

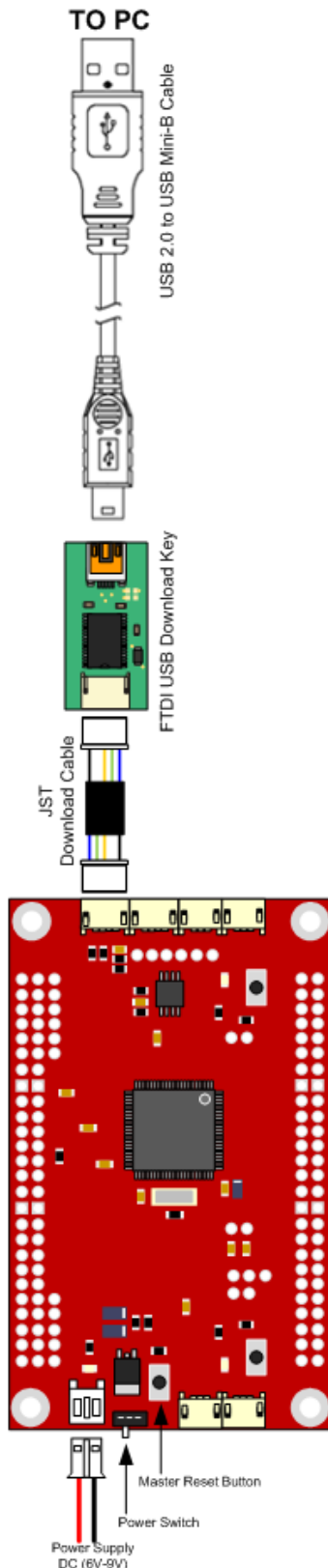


[THE BLAZINGCORE SERIES]

Getting Started

BASIC SETUP

BCORE100 REV.B



To perform the most basic of functions, you would require:

HARDWARE:

- BCore Microcontroller Board (Rev.A to Rev.D)
- A 6V-9V DC power supply
- FTDI USB Download Key

SOFTWARE:

- BCore IDE
- CommKey Driver
- .NET Platform

Note: All required software is available for download from www.aiscube.com.

Instruction Steps:

1. Setup the BCore with reference to the connections shown in the figure on the left.

2. Connect your Download Key to any USB Port on your PC.

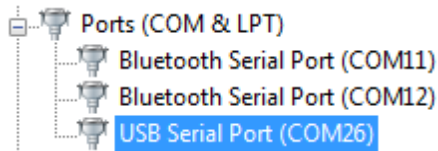
The download port is the JST connector located at the outermost edge of the board for all BCore boards. Please refer to the BCore User Manual for more details.

NOTE: Please install the Download Key driver **BEFORE** connecting the Download Key to the PC. If you have not already done so, please install the latest version of the BCore IDE, you may need to install the .NET Platform if your PC does not already have it in the system (Not required for Windows® Vista onwards). The driver and software can be obtained from our website.

3. Power up the Board.
4. Launch BCore IDE.

LAUNCHING BCORE IDE

Upon launching BCORE IDE (BlazingCore Integrated Development Environment), a dialog should pop up requesting the Comm. Number. The Comm. Number can be obtained from your Device Manager when you have plugged in your Download Key.



You can get the port numbers through:

Control Panel > System > Device Manager > Ports

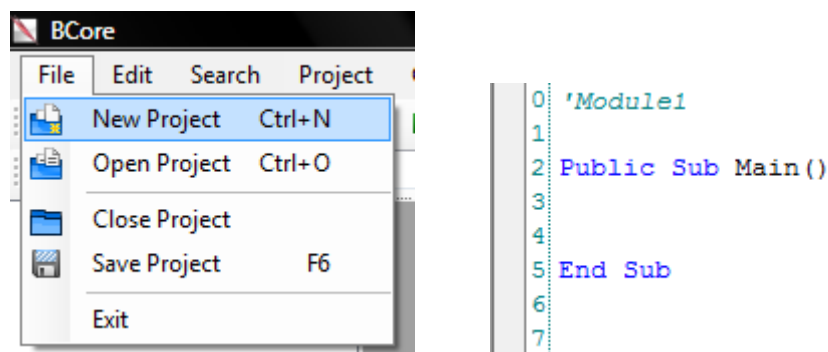
Look for **USB Serial Port**. In this case, the port number is COM26.

Press **OK** when you're done and the port should automatically open if the port number is valid. There are indicators on the bottom status bar stating "COM26 Open".

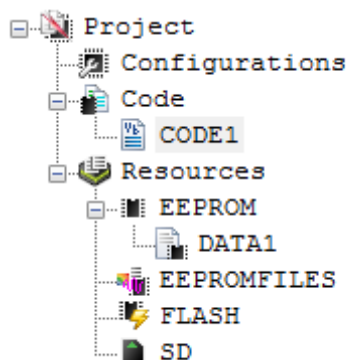
Note: You may locate the CommPort Setting under: **Comm > Comm Setting**

CREATING A NEW PROJECT

To create a new project, go to **File > New Project**. Keyboard Short-Cut: **CTRL + N**.



A new project will be automatically created for you, along with the standard structures of a program in the editor.




A project consists of your code, which is contained in Modules. You will be writing your program in Modules. The Module which contains the subroutine "Main" is your Main Module. Every Program **MUST** have a subroutine called Main(), and each subroutine has to have a unique name.

There should only be one Main Subroutine for every Project.


TIP: You can rename your modules to whatever you want to call it. This can be done under: **Project > Rename**.

SAVING YOUR PROJECT

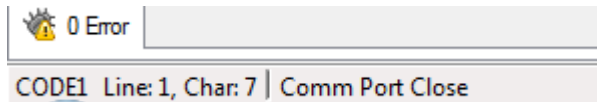
Before continuing, let's save this project. This can be done under **File > Save Project** or clicking on the  Save Icon on the toolbar. The Save Dialog should appear. Type in the project name that you would like to save as, e.g. "My Project" and hit the save button when you are done.

Keyboard Short-Cut: **CTRL + S**.


COMPILING YOUR PROJECT

You can now compile your project by selecting **Project > Compile** or clicking on the  Compile Icon on the toolbar. Keyboard Short-Cut: **F4**.

Compiling the project checks for any syntax errors in your program, you should not encounter any errors here. You can tell the number of errors you currently have, from the bottom indication in BcoreIDE. For now, you should see something as follow.



COMPILING AND DOWNLOADING YOUR PROJECT

Since there are no errors in our project, let's download it into the BCore. From the BCoreIDE menu choose: **Project > Compile And Download** or press the  icon on the tool bar. Keyboard Short-Cut: **F5**.

You may have to wait for a moment for the download process to complete. Your download is successful if there are no errors reported.

If you get an error message, such as: **"Download Error"**

Check the following:

- Make sure your power supply is on. The power indicator on your BCore board should light up.
- Make sure that the Download Key is connected properly to the BCore board and to your PC.
- Make sure you have the right Comm Port setting.

Important!

By default, programs downloaded are stored in the Core's SRAM. This memory is volatile, meaning that if you power off the board and power it up again, your program will be gone.

Once you are ready and you have finalized your program, you may save the program into the onboard DataFlash chip. On power up, the Core will take the program from the DataFlash chip, transfer it into its internal SRAM and run your program.

To save your program into the DataFlash Chip, you need to perform the following 2 steps.

1. **Project > Compile And Download**
2. **Project > Save Code Mem to Ext OS Mem**

HELLO WORLD!

THE HELLO WORLD PROGRAM

Congratulations, you have successfully created, compiled and downloaded your first FI Project. Now for more VB coding.

The language you will be using is VB.NET (Visual Basic); there are a number of differences as you will see when you work more with BCoreIDE. One of which is that the language is streamlined to be similar to that used when programming on the .NET platform and style instead of the conventional VB language.


```
0 'Module1
1
2 Public Sub Main()
3
4 Debug.Print "HELLO WORLD"
5
6
7 End Sub
8
9
```

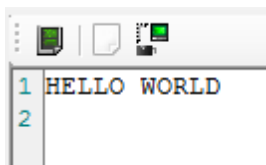
In the BCoreIDE Editor, type the following;
`Debug.Print "HELLO WORLD"`

Note: It is best to leave your Caps Lock ON. The Editor will automatically format your text as you type.

`Debug.Print` is a BCore OS command that sends the message between the quotes back to the PC through the Serial Download Port. In this case it sends "HELLO WORLD" to the PC.

Now Download and Compile your project.

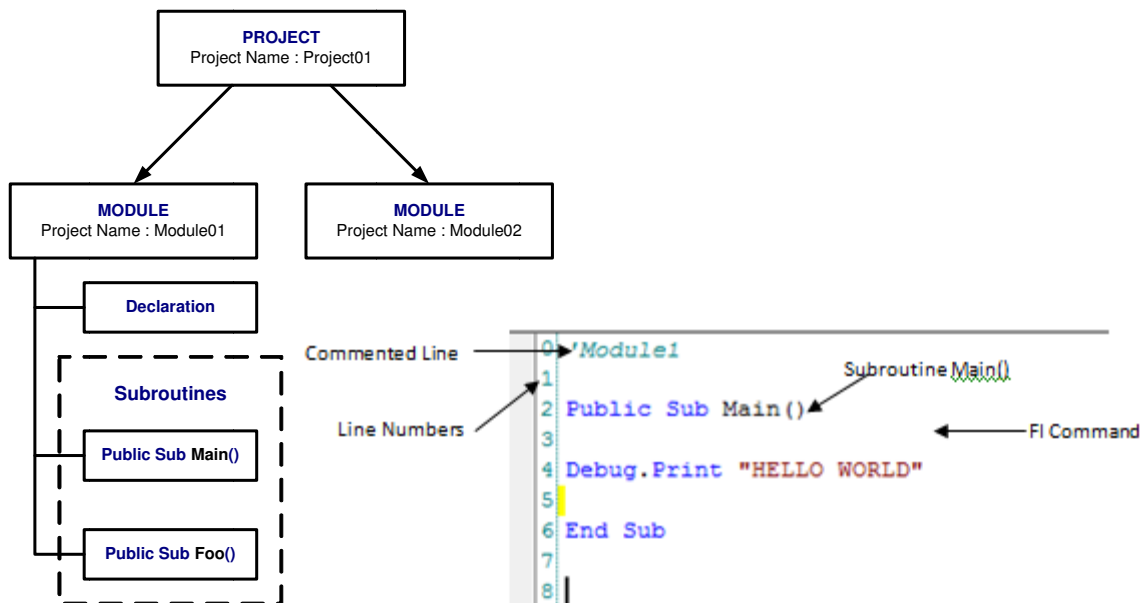
(Reminder: you can Download and Compile your project by pressing F5 or clicking the  icon.)



After Compiling and Downloading your Project, you should see the message "HELLO WORLD" appearing at the debug window. The debug window is located to the right of your editor.

UNDERSTANDING THE HELLO WORLD PROGRAM

A Project consists of one or more modules. Each Module consists of a declaration and one or more Subroutine. Every Project must have the Main() subroutine.



BLINKY LEDs!!

THE BLINKY LED PROGRAM

Here's another simple program. If you have an LED module, you can turn on and turn off the LED by using the following OS commands.

HIGH

Set logic 1 at the specified pin.

HIGH [pin no.]

LOW

Set logic 0 at the specified pin.

LOW [pin no.]

NOTE: I/O pins are tied low to the ground on the hardware therefore explaining the inverse result as opposed to the logic statements.

Write the following program and when you're done, Compile and Download it.

```
0  'Module1
1
2  Public Sub Main()
3
4  Debug.Print "HELLO WORLD"
5
6  Low(x)
7  Delay(200)
8  High(x)
9
10 End Sub
11
```

Replace 'x' with the pin number assigned to the IO pins on the BCore Board to which you connected the LED. You can obtain these numbers by referring to the BCore Reference Documentation (Hardware) for your respective BCore Board.

Hint: You can restart the program by pressing the master reset button on the BCore board.

After downloading the program, you will find that the LED blinks once for 200 msec.

UNDERSTANDING THE BLINKY LED PROGRAM

The Program starts by sending "HELLO WORLD" to the PC. Then the LED will be turned on. The program waits for 200 msec (or 0.2 sec) before turning the green LED off. And that's it!

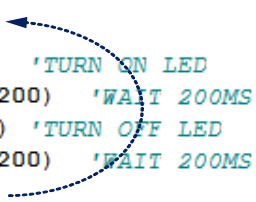
BLINKY LEDs!! (CONT'D)

BASIC STRUCTURE: DO... LOOP

In the previous program, we will need to type 3 times to blink the green LED 3 times. Or maybe we can just cut and paste the 3 lines. What do we need for the BCore to instruct the LED to blink indefinitely? What we need is to implement is a structure called Do...Loop.

Type in the following program and read the explanation before downloading it.

```
0 'Module1
1
2 Public Sub Main()
3
4 Debug.Print "HELLO WORLD"
5
6 Do
7     Low(x) 'TURN ON LED
8     Delay(200) 'WAIT 200MS
9     High(x) 'TURN OFF LED
10    Delay(200) 'WAIT 200MS
11 Loop
12
13 End Sub
14
15
```

A diagram illustrating the Do...Loop structure. A dashed arrow starts at the 'Do' keyword on line 6, loops around the four lines of code between 'Do' and 'Loop', and points back to the 'Do' keyword, indicating that the code between them is repeated.

The Do Loop repeats all commands between the Do and the Loop.

Note the *Delay(200)* Command After the *High* Command. This is required before it executes the next instruction, which is *Low(x)*, not having a delay would result in the LEDs turning on and off so fast so fast, (Remember, the BCore is able to execute instructions at up to 5 Million lines of instructions per second), that the human eye will not be able to see the LED Blinking.

LATEST DOCUMENTATION

All of our documentations are constantly updated to provide accurate and/or new information that we feel would help you with developing with our products.

The latest documentation may be obtained from our website: <http://www.aiscube.com/main/downloads.html>

HOW YOU CAN HELP

You can help us to improve our documentations by emailing to us or posting a thread in our forum, reporting any mistakes/typos or errata that you might spot while reading our documentation.

Email: TechSupport@aiscube.com

DISCLAIMER

All information in this documentation is provided 'as-is' without any warranty of any kind.

The products produced by AIS Cube are meant for rapid prototyping and experimental usage; they are not intended nor designed for implementation in environments that constitute high risk activities.

AIS Cube shall assume no responsibility or liability for any indirect, specific, incidental or consequential damages arising out of the use of this documentation or product.

COPYRIGHT© 2009 - 2011 AIS CUBE. ALL RIGHTS RESERVED.

ALL PRODUCT AND CORPORATE NAMES APPEARING IN THIS DOCUMENTATION MAY OR MAY NOT BE REGISTERED TRADEMARKS OR COPYRIGHTS OF THEIR RESPECTIVE COMPANIES. AND ARE ONLY USED FOR IDENTIFICATION OR EXPLANATION FOR THE OWNER'S BENEFIT. WITH NO INTENT TO INFRINGE.

BCORE IDE AND BLAZINGCORE(BCORE) ARE TRADEMARKS OF AIS CUBE IN SINGAPORE AND/OR OTHER COUNTRIES. ALL IMAGES DEPICTING THE BLAZINGCORE OR ANY PART OF IT IS COPYRIGHTED.

ALL OTHER TRADEMARKS OR REGISTERED TRADEMARKS ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS.