

Microchip Studio Tutorial

Author: Peter Sutton. Based on earlier versions by Len Payne, Peter Sutton, Ian Clough. Revised by Jarrod Bennett and Chamith Wijenayake – 2021.

Contents

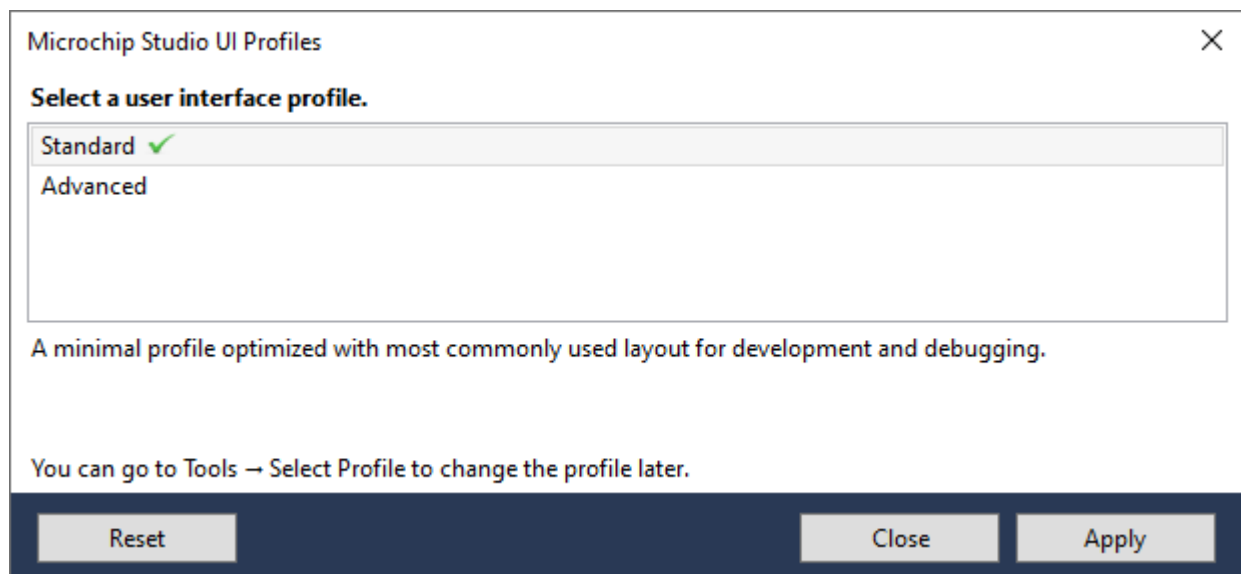
- [Starting Microchip Studio](#)
- [Creating a New Project](#)
- [Editing the Assembler file](#)
- [Assembling \(building\) the Source Code](#)
- [Simulating the Code](#)

Starting Microchip Studio

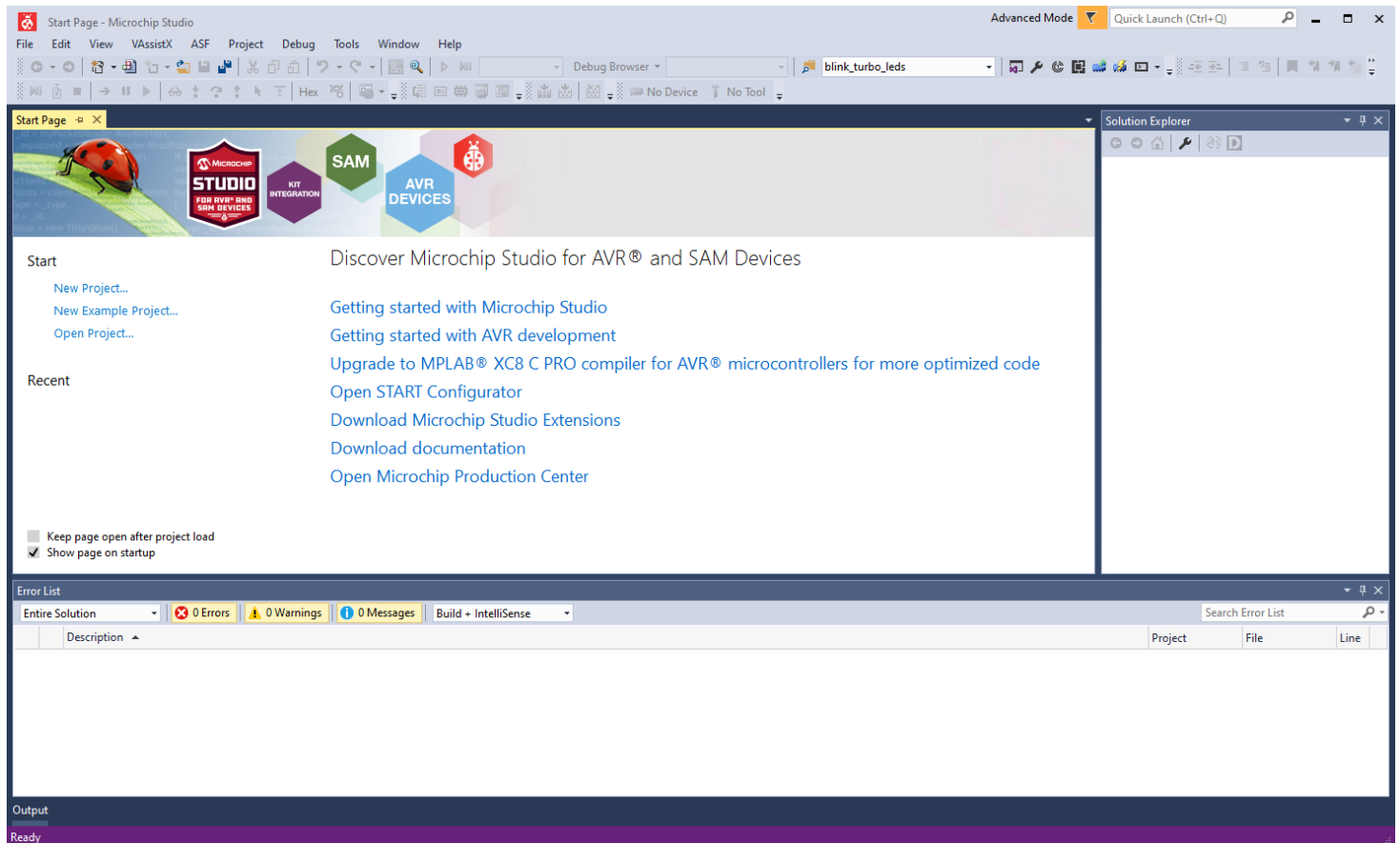
Start Microchip Studio from the Start Menu on Windows. While the program is starting you will see a screen like this:



You may get asked to select a user interface profile – select “Standard” and then “Apply”



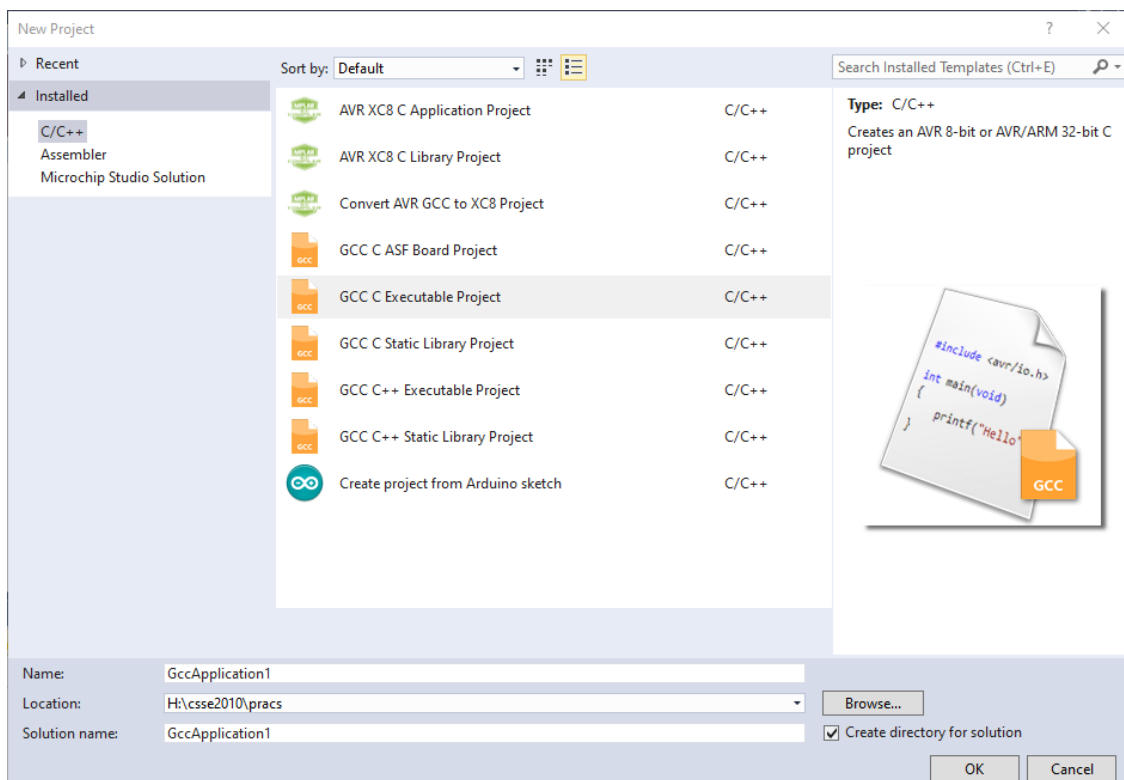
Once the program has started, you will be looking at a screen like this:



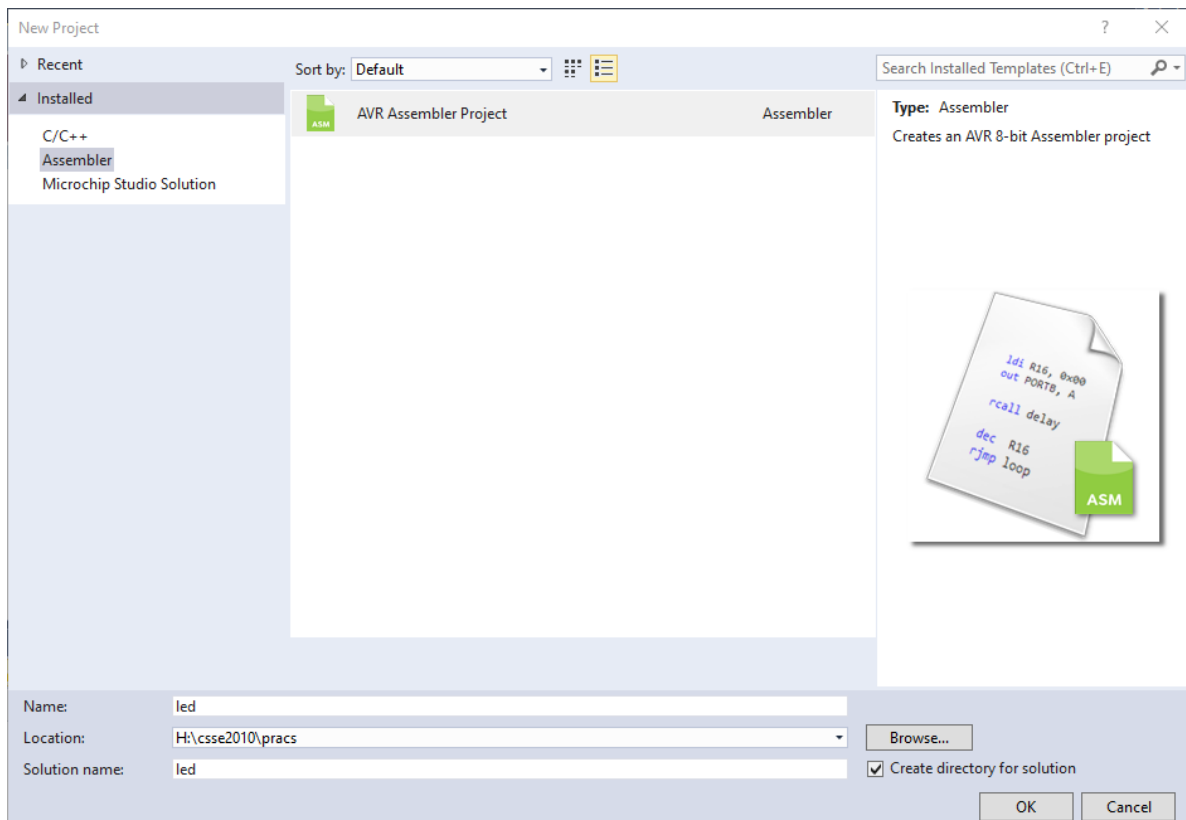
Creating a New Project

In this tutorial we will make a simple program that increases the value of one of the PORT registers, making a binary counter.

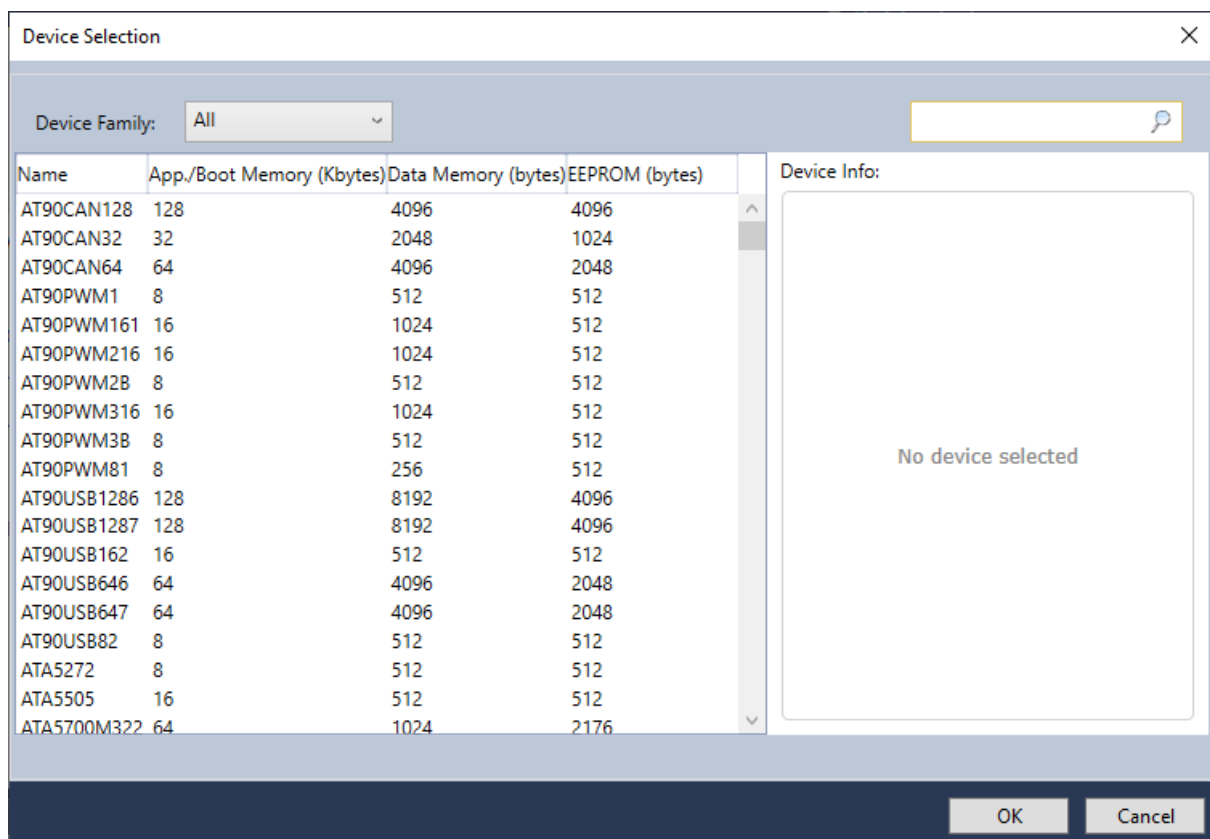
To create a new project, click on "New Project..." on the Start Page or go to the "File" menu and select "New" -> "Project...". The dialog box shown in the next figure appears.



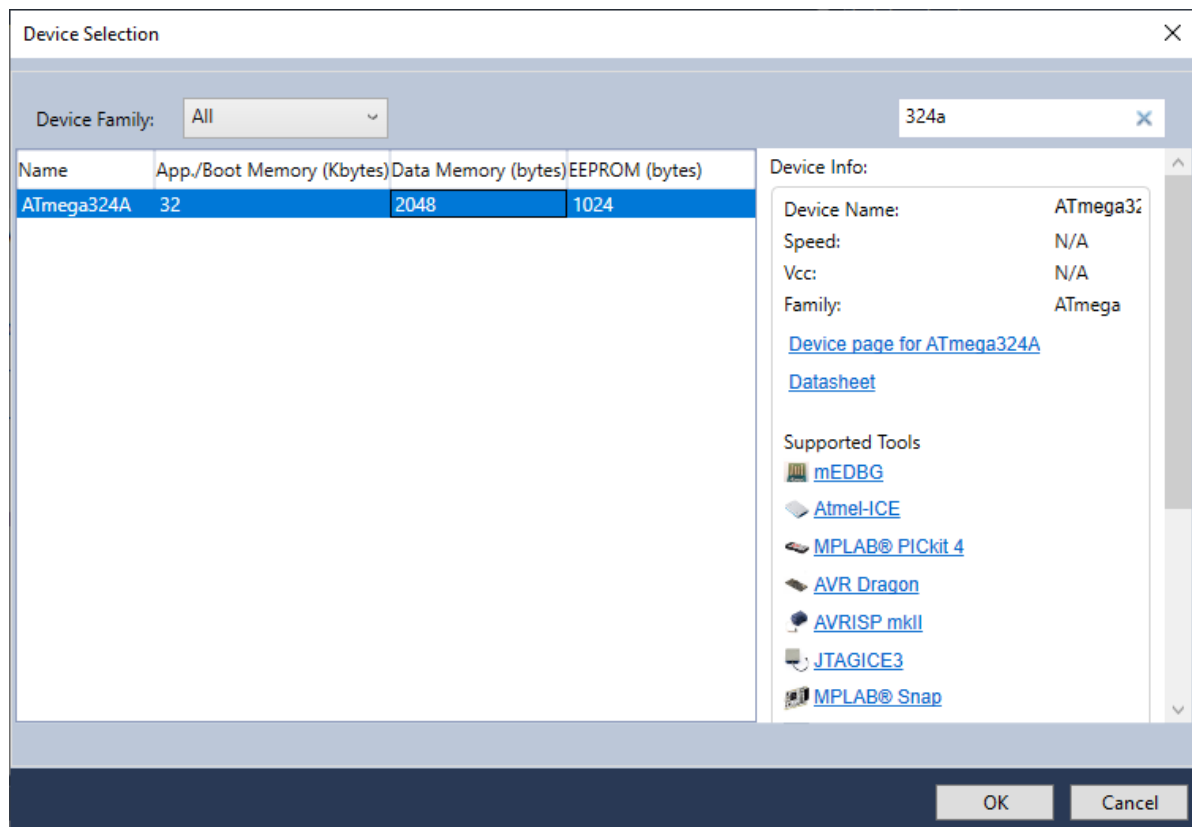
In this dialog box you should select "Assembler" on the left pane and enter the project name. We choose the name "led" here, but this could of course be an arbitrary name. Next, you'll have to select the project location. This is the location where Microchip Studio will store all files associated with the project. We have used the location H:\csse2010\pracs\ as the folder. If the folder does not exist, Microchip Studio should create it for you.



Click **OK** and you will see the Device Selection window:

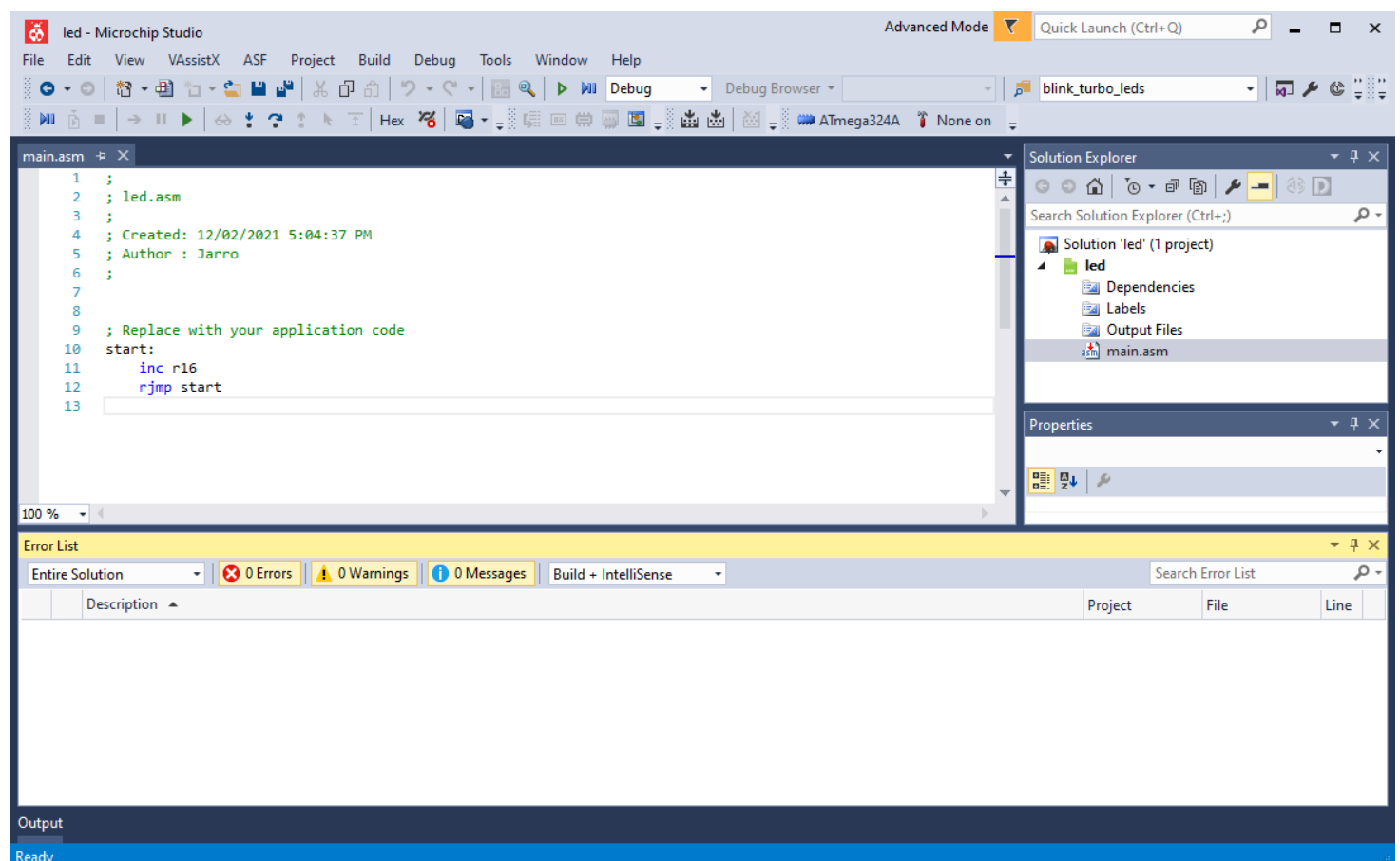


Type "324a" into the search window in the top right of the dialog window and then select "ATmega324A" as the device to use:

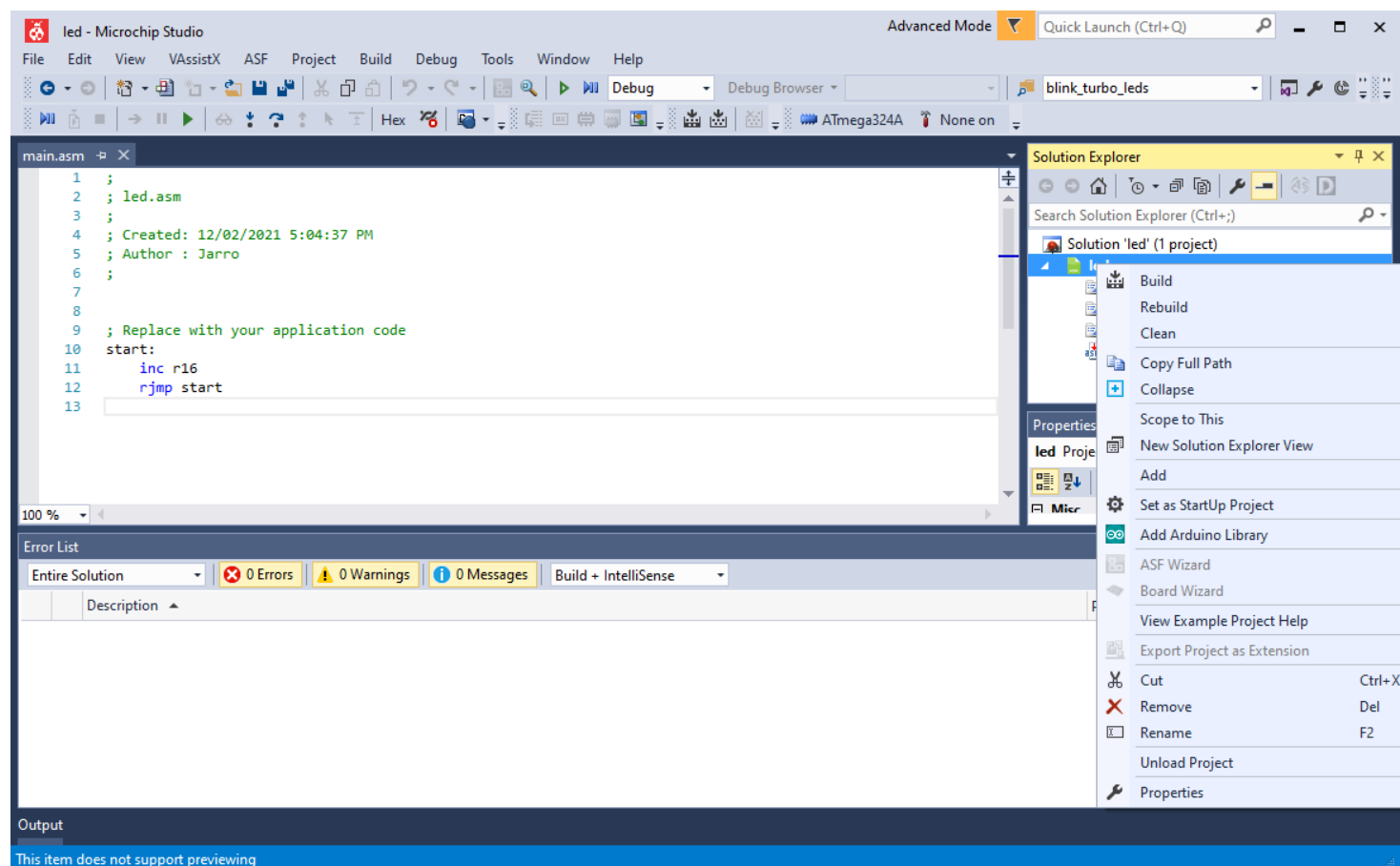


Click "OK"

The main AVR window will look something like this:



We can now enter the Assembly code of the program. (Microchip Studio has opened an editor window called main.asm). Alternatively, you can add or create other assembly language files (items) by right clicking on "led" in the Solution Explorer pane:



Editing the Assembler file

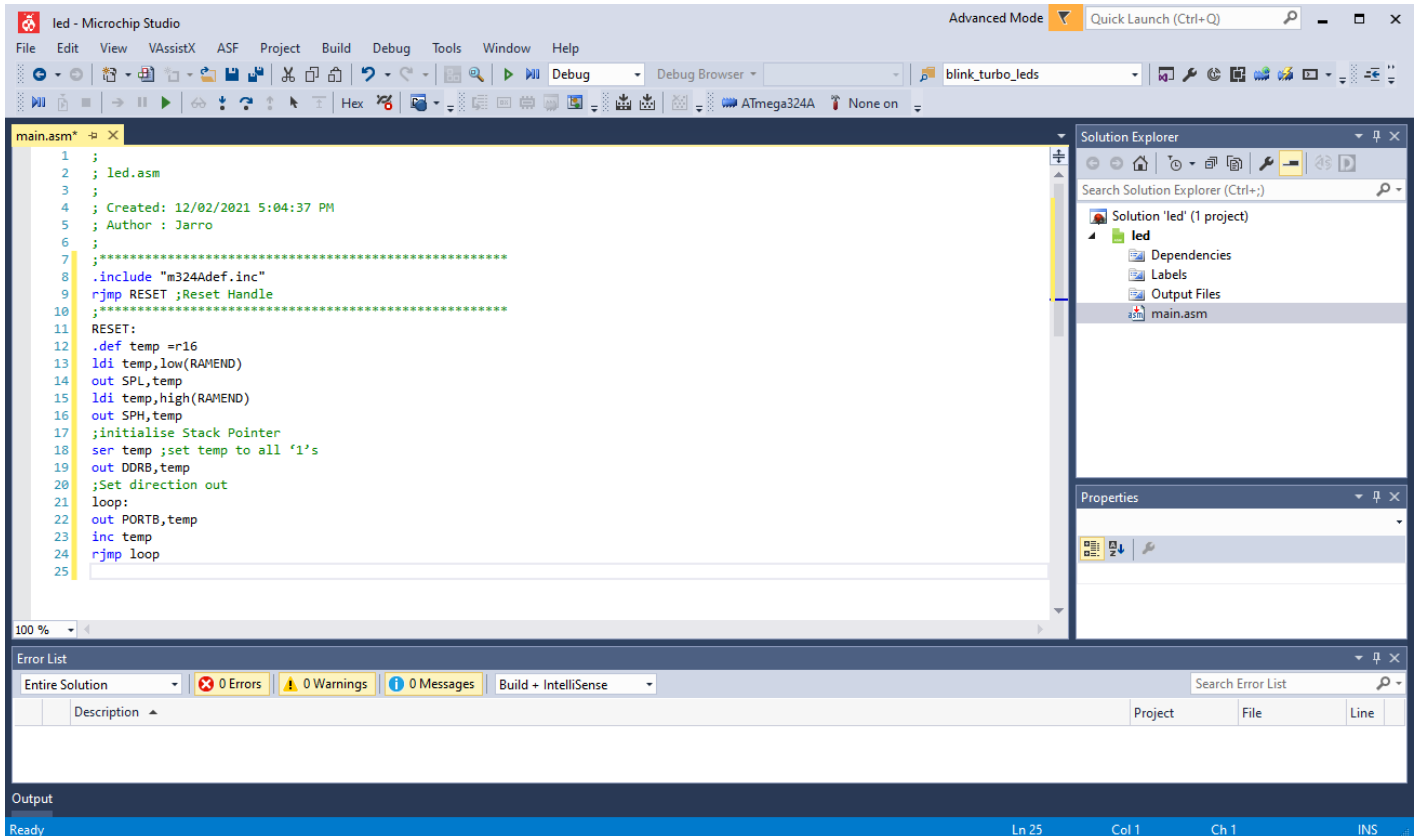
We have just added a new but (mostly) empty file to our project. The next step is to fill this file with our code. Manually enter the following code: (or you may Copy and Paste the code below directly into the editor window.)

```

;*****
.include "m324Adef.inc"
rjmp RESET ;Reset Handle
;*****
RESET:
.def temp =r16
ldi temp,low(RAMEND)
out SPL,temp
ldi temp,high(RAMEND)
out SPH,temp
;initialise Stack Pointer
ser temp ;set temp to all '1's
out DDRB,temp
;Set direction out
loop:
out PORTB,temp
inc temp
rjmp loop

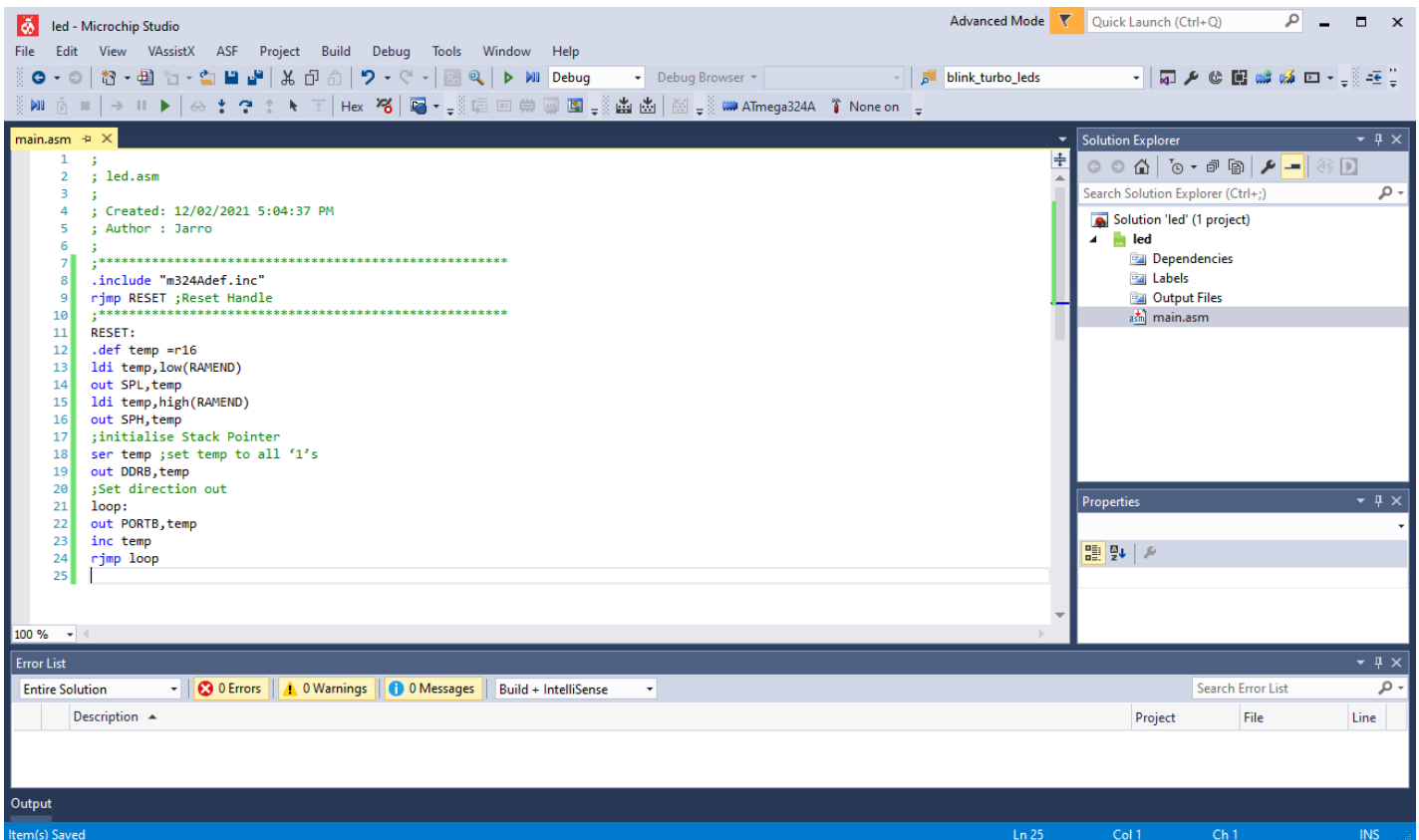
```

The Microchip Studio window should now look something like the following picture:



The Yellow bar down the left side indicates that the file has been changed but not yet saved.

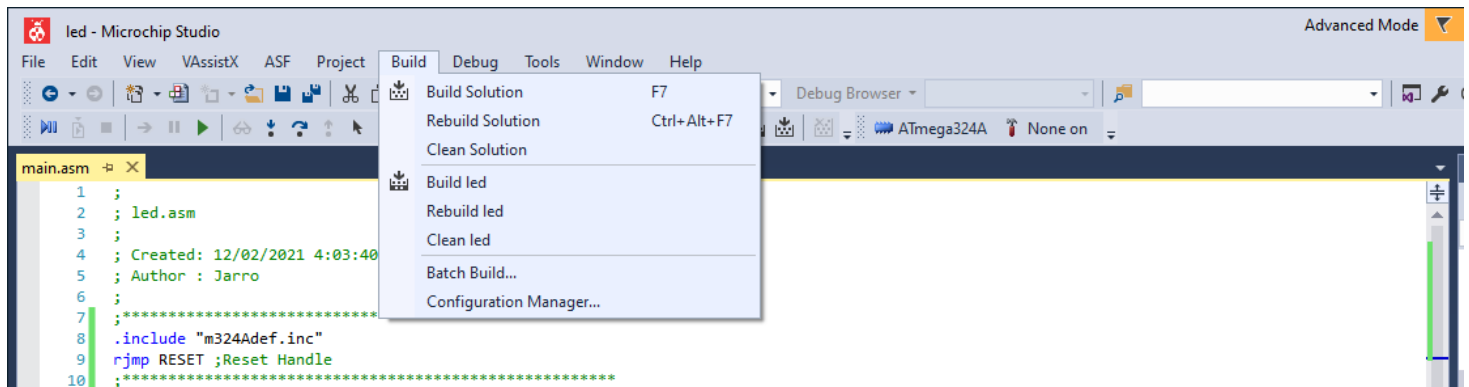
Save the file (Select "Save" from the "File" menu) and the Microchip Studio window should now look something like the following picture: Note the line has now turned Green.



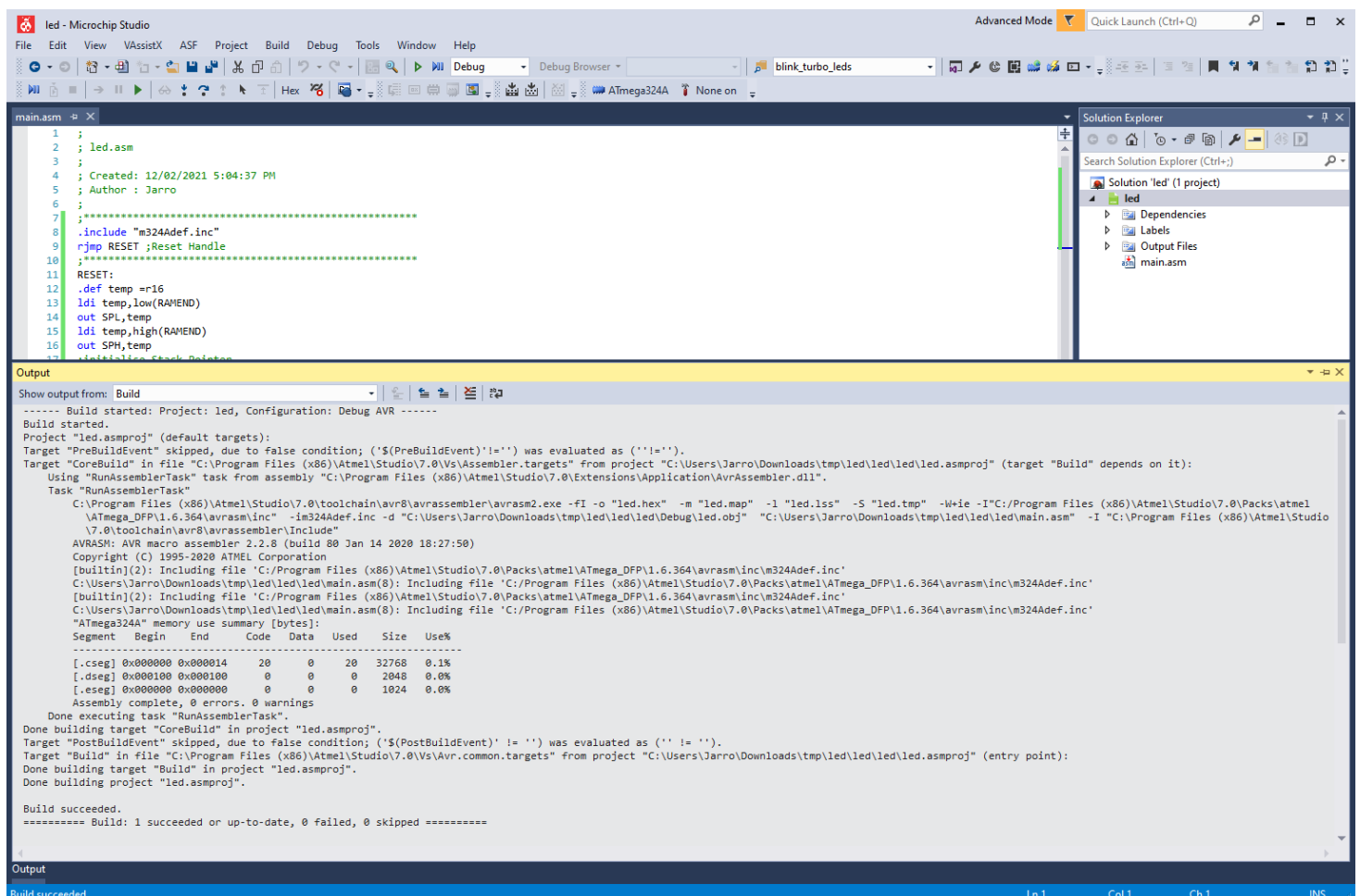
We are going to build the project and simulate the operation of the code.

Assembling (building) the Source Code

To assemble the code, we need to build the project. Select "Build Solution" from the Build menu (or press F7):




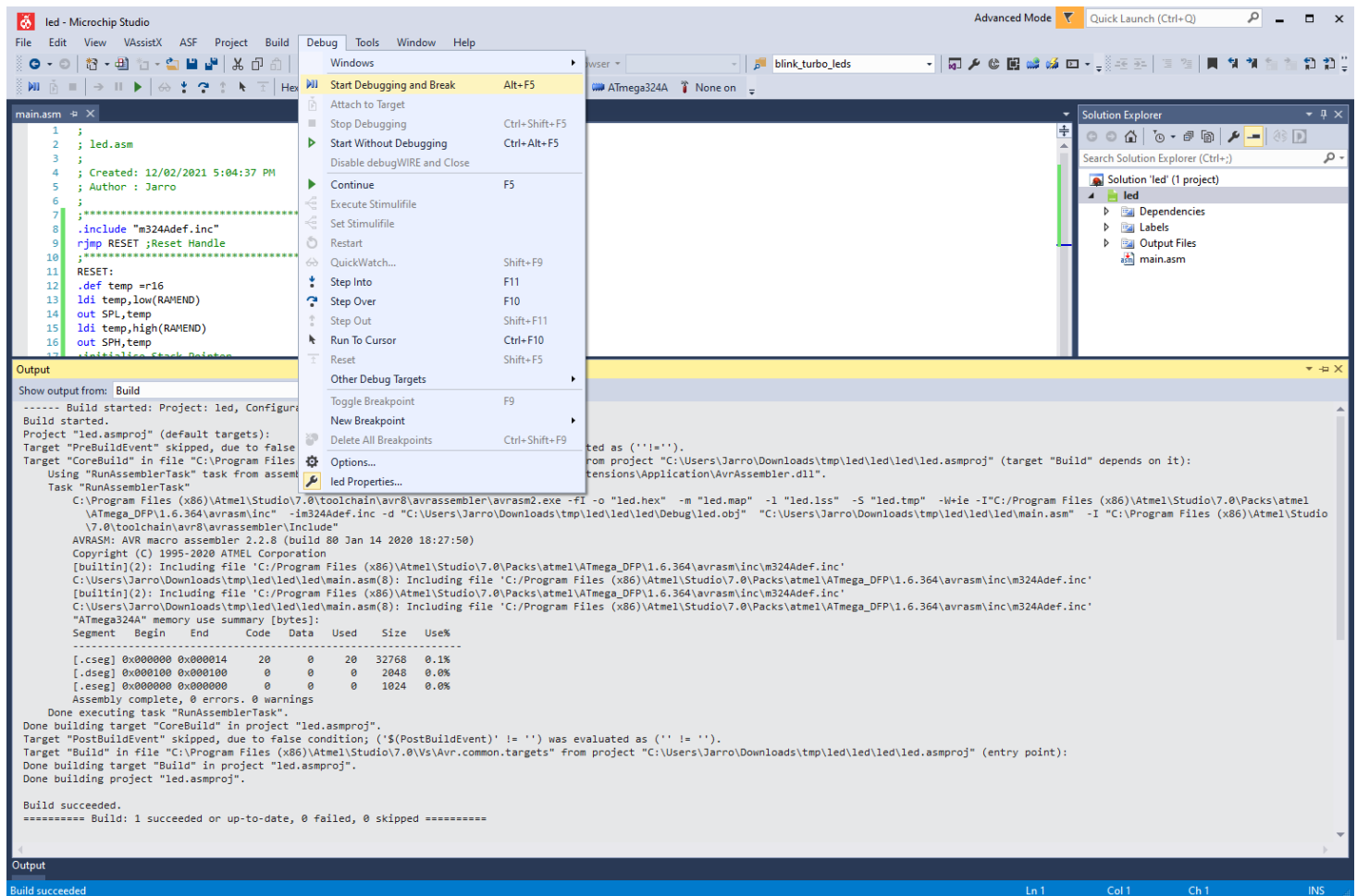
The result of building the project will be shown in the "Build" pane at the bottom of the window and will be something like:



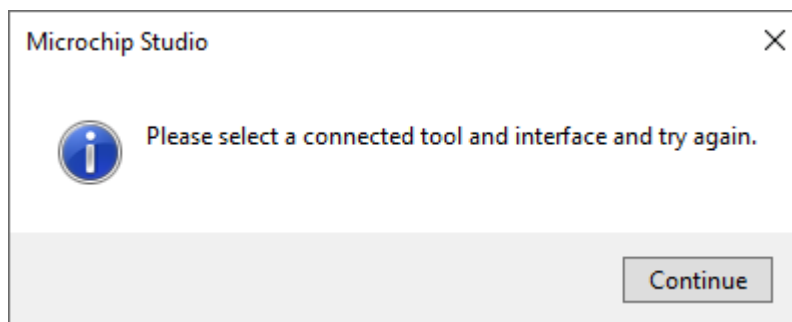
From this window we can see that the code is 20 bytes, and that assembly was completed with no errors. We are now ready to advance to the next step, which is running the code in simulator mode.

Simulating the Code

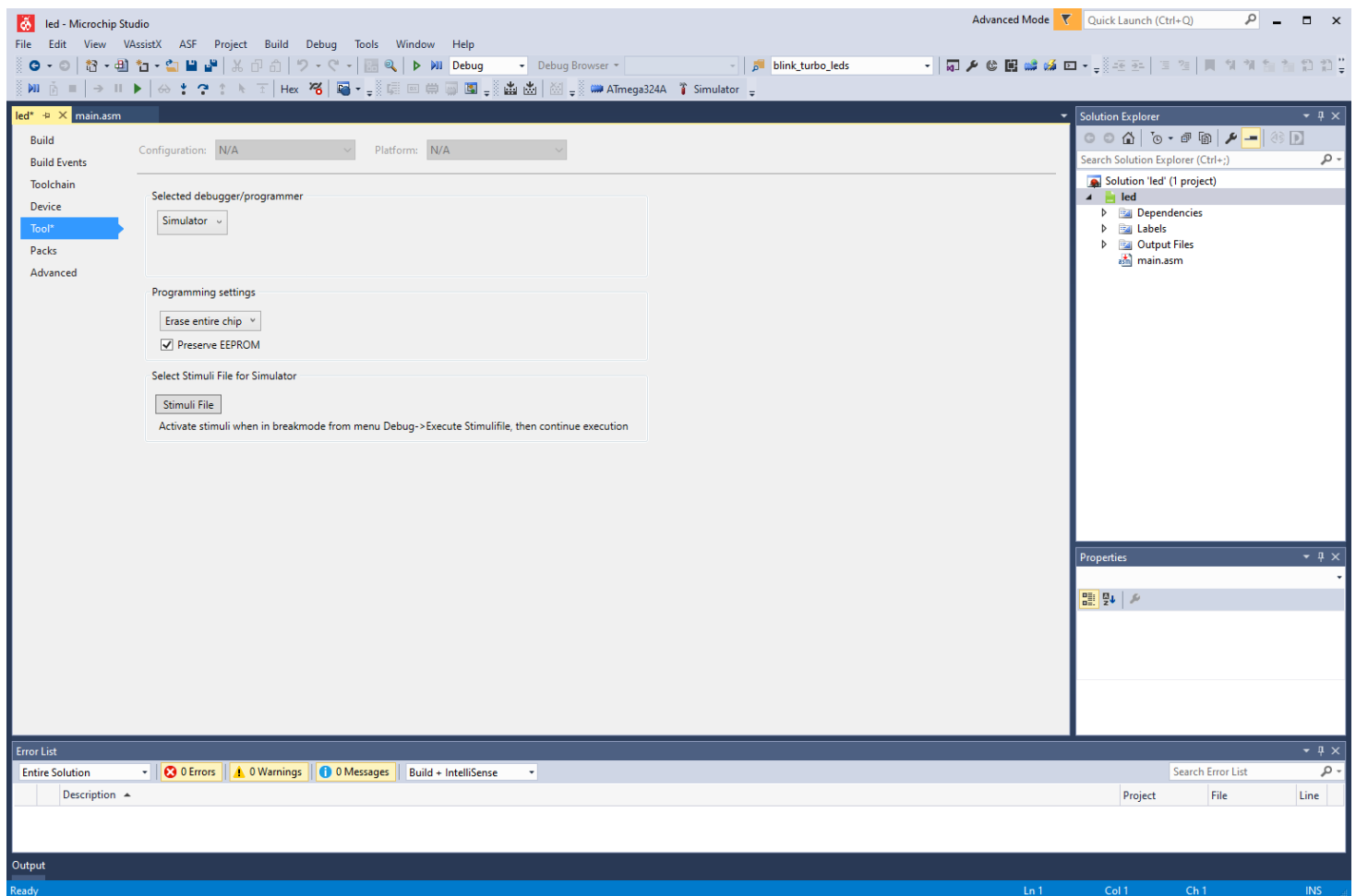
At this point we have generated the files needed to simulate the code. To start running the code, select "Start Debugging" from the "Debug" menu, press the  icon, (or press Alt+F5):



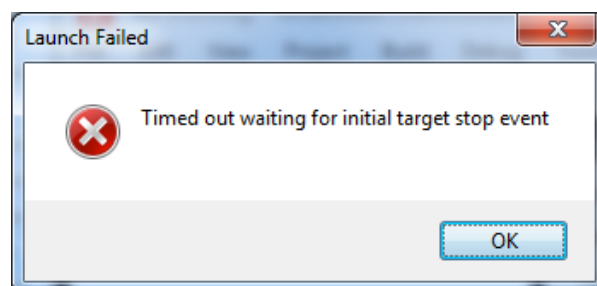
If you haven't previously selected the simulator, you may get the following message:



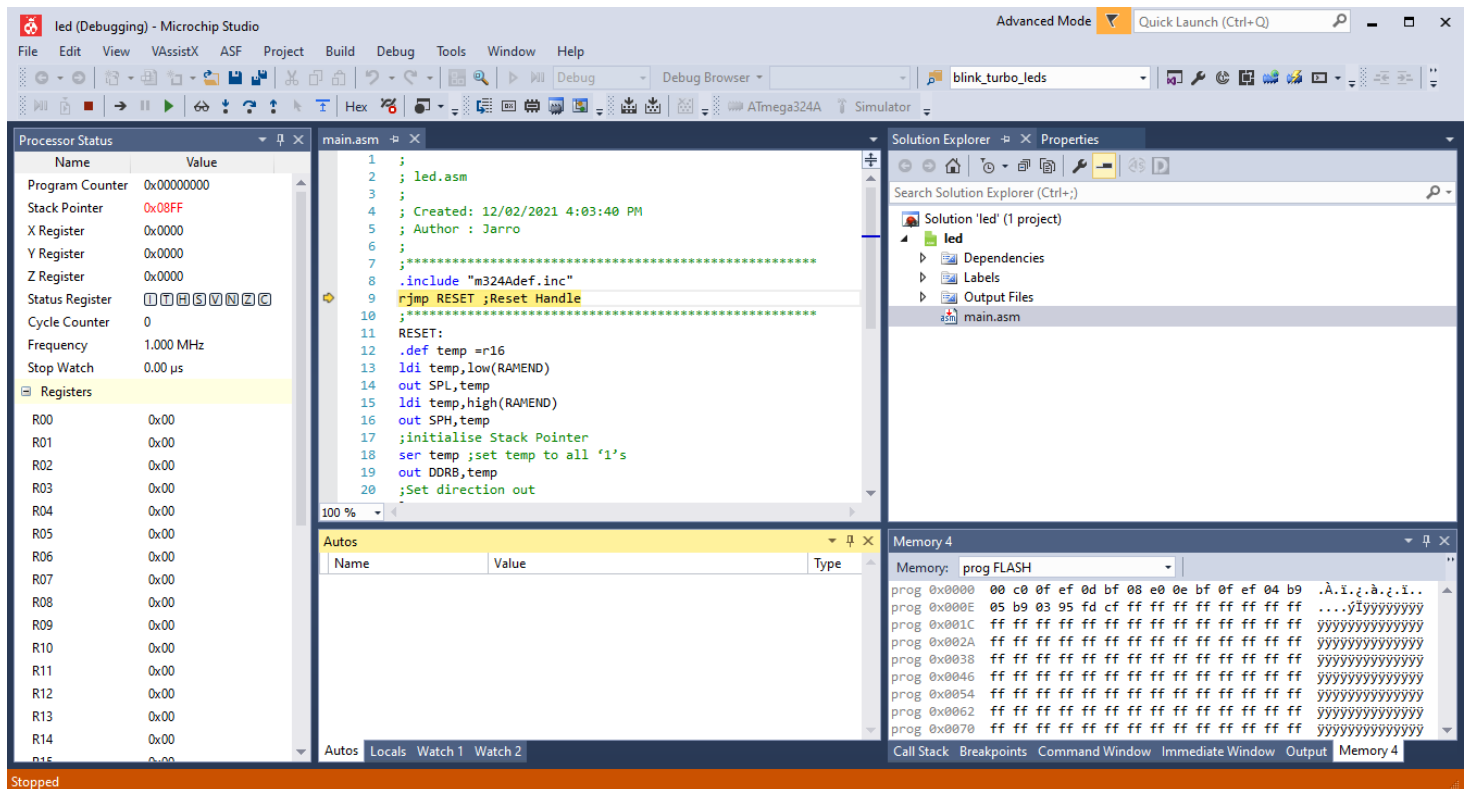
Click "Continue", then select "Simulator" under "Selected debugger/programmer":



Start debugging again, and the simulation will start. If you get the following message, press OK and try again.

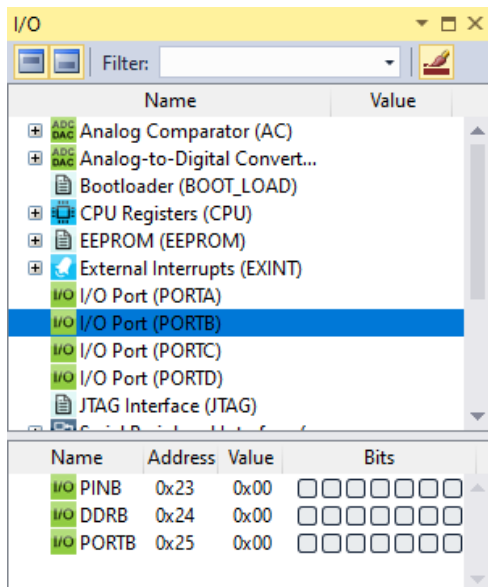


Once the simulation starts, you should see the following windows:



Instruction Pointer

Now look in the editor view for “main.asm”. You'll see that a yellow right-arrow has appeared in the left margin of the code. This arrow indicates the position of the Program Counter (PC). In other words, it points to the next instruction to be executed.



We want to set the I/O View so that we can have a closer look at what is happening on the Port B registers during program execution. If the I/O View is not visible, in the “Debug” menu select “Windows” -> “I/O”.

In the "I/O" Window, click on the green I/O symbol next to "PORTB".

Similarly, you can expand other views if desired.

This shows all registers associated with Port B.

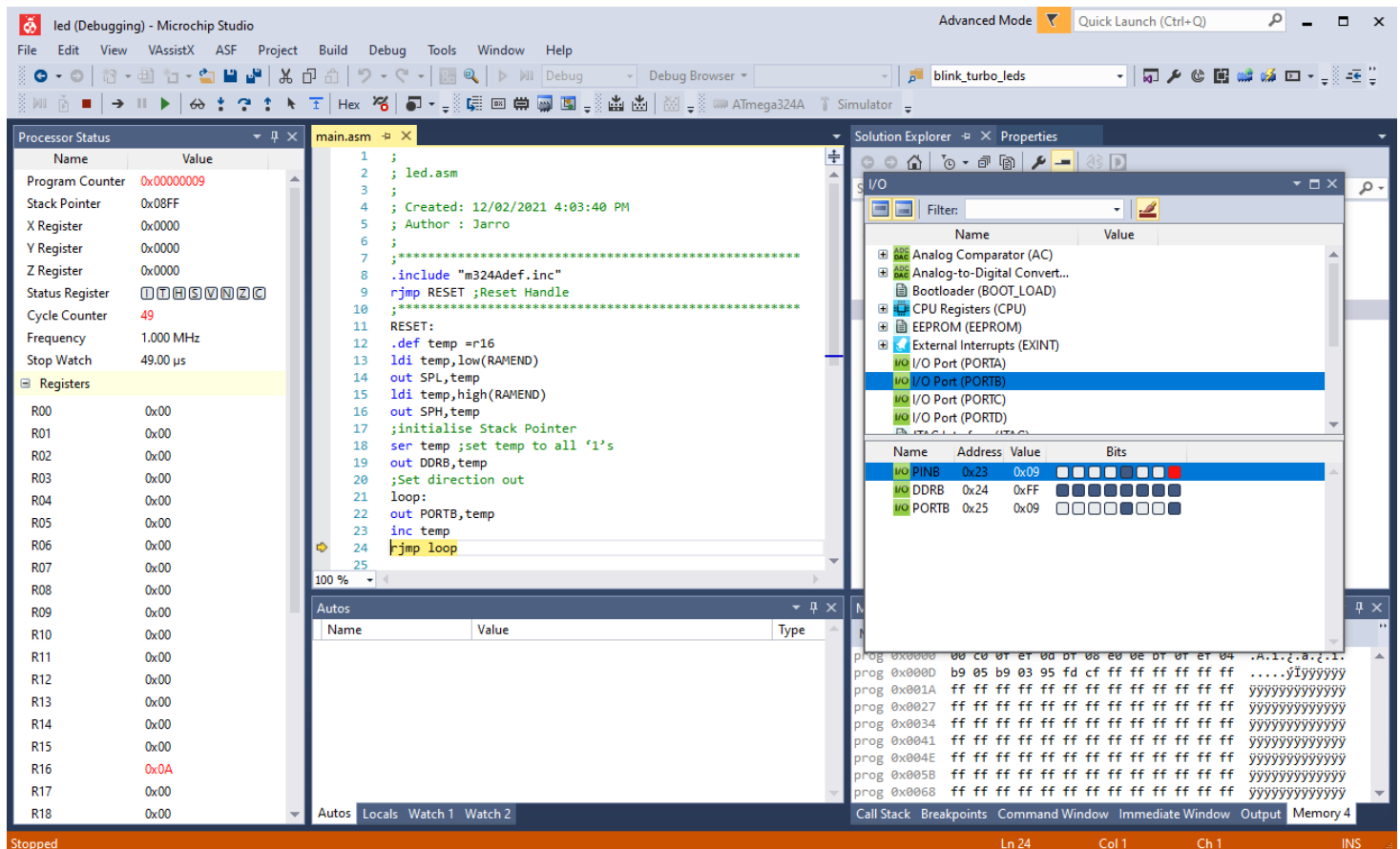
These are: Port B Data register (PORTB), Data Direction (DDRB) and Input Pins (PINB). As shown, each bit in the registers is represented by a checkbox in the panel below.

A logical 'zero' (0) is represented by an empty checkbox and a logical 'one' (1) is represented by a filled-in checkbox. These checkboxes will be updated during program execution and show the current state of every bit. You may also set and clear these bits by clicking on the appropriate checkbox at any time during the program execution. You can also monitor the hexadecimal equivalent representation.

Single Stepping the Program

There are two commands to single step through the code. These are "Step Over" <F10> and "Step Into" <F11>. The difference between these commands is that "Step Over" does not trace into subroutines (which may have hundreds of lines of code, so you generally want to avoid going through each line individually). Since our example does not contain any subroutines, there is no difference between the operation of these commands in this example.

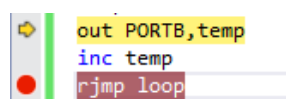
Now single step down to the last line of code (`rjmp loop`) by repeatedly pressing the <F11> key or by selecting "Step Into" from the "Debug" menu. Notice how the colour changes to red on the registers that change value. This makes it easier to identify which registers change value on each instruction. Continue pressing the <F11> key and see how the binary value in Port B is increased.



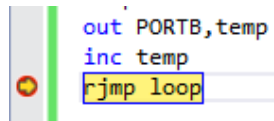
Setting Breakpoints

Breakpoints are a method of halting execution flow. By adding a breakpoint in the assembly code, we can run the program at full speed, and it will be stopped at the line with the breakpoint. By now you have noticed that you have to press <F11> three times to go through the loop once. We will add a breakpoint at the `rjmp loop` instruction to show how this can be used to speed up the debug process.

Place the cursor on the `rjmp loop` instruction in the source view window and press <F9> (or the "Toggle Breakpoint" in the "Debug" menu or right-click on the line of code and select "Toggle Breakpoint"). A red circle will appear in the left margin of the source view window as shown:

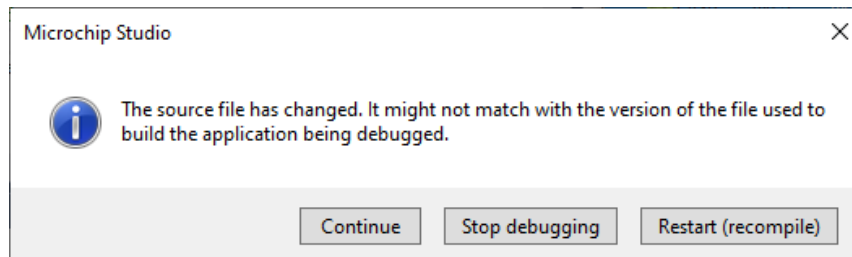


in the left image. By pressing <F5> or "Run" from the "Debug" menu the program will start running and break (stop) at the instruction with the breakpoint:



Modifying the Code

Now we want the program to count down instead of up. To make this change we'll have to edit the source code. Place the cursor in the source view, and change the `inc` to a `dec` instruction. If you now press <F5> (Run), the following dialog will appear:



This box indicates that one of the source files has been changed, and that the project should be rebuilt. Press "Restart (recompile)".

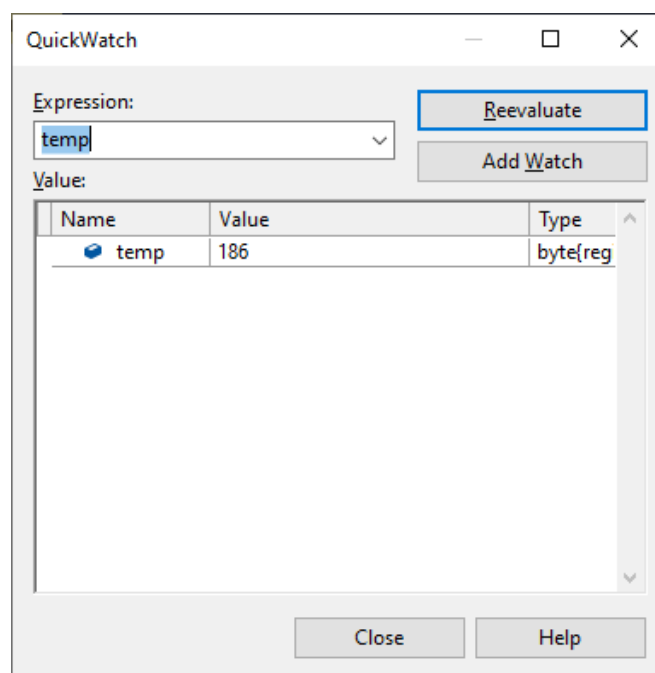
The Project will now be rebuilt. Notice how the breakpoint is remembered.

Opening the Watch View (while still in debug mode)

Select the Watch window (Watch 1 or Watch 2) at the bottom left corner of the project view:

Variables that are defined (by the `.def` directive) can be placed in the Watch view. The only defined variable in this code is the `temp` variable.

Select the variable name `temp` with the cursor and then press the QuickWatch icon, or press Shift+F9. The following window will appear:



Now click on "Add Watch" then "Close". As we continue to run through the program the `temp` variable will constantly be updated during program execution.

Setting up the Processor View

Now we will examine the Processor view. If you cannot see the "Processor Status" pane, open this pane from the "Debug" window by selecting "Windows" -> "Processor Status".

This view shows processor specific information like the value of the Program Counter. In this view you will also find a Cycle Counter and a Stopwatch. These are very useful if you wish to measure the length of a loop or how much time a specific subroutine uses (provided you set the clock frequency correctly). We will not use this view directly in this example, but it provides a lot of useful information during debugging of a project. You can also view the contents of the individual registers.

Saving the Project

Before exiting Microchip Studio, we will save our project. Microchip Studio will remember where the views are placed and will use this setting when opening the project later. To save the project, select "Save All" from the "File" menu.

Starting Over

To return to the start page at any time use the Start Page Icon, or select View -> StartPage. From there you can follow the multitude of links available.