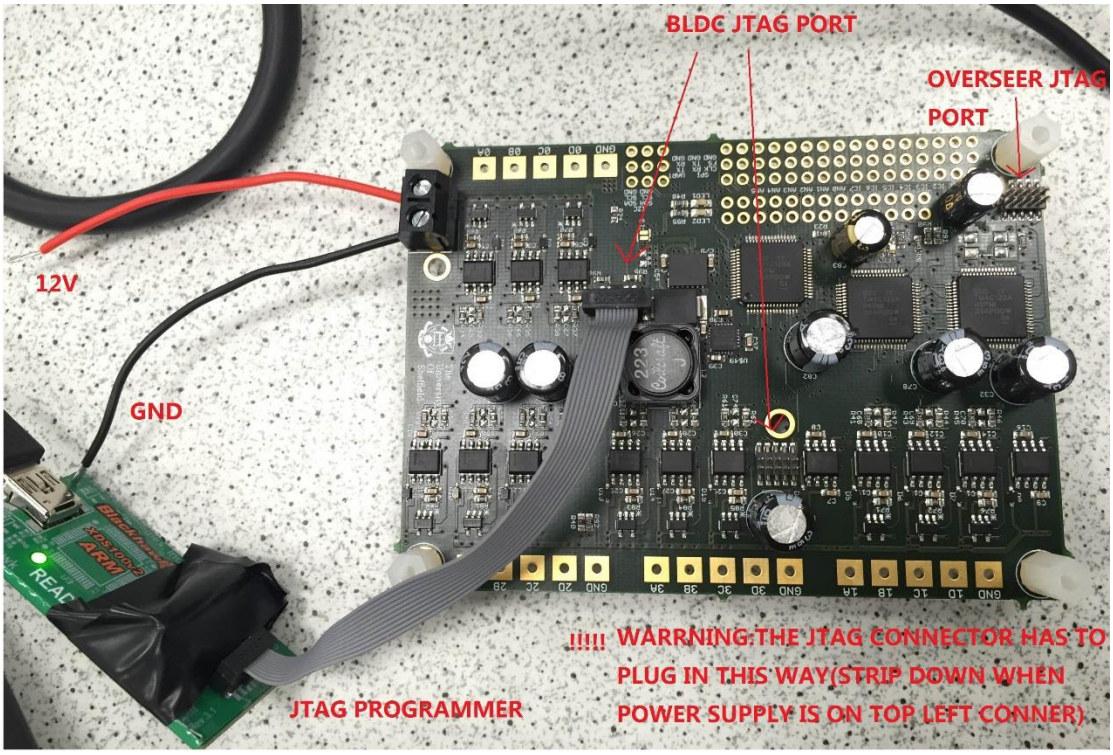
OptimusPi Board and project set-up and using I2C libary

Here is a tutorial how to set up the project in CCS, program the board and how to use the code I written for I2C bus class on board to extract sensored data out from MPU9150.

i) Prgromming Board

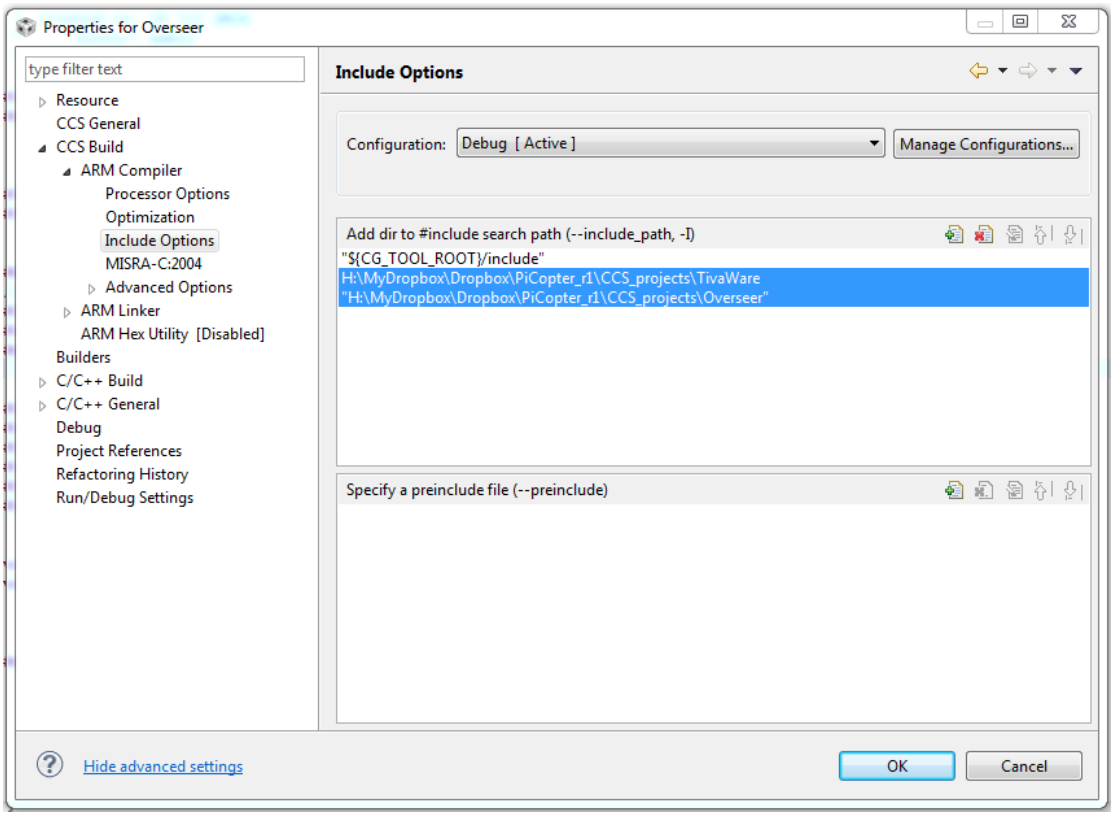
The board has 3 JTAG ports, two for bottom-level controllers, BLDC controllers in this case, one for top-level controllers, i.e. Overseer here. To program any of controllers, the JTAG cable has to be plugged in with strip down, and the USB cable should not be plugged in until the JTAG cable connected board. More detail is shown in the following diagram.



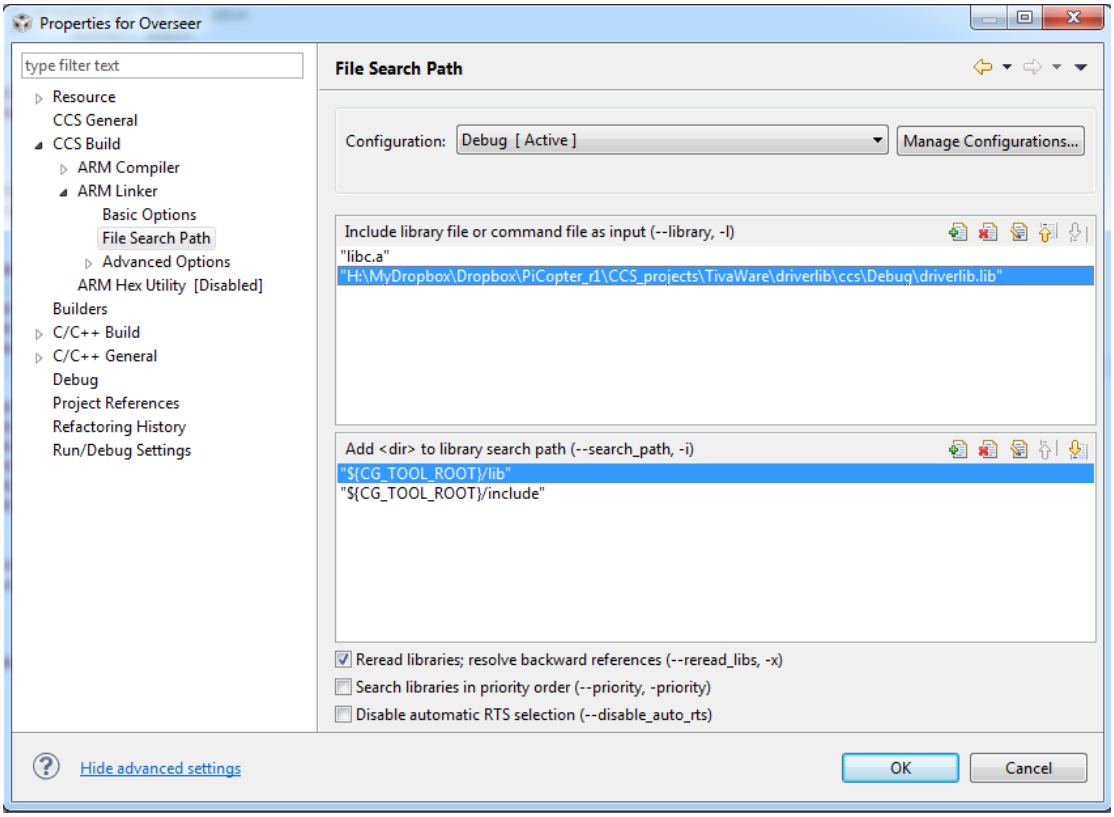
ii) CCS set-up

1) Download the CCS from ti.com and load a copy of Matt's CCS project code as the workspace.
2) Go to "Project Explorer", right-click on "Overseer" project, click "Properties" at the bottom.
3) Go to "CCS Build" -> "ARM Compiler" -> "Include Options", make sure in "Configuration" tag, you choose "Debug".
4) In the "Add dir to #include search path (--include_path, -I)" window, amend the directories to the path that "Overseer" folder and "Tivaware" folder locates in your current CCS project directory. For example, my CCS project path is: H:\MyDropbox\Dropbox\PiCopter_r1\CCS_projects, so I added

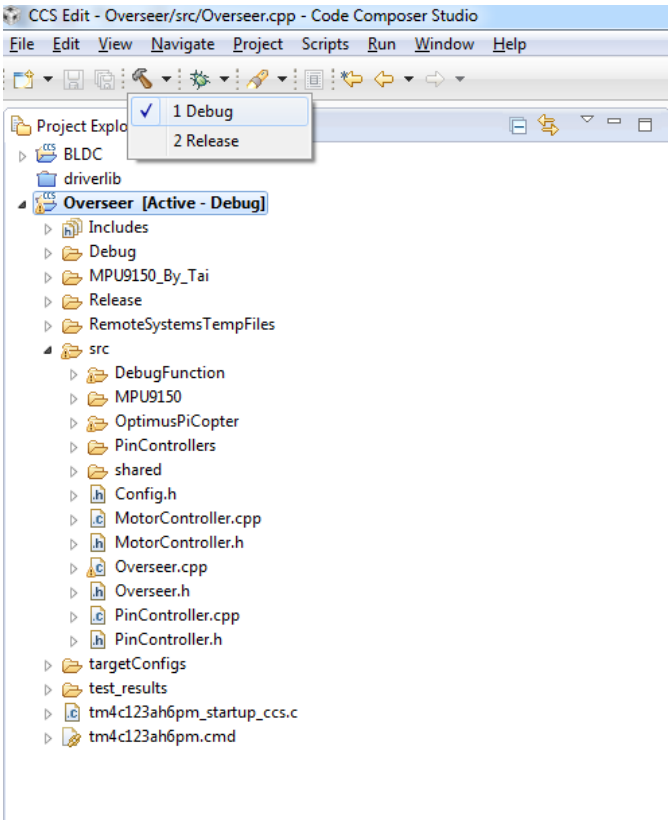
“H:\MyDropbox\Dropbox\PiCopter_r1\CCS_projects\TivaWare” and “H:\MyDropbox\Dropbox\PiCopter_r1\CCS_projects\Overseer” as search path.



5) Go to “CCS Build” -> “ARM Linker” -> “File Search Path”, make sure in “Configuration” tag, you choose “Debug”.
6) In the “Include Library file or command file as input (--library, -l)” window, amend the directories to the path that TivaWare driver library locates in your current CCS project directory. For example, my CCS project path is: H:\MyDropbox\Dropbox\PiCopter_r1\CCS_projects, so I added "H:\MyDropbox\Dropbox\PiCopter_r1\CCS_projects\TivaWare\driverlib\ccs\Debug\driverlib.lib” as the path.



7) Build project “Overseer [Debug - Active]” (Clean the project first before you build it first time)



iii) I2C driver, tests against MPU9150 Class and tests results

As stated in previous email, I have written an I2C driver on the board to get access to the MPU9150 and written tests to verify ability to get data out of MPU9150 and prints out tests results. By reading through the test code I wrote, it may help you understand how to invoke MPU9150 Class. If you want to use this class, you still need to learn a little bit about how to write C++ and some fundamentals of objective-orientated language, though. (Google is always your friend) The following is the directory tree of two folders in the attachments.

/MPU9150
- /I2C.cpp # Overseer I2C driver C++ file
- /I2C.h # Overseer I2C driver header file
- /MPU9150RegisterMap.h # Generic MPU9150 register map macros by MattW
- /MPU9150.cpp # Generic MPU9150 C++ file by MattW
- /MPU9150.h # Generic MPU9150 header file by MattW
  # YOU SHOULD INCLUDE THIS HEADER FILE WHERE IT IS BEING USED TO ACCESS MPU9150 DATA. If you are going to use
  # MattW's code for data calibration, and If I am right, this header file should be included from MattW's Raspberry Pi code
  # with path "OptimusPI Data\Eclipse Linux Projects\OptimusPiCopter\src\AHRS.h"
- /test
  - /I2CTest.cpp # Test for I2C driver. It is not necessary for you, but reading the code may give you much more insight of how to use I2C method.
  - /I2CTest.h
  - /MPU9150Test.cpp # Test for MPU9150 Class. It is not necessary for you, but reading the code may give you much more insight of how to use MPU9150 Class.
  - # N.B. THIS TEST NEEDS PRINTF SUPPORT TO BE RUN. IF YOU DON’T HAVE PRINTF SUPPORTED, YOU CANNOT SEE THE TEST RESULTS.
  - /MPU9150Test.h

/test_results
- /3_MPU9150_tests/
  - /3_1_MPU_test_results_RawData_Readings.txt
  - # test results reference for raw data reading from MPU9150
  - / 3_2_MPU_test_results_ScaledSensorData_Readings.txt
  - # test results reference for raw data reading from MPU9150

An example of pieces of the test results would be like the following:

[CORTEX_M4_0]

3. Start Continuous MPU Tests.

MPU9150 connection verified

2) MPU ScaledSensorData Reading Test. It will take the sensored data five times and move to next test. Please shake, rotate the board and warm up the sensor while the test is running in order to observe a clear change on raw data.

Reading the sensor data for 40 times, printing out the 40th reading of data.
scaledSensorData.x is 0.021729.
scaledSensorData.y is 0.004395.
scaledSensorData.z is 0.980225.
scaledSensorData.temp is 30.205883.
scaledSensorData.p is -1.236641.
scaledSensorData.q is -0.992366.
scaledSensorData.r is 0.625954.
scaledSensorData.magx is 0.000000.
scaledSensorData.magy is 0.000000.
scaledSensorData.magz is -8.015640.
