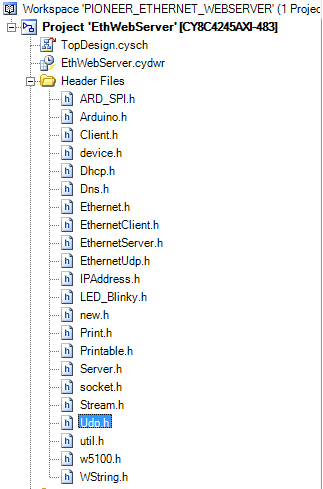
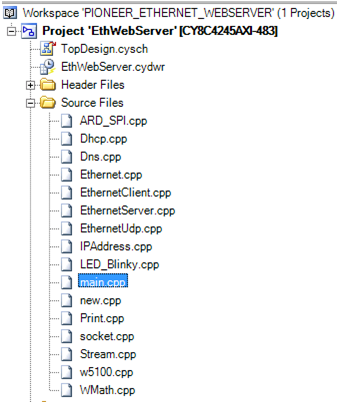
**Firmware**

Please review the file describing how to load/build C++ files in a Creator project.

The Arduino firmware version 1.0.4 was downloaded from <http://arduino.cc/en/Main/Software>. Ethernet Library can be found in \arduino-1.0.4\libraries\Ethernet. The Cores library is found at \arduino-1.0.4\hardware\arduino\cores\arduino\

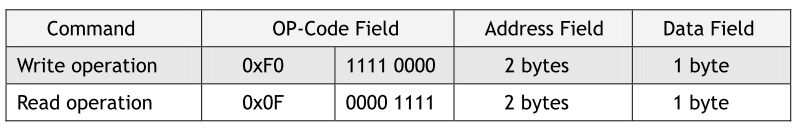
The library files needed for Ethernet were added to the PSoC Creator project. These are the files needed:

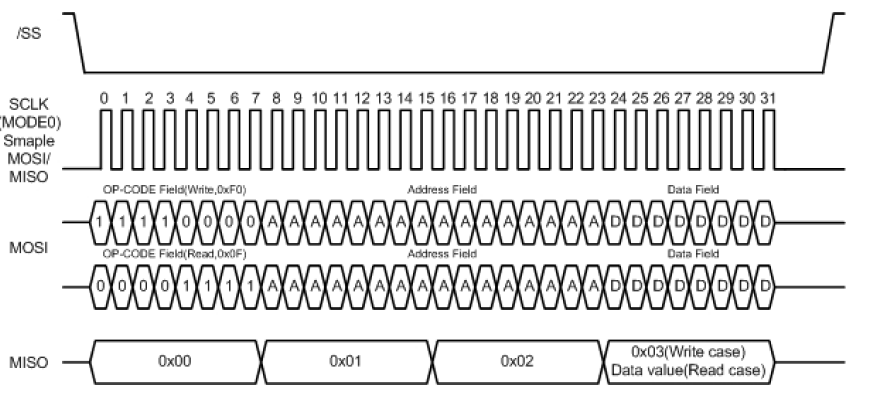




Whenever a Cpp file is added to the Creator, the user has to right click on that file -> click on Properties -> Change the File Type to ‘C\_FILE’

The Arduino Ethernet Shield uses SPI as the physical layer. The command format is as shown below





This requires a 32-bit command format. PSoC4 SPI generates the Slave Select (SS) for every byte. To achieve the above format, the SS is ignored and the ETH\_CS is driven from a GPIO (P3.4). This modification is done in w5100.h

inline static void initSS() { \*(unsigned int \*)0x40040300 |= 0x10; };

inline static void setSS() { \*(unsigned int \*)0x40040300 &= 0xFFFFFFEF; };

inline static void resetSS() { \*(unsigned int \*)0x40040300 |= 0x10; };

The SPI transfer function is modified to use the PSoC SPI component in ARD\_SPI.h

byte SPIClass::transfer(byte \_data) {

SPI\_SpiUartWriteTxData(\_data);

CyDelay(2); // SHAS - without this delay it does not work !!

// SPI\_SpiUartReadRxData() just returns the RX FIFO content

// It needs to check if Data Available in RX FIFO

return ((byte)SPI\_SpiUartReadRxData());

}

SPI is started in the ARD\_SPI.cpp file

void SPIClass::begin() {

SPI\_Start();

}

SPI component is configured for Mode 0, 4Mbps data rate, 8 data bits. The PSoC4 SPI supports data bits range of 4 to 16. In order to achieve 32 bits transfer in this project, the data bit width is set to 8 and 4 bytes will be sent. The SS will be ignored since it will be generated every byte and instead ETH\_CS will be driven from a GPIO for every 32-bits.

The Arduino Ethernet library uses a millisecond counter for checking the timeout operations. This was implemented using a 16-bit counter in PSoC in Arduino.h as follows:

#define millis() Counter\_ReadCounter()

Apart from the modifications described above, there were some changes done to fix the compilation errors such as undefined references, type not found, etc.

Each Arduino Ethernet board has a unique MAC Id which is denoted with a sticker pasted at the back of the shield. This needs to be configured in the main.cpp.

byte mac[] = {

0x90, 0xA2, 0xDA, 0x0D, 0x44, 0xFD };

IPAddress ip(192,168,0, 102);

IPAddress gateway(192,168,0,1);

Configure the IP address and the gateway address to the appropriate addresses.