SABT Programming Guide v2

21st April, 2015

# Table of Contents

[Table of Contents](#h.k9ag67hlxu1m)

[Overview](#h.u006zmmecvqi)

[Development Environment](#h.b3nbbnjg78b4)

[Operating System](#h.i20wqb2xnnyz)

[Development IDE](#h.ch660ygx1gp2)

[Development Hardware](#h.97xqn7b2cwu4)

[Codebase Management](#h.ehp05tt70o94)

[Programming the SABT](#h.henv3kg0pwd5)

[Debugging the SABT](#h.1fpnwughbry5)

# Overview

This guide briefly documents how to compile the SABT firmware and program it on to the device. This can be done in either a Windows, OS X or Linux environment. This guide only covers the Windows development environment. However, even if you are using a different environment, certain device-specific settings are still applicable.

# 

# Development Environment

## Operating System

As mentioned before, this guide primarily focuses on setting up a development environment in Windows. If you do not have a Windows environment handy, you can either choose to use third party and open source tools such as avr-gcc, avrdude and minicom to achieve results similar to those discussed below. Another option is to install Windows on your machine and proceed. A third option is to use a virtual machine on top of your existing environment and install Windows on that VM. All of these approaches have their own pros and cons. Ultimately, it comes down to personal preferences, performance issues and license limitations. If you choose to go one of the undocumented routes, please expand this guide with your findings.

## Development IDE

Since the SABT’s primary microcontrollers are manufactured by Atmel Corporation, Atmel Studio is the IDE of choice. At the time of writing this guide, version 6.2 was the latest release and will be the one this guide recommends. However, this is not to say that similar settings will not exist in future iterations of the IDE. The IDE can be downloaded from the following link: <http://www.atmel.com/tools/atmelstudio.aspx>

## Development Hardware

In addition to a SABT and development environment, you will need to make sure you have the following equipment:

* Atmel JTAGICE mkII
* Atmel AVRISP mkII (or a JTAG to SPI adapter)
* 1 USB B-to-A male-male cable (2 if you want to debug over USB)

## Codebase Management

The most recent stable release of the firmware can be found here: <https://github.com/TechBridgeWorld/SABTSoftware>. It is suggested that you fork the repository for development and issue consolidated single-commit pull requests instead of cluttering the repository’s commit history with several commits, many of which may be trivial. In order to do this, once you’re done with a logical set of commits, such as a new feature or mode, rebase them into a single commit and issue a pull request that describes the changes in brief.

**Debugging**

The SABT is debugged via print statements delivered to the computer via USB and a serial port client. You need to download PuTTY or another equivalent.

# 

# Programming the SABT

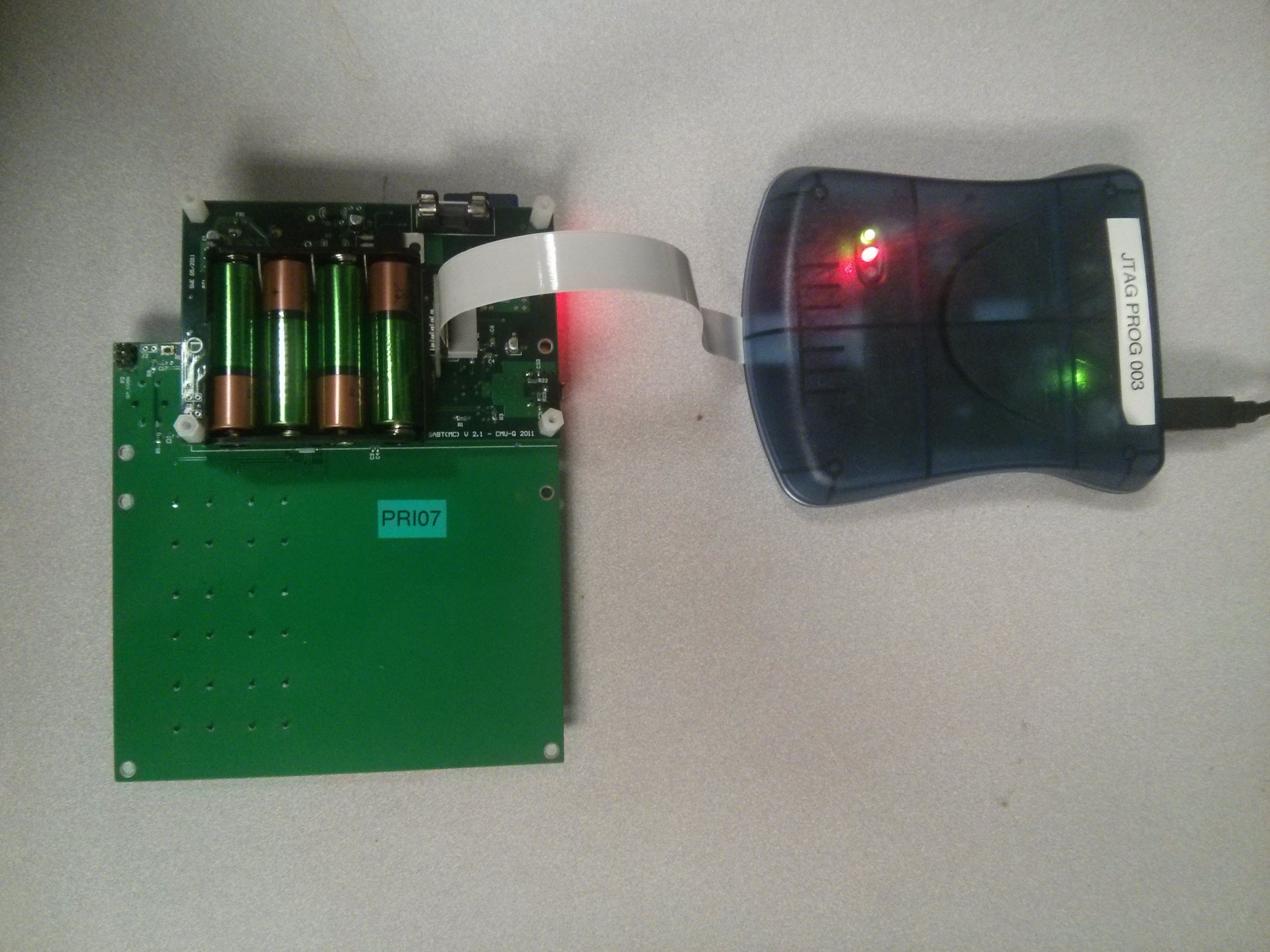
1. Open the solution file (SABT.atsln) in Atmel Studio.
2. Verify that you can compile the MCU firmware. To do this, open the appropriate project in the *Solution Explorer* pane on the right.
   * For the MCU, open the *SABT\_MainUnit* project.
   * For the user interface boards, open the corresponding *UserInterface* project.

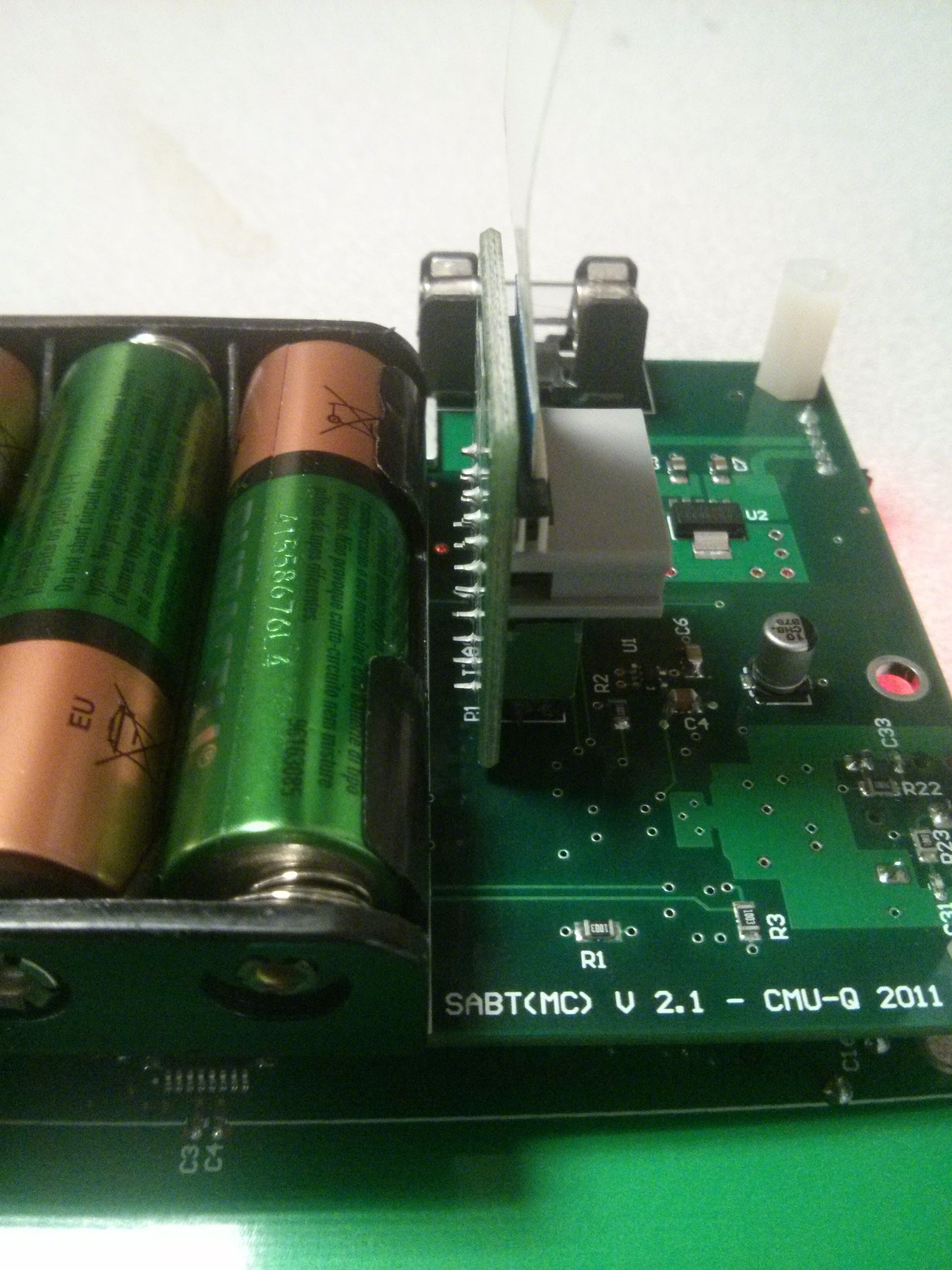
Then select *Build Solution* from the *Build* menu. All projects should ideally compile without fatal errors. If there are any errors, fix them and try again.  
  
**NOTE:** At the time of the writing of this guide, the intermediate and advanced interface boards have not been fully developed and are therefore not added as projects to the SABT solution.

1. Connect the appropriate programmer to your computer via USB.
   * MCU: Use the JTAGICE mkII.
   * Interface boards: Use the AVRISP mkII. If an AVRISP is not available, it is possible to use an adapter or breakout headers to program the user interface boards using the JTAGICE.
2. If using the JTAGICE, turn it off using the switch at the back of the programmer.
3. Connect the device to the programmer.

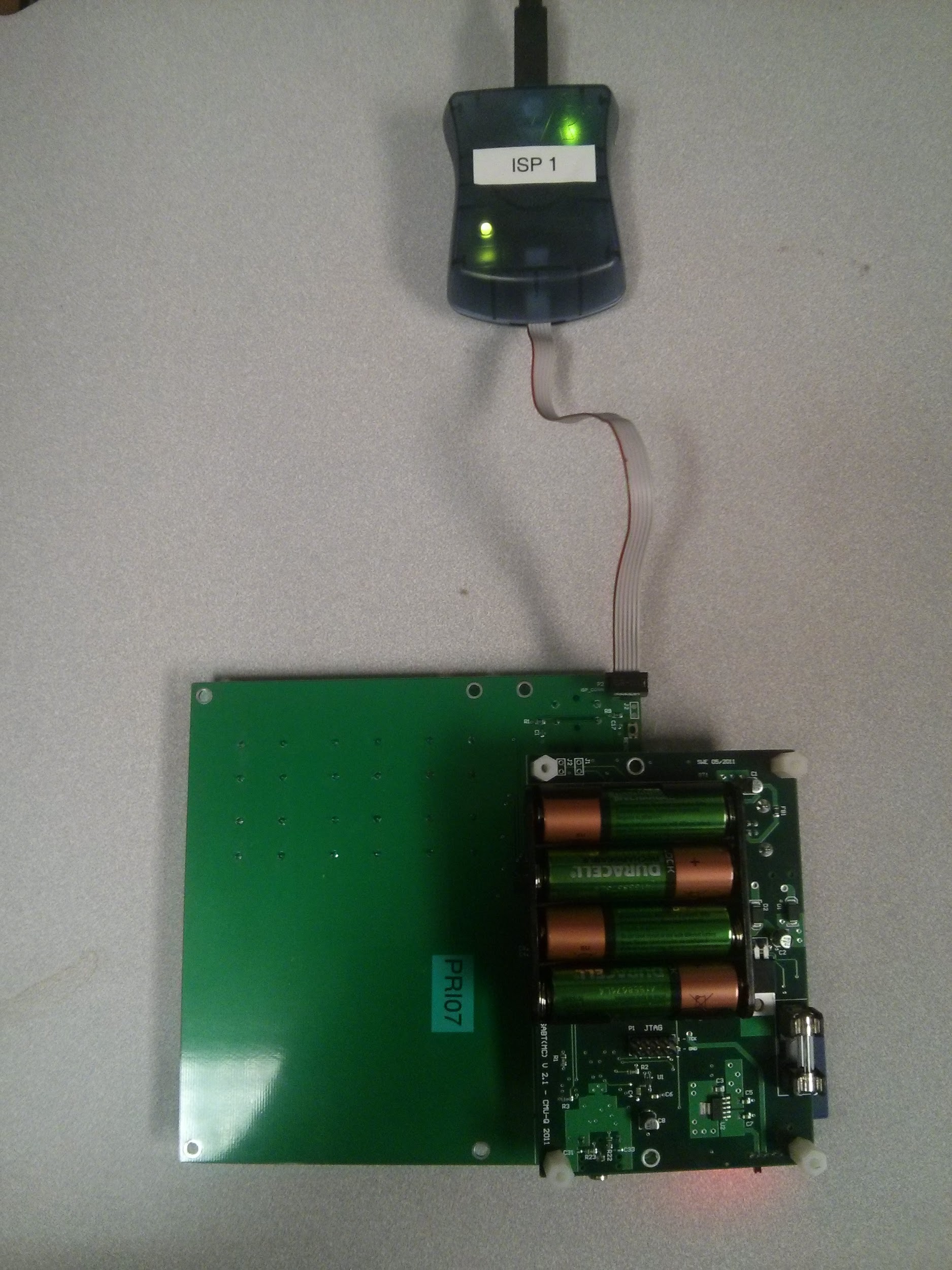
**NOTE:** It is very important to be careful while handling the programmers and their ribbon cables. They are delicate and difficult to replace.

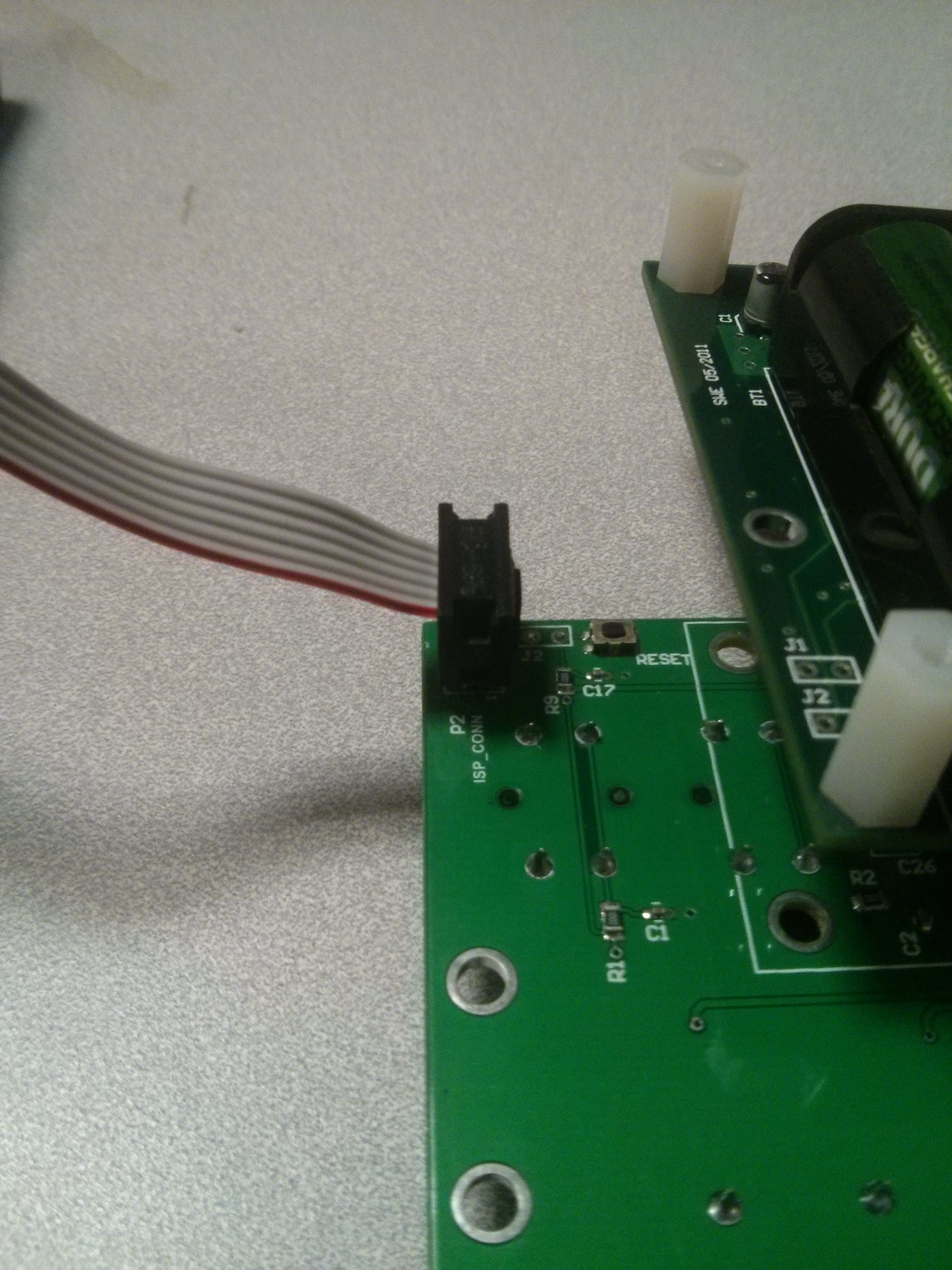
* + MCU: Connect the JTAG cable to the 10-pin JTAG connector on the underside of the MCU next to the batteries.



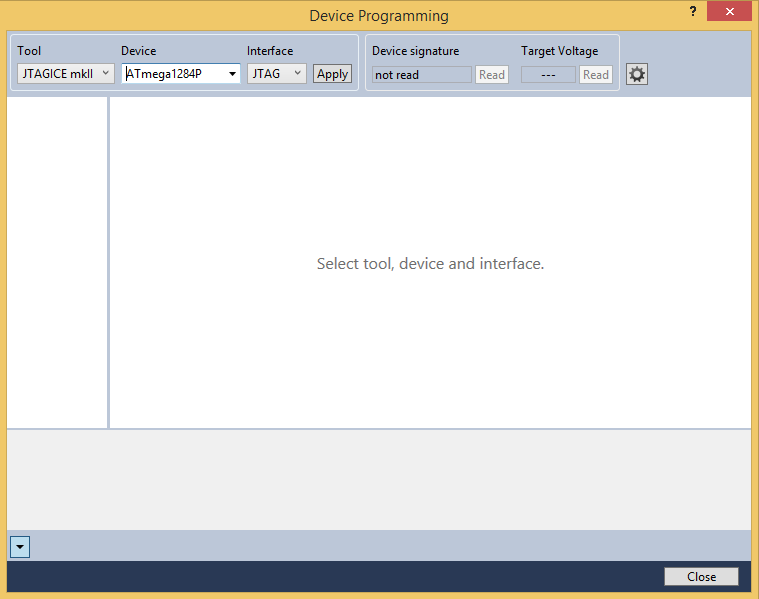


* + Interface boards: Connect the ISP cable to the 6-pin ISP connector on the underside of the board close to the top-left corner of the board. If using the JTAGICE with a JTAG to ISP adapter, make sure to position the programmer in a way that doesn’t stress the ribbon cable.



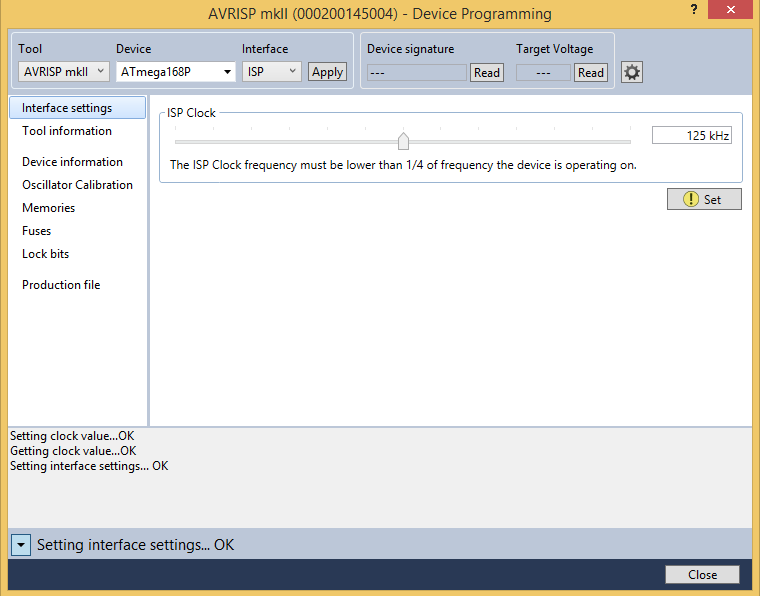


1. If using the JTAGICE, turn it back on and then turn on the SABT. Note that neither programmer cannot supply power to the SABT.
2. In Atmel Studio, select *Device Programming* under the *Tools* menu. Choose the following settings and *Apply* them.
   * MCU
     + Tools: JTAGICE mkII
     + Device: ATmega1284P
     + Interface: JTAG

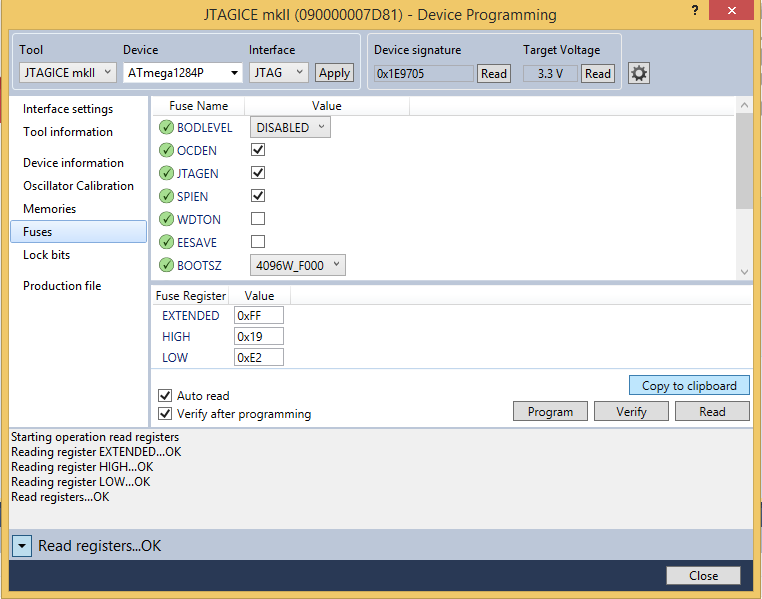


* + Interface boards
    - Tools: AVRISP mkII (or JTAGICE mkII)
    - Device: ATmega168PA
    - Interface: ISP

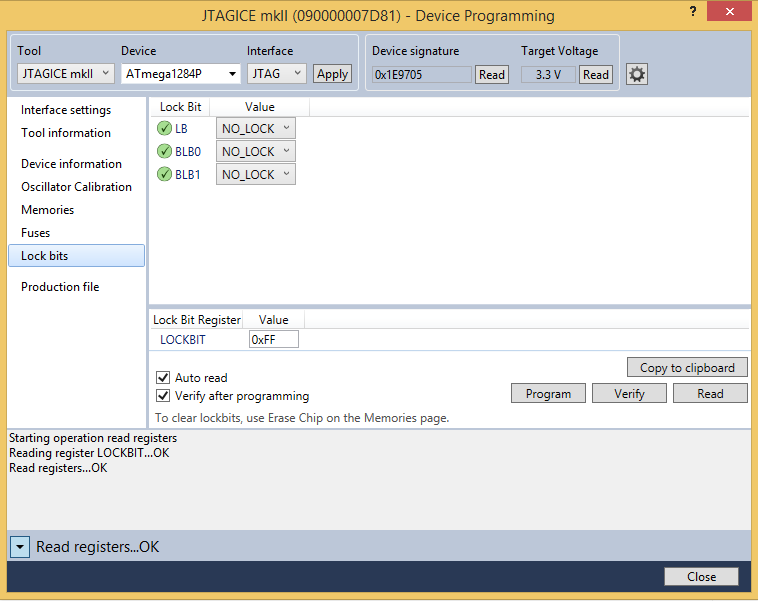
1. Try to read the device signature to verify that the programmer can identify the chip.
   * If you receive a *Device voltage warning* dialog at this point, it probably means that the SABT’s batteries are missing or discharged.
   * If you receive an *Info* dialog telling you the programmer is *unable to enter programming mode*, check the orientation of your JTAG or ISP cable.
   * Most miscellaneous programming errors can be fiexd by disconnecting the programmer, computer and SABT from one another, power cycling the programmer and SABT and trying again.
2. For the user interface boards, set the ISP clock to 125 kHz.



1. Verify the fuses and lockbit on the chip you are about to program. This can be done via the *Fuses* and *Lockbits* panes. If the values on the chip do not match the ones below, program new values onto the chip.  
     
   **NOTE:** It is very important not to program incorrect fuses onto the chip since this has the potential to disable the communication buses that are used to program it.  
   * MCU
     + Fuses
       - Extended: 0xFF
       - High: 0x19
       - Low: E2

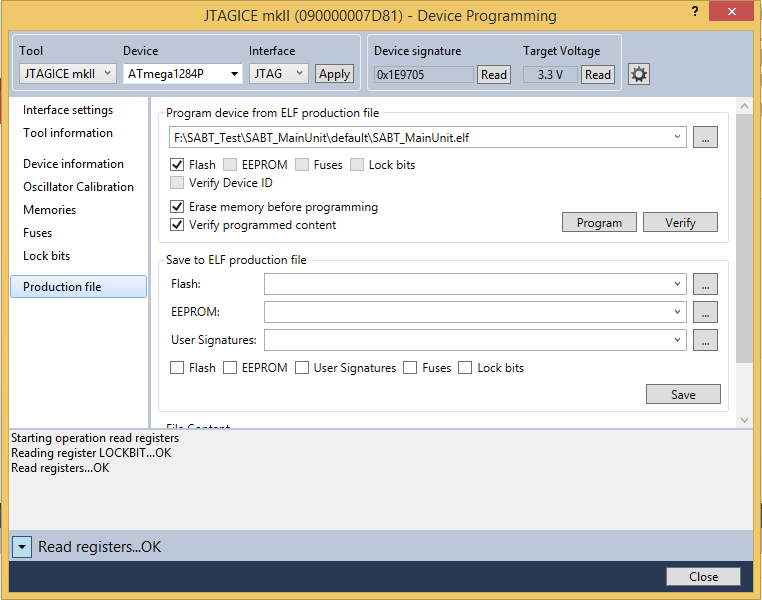


* + - Lockbit: 0xFF



* + Interface boards
    - Fuses
      * Extended: 0xF9
      * High: 0xDF
      * Low: E2
    - Lockbit: 0xFF

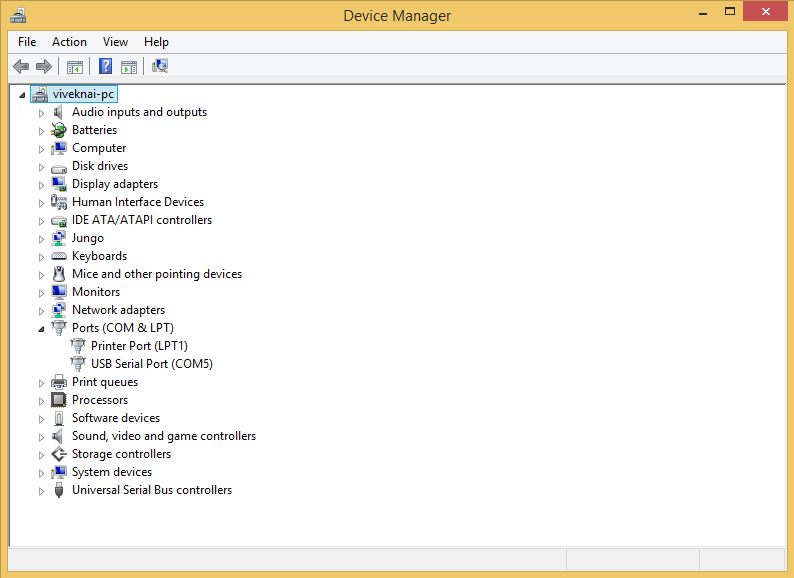
1. Program the compiled code onto the chip. Open the *Production file* pane, select the appropriate production file in the file, check *Flash, Erase memory before programming* and *Verify programmed content* and select *Program.*
   * MCU: SABT\_MainUnit/default/SABT\_MainUnit.elf



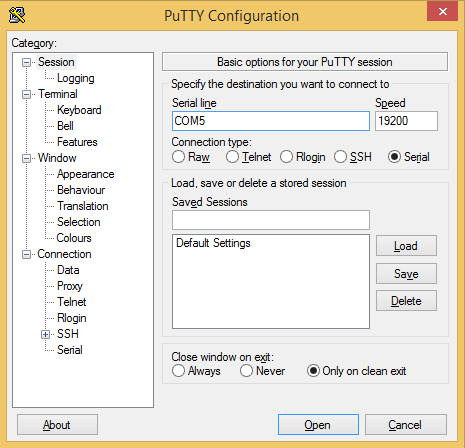
* + Interface board: UserInterface(Primary)/deafult/UserInterface.elf

# Debugging the SABT

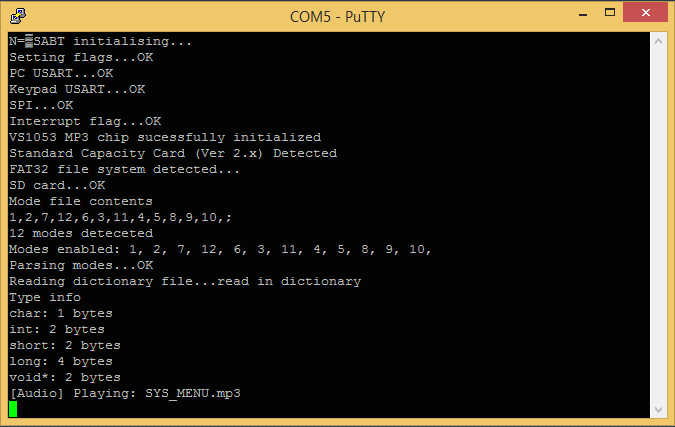
1. Connect the SABT to the computer using a USB cable.
2. Open *Device Manager* to figure out which USB serial port the SABT enumerates to.



1. Open *PuTTY* and use the following settings.
   * Select *Serial* as the *Connection type*.
   * Use the *Serial line* identified as the SABT in *Device Manager*.
   * Set the *Speed* to 19200.



1. When you restart the SABT after opening a PuTTY terminal, you should see bootup debugging information.



1. You can add more debug statements using the PRINTF macro.  
     
   **NOTE:** Adding print statements has the potential to break code in a real-time embedded system like the SABT. They should be used sparingly.
2. The following commands can be used to control the modes available on the SABT via PuTTY. These commands used to modify the mode file:
   * PCx - Initializes the communication between the computer and the SABT.
   * PCM<1><2>...<n>$ - Writes the line into the mode file. Use to specify which modes are available on the SABT.