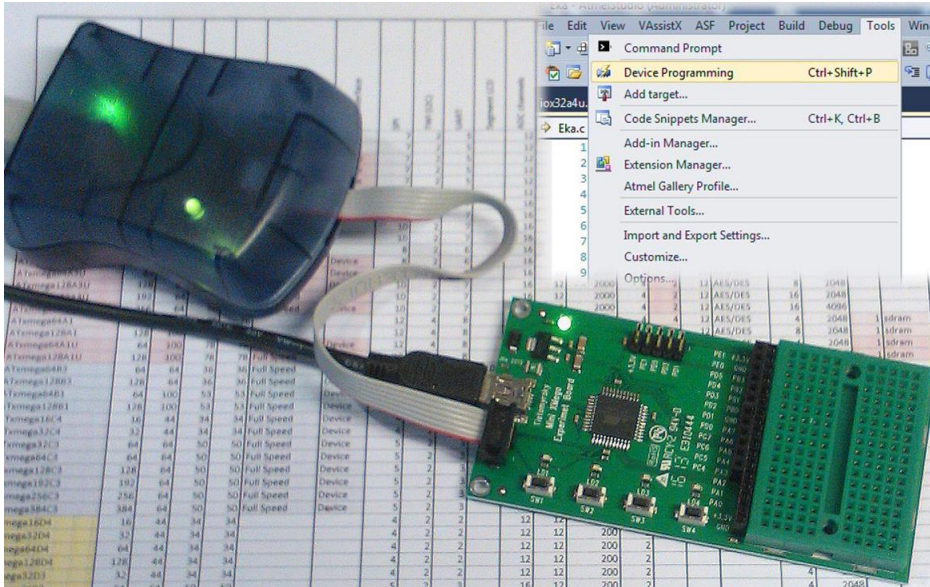
6. To compile the program just press **F7** key (Build → Build Solution)



If there are any errors in Output window, always start fixing the errors from first one on the list, usually the first error generates rest of the errors.

Program the device:

7. Connect power via USB cable to the MXMEXB board. Connect programming device to the PDI connector of the MXMEXB board.
8. Tools → Device Programming



9. Select the programming tool, device (microcontroller type) and programming interface.

Device Programming

Tool: AVRISP mkII | Device: ATxmega32A4U | Interface: PDI | **Apply**

Device signature: not read | Read | Target Voltage: --- | Read

1. Select Tool
2. Select Device
3. Select Interface
4. **Press Apply button**

Target Voltage -> Read

Device ID -> Read

If you can read Device ID, then programming interface works.

AVRISP mkII (0000A0021224) - Device Programming

Tool: AVRISP mkII | Device: ATxmega32A4U | Interface: PDI | **Apply**

Device signature: 0x1E9541 | **Read** | Target Voltage: 3.0 V | **Read**

Interface settings

- Tool information
- Device information
- Memories
- Fuses
- Lock bits
- Production Signatures
- Production file
- Reading device ID...OK

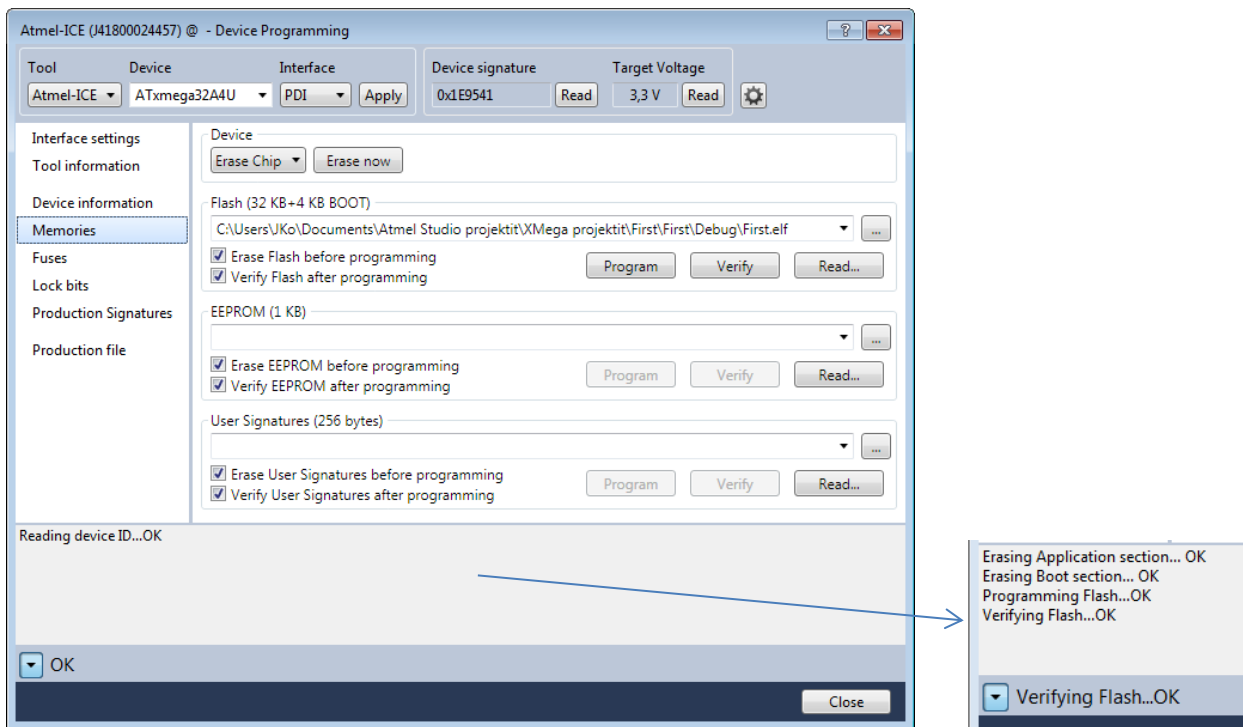
Functions of Programming Interface

Options

☒ Reading device ID...OK

Close

10. Program the device. Select **Memories** from Interface settings.



11. Click **Program** from Flash section.

Now all four leds of the MXMEXB board should blink.

MegaAVR vs. Xmega

MegaAVR	Xmega	Register
DDRA	PORTA.DIR	Data Direction Register
PORTA	PORTA.OUT	Output Register
PINA	PORTA.IN	Input Register

More example programs:

```
/*
 * Knight Rider light
 *
 * Created: 24.10.2015
 * Author: JKo, Tietomysrsky
 */

#include <avr/io.h>
#define F_CPU 2.0E6 // default CPU clock frequency is 2 MHz
#include <util/delay.h>

int main(void)
{
    uint8_t leds = 0x01; // start from right
    uint8_t i;

    PORTC.DIR = 0x0F; // set four lower bits of PORTC to output

    while(1)
    {
        for (i=0; i<4; i++)
        {
            leds <<= 1; // rotate left
            PORTC.OUT = leds; // show next figure
            _delay_ms(100); // 100 ms delay
        }

        for (i=0; i<4; i++)
        {
            leds >>= 1; // rotate right
            PORTC.OUT = leds; // show next figure
            _delay_ms(100); // 100 ms delay
        }
    }
}
```