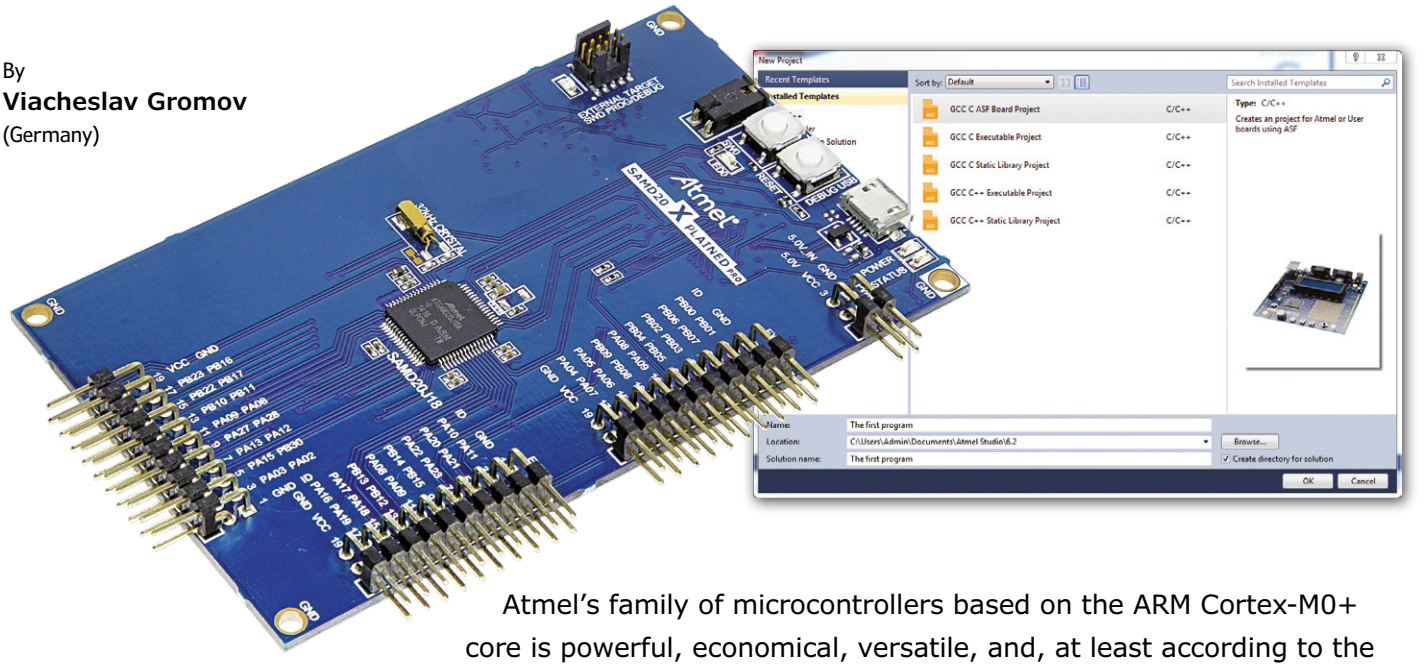
From 8 to 32 bits: ARM Microcontrollers for Beginners (1)

The board, the software and our first program

By
Viacheslav Gromov
(Germany)



Atmel's family of microcontrollers based on the ARM Cortex-M0+ core is powerful, economical, versatile, and, at least according to the manufacturer, easy to use. So why not take the plunge now and explore the world of 32-bit microcontrollers? Our course, designed for those with a little experience of 8-bit devices, will help you on the way.

This programming course will introduce you to the world of ARM Cortex-M0+ microcontrollers, and, as always at Elektor, our emphasis is on practice rather than theory. Many free development environments and low-cost development boards are available: for our course we have chosen the 'SAM D20 Xplained Pro', which is based around the SAM D20 low power microcontroller. Thanks to support from the manufacturer Atmel, we are able to make a thousand of

these boards available to interested readers at a reduced price: see the text box for more details. The course will start with a brief overview of the board and the microcontroller device, followed by the installation of Atmel Studio 6.2. Then, for a little bit of instant gratification, we will build our first project. We will compare the device with eight-bit microcontrollers, to which it is similar in many ways. Then, in the next installment we will describe the main peripheral elements and how they can be used in simple projects.

The board

The block diagram of the board (Figure 1) might not look too exciting at first glance. As you can see, most of the 64 pins of the microcontroller are brought out to headers, and tables are provided giving the pinouts of these headers and the other connectors.

Power can be supplied to the board at 5 V using either the USB socket or the PWR header. If USB power is selected then the PWR header can supply power to an external circuit at either 5 V or 3.3 V; if, on the other hand, power is applied at PWR, the EDBG on-board debugger (see text box) is automatically switched off to reduce current consumption. It is nevertheless recommended to use a power supply capable of delivering at least 500 mA, whichever power input is used.
Headers EXT1 to EXT3 also carry power at 3.3 V to supply expansion boards. On each extension header pin 1 (called 'ID') is reserved for the connection of an ID chip on the expansion board. The ID is used by the EDBG to determine what type of expansion board has been plugged in, and this information can be displayed on the screen of the PC running the development environment software.

The board also includes a 32 kHz quartz crystal (which forms one of the clock sources for the main microcontroller), the DEBUG USB connection for an external debugger, buttons labeled RESET and SW0, and LED0, a yellow LED. SW0 and LED0 are connected to PA15 and PA14 respectively and may be used freely by the developer. The jumper next to SW0 connects the output of the on-board voltage regulator to the microcontroller: the current consumption of the device can be measured by removing the jumper and connecting an ammeter between the pins.
The power and status LEDs (not shown) next to the USB socket are both connected to the EDBG circuit. The power LED lights when the board is supplied with power, and the status LED flashes when the main SAM D20 microcontroller is being debugged over the EDBG or is in some other special state. Both LEDs flash simultaneously when the debugger's firmware is updated. The user manual for the board can be found at [1].

The microcontroller

The SAM D20J18 is an interesting member of the ARM Cortex-M0+ family. It offers 64 pins, 256 kB of flash memory, 32 kB of SRAM, and a wide range of peripherals, and can run at a maximum clock frequency of 48 MHz. It is pow-

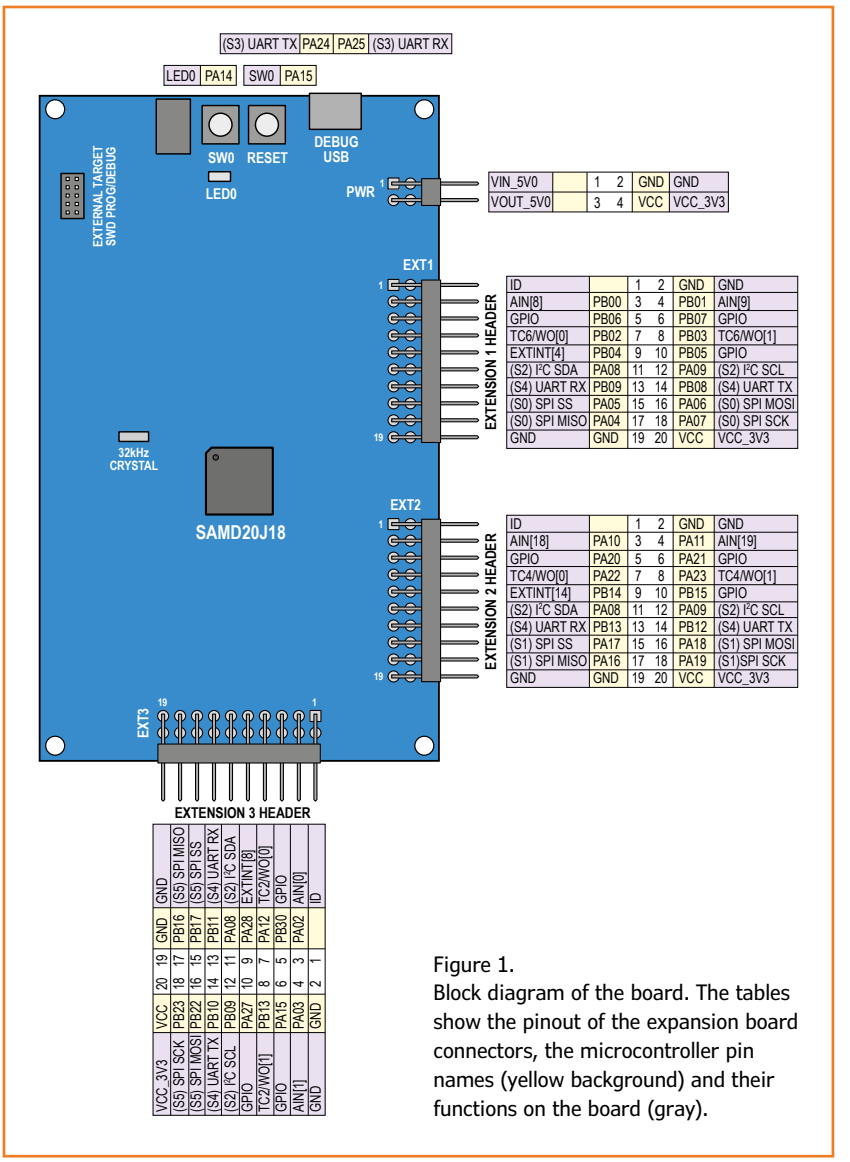


Figure 1.
Block diagram of the board. The tables show the pinout of the expansion board connectors, the microcontroller pin names (yellow background) and their functions on the board (gray).

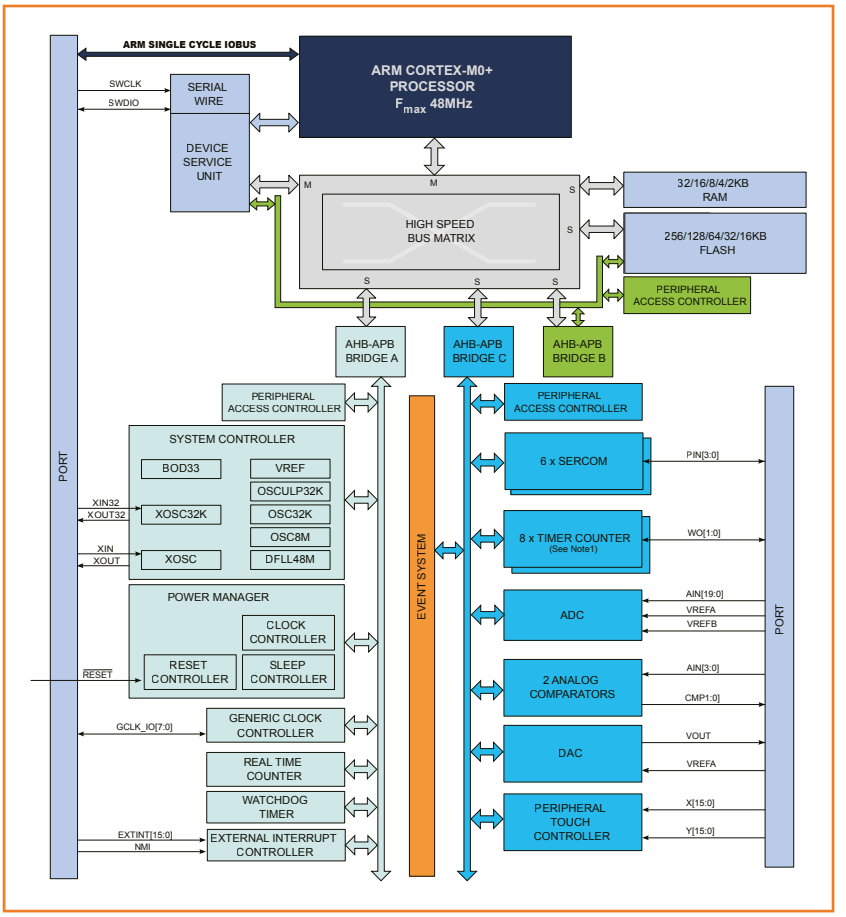


Figure 2.
Block diagram of the SAM D20 microcontroller (courtesy Atmel).

er-efficient and fast, and is therefore suitable for many different applications. Its current draw is only 70 µA/MHz and can run from a supply voltage of between 1.62 V and 3.63 V. Of particular interest are its peripheral touch controller (PTC) and its event system. We will describe the PTC later in this course, and we will show how it can be used in practice. The event system, as in the ATxmega microcontroller series, can be configured for example to wake the CPU from sleep when a peripheral unit such as the ADC triggers an event; however, not all peripherals are supported in this way. The microcontroller has two sleep modes: in idle mode only the CPU is powered down, while in standby mode the clock source and all peripheral units (except those otherwise configured in software) are put to sleep.

Figure 2 shows a block diagram of the microcontroller family. On the left is the 'ARM single-cycle I/O bus', which allows the processor to have fast access to the GPIOs. Below that is the serial debug interface, which has direct access to the processor core. Below the 'high speed bus matrix', which connects the core to the memories on the right via slave ports, you can see several data buses and the peripheral access controller: this is in contrast to the arrangement in conventional eight-bit microcontrollers. The peripheral access controller can prevent peripheral registers from being written to if necessary. The most important peripherals are connected to the APB-C bus: among these the most interesting are the six SERCOM blocks which provide for serial communications using a range of different protocols including USART, I²C and SPI. The pins used by these blocks are configurable. With the exception of the PTC, the remaining peripheral blocks will be familiar from eight-bit microcontroller designs, although the versions here are typically more powerful and present in greater numbers. Each of the eight timer/counters can be used in 2 x 8 bit configuration, 1 x 16 bit, or alternatively two counters can be chained together to form a 32-bit counter. The left-hand side of the block diagram is less exciting, being mainly concerned with power management and clock generation.

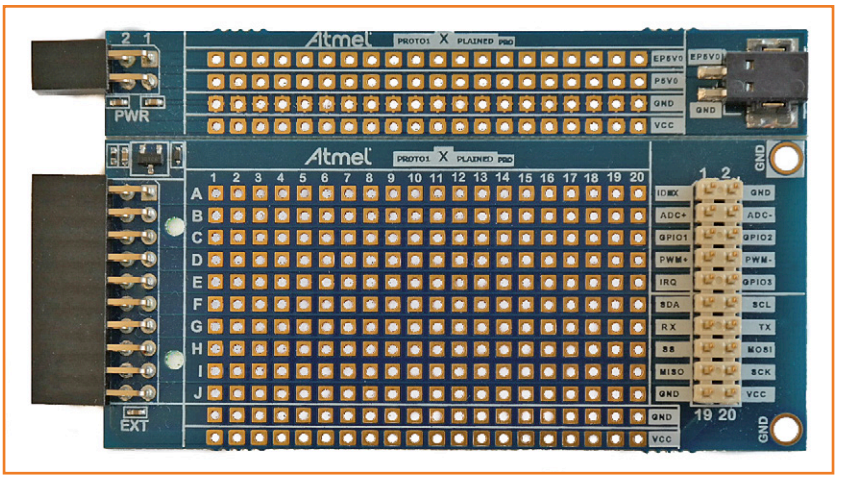


Figure 3.
The simplest expansion board available from Atmel is a prototyping board with a grid of uncommitted solder pads.

We will look in more detail at the possibilities offered by the various peripheral blocks in the next installment in this series, and show step by step how they are configured and used in practice. The data sheet for the device, which runs to some 700 pages, can be found at [3].

The expansion boards

Atmel has developed several expansion boards for the Xplained Pro board. They are designed to help developers new to the device rapidly get to the point of having a working prototype, and to help them learn about the microcontroller. The expansion boards conveniently plug directly into the headers on the main circuit board. Each expansion board includes an ATSHA204 'CryptoAuthentication' chip, which provides information to the EDBG chip on the Xplained Pro board, for example regarding the allowable supply voltage range and maximum current consumption. This information is then passed on to Atmel Studio, which allows the development environment to offer links to data sheets, libraries and example programs.

If you want to build your own expansion board and connect it to the Xplained Pro board, the PROTO1 Xplained Pro [4] (Figure 3) provides the answer. It includes a total of 200 solder pads for prototyping and is connected to headers EXT1 and PWR. On the right the same pins are brought out in a different order to provide a connection for an 'Xplained Top Module'. A groove allows the upper part of the board, which includes the power supply connections, to be broken off if it is not wanted.

The expansion board shown in Figure 4, the IO1 Xplained Pro [5], is designed to help developers understand the most important peripheral blocks of the microcontroller. It includes an LED, a light sensor, a low-pass filter to allow testing of the PWM and ADC blocks, a 12-bit temperature sensor with 8 kB of EEPROM connected over an I²C bus, and a microSD card socket, connected over an SPI bus. A microSD card is included with the board. A couple of spare pins are also brought out.
If you wish to output something to a display, you

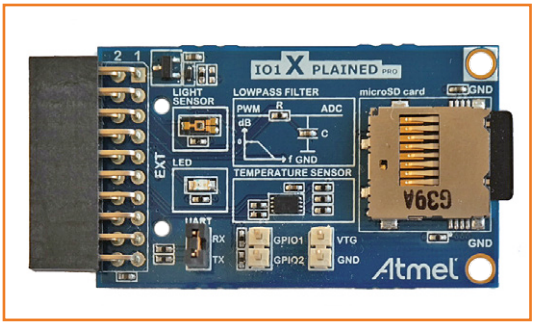


Figure 4.
Universal expansion board for testing and training.

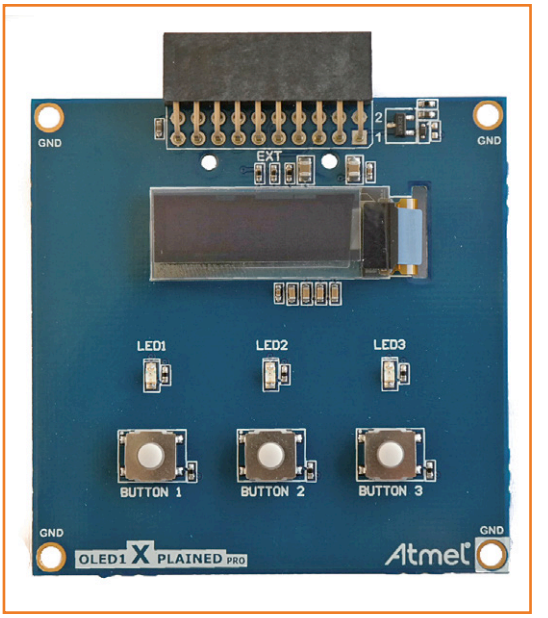


Figure 5.
OLED display expansion board for more advanced projects.

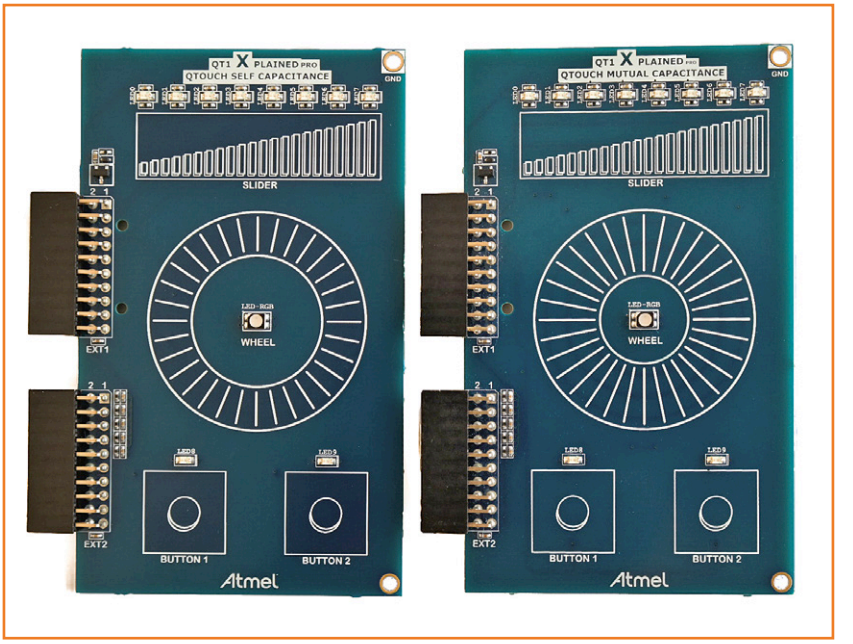


Figure 6.
The two QTouch boards will be used in a later installment of this series.

Figure 7.
Click on the upper icon if you wish to install Atmel Studio on a computer that does not have a permanent Internet connection (all screenshots courtesy Atmel).

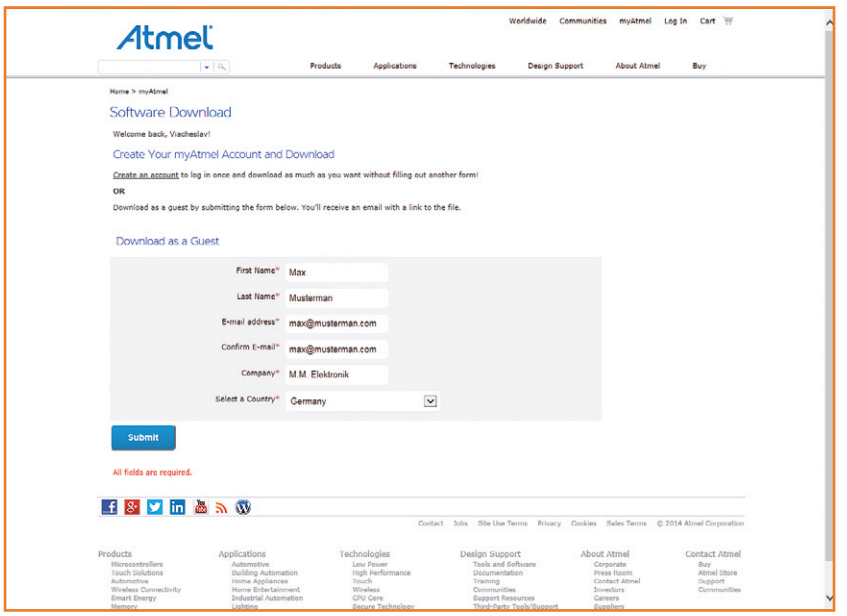


Figure 8.
At this point you should either click on 'Create an account' or enter your personal details.

can use the OLED1 Xplained Pro [6] expansion board: see Figure 5. The board is fitted with a 128-by-32 pixel OLED display with an SPI interface. The board also provides three LEDs and three buttons. This board is designed to be connected to header EXT3.

A special feature of the SAM D20, like certain other microcontrollers by the same manufacturer, is the PTC. Atmel offers a kit (Figure 6) called the QT1 Xplained Pro [7] comprising two expansion boards. Externally the two boards appear identical, but they use different touch detection technologies: 'QTouch self capacitance' and 'QTouch mutual capacitance'. We will examine the exact differences between these technologies and their advantages and disadvantages in a later installment of this series. Each board includes a touch

Debugging and the EDBG

The idea of using a debugger to seek out and eradicate software bugs is not very widespread in the world of eight-bit microcontrollers, and so we shall say a few words about the process here.

Two components are essential to the debugging process: a software tool, forming part of the development environment, and the debugger itself, a hardware bridge between the PC and the target microcontroller system. The debugger can be used to examine and alter variables, memory contents, and often even processor registers during program execution. Breakpoints can be set to halt execution at certain points in the code so that the status of the hardware can be examined. These facilities make it easier and quicker to find bugs, even when the microcontroller is built in to a functioning prototype, as long as the debugger remains connected to it [9].
The EDBG (Embedded Debugger) is a feature not only of all boards in the Xplained Pro series, but is also frequently found on AVR boards. It takes the form of debug hardware, specially developed by Atmel for its development kits, integrated onto the board. As well as offering facilities for programming and debugging the connected microcontroller, it has an extra feature that will come in very handy later on: the Data Gateway Interface, which provides a bridge between the PC and several of the interfaces and GPIOs on the microcontroller. While the microcontroller is running, the state of selected GPIOs can be viewed and data can be received over selected interfaces: this can make development much easier. On the SAM D20 Xplained Pro board this interface is connected to the SPI pins of SERCOM5, the I^2C pins of SERCOM2 and GPIO pins PA27, PA28, PA20 and PA21.

The debugger chip also controls the status LEDs and reads ID codes from the expansion boards. And finally, the EDBG can also emulate a COM port over USB, as it is connected to the UART pins of SERCOM3 on the SAM D20 [2].

wheel, a slider and two buttons, as well as ten yellow LEDs and one RGB LED.

The development environment

Atmel microcontrollers can be programmed and debugged using a suitable programmer/debugger and Atmel Studio. This integrated development environment (IDE) is free, comes directly from the manufacturer, and provides many useful functions: it is therefore an ideal tool with which to start. You may even already be familiar with it.

In order to install the most recent version (version 6.2), go to [8] and click on the CD icon labeled 'Atmel Studio 6.2 sp1 (build 1502) Installer' (see Figure 7). A window will appear as shown in Figure 8, in which you must either create an account or provide details to continue as a guest before downloading the software. It is advisable to create an account, as registration will be required again at various points later in this course. If you already have an Atmel account you can log in using the button at the top right. A link will now appear which, when clicked, will start the download. When the browser pops up a message asking if you wish only to download the program or if you wish to run it automatically, it is best to select the latter option. After a pause a Windows security message will appear: click on 'Always trust software from Atmel Norway' and then click on the 'Install' button. If you do not have Microsoft's .NET framework or Visual Studio installed on your machine, Atmel Studio will prompt you to install them: follow the instructions provided. In both cases you will need to accept the license terms, and in both cases you should install the full versions of the programs. You should use the suggested installation paths unless you have a reason to change them. The InstallShield Wizard will then present a window suggesting the installation of the USB driver, which you should accept by clicking 'Install' (see Figure 9).

After accepting the license in the next window (Figure 10) installation of the driver will begin, which can take a couple of minutes. Then, with any luck, a message will appear confirming successful installation of the driver. Again, you should confirm this.

Installation of Atmel Studio itself can now begin. In the first window (Figure 11) click on 'Next'; in the second, accept the license and again click on

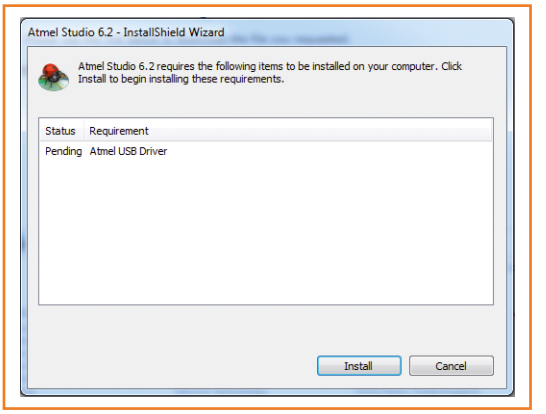


Figure 9.
Atmel Studio requires the installation of the USB driver.

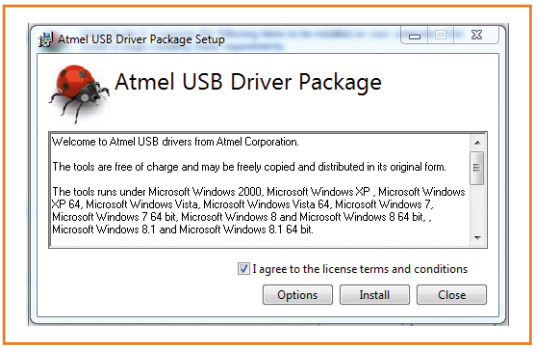


Figure 10.
Read the license text carefully before accepting it.

Figure 11.
The main part of the installation process begins.

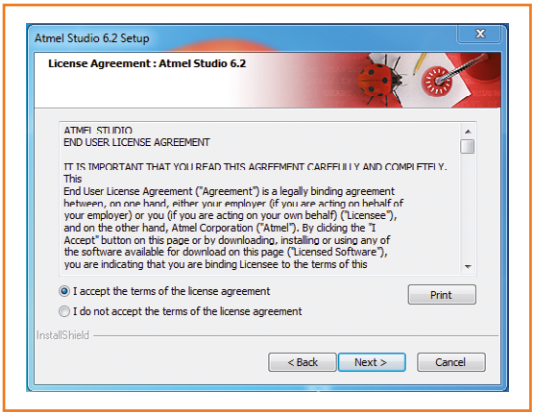


Figure 12.
You cannot proceed further until you have accepted the license.

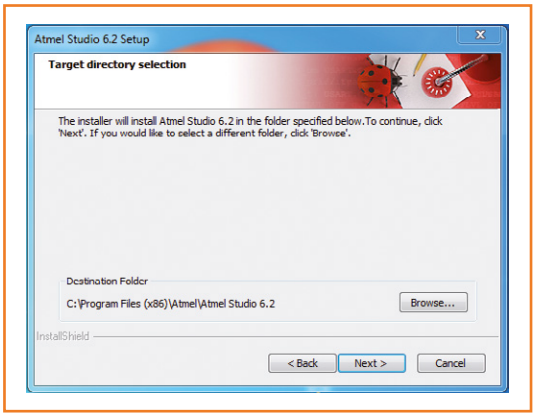


Figure 13.
The suggested installation path is normally appropriate.

Figure 14.
The installation is complete!

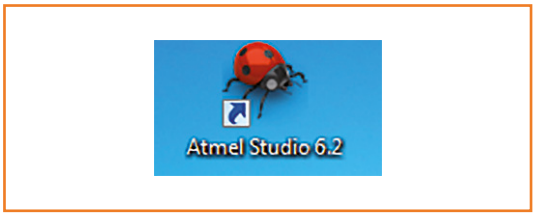


Figure 15.
The Atmel Studio 6.2 icon.

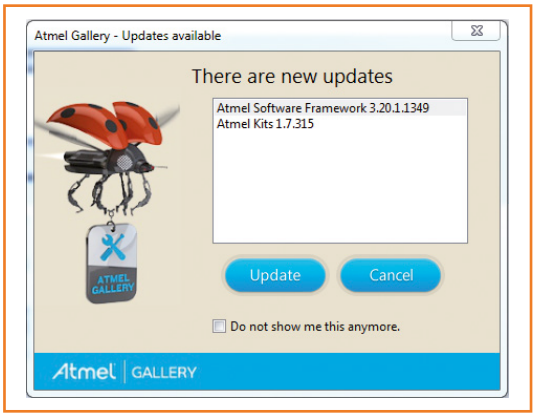


Figure 16.
These updates should definitely be installed.

'Next' (Figure 12). In the next step (Figure 13) select the installation path for Atmel Studio. If you wish to use a different path from the one suggested, click on 'Browse...'. Then click on 'Next'. The last step in the installation process is a summary of what will be installed and where. Confirm the details by clicking 'Next'. It will now take some time for installation to complete (Figure 14). Tick the check box (before clicking on 'Finish') if you do not have any other platforms installed on your machine that use files with the same conventional extensions as those listed. The InstallShield Wizard will now disappear and the icon shown in Figure 15 should appear on your desktop. It is now a good idea to reboot the machine to ensure that it is in a clean state.

Our first program

The development board is connected to the PC using a USB-A to USB-B male-to-male cable. The device manager should automatically recognize the device and install the driver that was included with Atmel Studio. If the device manager fails to find the driver, you must tell it the necessary path by hand.

Now we launch Atmel Studio for the first time by double-clicking on its icon on the desktop. After a brief delay the welcome window should appear. You will see a message like the one shown in Figure 16, which invites you to update some of the tools.

Among these is the Atmel Software Framework (ASF) which we will be making heavy use of in this series. So click on 'Update', which will take you to the 'Extension Manager' (Figure 17), where the necessary updates can be carried out one by one. At the outset we recommended that you register on the Atmel website so that you can install the updates. Since the installation process for updates can vary, we will not describe it in detail here. Depending on the individual update, you will need to respond to security messages, accept licenses, and link the downloaded update with Atmel Studio. The simplest and most stress-free approach is always to accept default options recommended by the installation program. Once all the updates are done, close the Extension Manager and follow its recommendation to restart Atmel Studio. And now finally we can make a start on our first project! The first thing to do is to become familiar with the IDE. Having restarted Atmel Studio, select the start page at the top of the screen and click on 'New Project...', which will begin the

About the author

At fifteen years of age, Viacheslav Gromov is one of the youngest ever Elektor authors, but nevertheless has been working with analog and digital electronics for several years, mostly from his well-equipped basement workshop. He has already had a few short articles published in Elektor, and has written books, including about ARM Cortex microcontrollers. He would like to take this opportunity to thank his family for their support of his hobby, as well as Andreas Riedenauer of Ineltek Mitte GmbH for supplying information and boards.

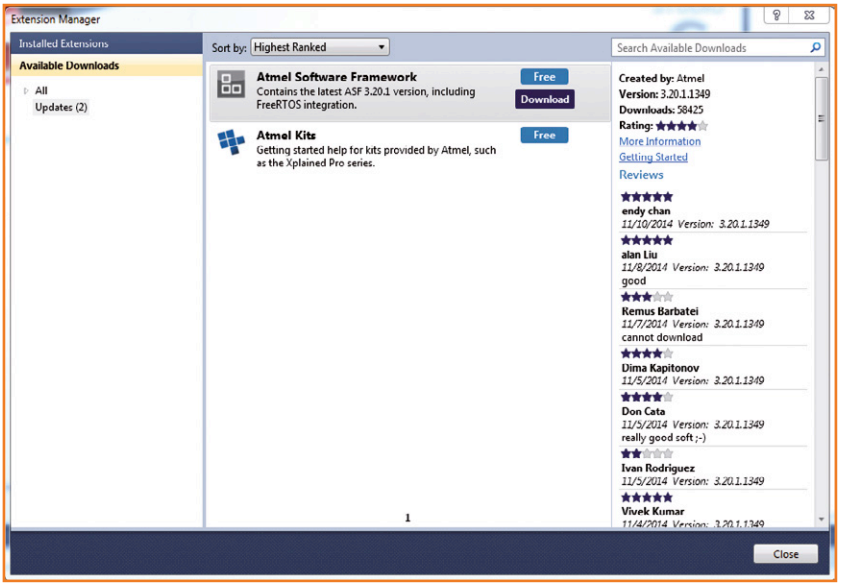


Figure 17.
The extension manager includes many additional software tools that you may wish to install.

process of creating a new project (Figure 18). Alternatively you can click on 'Project...' under 'Folder/New'.

Atmel Studio's 'New Project' window will now appear, which asks you for the type, name and path of the program you are about to write. As shown in Figure 19, select the type 'GCC C ASF Board Project', which, as the name indicates, means that we will be using the C programming language and the Atmel Software Framework. We will look at the ASF in more detail next month. In the next window (Figure 20) choose the correct board type with the help of the search function. There will now be a delay while the project is initialized, and then you will be able to open the file main.c in the project directory. There you will see that the source code for a mini-application has already been generated (Listing 1).

At the beginning of the program the header file for the ASF is included; the ASF system is initialized in the main routine. The program then drops into an infinite loop in which an if-statement checks whether button SW0 is pressed and turns LED0 on or off accordingly. This example file is thus a complete project in itself, and its function is easy to understand.

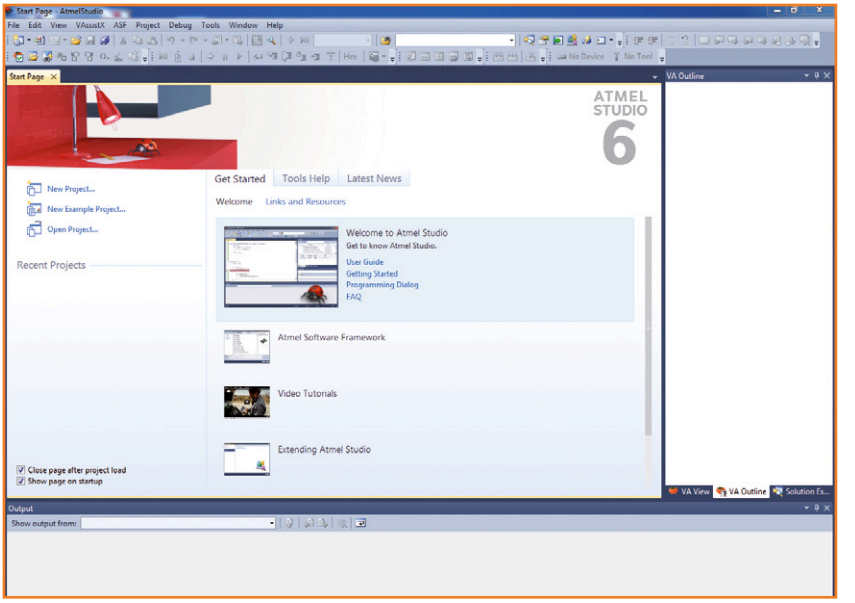


Figure 18.
The welcome screen of Atmel Studio 6.2.

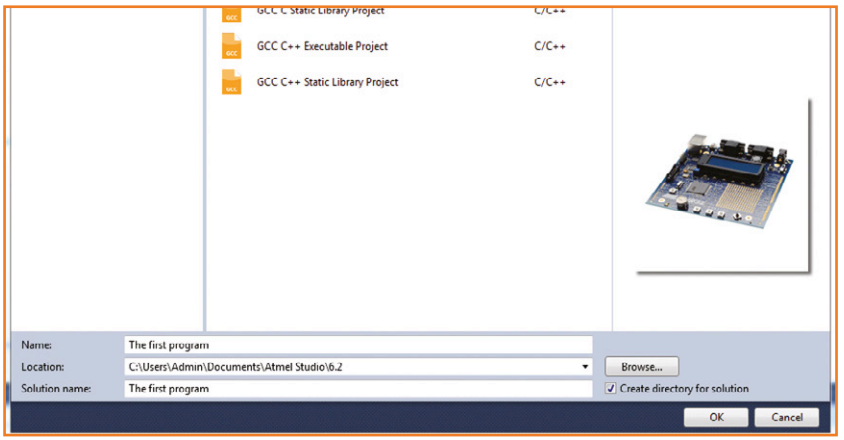


Figure 19.
At this point you must give your project a name.

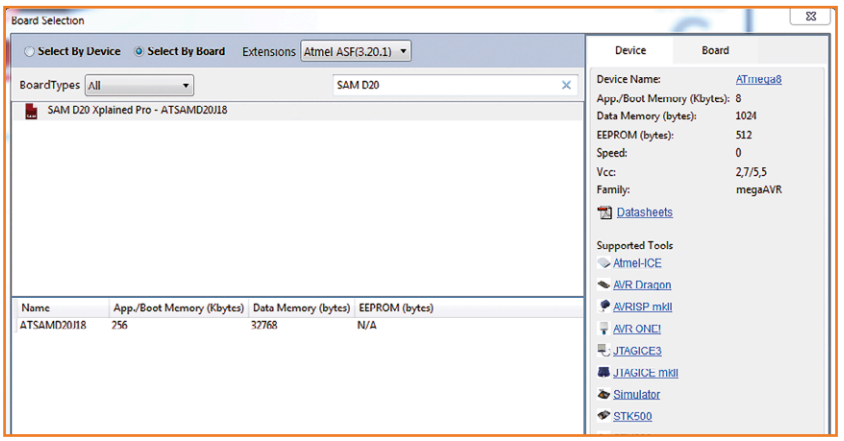


Figure 20.
Use the 'Select By Board' search facility.

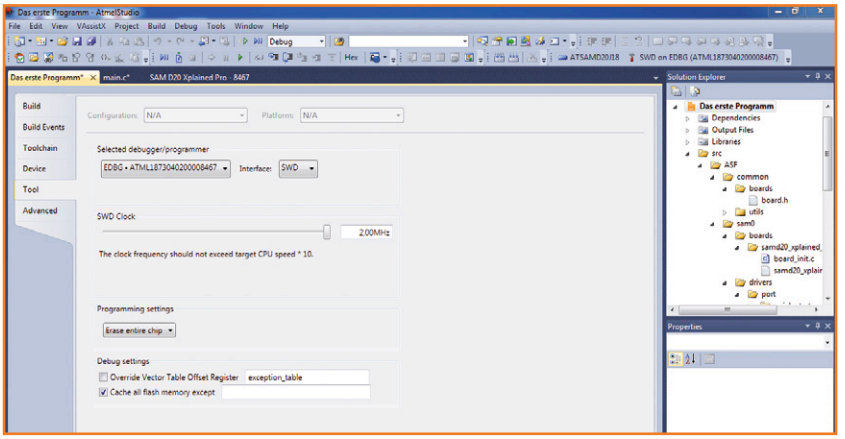


Figure 21.
Atmel Studio should recognize the debugger on the board automatically.

Let's quickly take a look at the other files. The file asf.h simply provides a means to include other header files, which avoids having to clutter up the main program file with a large number of include directives. The file samd20_xplained_pro.h is located in the directory src\ASF\sam0\boards\samd20_xplained_pro and defines names for the most important pins on the board (see Listing 2). This file is automatically generated by Atmel Studio when a project is created based on a particular board.

Other important files in this project are system.c where the function system_init() is defined, and port.h, where the GPIO functions are declared: it is worth taking a look at the contents of these files.

We will be looking more closely at the individual functions in the next installment of this series, but for now we are just interested in compiling the project and running it on the board. Click on the green arrow 'Start Without Debugging' and the project will be compiled and transferred to the board, but the debugger will not be enabled. If beforehand you select a debugger, as shown at the top right of Figure 21, leaving the other settings as they are, then after compilation a message like the one in Figure 22 may appear, inviting you to update the debugger's firmware. Click on 'Upgrade' and wait while the new debugger firmware is loaded onto the board.

Then click again on the green arrow so that our program is finally transferred to the board. You should see the following success report in the output window at the bottom of the screen (here abbreviated):

```
Program Memory Usage : 1628 bytes 0,6 % Full
Data Memory Usage : 8256 bytes 25,2 % Full

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

Our first 32-bit ARM microcontroller program can now be tested by pressing the reset button.

A bright outlook

We hope you have enjoyed this first installment of the series. For any comments or questions, please get in touch via the Books | DVDs | Videos | Courses | Seminars | Webinars topic on

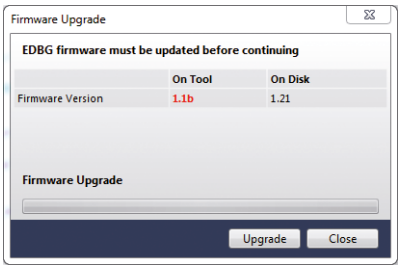


Figure 22.
Both status LEDs on the board flash when the debugger firmware is being upgraded.

the Elektor forum [10]. In the next installment we will look at how to control the GPIOs and the U(S)ART interfaces. Until then, you can get to know the board, the microcontroller and Atmel Studio a little better and try out some of the many example projects available: these can be accessed under 'File/New/Example Project...' or 'New Example Project...' in the opening screen. Have fun!

(140037)

Web Links

[1] www.atmel.com/Images/Atmel-42102-SAMD20-Xplained-Pro_User-Guide.pdf
[2] www.atmel.com/Images/Atmel-42096-Micro-controllers-Embedded-Debugger_User-Guide.pdf
[3] www.atmel.com/images/Atmel-42129-SAM-D20_Datasheet.pdf
[4] www.atmel.com/tools/atproto1-xpro.aspx
[5] www.atmel.com/tools/atio1-xpro.aspx
[6] www.atmel.com/tools/atoled1-xpro.aspx
[7] www.atmel.com/tools/ATQT1-XPRO.aspx
[8] www.atmel.com/tools/atmelstudio.aspx
[9] http://en.wikipedia.org/wiki/Debugger
[10] http://forum.elektor.com

Listing 1. Our first program (excerpt).

```
#include <asf.h>

int main (void)
{
  system_init();

  // Insert application code here, after the board has been
  // initialized.

  // This skeleton code simply sets the LED to the state of
  // the button.
  while (1) {
    // Is button pressed?
    if (port_pin_get_input_level(BUTTON_0_PIN) == BUTTON_0_ACTIVE) {
      // Yes, so turn LED on.
      port_pin_set_output_level(LED_0_PIN, LED_0_ACTIVE);
    } else {
      // No, so turn LED off.
      port_pin_set_output_level(LED_0_PIN, !LED_0_ACTIVE);
    }
  }
}
```

Listing 2. Definitions required for simple access to the LED and button.

```
#define LED0_PIN                PIN_PA14
#define LED0_ACTIVE             false
#define LED0_INACTIVE           !LED0_ACTIVE

#define SW0_PIN                 PIN_PA15
#define SW0_ACTIVE              false
#define SW0_INACTIVE            !SW0_ACTIVE
#define SW0_EIC_PIN             PIN_PA15A_EIC_EXTINT15
#define SW0_EIC_MUX             MUX_PA15A_EIC_EXTINT15
#define SW0_EIC_PINMUX          PINMUX_PA15A_EIC_EXTINT15
#define SW0_EIC_LINE            15

#define LED_0_NAME              "LED0 (yellow)"
#define LED_0_PIN               LED0_PIN
#define LED_0_ACTIVE            LED0_ACTIVE
#define LED_0_INACTIVE          LED0_INACTIVE
#define LED0_GPIO               LED0_PIN

#define BUTTON_0_NAME           "SW0"
#define BUTTON_0_PIN            SW0_PIN
#define BUTTON_0_ACTIVE         SW0_ACTIVE
#define BUTTON_0_INACTIVE       SW0_INACTIVE
#define BUTTON_0_EIC_PIN        SW0_EIC_PIN
#define BUTTON_0_EIC_MUX        SW0_EIC_MUX
#define BUTTON_0_EIC_PINMUX     SW0_EIC_PINMUX
#define BUTTON_0_EIC_LINE       SW0_EIC_LINE
```

elektor PCB Service
powered by Eurocircuits

PCBs Multilayer Prototypes Small series

Professional Quality
Trusted Service
Secure Ordering

Elektor PCB Service at a glance:

- 4 Targeted pooling services and 1 non-pooling service
- Free online PCB data verification service
- Online price calculator available
- No minimum order value
- No film charges or start-up charges

Delivery from 2 working days

Further information and ordering at www.elektorpcbservice.com