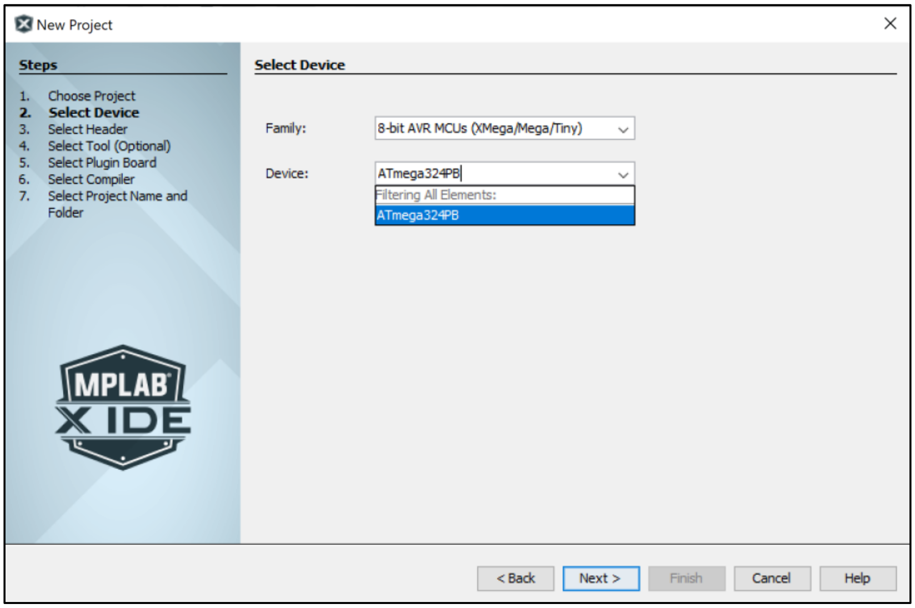
ASM PROGRAMMING IN MPLAB X

Contents

Create your Project and Source files

a. Create a new project by clicking the icon

b. Set the Family as 8-bit AVR MCUs and Device as ATmega324PB



Steps

1. Choose Project
2. **Select Device**
3. Select Header
4. Select Tool (Optional)
5. Select Plugin Board
6. Select Compiler
7. Select Project Name and Folder

Select Device

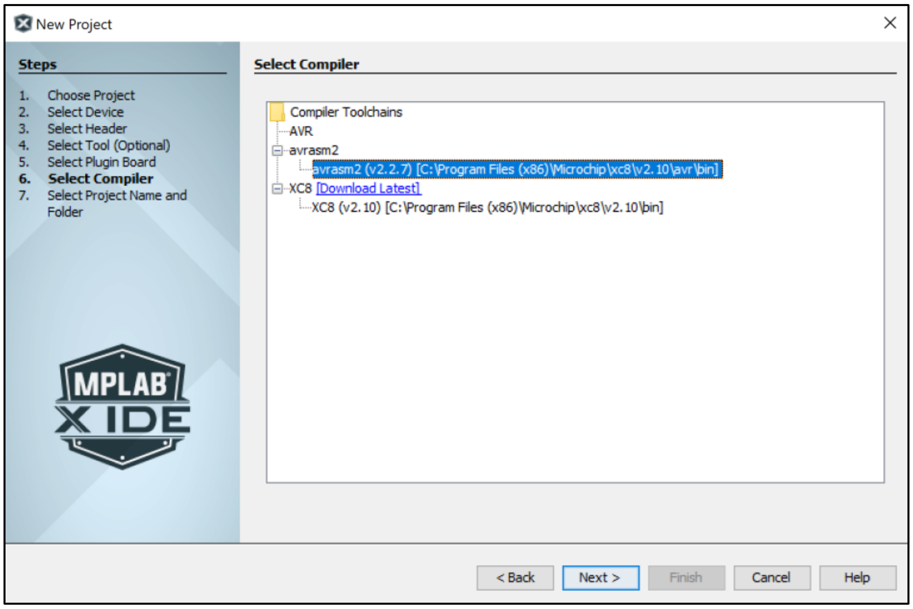
Family: 8-bit AVR MCUs (XMega/Mega/Tiny)

Device: ATmega324PB

Filtering All Elements:

ATmega324PB

c. Set the toolchain as avrasm2



Steps

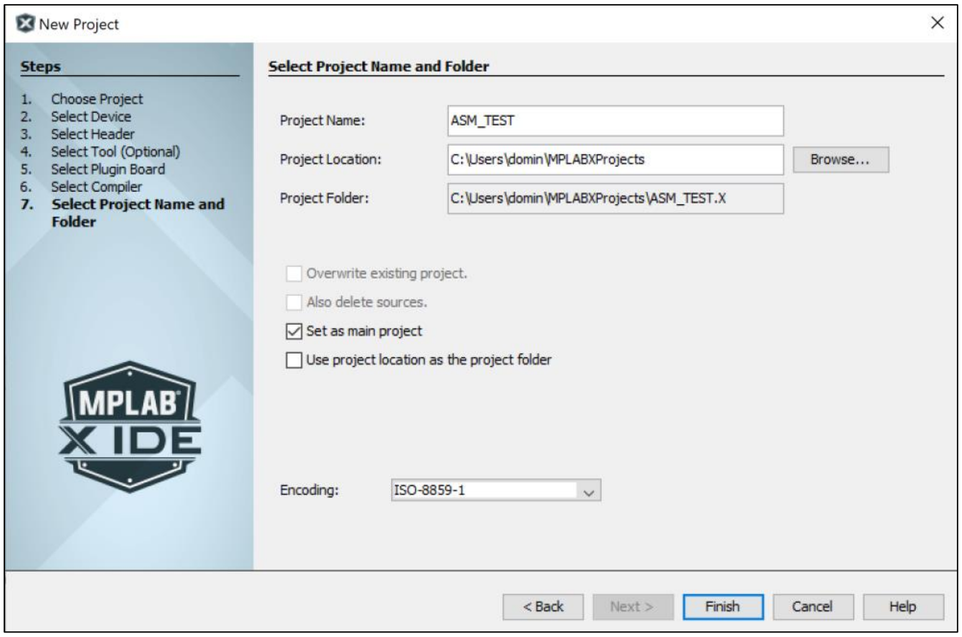
1. Choose Project
2. Select Device
3. Select Header
4. Select Tool (Optional)
5. Select Plugin Board
6. **Select Compiler**
7. Select Project Name and Folder

Select Compiler

```
Compiler Toolchains
 AVR
 avrasm2
   avrasm2 (v2.2.7) [C:\Program Files (x86)\Microchip\xc8\v2.10\avr\bin]
 XC8 [Download Latest]
   XC8 (v2.10) [C:\Program Files (x86)\Microchip\xc8\v2.10\bin]
```

i. If you do not have this toolchain available, follow the Getting Started with MPLAB Tutorial.

d. Set the project directory to a reasonable location in your file system.



New Project

Steps

1. Choose Project
2. Select Device
3. Select Header
4. Select Tool (Optional)
5. Select Plugin Board
6. Select Compiler
7. **Select Project Name and Folder**

Select Project Name and Folder

Project Name: ASM_TEST

Project Location: C:\Users\domin\MPLABXProjects

Project Folder: C:\Users\domin\MPLABXProjects\ASM_TEST.X

□ Overwrite existing project.

□ Also delete sources.

☑ Set as main project

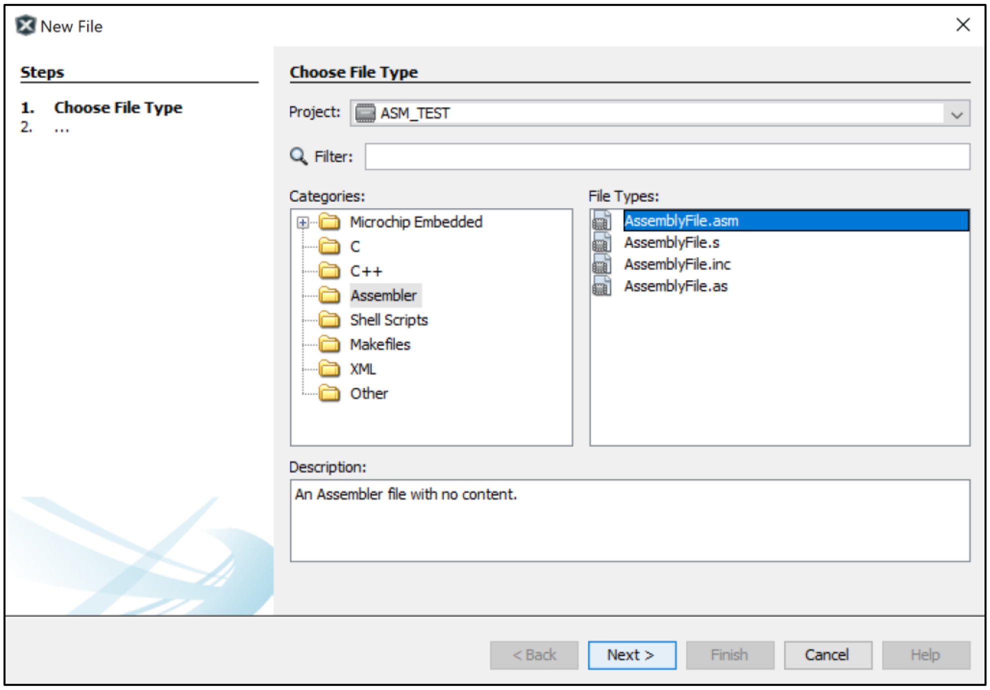
□ Use project location as the project folder

Encoding: ISO-8859-1

i. Be sure the path name contains no spaces!

e. Create a new source file by clicking the 🗋 icon

f. Chose Assembler > Assembly File



New File

Steps

1. **Choose File Type**
2. ...

Choose File Type

Project: ASM_TEST

Filter:

Categories:

- Microchip Embedded
- C
- C++
- Assembler
- Shell Scripts
- Makefiles
- XML
- Other

File Types:

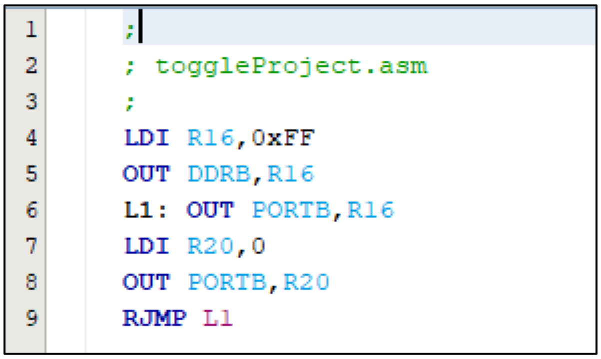
- AssemblyFile.asm
- AssemblyFile.s
- AssemblyFile.inc
- AssemblyFile.as

Description:

An Assembler file with no content.

Writing the First Assembly Project

g. Type the following code. This program sets the **Data Direction Register** for **PORTB (DDRB)** by writing 0xFF, then it loops and toggles the outputs at **PORTB**.



```
;
; toggleProject.asm
;
LDI R16,0xFF
OUT DDRB,R16
L1: OUT PORTB,R16
LDI R20,0
OUT PORTB,R20
RJMP L1
```

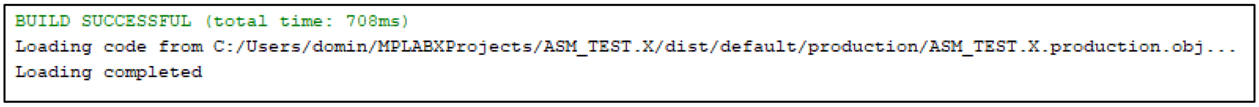
Building your Project

h. Building is like compiling in on other platforms.

i. Press the 🔨 icon to Build or the 🔨🧹 icon to clean and build.

j. Like compiling, Building will tell you if your program has any errors.

k. Building is complete when you see this message

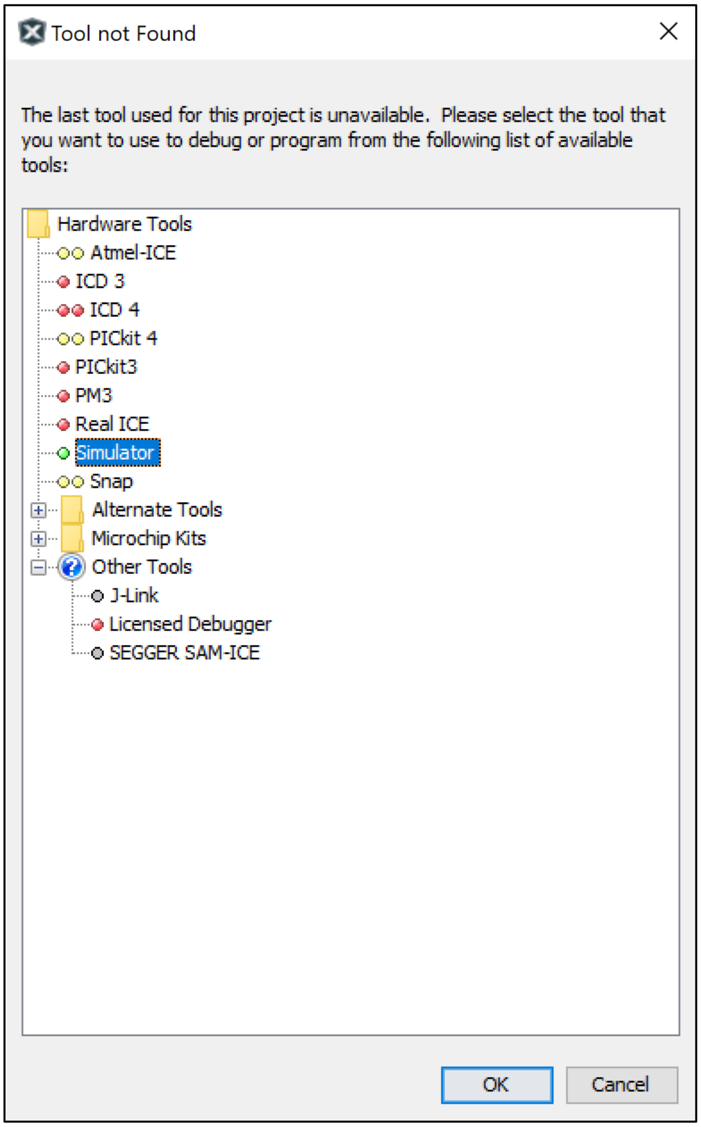


```
BUILD SUCCESSFUL (total time: 708ms)
Loading code from C:/Users/domin/MPLABXProjects/ASM_TEST.X/dist/default/production/ASM_TEST.X.production.obj...
Loading completed
```

Using the Debugger

l. Click the icon to start the debugger

m. The following dialog box will appear. Chose the Simulator.



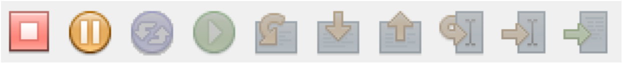
Tool not Found

The last tool used for this project is unavailable. Please select the tool that you want to use to debug or program from the following list of available tools:

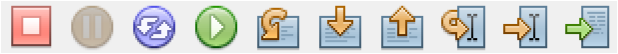
```
Hardware Tools
  Atmel-ICE
  ICD 3
  ICD 4
  PICkit 4
  PICkit3
  PM3
  Real ICE
  Simulator
  Snap
Alternate Tools
Microchip Kits
Other Tools
  J-Link
  Licensed Debugger
  SEGGER SAM-ICE
```

Note: Simulator vs. debugger

Using the simulator, you can execute the instructions, and watch the registers and variables. If you have a debugger, e.g. AVRISP mkII or Atmel-ICE, you can connect a trainer board to your computer. In the case, the microcontroller of the board executes the same instructions, when you trace the program. This facilitates you to check the hardware while monitoring the variables in the IDE.

n. This  is the debugger toolbar. While the debugger is running the only options available are to exit or pause the debugger.

o. With the debugger paused, the other options become available



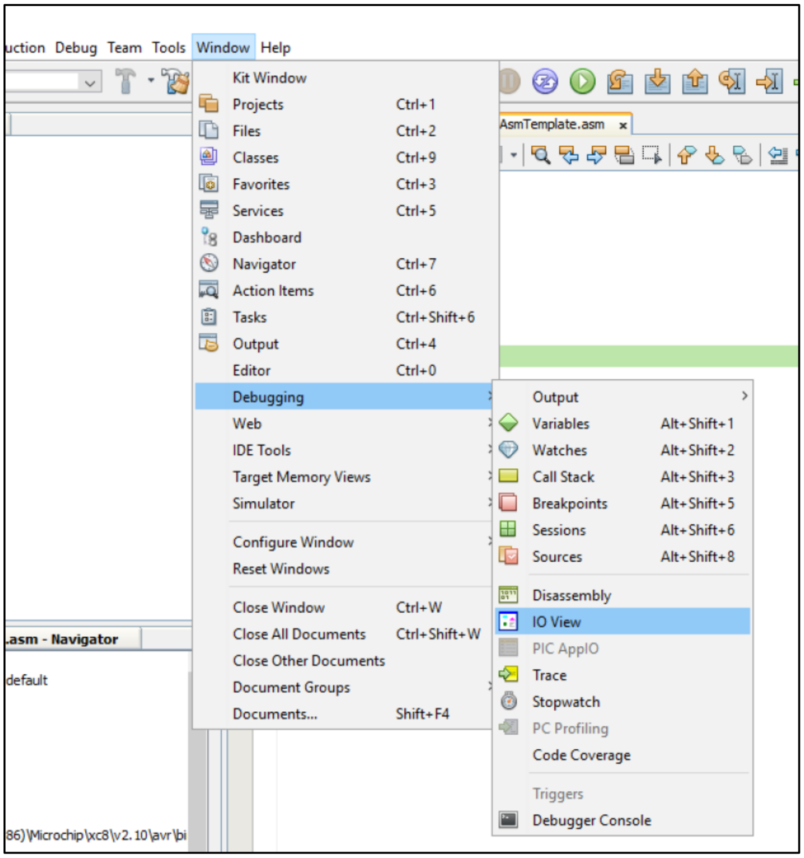
p. The buttons from left to right are Quit, Pause, Reset, Continue, Step Over, Step Into, Step Out, Run to Cursor, Set Program Counter to Cursor, Focus Cursor at Program Counter

Step Into vs. Step Over
Both F8 (Step over) and F7 (Step into) execute one instruction and go to the next instruction. But they work differently when the cursor is on a function call. If the cursor is on the function call, Step into goes into the first instruction of the function, but Step Over executes the whole function and goes to the next instruction.

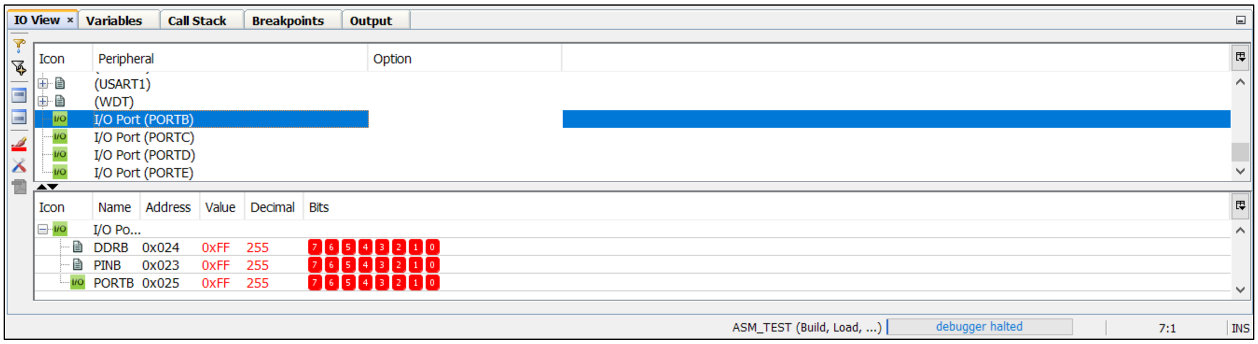
Step Out
If the execution is in a function, you can execute the function to the end by pressing the Step Out.

Run to Cursor
You can put the cursor on an instruction and then press the Run to Cursor button. In the case, the program runs until it reaches the instruction which the cursor is on it.

q. To monitor peripherals like I/O, click window, debugging, I/O view



r. The **I/O** tab appears on the bottom which shows the peripherals of the microcontroller, including the **I/O** ports. Select **PORTB**. The values of the related registers (**PINB**, **DDRB**, and **PORTB**) will be shown below.



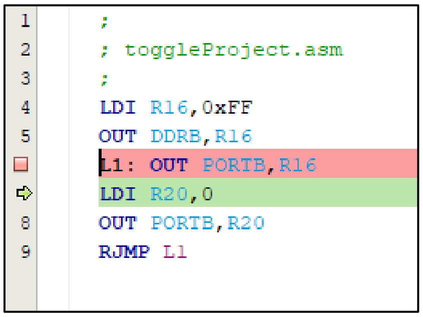
Icon	Name	Address	Value	Decimal	Bits
I/O	I/O Po...				
	DDRB	0x024	0xFF	255	7 6 5 4 3 2 1 0
	PINB	0x023	0xFF	255	7 6 5 4 3 2 1 0
I/O	PORTB	0x025	0xFF	255	7 6 5 4 3 2 1 0

s. Press **F8 (Step Over)** a few times and see the **PORTB** register changes in the **I/O** window.

Using Breakpoints

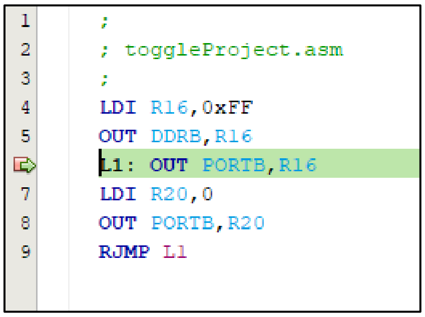
t. If you want to debug a portion of a program, add a breakpoint to the beginning of this part of the code and press the run button. The IDE runs the program and when it reaches the breakpoint, it stops running and the yellow cursor is shown on the breakpoint line. Below, you see the steps in detail.

u. With the debugger active and halted, click on the line you wish to insert a breakpoint.



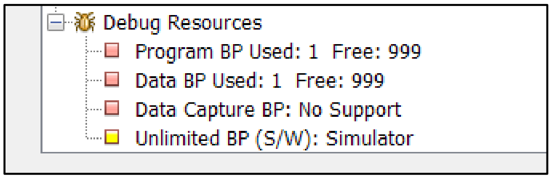
```
;
; toggleProject.asm
;
LDI R16,0xFF
OUT DDRB,R16
L1: OUT PORTB,R16
LDI R20,0
OUT PORTB,R20
RJMP L1
```

v. Press **Continue (F5)** to resume the debugger. The debugger will halt when the Program Counter reaches that instruction.



```
;
; toggleProject.asm
;
LDI R16,0xFF
OUT DDRB,R16
L1: OUT PORTB,R16
LDI R20,0
OUT PORTB,R20
RJMP L1
```

w. The bottom left corner of the screen shows how many breakpoints are in use and available for use.



Debug Resources
- Program BP Used: 1 Free: 999
- Data BP Used: 1 Free: 999
- Data Capture BP: No Support
- Unlimited BP (S/W): Simulator