Peidong Qi
03/13/2018
AMRUPT, Spring 2018

The Communication between CC1310 and Raspberry-pi update

Goals

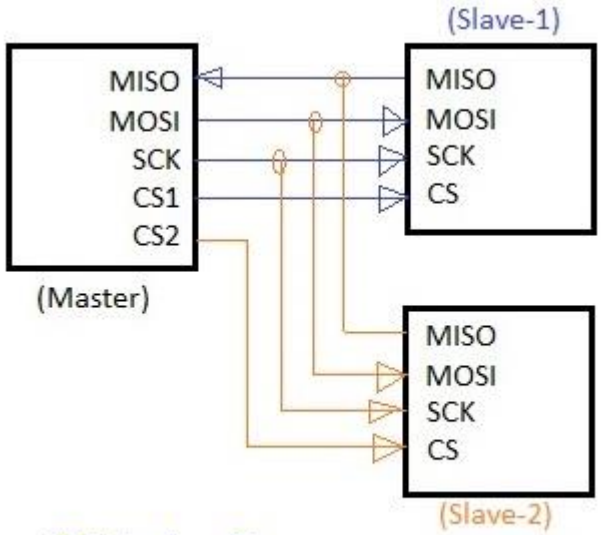
We continue working on SPI. After we decide to use SPI to transfer the data from CC1310 to Raspberry-PI. We need to figure out how the SPI work and how to set up SPI on both Raspberry-pi and CC1310. For this week, we mainly focus on SPI setup on Raspberry Pi. Our goal is to able to connect SPI on Raspberry PI

Problem

Understand to master and slave relation between CC1310 and Raspberry-pi. Hardware and software setup need to be studied,

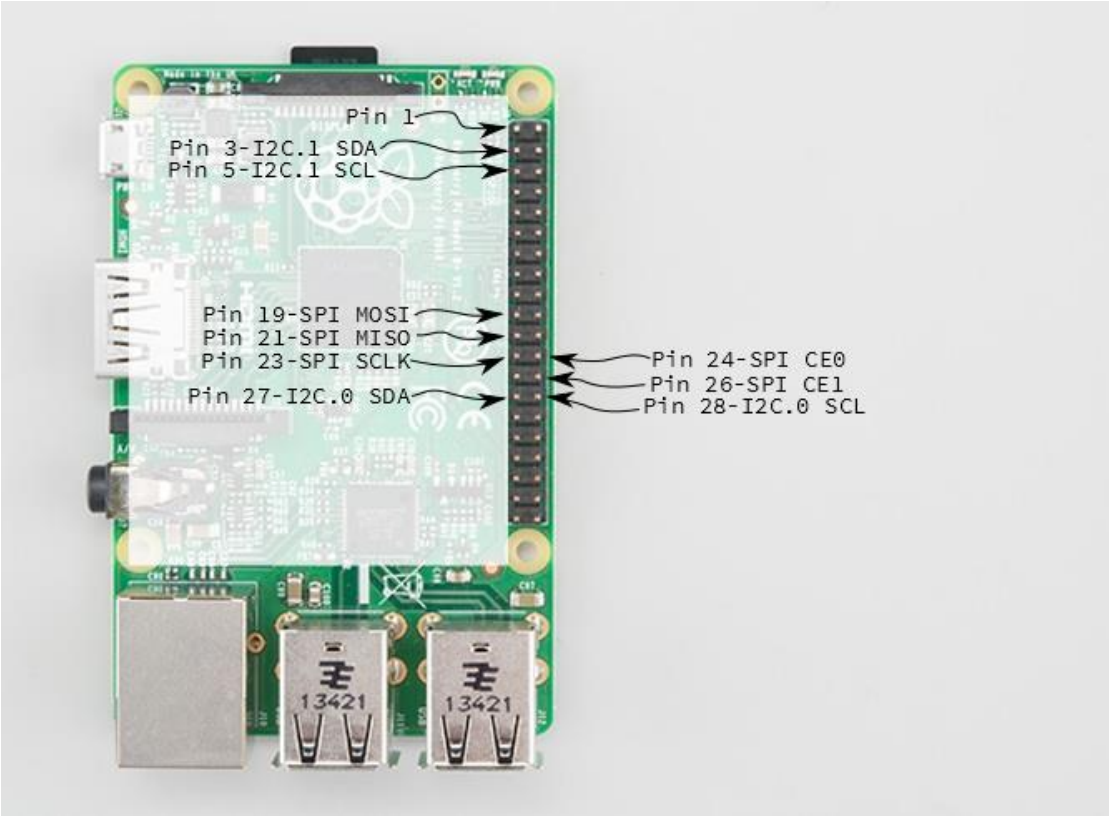
General approach

This week's work is to find out how to setup SPI on Raspberry-pi. The Raspberry Pi is equipped with one SPI bus that has 2 chip selects



SPI Interface Diagram

As the figure shown to use, the master device need to setup MISO, MOSI, SCK, CS1 and CS2 to enable SPI.



The picture shows there are already setup the ports for the SPI. After we setup the system and GPIO pins. We are able to use those ports. The online tutorial has every steps to setup the SPI. By following those steps, we are able to setup the SPI and run the example code. However, the setup on CC1310 is not started. So we are not able to test SPI.

Code-level problems and solutions, and empirical testing

Example code is shown below:

```
#include < iostream >
#include < errno.h >
#include < wiringPiSPI.h >
#include < unistd.h >
using namespace std; // channel is the wiringPi name for the chip select (or chip enable) pin. // Set this to 0 or 1, depending on how it's connected.
static  const int CHANNEL = 1;
int main() {
    int fd, result;
    unsigned char buffer[100];
    cout << "Initializing" << endl; // Configure the interface. // CHANNEL insicates chip select, // 500000 indicates bus speed.
    fd = wiringPiSPISetup(CHANNEL, 500000);
    cout << "Init result: " << fd << endl; // clear display
    buffer[0] = 0x76;
```

```
14.     wiringPiSPIDataRW(CHANNEL, buffer, 1);
15.     sleep(5); // Do a one-
    hot bit selection for each field of the display // It displays gibberish, but tells
     us that we're correctly addressing all  // of the segments.
16.     for (int i = 1; i <= 0x7f; i <<= 1) { // the decimals, colon and apostrophe dot
    s
17.         buffer[0] = 0x77;
18.         buffer[1] = i;
19.         result = wiringPiSPIDataRW(CHANNEL, buffer, 2); // The first character
20.         buffer[0] = 0x7b;
21.         buffer[1] = i;
22.         result = wiringPiSPIDataRW(CHANNEL, buffer, 2); // The second character
23.         buffer[0] = 0x7c;
24.         buffer[1] = i;
25.         result = wiringPiSPIDataRW(CHANNEL, buffer, 2); // The third character
26.         buffer[0] = 0x7d;
27.         buffer[1] = i;
28.         result = wiringPiSPIDataRW(CHANNEL, buffer, 2); // The last character
29.         buffer[0] = 0x7e;
30.         buffer[1] = i;
31.         result = wiringPiSPIDataRW(CHANNEL, buffer, 2); // Pause so we can see them


32.         sleep(5);
33.     } // clear display again
34.     buffer[0] = 0x76;
35.     wiringPiSPIDataRW(CHANNEL, buffer, 1);
36. }
```

Planned Course of Action

Keep working on SPI, Mainly focus on SPI on CC1310 next week. Finished SPI setup on CC1310 and start to test SPI.

Resources and relevant Forum Posts

[1] “SPI setup on Raspberry PI” https://www.raspberrypi.org/documentation/hardware/raspberrypi/spi/README.md.

[2] “UART vs SPI vs I2C,” http://www.rfwireless-world.com/Terminology/UART-vs-SPI-vs-I2C.html.

[3] Raspberry PI SPI and I2C Tutorial” https://learn.sparkfun.com/tutorials/raspberry-pi-spi-and-i2c-tutorial