The Australian National University
ENGN 3213 Digital Systems and Microprocessors

HLAB06
Atmega328 Kit Soldering and Testing

1. Introduction

This lab will introduce an Atmega328P microprocessor, manufactured from Atmel Corporation (which is now merged to Microchip Technology) It is a family of Atmel microprocessors, called AVR. The Atmega328 chip is an 8-bit AVR with 32K internal flash memory, thus giving the name “328.” The suffix ‘P’ stands for pico-power. Atmega8 has been one of the dominating 8-bit microprocessors in the embedded systems market together with the 8-bit PIC-microcontroller from Microchip. Automotive industry drives their mass production, and the unit price has become very chip (less than AU$4). There are also other brands such as ARM processors which have been dominating the 32-bit microprocessor market (e.g. many mobile phones are adopting 32-bit ARM Cortex chips).

In the first part of the lab, we will assemble an Atmega328 Kit and connect it to a USB programmer ready for use. In the second part, we will use Atmel Studio, an IDE tool to develop C-codes, compile/link, and download to the board, to run a test program which can blink an LED.

NOTE: We will keep using the same board throughout the semester, and each group will be responsible for keeping the board and all electronics parts safely. It will be in your interest to take as much care as possible to create a well-soldered circuit so that you can reliably use it towards the final demonstration in week 11.

Safety Notes

- **Dangerous**: **Electrolytic capacitor has polarity.** The capacitors connected to the power regulator in the board can explode if they are connected with reverse polarity, which can cause serious damage to eyes***.*** You must get the soldered board checked by the tutor before connecting the external power. It is recommended to wear a safety goggle when you are soldering and testing the board.

- **Dangerous**: **Flying pieces of leads**. When you are trimming any excess leads from through-hole mountings, make sure you face the trimming-side down so that the pieces of leads won’t fly across the bench, which can accidentally hit other students, for example to the eyes. So wearing safety goggles is for your protection.

2. Aims

- To gain hands-on skills on soldering an Atmega328 microprocessor kit.
- To be able to set up the embedded-system development environment.
- To be able to run a test program (blinking LED) using Atmel Studio.

3. Assembling and Soldering an Atmega328 Board

You need to refer to the datasheet and user manuals of the Atmega328 protoboard and USB programmer which you can download from Wattle site.

- *ATmega328 Protoboard User Guide (PB-MC-AVR28-UG-V1.6.pdf)* – It contains the schematic and soldering instructions of the board.
- *ATmega328P_Datasheet.pdf* – It contains comprehensive technical information of the microprocessor. Some relevant information are copied and provided on the lab sheet, but you need to refer to this datasheet for more detailed information.
- *Pololu_usb_avr_programmer_v2.pdf* – It contains how to set up the USB programmer (if it has not been installed in Windows 10) and to check the serial port numbers. There are two serial ports – one for the USB programming and another for a general serial.

The Atmega328P used in this lab comes in a 28-pin PDIP (plastic dual-in-line package) as shown in Figure 1. PDIP has the largest pin spacings of all packages which makes it easier for soldering or wiring up the circuits. Other packages with much smaller footprints do exist such as SMD (surface mount device). Note a ***half-moon mark*** at the top of the chip indicating the orientation, and a ***dot mark*** next to it showing the first pin. The Atmega8 has many internal peripherals. For example, 23-GPIO (general purpose digital I/O) on Port B/C/D, 3 serial communications (UART, SPI and I2C), 3 timers, and 6-channel ADC (analogues to digital converter) (see page 13 of Atmega328 datasheet). The SPI (serial peripheral interface) pins will be used for the USB programmer to download the user program to the flash memory. Note that many of the pins on the Atmega8 PDIP package are *multiplexed* allowing the pins to be programmed for other purposes. For example, a GPIO PB1 (pin 15) can be used as an OC1A (timer waveform output) during the execution of a program.

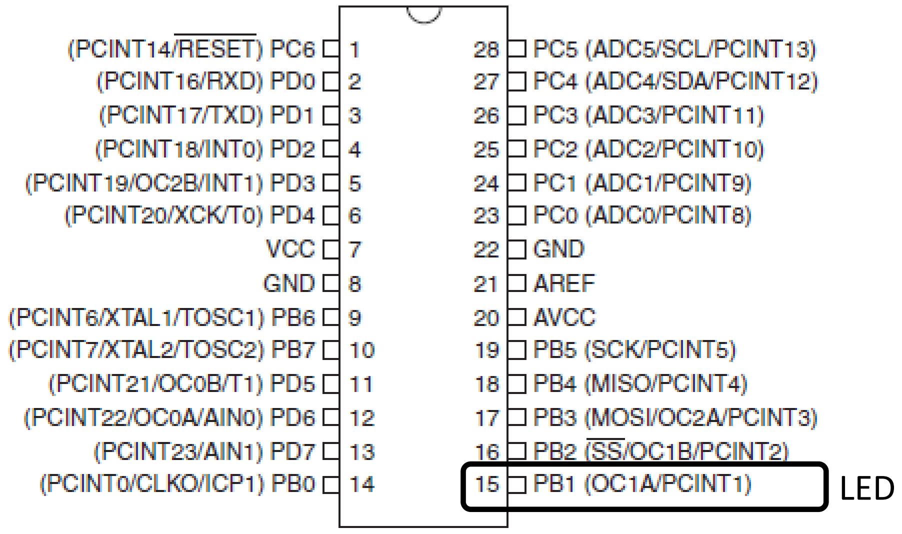


Figure 1 Atmega328P PDIP pin-outs. PB1 pin will be used to connect a LED.

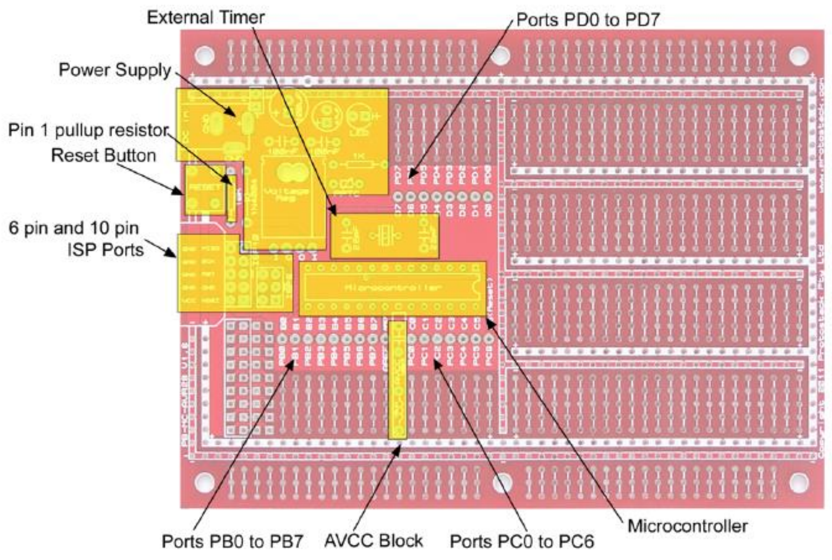


Figure 2 Protoboard layout (from Protoboard User Guide).

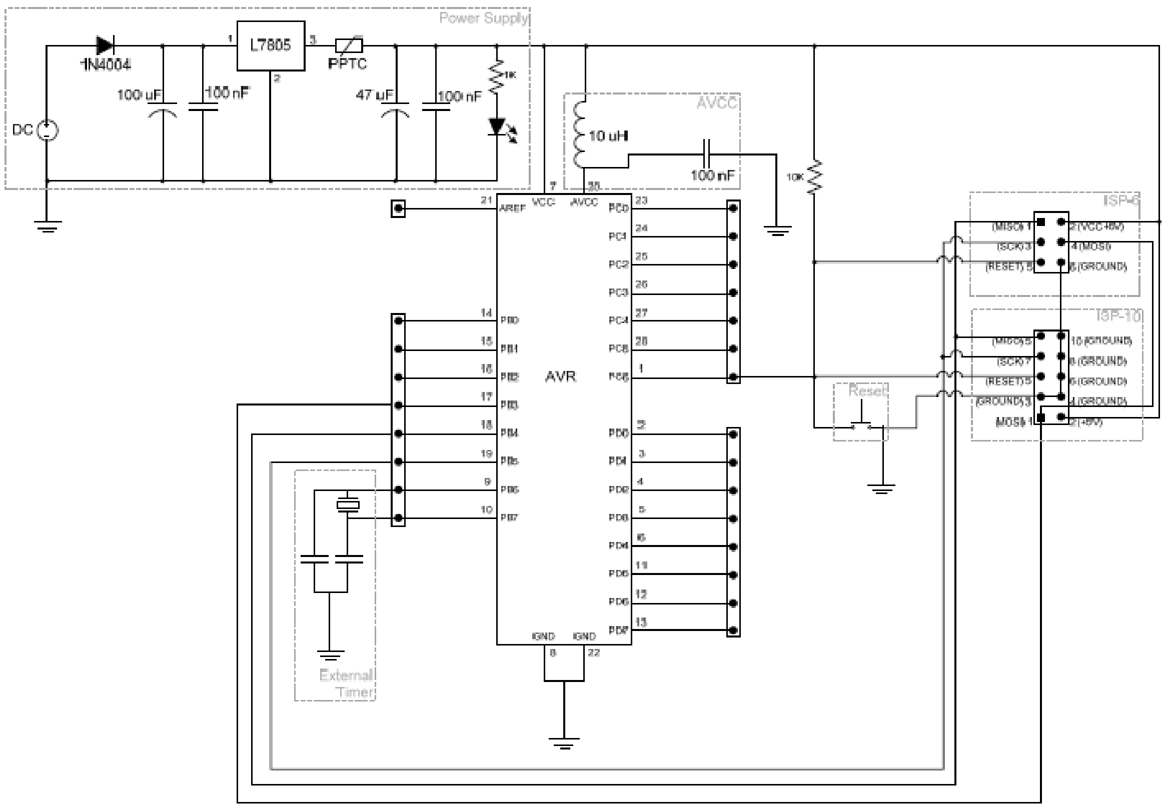


Figure 3 Protoboard schematic (from Protoboard User Guide).

- Follow the Protoboard User Guide, to solder the parts. Note also that
    - The 6-pin ISP (in-system programmer) connector will be used, and so the ISP-10 connection is not required.
    - Pay attention to two electrolytic capacitors in power section which have polarity. The LED also has the polarity (a longer-side lead is a positive side

terminal). When soldering, it would be helpful to solder just one pin of a part. Check the position of the part on the other side of the board and adjust if required. Then solder the remaining leads. If you solder all leads of a part together, it is typically hard to re-adjust the part position.

  - Make sure the 5V regulator (L7805) is inserted properly so that its ground pin in the middle matches to the G-label on the board.
  - It is a good idea to solder some header pins on the port outputs, which later can be connected using jumper wires.

- To test a blinking LED code, connect an LED (to the PB1 pin) with a series pull-up resistor of 620Ω connected to 5V. The pull-up resistor value determines the amount of current (so the brightness) and is not necessarily accurate.
- In the HLAB09 lab, we will connect a PNAV IMU sensor which requires a 3.3V regulator (the connection diagrams can be found in the HLAB09 manual or you can refer to the datasheet of LM2936-3.3).
- Insert the Atmega328P chip onto the holder, paying attention to the direction mark (Pin1 faces downwards in the figure below). The PDIP pins typically need to be bent inward gently to be inserted into the holder.
- The final soldered board will look like Figure 4 shown below.

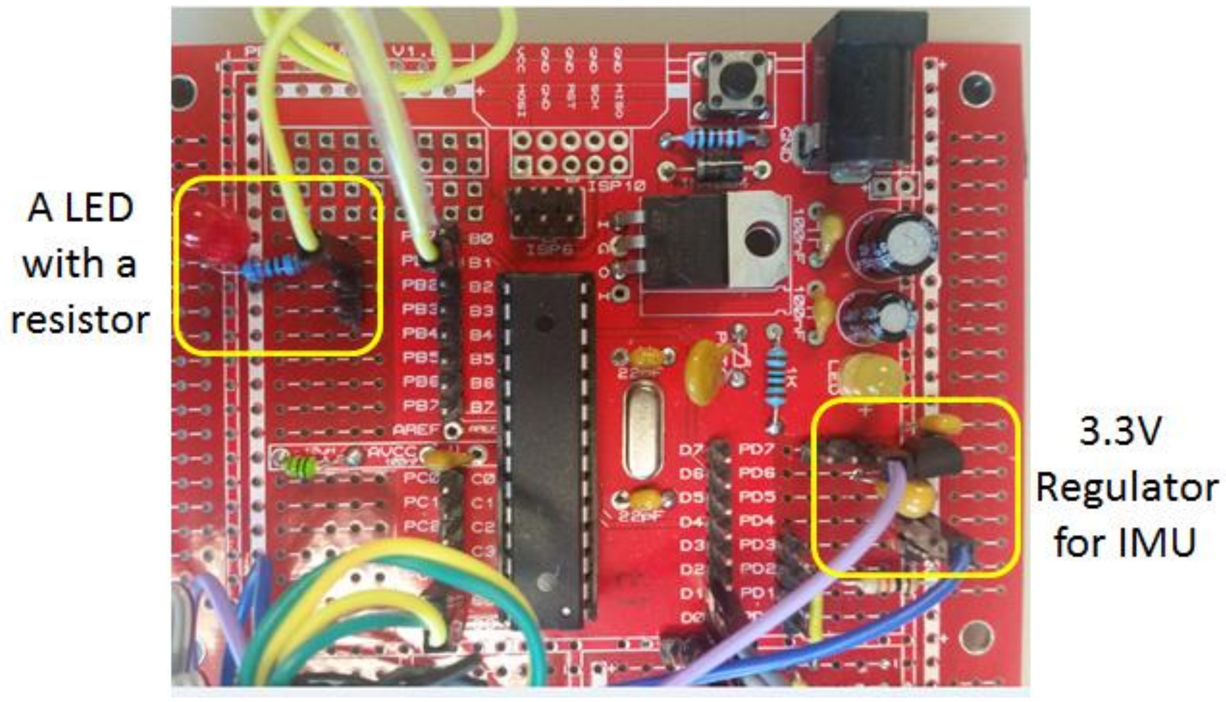


Figure 4 An example of soldered board (the 3.3V regulator can be done in HLAB09).

- Trim any excessive leads under the board. Use a multimeter to check any short connections between the power rail and ground.
- Now it's time to connect the power. Connect the 9V external power plug and check the power-on LED. If the LED doesn't light, unplug the power and check the LED polarity and power rail.
- Leave the power on for a while and examine the board carefully. Make sure the Electrolytic capacitors are not swallowing, and the regulator is getting warm but not too hot. Use the multimeter or oscilloscope to check the voltage, for example, the regulator's 5V output pin and the VCC pin-7 of Atmega328 chip.
- Note that the Atmega328 chip uses an internal oscillator (1 MHz clock). The external crystal oscillator soldered is not used until we programmed the internal fuse in next lab HLAB07. For this lab, we will use the default internal 1 MHz clock.
- **Congratulation**! You've made an Atmega328 board which will be used throughout the labs.

4. The USB programmer

The powered-on microcontroller is in an idle state, as there is no executable code in the flash memory. We will download the executable codes onto the Atmega's flash memory by using an ISP (in-system programmer) which uses SPI serial protocol. We will be using a Pololu USB programmer as shown in Figure 5, which supports two serial ports: one for the ISP programming and another for serial (TTL) communication. Refer to the Pololu USB AVR Programmer v2 User's Guide to setup the board.

- Connect the USB cable to the desktop PC and the ISP cable to the Atmega board. If the Atmega board is externally powered at 5V, both yellow LEDs will flash together (see the datasheet for more LED indicators).
- The windows device drivers have been installed in the desktop PCs. To check the serial port number of the programmer, type "pavr2cmd --prog-port" or "pavr2cmd --ttl-port" in a Windows command prompt. Each of this command will print the serial port number assigned. Now it is ready for programming.

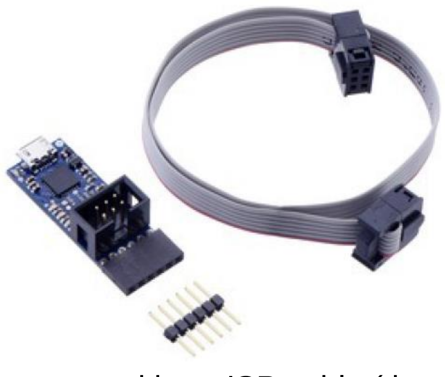
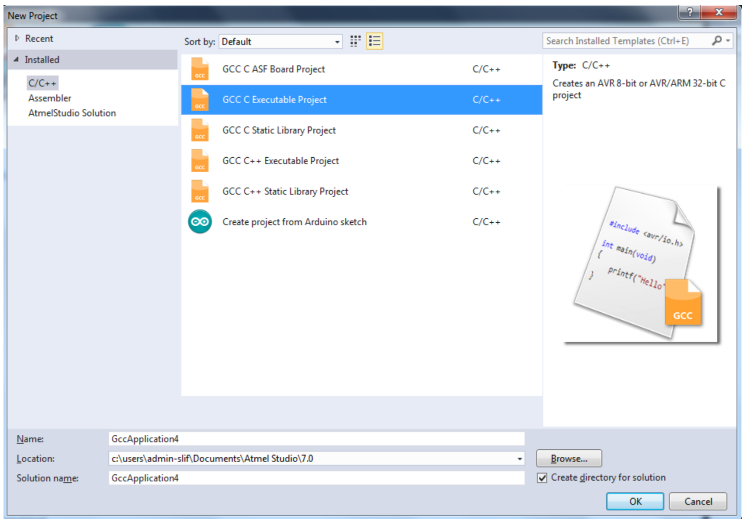


Figure 5 A Pololu USB programmer with an ISP cable (the cable for the extra serial port are not shown here).

5. Atmel Studio 7

Now we will briefly introduce the Atmel Studio 7, and you will be able to compile the first program and finally bring your board to life! This IDE program has many features, but you will only be using the editor, the compiler, and the programming tools.

- Start Atmel Studio 7 by clicking a ladybeetle icon ( ) on the Windows desktop. In the main screen which opens up, click *New Project.* In the New Project window that appears (see Figure 6 below), select *GCC C Executable Project* if you want to work with C language project. Enter a meaningful project name and location for your files and press OK.



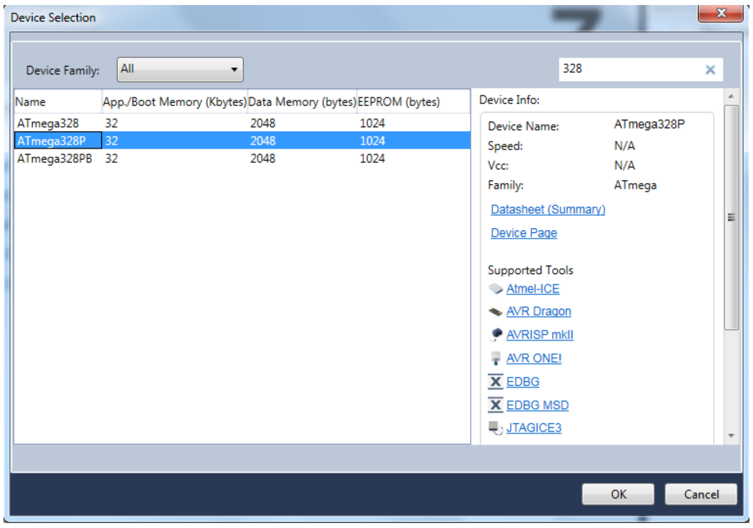
GCC C ASF Board Project	C/C++
GCC C Executable Project	C/C++
GCC C Static Library Project	C/C++
GCC C++ Executable Project	C/C++
GCC C++ Static Library Project	C/C++
Create project from Arduino sketch	C/C++

Type: C/C++
Creates an AVR 8-bit or AVR/ARM 32-bit C project

Name: GccApplication4
Location: c:\users\admin-slif\Documents\Atmel Studio\7.0
Solution name: GccApplication4
[✓] Create directory for solution

Figure 6 New project dialogue.

- In the *Device Selection* window that appears in Figure 7 below, type "328" in the search field, and select the ATmega328P.



Device Family: All
328

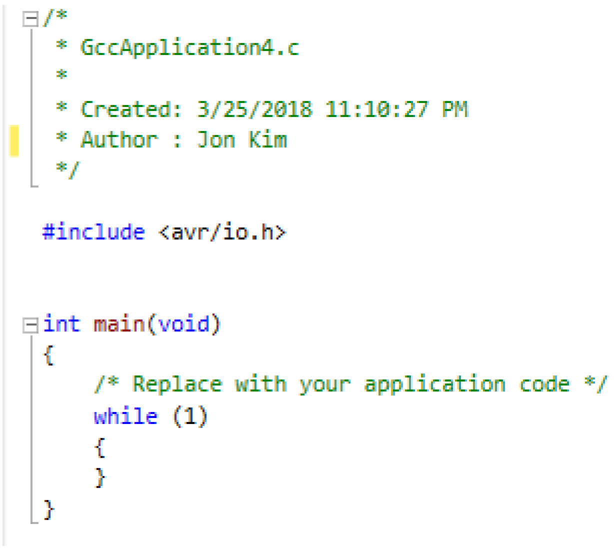
Name	App./Boot Memory (Kbytes)	Data Memory (bytes)	EEPROM (bytes)
ATmega328	32	2048	1024
ATmega328P	32	2048	1024
ATmega328PB	32	2048	1024

Device Info:

Device Name:	ATmega328P
Speed:	N/A
Vcc:	N/A
Family:	ATmega

Figure 7 Device selection dialogue.

- Atmel Studio will then bring up your editor window with a default program, which will look similar to the shown below

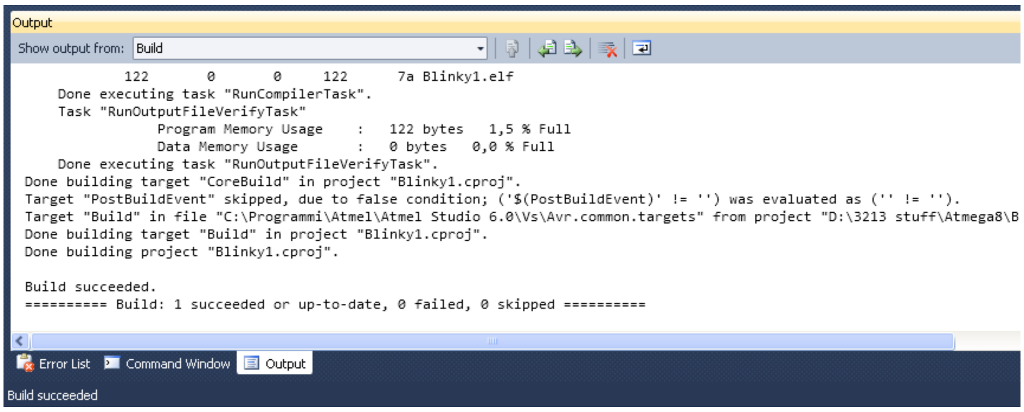


```
/*
 * GccApplication4.c
 *
 * Created: 3/25/2018 11:10:27 PM
 * Author : Jon Kim
 */

#include <avr/io.h>


int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

- Download and open the sample code (*1_blink_LED.c*) in the wattle. You can add the *1_blink_LED.c* file to the project in the Solution Explorer (in the right window pane) and remove the default *main.c* from the project (there should be only one main() function in a project). Alternatively you can just copy and paste the code onto your editor.
- Now you are now ready to compile. Go to the *Build* menu in the menu bar and select *Build Solution* (Shortcut key F7). The system will run the compiler and linker, and if no errors are detected it will generate the executable file and a "Build succeeded" message will appear in the bottom left corner of the Atmel Studio window. The screenshot below shows the output from the build command.

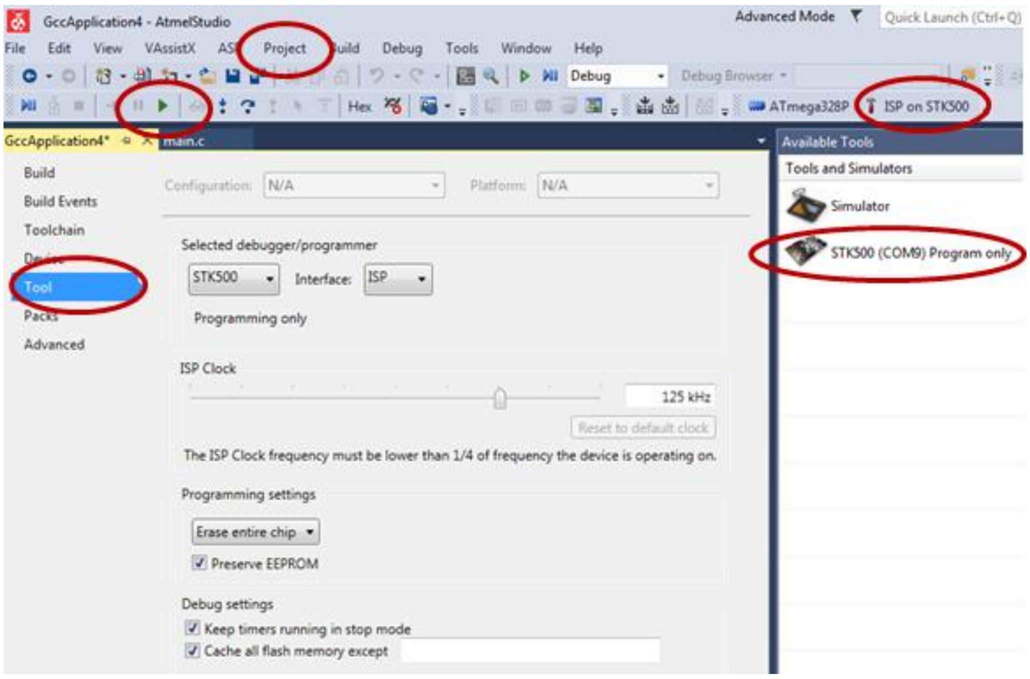


```
             122        0        0      122       7a Blinky1.elf
     Done executing task "RunCompilerTask".
     Task "RunOutputFileVerifyTask"
                 Program Memory Usage    :   122 bytes   1,5 % Full
                 Data Memory Usage       :   0 bytes   0,0 % Full
     Done executing task "RunOutputFileVerifyTask".
 Done building target "CoreBuild" in project "Blinky1.cproj".
 Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated as ('' != '').
 Target "Build" in file "C:\Programmi\Atmel\Atmel Studio 6.0\Vs\Avr.common.targets" from project "D:\3213 stuff\Atmega8\B
 Done building target "Build" in project "Blinky1.cproj".
 Done building project "Blinky1.cproj".

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

Figure 8 A screenshot of build results

- The built project can now be written into the microprocessor's flash memory. Click *Project* in the menu and select *Properties* which will open a page as shown in Figure 9 below. Click *Tool* and select the *STK500* (compatible to the Pololu), the device is ATmega328P, and the Interface is ISP. Save the page. You can also use the button "*ISP on STK500*" shown in the right-top corner of the figure.



Configuration: N/A Platform: N/A

Selected debugger/programmer: STK500 Interface: ISP

Programming only

ISP Clock: 125 kHz

The ISP Clock frequency must be lower than 1/4 of frequency the device is operating on.

Programming settings: Erase entire chip

☑ Preserve EEPROM

Debug settings

☑ Keep timers running in stop mode

☑ Cache all flash memory except

Figure 9 ISP programmer selection

- You can now download the hex file from the *Device Programming* in the *Tools* menu. However, Atmel Studio 7 supports a short-key in the *Debug* menu and *Continue (Shortcut key F5)* which is shown as a play-icon (▶) in Figure 9. By clicking this button (or F5), the code will be *built* and *downloaded* using the selected programmer, which is quite convenient. The Pololu programmer LEDs will be busy during downloading.
- Show your working board with a blinking LED at 1 Hz to your demonstrator to get marks.