**Smart Doorlock Laboratory Notebook**

**Primary Author: Sokwhan Huh**

*This space is intentionally left blank*

# August 31st, 2016

Project Requirements

The core project requirements are as follows:

* Demonstrate NFC P2P capability
  + Be able to open the door lock with NFC using a smartphone app under P2P mode. The phone must be registered in advance to open the door lock.
* Demonstrate IoT capability
  + Be able to open or permanently lock (lockdown mode) the door lock over the internet using a smartphone app or by accessing Smart Doorlock’s web server with a browser. The phone must be registered in advance to open the door lock.
* Demonstrate enhanced security system
  + Be able to register phones for access.
  + Be able to clear existing registered phones.

Additional requirements are as follows:

* Implement full logging system for entries.
* Have the door lock function with a battery backup in case of power outage.

Technical Specifications

* Interfacing via NFC in P2P mode using Android Beam technology
* Interfacing via IoT to a cloud server over MQTT Protocol
* NFC communication range: 5 cm maximum (from smartphone to Smart Doorlock)
* Wi-Fi communication range: 50 meters maximum (TX 14.5dBM, RX -74.0dBM at 54 OFDM)
* Microcontroller: Texas Instruments CC3200
* NFC Transceiver IC: Texas Instruments TRF7970A
* AC power with DC battery backup
* Minimum backup battery life of 24 hours
* Keypad: Storm Interface’s Storm 720GFXI Series (7207-121W20)
* LCD: Electronic Assembly’s EA DIP203G-4NLED LCD
* Programming Language: C (Microcontroller), Java (Android Application), C# (Web Server)
* Supported Android versions: 4.0 (Ice Cream Sandwich) and above
* Web Server Framework: ASP.NET MVC 5
* Web Server Database: MySQL

**Source Control (Mercurial):** [**https://bitbucket.org/l46kok/smart-doorlock**](https://bitbucket.org/l46kok/smart-doorlock)

Source code of the project, along with design documents and detailed diagrams can be found under the source control link (inside document folder).

Started a new project and pushed a new source base into repository. To test that CC3200’s basic functionality works as intended, the following simple program was made, which is to be served as the skeleton of the entire project. The following code currently only does two things: initialize the microcontroller and send ASCII strings via UART TX.

|  |
| --- |
| **main.c** |
| /\*  \* Smart Doorlock  \* main.c  \*  \* June 15th, 2016  \*  \* Author: Sokwhan Huh  \*/  // Driverlib includes  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_common\_reg.h"  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "uart.h"  **#include** "interrupt.h"  **#include** "pinmux.h"  **#include** "utils.h"  **#include** "prcm.h"  // Common interface include  **#include** "uart\_if.h"  **#define** APPLICATION\_VERSION "1.1.1"  **#define** APP\_NAME "Smart Doorlock"  **#define** CONSOLE UARTA0\_BASE  **#define** UartGetChar() MAP\_UARTCharGet(CONSOLE)  **#define** UartPutChar(c) MAP\_UARTCharPut(CONSOLE,c)  **#define** MAX\_STRING\_LENGTH 80  **static** **void** **DisplayBanner**(**char** \* AppName)  {  **Report**("\n\n\n\r");  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\t\t \n\r", AppName);  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\n\n\n\r");  }  **static** **void** **BoardInit**(**void**)  {  /\* In case of TI-RTOS vector table is initialize by OS itself \*/  **#ifndef** USE\_TIRTOS  //  // Set vector table base  //  **#if** defined(ccs)  MAP\_IntVTableBaseSet((**unsigned** **long**)&g\_pfnVectors[0]);  **#endif**  **#if** defined(ewarm)  MAP\_IntVTableBaseSet((**unsigned** **long**)&\_\_vector\_table);  **#endif**  **#endif**  //  // Enable Processor  //  MAP\_IntMasterEnable();  MAP\_IntEnable(FAULT\_SYSTICK);  **PRCMCC3200MCUInit**();  }  **int** **main**(**void**) {  //  // Initailizing the board  //  BoardInit();  //  // Muxing for Enabling UART\_TX and UART\_RX.  //  PinMuxConfig();  //  // Initialising the Terminal.  //  **InitTerm**();  //  // Clearing the Terminal.  //  **ClearTerm**();  DisplayBanner(APP\_NAME);  **return** 0;  } |

|  |
| --- |
| **uart\_if.c** |
| /\*  \* Smart Doorlock  \*  \* uart\_if.c  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  // Standard includes  **#include** <stdarg.h>  **#include** <stdlib.h>  **#include** <stdio.h>  **#include** <string.h>  // Driverlib includes  **#include** "hw\_types.h"  **#include** "hw\_memmap.h"  **#include** "prcm.h"  **#include** "pin.h"  **#include** "uart.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  **#include** "osi.h"  **#endif**  **#include** "uart\_if.h"  **#define** IS\_SPACE(x) (x == 32 ? 1 : 0)  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Global variable indicating command is present  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **static** **unsigned** **long** \_\_Errorlog;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Global variable indicating input length  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **int** ilen=1;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Initialization  //!  //! This function  //! 1. Configures the UART to be used.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **InitTerm**()  {  **#ifndef** NOTERM  MAP\_UARTConfigSetExpClk(CONSOLE,MAP\_PRCMPeripheralClockGet(CONSOLE\_PERIPH),  UART\_BAUD\_RATE, (UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE |  UART\_CONFIG\_PAR\_NONE));  **#endif**  \_\_Errorlog = 0;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Outputs a character string to the console  //!  //! \param str is the pointer to the string to be printed  //!  //! This function  //! 1. prints the input string character by character on to the console.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **Message**(**const** **char** \*str)  {  **#ifndef** NOTERM  **if**(str != NULL)  {  **while**(\*str!='\0')  {  MAP\_UARTCharPut(CONSOLE,\*str++);  }  }  **#endif**  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Clear the console window  //!  //! This function  //! 1. clears the console window.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **ClearTerm**()  {  Message("\33[2J\r");  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Error Function  //!  //! \param  //!  //! \return none  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **Error**(**char** \*pcFormat, ...)  {  **#ifndef** NOTERM  **char** cBuf[256];  va\_list list;  va\_start(list,pcFormat);  **vsnprintf**(cBuf,256,pcFormat,list);  Message(cBuf);  **#endif**  \_\_Errorlog++;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Get the Command string from UART  //!  //! \param pucBuffer is the command store to which command will be populated  //! \param ucBufLen is the length of buffer store available  //!  //! \return Length of the bytes received. -1 if buffer length exceeded.  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int**  **GetCmd**(**char** \*pcBuffer, **unsigned** **int** uiBufLen)  {  **char** cChar;  **int** iLen = 0;  //  // Wait to receive a character over UART  //  **while**(MAP\_UARTCharsAvail(CONSOLE) == false)  {  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  osi\_Sleep(1);  **#endif**  }  cChar = MAP\_UARTCharGetNonBlocking(CONSOLE);  //  // Echo the received character  //  MAP\_UARTCharPut(CONSOLE, cChar);  iLen = 0;  //  // Checking the end of Command  //  **while**((cChar != '\r') && (cChar !='\n') )  {  //  // Handling overflow of buffer  //  **if**(iLen >= uiBufLen)  {  **return** -1;  }  //  // Copying Data from UART into a buffer  //  **if**(cChar != '\b')  {  \*(pcBuffer + iLen) = cChar;  iLen++;  }  **else**  {  //  // Deleting last character when you hit backspace  //  **if**(iLen)  {  iLen--;  }  }  //  // Wait to receive a character over UART  //  **while**(MAP\_UARTCharsAvail(CONSOLE) == false)  {  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  osi\_Sleep(1);  **#endif**  }  cChar = MAP\_UARTCharGetNonBlocking(CONSOLE);  //  // Echo the received character  //  MAP\_UARTCharPut(CONSOLE, cChar);  }  \*(pcBuffer + iLen) = '\0';  Report("\n\r");  **return** iLen;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Trim the spaces from left and right end of given string  //!  //! \param Input string on which trimming happens  //!  //! \return length of trimmed string  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** **TrimSpace**(**char** \* pcInput)  {  size\_t size;  **char** \*endStr, \*strData = pcInput;  **char** index = 0;  size = **strlen**(strData);  **if** (!size)  **return** 0;  endStr = strData + size - 1;  **while** (endStr >= strData && IS\_SPACE(\*endStr))  endStr--;  \*(endStr + 1) = '\0';  **while** (\*strData && IS\_SPACE(\*strData))  {  strData++;  index++;  }  **memmove**(pcInput,strData,**strlen**(strData)+1);  **return** **strlen**(pcInput);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! prints the formatted string on to the console  //!  //! \param format is a pointer to the character string specifying the format in  //! the following arguments need to be interpreted.  //! \param [variable number of] arguments according to the format in the first  //! parameters  //! This function  //! 1. prints the formatted error statement.  //!  //! \return count of characters printed  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** **Report**(**const** **char** \*pcFormat, ...)  {  **int** iRet = 0;  **#ifndef** NOTERM  **char** \*pcBuff, \*pcTemp;  **int** iSize = 256;  va\_list list;  pcBuff = (**char**\*)**malloc**(iSize);  **if**(pcBuff == NULL)  {  **return** -1;  }  **while**(1)  {  va\_start(list,pcFormat);  iRet = **vsnprintf**(pcBuff,iSize,pcFormat,list);  va\_end(list);  **if**(iRet > -1 && iRet < iSize)  {  **break**;  }  **else**  {  iSize\*=2;  **if**((pcTemp=**realloc**(pcBuff,iSize))==NULL)  {  Message("Could not reallocate memory\n\r");  iRet = -1;  **break**;  }  **else**  {  pcBuff=pcTemp;  }  }  }  Message(pcBuff);  **free**(pcBuff);  **#endif**  **return** iRet;  } |

|  |
| --- |
| **uart\_if.c** |
| /\*  \* Smart Doorlock  \*  \* uart\_if.c  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  // Standard includes  **#include** <stdarg.h>  **#include** <stdlib.h>  **#include** <stdio.h>  **#include** <string.h>  // Driverlib includes  **#include** "hw\_types.h"  **#include** "hw\_memmap.h"  **#include** "prcm.h"  **#include** "pin.h"  **#include** "uart.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  **#include** "osi.h"  **#endif**  **#include** "uart\_if.h"  **#define** IS\_SPACE(x) (x == 32 ? 1 : 0)  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Global variable indicating command is present  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **static** **unsigned** **long** \_\_Errorlog;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Global variable indicating input length  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **int** ilen=1;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Initialization  //!  //! This function  //! 1. Configures the UART to be used.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **InitTerm**()  {  **#ifndef** NOTERM  MAP\_UARTConfigSetExpClk(CONSOLE,MAP\_PRCMPeripheralClockGet(CONSOLE\_PERIPH),  UART\_BAUD\_RATE, (UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE |  UART\_CONFIG\_PAR\_NONE));  **#endif**  \_\_Errorlog = 0;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Outputs a character string to the console  //!  //! \param str is the pointer to the string to be printed  //!  //! This function  //! 1. prints the input string character by character on to the console.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **Message**(**const** **char** \*str)  {  **#ifndef** NOTERM  **if**(str != NULL)  {  **while**(\*str!='\0')  {  MAP\_UARTCharPut(CONSOLE,\*str++);  }  }  **#endif**  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Clear the console window  //!  //! This function  //! 1. clears the console window.  //!  //! \return none  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **ClearTerm**()  {  Message("\33[2J\r");  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Error Function  //!  //! \param  //!  //! \return none  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **Error**(**char** \*pcFormat, ...)  {  **#ifndef** NOTERM  **char** cBuf[256];  va\_list list;  va\_start(list,pcFormat);  **vsnprintf**(cBuf,256,pcFormat,list);  Message(cBuf);  **#endif**  \_\_Errorlog++;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Get the Command string from UART  //!  //! \param pucBuffer is the command store to which command will be populated  //! \param ucBufLen is the length of buffer store available  //!  //! \return Length of the bytes received. -1 if buffer length exceeded.  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int**  **GetCmd**(**char** \*pcBuffer, **unsigned** **int** uiBufLen)  {  **char** cChar;  **int** iLen = 0;  //  // Wait to receive a character over UART  //  **while**(MAP\_UARTCharsAvail(CONSOLE) == false)  {  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  osi\_Sleep(1);  **#endif**  }  cChar = MAP\_UARTCharGetNonBlocking(CONSOLE);  //  // Echo the received character  //  MAP\_UARTCharPut(CONSOLE, cChar);  iLen = 0;  //  // Checking the end of Command  //  **while**((cChar != '\r') && (cChar !='\n') )  {  //  // Handling overflow of buffer  //  **if**(iLen >= uiBufLen)  {  **return** -1;  }  //  // Copying Data from UART into a buffer  //  **if**(cChar != '\b')  {  \*(pcBuffer + iLen) = cChar;  iLen++;  }  **else**  {  //  // Deleting last character when you hit backspace  //  **if**(iLen)  {  iLen--;  }  }  //  // Wait to receive a character over UART  //  **while**(MAP\_UARTCharsAvail(CONSOLE) == false)  {  **#if** defined(USE\_FREERTOS) || defined(USE\_TI\_RTOS)  osi\_Sleep(1);  **#endif**  }  cChar = MAP\_UARTCharGetNonBlocking(CONSOLE);  //  // Echo the received character  //  MAP\_UARTCharPut(CONSOLE, cChar);  }  \*(pcBuffer + iLen) = '\0';  Report("\n\r");  **return** iLen;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Trim the spaces from left and right end of given string  //!  //! \param Input string on which trimming happens  //!  //! \return length of trimmed string  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** **TrimSpace**(**char** \* pcInput)  {  size\_t size;  **char** \*endStr, \*strData = pcInput;  **char** index = 0;  size = **strlen**(strData);  **if** (!size)  **return** 0;  endStr = strData + size - 1;  **while** (endStr >= strData && IS\_SPACE(\*endStr))  endStr--;  \*(endStr + 1) = '\0';  **while** (\*strData && IS\_SPACE(\*strData))  {  strData++;  index++;  }  **memmove**(pcInput,strData,**strlen**(strData)+1);  **return** **strlen**(pcInput);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! prints the formatted string on to the console  //!  //! \param format is a pointer to the character string specifying the format in  //! the following arguments need to be interpreted.  //! \param [variable number of] arguments according to the format in the first  //! parameters  //! This function  //! 1. prints the formatted error statement.  //!  //! \return count of characters printed  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **int** **Report**(**const** **char** \*pcFormat, ...)  {  **int** iRet = 0;  **#ifndef** NOTERM  **char** \*pcBuff, \*pcTemp;  **int** iSize = 256;  va\_list list;  pcBuff = (**char**\*)**malloc**(iSize);  **if**(pcBuff == NULL)  {  **return** -1;  }  **while**(1)  {  va\_start(list,pcFormat);  iRet = **vsnprintf**(pcBuff,iSize,pcFormat,list);  va\_end(list);  **if**(iRet > -1 && iRet < iSize)  {  **break**;  }  **else**  {  iSize\*=2;  **if**((pcTemp=**realloc**(pcBuff,iSize))==NULL)  {  Message("Could not reallocate memory\n\r");  iRet = -1;  **break**;  }  **else**  {  pcBuff=pcTemp;  }  }  }  Message(pcBuff);  **free**(pcBuff);  **#endif**  **return** iRet;  } |

|  |
| --- |
| **uart\_if.h** |
| /\*  \* Smart Doorlock  \*  \* uart\_if.h  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  **#ifndef** \_\_uart\_if\_H\_\_  **#define** \_\_uart\_if\_H\_\_  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // If building with a C++ compiler, make all of the definitions in this header  // have a C binding.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **#ifdef** \_\_cplusplus  **extern** "C"  {  **#endif**  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* MACROS \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  **#define** UART\_BAUD\_RATE 115200  **#define** SYSCLK 80000000  **#define** CONSOLE UARTA0\_BASE  **#define** CONSOLE\_PERIPH PRCM\_UARTA0  //  // Define the size of UART IF buffer for RX  //  **#define** UART\_IF\_BUFFER 64  //  // Define the UART IF buffer  //  **extern** **unsigned** **char** g\_ucUARTBuffer[];  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* FUNCTION PROTOTYPES \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  **extern** **void** **DispatcherUARTConfigure**(**void**);  **extern** **void** **DispatcherUartSendPacket**(**unsigned** **char** \*inBuff, **unsigned** **short** usLength);  **extern** **int** **GetCmd**(**char** \*pcBuffer, **unsigned** **int** uiBufLen);  **extern** **void** **InitTerm**(**void**);  **extern** **void** **ClearTerm**(**void**);  **extern** **void** **Message**(**char** \*format);  **extern** **void** **Error**(**char** \*format,...);  **extern** **int** **Report**(**char** \*format, ...);  **void** **UartPutChar**(**unsigned** **char** ch);  **void** **UartPutCrlf**(**void**);  **extern** **void** **UartSendCString**(**char** \*format);  **void** **UartPutByte**(**unsigned** **char** ch);  **void** **UartPutByteHex**(**unsigned** **char** ch);  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // Mark the end of the C bindings section for C++ compilers.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **#ifdef** \_\_cplusplus  }  **#endif**  **#endif** |

|  |
| --- |
| **pinmux.c** |
| /\*  \* Smart Doorlock  \*  \* pinmux.c  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  **#include** "pinmux.h"  **#include** "hw\_types.h"  **#include** "hw\_memmap.h"  **#include** "hw\_gpio.h"  **#include** "pin.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "gpio.h"  **#include** "prcm.h"  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **PinMuxConfig**(**void**)  {  //  // Enable Peripheral Clocks  //  MAP\_PRCMPeripheralClkEnable(PRCM\_UARTA0, PRCM\_RUN\_MODE\_CLK);  //  // Configure PIN\_55 for UART0 UART0\_TX  //  MAP\_PinTypeUART(PIN\_55, PIN\_MODE\_3);  //  // Configure PIN\_57 for UART0 UART0\_RX  //  MAP\_PinTypeUART(PIN\_57, PIN\_MODE\_3);  } |

|  |
| --- |
| **pinmux.h** |
| /\*  \* Smart Doorlock  \*  \* pinmux.h  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  **#ifndef** PINMUX\_H\_  **#define** PINMUX\_H\_  **extern** **void** **PinMuxConfig**(**void**);  **#endif** /\* PINMUX\_H\_ \*/ |

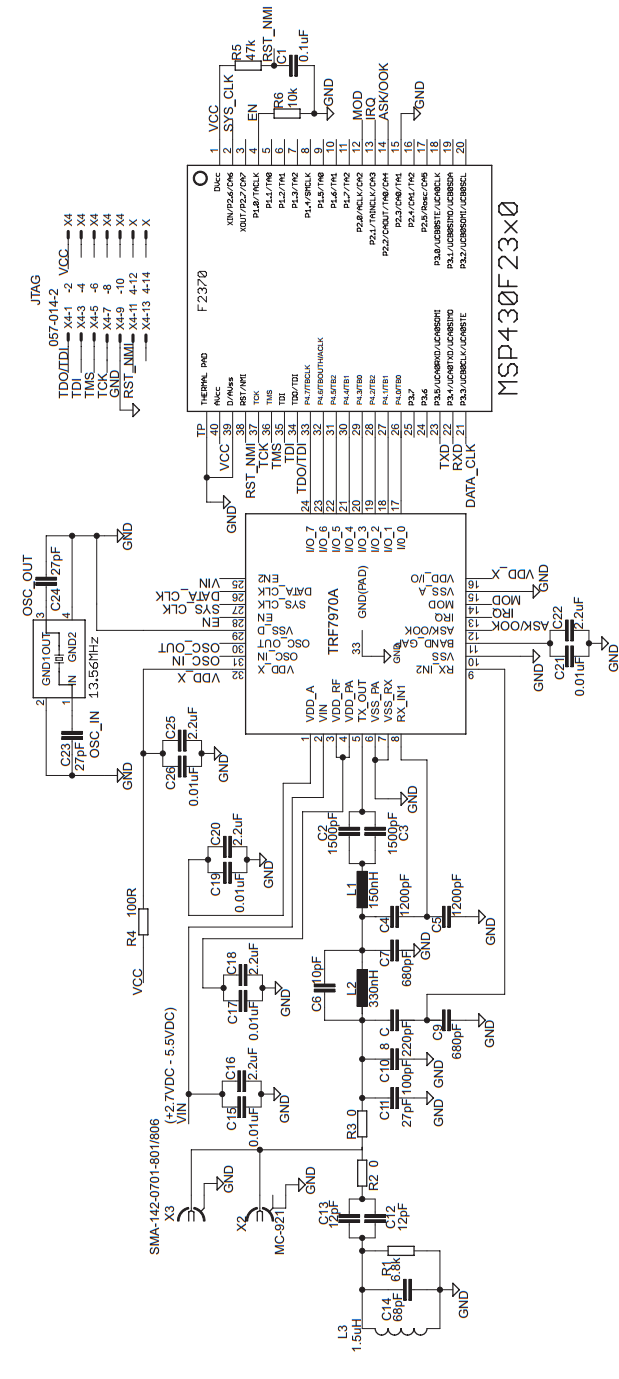
# September 1st, 2016

Further research was performed on NFC communication via tag on TRF7970A. The demo NFCReaderApp from TI would not work for some reason, potentially due to version difference in TRF7970A. Online findings indicated that they’ve changed the pin configurations on the board and the code would have to match the newly set pins. After changing the pin configurations, still no luck so I made a post on TI forums and see if anyone else has an idea. A TI employee answered the post, looked over my code modifications and saw no issues with it. He advised that I solder the jumper pin on IRQ to revert the pin settings back to old version. Will have to try this in lab later. The actual forum link is the following: <http://e2e.ti.com/support/wireless_connectivity/nfc_rfid/f/667/p/518595/1905610#1905610>

Another option is to just jump directly to NFC P2P, but I’m a bit skeptical on whether this approach is sound or not as NFC P2P tends to be a lot more complicated than simple tag read. If I can’t get the NFC tag to work, then I guess this is the alternative route I have to take.

If only I could get my hands on the TRF7970A breakout board, life would be so much easier but they are way too expensive…

# September 4th, 2016

There are couple of things I missed in the initial schematic of this project. Below is a sample schematic of TRF7970A from documentation:

**2**

**1**

First, TRF7970A requires a separate crystal oscillator clocked at 13.56MHz. The development board comes with one, so I mistakenly assumed that the actual chip has an internal clock integrated with it [1]. The second, and probably the bigger problem is that I need to build an entire RF circuitry for the NFC communication to work [2]. I have no prior background in RF so this will pose a huge challenge for me, but it must be catered for before the PCB design. I’m pretty sure CC3200 would require a separate RF circuitry design for Wi-Fi communication but this has not been verified yet. The example seems to be using 50-ohm female-type antenna (with the actual “pole?” thing to be plugged in later), if I am not mistaken, but I will have to find a different schematic as I’ll be using a flat card-like antenna found in most NFC devices.

There are no DIP modules for TRF7970A either so I basically have zero way of testing the RF stuff. I will also have to find out a way of testing this RF circuitry before printing the PCB (if possible).

# September 7th, 2016

RTOS functionality has been included to the project. CC3200 will now spawn a separate thread for executing the main logic of the application. GPIO code was also added for blinking LEDs and handling interrupts in the future.

|  |
| --- |
| **main.c** |
| /\*  \* Smart Doorlock  \* main.c  \*  \* Created on: 2016. 8. 27.  \*  \* Author: Sokwhan Huh  \*/  // C-Library includes  **#include** <string.h>  // Driverlib includes  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_common\_reg.h"  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "uart.h"  **#include** "interrupt.h"  **#include** "pinmux.h"  **#include** "utils.h"  **#include** "prcm.h"  **#include** "simplelink.h"  // Common interface include  **#include** "uart\_if.h"  **#include** "gpio\_if.h"  **#define** APP\_NAME "Smart Doorlock"  //RTOS Related Defines  **#define** OSI\_STACK\_SIZE 4096 /\* 2048 \*/  **#define** SPAWN\_TASK\_PRIORITY 9  **static** **void** **DisplayBanner**(**char** \* AppName)  {  **Report**("\n\n\n\r");  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\t\t %s \n\r", AppName);  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\n\n\n\r");  }  **static** **void** **BoardInit**(**void**)  {  // Enable Processor  MAP\_IntMasterEnable();  MAP\_IntEnable(FAULT\_SYSTICK);  **PRCMCC3200MCUInit**();  }  /\*\*  \*  \*/  **void** **SmartDoorlockApp**(**void** \*pvParameters) {  GPIO\_IF\_LedConfigure(*LED1*|*LED2*|*LED3*);  **for** (;;) {  GPIO\_IF\_LedOn(*MCU\_RED\_LED\_GPIO*);  **osi\_Sleep**(500);  GPIO\_IF\_LedOff(*MCU\_RED\_LED\_GPIO*);  **osi\_Sleep**(500);  }  }  **int** **main**(**void**) {  // Initailizing the board  BoardInit();  // Muxing for Enabling GPIO, UART\_TX and UART\_RX.  PinMuxConfig();  // Init Terminal  **InitTerm**();  **ClearTerm**();  DisplayBanner(APP\_NAME);  //Start the simplelink host  **VStartSimpleLinkSpawnTask**(SPAWN\_TASK\_PRIORITY);  // Start the SmartDoorlock task  **osi\_TaskCreate**( SmartDoorlockApp,  (**const** **signed** **char**\*)"Smart Doorlock App",  OSI\_STACK\_SIZE, NULL, 1, NULL );  **osi\_start**();  **return** 0;  } |

|  |
| --- |
| **gpio\_if.c** |
| /\*  \* gpio\_if.c  \*  \* Created on: 2016. 9. 7.  \* Author: Sokwhan  \*/  **#include** <stdio.h>  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "hw\_memmap.h"  **#include** "interrupt.h"  **#include** "pin.h"  **#include** "gpio.h"  **#include** "hw\_apps\_rcm.h"  **#include** "prcm.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "gpio\_if.h"  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // GLOBAL VARIABLES  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **static** **unsigned** **long** ulReg[]=  {  GPIOA0\_BASE,  GPIOA1\_BASE,  GPIOA2\_BASE,  GPIOA3\_BASE  };  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // Variables to store TIMER Port,Pin values  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **int** g\_uiLED1Port = 0,g\_uiLED2Port = 0,g\_uiLED3Port = 0;  **unsigned** **char** g\_ucLED1Pin, g\_ucLED2Pin, g\_ucLED3Pin;  **extern** **unsigned** **char** g\_heart\_beat, g\_state;  **#define** PIN\_LED1 9  **#define** PIN\_LED2 10  **#define** PIN\_LED3 11  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  // LOCAL FUNCTION DEFINITIONS  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! GPIO Enable & Configuration  //!  //! \param ucPins is the bit-pack representation of 3 LEDs  //! LSB:GP09-GP10-GP11:MSB  //!  //! \return None  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_LedConfigure**(**unsigned** **char** ucPins)  {  **if**(ucPins & *LED1*)  {  GPIO\_IF\_GetPortNPin(PIN\_LED1,  &g\_uiLED1Port,  &g\_ucLED1Pin);  }  **if**(ucPins & *LED2*)  {  GPIO\_IF\_GetPortNPin(PIN\_LED2,  &g\_uiLED2Port,  &g\_ucLED2Pin);  }  **if**(ucPins & *LED3*)  {  GPIO\_IF\_GetPortNPin(PIN\_LED3,  &g\_uiLED3Port,  &g\_ucLED3Pin);  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Turn LED On  //!  //! \param ledNum is the LED Number  //!  //! \return none  //!  //! \brief Turns a specific LED Off  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_LedOn**(**char** ledNum)  {  **switch**(ledNum)  {  **case** *MCU\_GREEN\_LED\_GPIO*:  {  /\* Switch ON GREEN LED \*/  GPIO\_IF\_Set(PIN\_LED3, g\_uiLED3Port, g\_ucLED3Pin, 1);  **break**;  }  **case** *MCU\_ORANGE\_LED\_GPIO*:  {  /\* Switch ON ORANGE LED \*/  GPIO\_IF\_Set(PIN\_LED2, g\_uiLED2Port, g\_ucLED2Pin, 1);  **break**;  }  **case** *MCU\_RED\_LED\_GPIO*:  {  /\* Switch ON RED LED \*/  GPIO\_IF\_Set(PIN\_LED1, g\_uiLED1Port, g\_ucLED1Pin, 1);  **break**;  }  **case** *MCU\_ALL\_LED\_IND*:  {  /\* Switch ON ALL LEDs LED \*/  GPIO\_IF\_Set(PIN\_LED3, g\_uiLED3Port, g\_ucLED3Pin, 1);  GPIO\_IF\_Set(PIN\_LED2, g\_uiLED2Port, g\_ucLED2Pin, 1);  GPIO\_IF\_Set(PIN\_LED1, g\_uiLED1Port, g\_ucLED1Pin, 1);  **break**;  }  **default**:  **break**;  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Turn LED Off  //!  //! \param ledNum is the LED Number  //!  //! \return none  //!  //! \brief Turns a specific LED Off  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_LedOff**(**char** ledNum)  {  **switch**(ledNum)  {  **case** *MCU\_GREEN\_LED\_GPIO*:  {  /\* Switch OFF GREEN LED \*/  GPIO\_IF\_Set(PIN\_LED3, g\_uiLED3Port, g\_ucLED3Pin, 0);  **break**;  }  **case** *MCU\_ORANGE\_LED\_GPIO*:  {  /\* Switch OFF ORANGE LED \*/  GPIO\_IF\_Set(PIN\_LED2, g\_uiLED2Port, g\_ucLED2Pin, 0);  **break**;  }  **case** *MCU\_RED\_LED\_GPIO*:  {  /\* Switch OFF RED LED \*/  GPIO\_IF\_Set(PIN\_LED1, g\_uiLED1Port, g\_ucLED1Pin, 0);  **break**;  }  **case** *MCU\_ALL\_LED\_IND*:  {  /\* Switch OFF ALL LEDs LED \*/  GPIO\_IF\_Set(PIN\_LED3, g\_uiLED3Port, g\_ucLED3Pin, 0);  GPIO\_IF\_Set(PIN\_LED2, g\_uiLED2Port, g\_ucLED2Pin, 0);  GPIO\_IF\_Set(PIN\_LED1, g\_uiLED1Port, g\_ucLED1Pin, 0);  **break**;  }  **default**:  **break**;  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! \brief This function returns LED current Status  //!  //! \param[in] ucGPIONum is the GPIO to which the LED is connected  //! MCU\_GREEN\_LED\_GPIO\MCU\_ORANGE\_LED\_GPIO\MCU\_RED\_LED\_GPIO  //!  //!  //! \return 1: LED ON, 0: LED OFF  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **char**  **GPIO\_IF\_LedStatus**(**unsigned** **char** ucGPIONum)  {  **unsigned** **char** ucLEDStatus;  **switch**(ucGPIONum)  {  **case** *MCU\_GREEN\_LED\_GPIO*:  {  ucLEDStatus = GPIO\_IF\_Get(ucGPIONum, g\_uiLED3Port, g\_ucLED3Pin);  **break**;  }  **case** *MCU\_ORANGE\_LED\_GPIO*:  {  ucLEDStatus = GPIO\_IF\_Get(ucGPIONum, g\_uiLED2Port, g\_ucLED2Pin);  **break**;  }  **case** *MCU\_RED\_LED\_GPIO*:  {  ucLEDStatus = GPIO\_IF\_Get(ucGPIONum, g\_uiLED1Port, g\_ucLED1Pin);  **break**;  }  **default**:  ucLEDStatus = 0;  }  **return** ucLEDStatus;  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Toggle the Led state  //!  //! \param ledNum is the LED Number  //!  //! \return none  //!  //! \brief Toggles a board LED  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void** **GPIO\_IF\_LedToggle**(**unsigned** **char** ucLedNum)  {  **unsigned** **char** ucLEDStatus = GPIO\_IF\_LedStatus(ucLedNum);  **if**(ucLEDStatus == 1)  {  GPIO\_IF\_LedOff(ucLedNum);  }  **else**  {  GPIO\_IF\_LedOn(ucLedNum);  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Get the port and pin of a given GPIO  //!  //! \param ucPin is the pin to be set-up as a GPIO (0:39)  //! \param puiGPIOPort is the pointer to store GPIO port address return value  //! \param pucGPIOPin is the pointer to store GPIO pin return value  //!  //! This function  //! 1. Return the GPIO port address and pin for a given external pin number  //!  //! \return None.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_GetPortNPin**(**unsigned** **char** ucPin,  **unsigned** **int** \*puiGPIOPort,  **unsigned** **char** \*pucGPIOPin)  {  //  // Get the GPIO pin from the external Pin number  //  \*pucGPIOPin = 1 << (ucPin % 8);  //  // Get the GPIO port from the external Pin number  //  \*puiGPIOPort = (ucPin / 8);  \*puiGPIOPort = ulReg[\*puiGPIOPort];  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Configures the GPIO selected as input to generate interrupt on activity  //!  //! \param uiGPIOPort is the GPIO port address  //! \param ucGPIOPin is the GPIO pin of the specified port  //! \param uiIntType is the type of the interrupt (refer gpio.h)  //! \param pfnIntHandler is the interrupt handler to register  //!  //! This function  //! 1. Sets GPIO interrupt type  //! 2. Registers Interrupt handler  //! 3. Enables Interrupt  //!  //! \return None  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_ConfigureNIntEnable**(**unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **int** uiIntType,  **void** (\*pfnIntHandler)(**void**))  {  //  // Set GPIO interrupt type  //  MAP\_GPIOIntTypeSet(uiGPIOPort,ucGPIOPin,uiIntType);  //  // Register Interrupt handler  //  MAP\_GPIOIntRegister(uiGPIOPort,pfnIntHandler);  //  // Enable Interrupt  //  MAP\_GPIOIntClear(uiGPIOPort,ucGPIOPin);  MAP\_GPIOIntEnable(uiGPIOPort,ucGPIOPin);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Set a value to the specified GPIO pin  //!  //! \param ucPin is the GPIO pin to be set (0:39)  //! \param uiGPIOPort is the GPIO port address  //! \param ucGPIOPin is the GPIO pin of the specified port  //! \param ucGPIOValue is the value to be set  //!  //! This function  //! 1. Sets a value to the specified GPIO pin  //!  //! \return None.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **GPIO\_IF\_Set**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **char** ucGPIOValue)  {  //  // Set the corresponding bit in the bitmask  //  ucGPIOValue = ucGPIOValue << (ucPin % 8);  //  // Invoke the API to set the value  //  MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,ucGPIOValue);  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! Set a value to the specified GPIO pin  //!  //! \param ucPin is the GPIO pin to be set (0:39)  //! \param uiGPIOPort is the GPIO port address  //! \param ucGPIOPin is the GPIO pin of the specified port  //!  //! This function  //! 1. Gets a value of the specified GPIO pin  //!  //! \return value of the GPIO pin  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **unsigned** **char**  **GPIO\_IF\_Get**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin)  {  **unsigned** **char** ucGPIOValue;  **long** lGPIOStatus;  //  // Invoke the API to Get the value  //  lGPIOStatus = MAP\_GPIOPinRead(uiGPIOPort,ucGPIOPin);  //  // Set the corresponding bit in the bitmask  //  ucGPIOValue = lGPIOStatus >> (ucPin % 8);  **return** ucGPIOValue;  }  **void** **GPIO\_Set**(**unsigned** **char** ucGPIONum){  **int** ucGPIOPin, uiGPIOPort;  ucGPIOPin = 1 << (ucGPIONum % 8);  uiGPIOPort = ulReg[ucGPIONum / 8];  MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,0xff);  }  **void** **GPIO\_Clear**(**unsigned** **char** ucGPIONum){  **int** ucGPIOPin, uiGPIOPort;  ucGPIOPin = 1 << (ucGPIONum % 8);  uiGPIOPort = ulReg[ucGPIONum / 8];  MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,0x0);  }  **void** **GPIOIntInit**(**unsigned** **long** ulPort, **unsigned** **char** ucPin, **unsigned** **long** ulInterrupt, **void** (\*pfnHandler)(**void**), **unsigned** **long** ulIntType, **unsigned** **char** ucPriority) {  **IntRegister**(ulInterrupt, pfnHandler);  **IntPrioritySet**(ulInterrupt, INT\_PRIORITY\_LVL\_1);  **GPIOIntTypeSet**(ulPort,ucPin,ulIntType);  **GPIOIntClear**(ulPort,0xFF);  **GPIOIntEnable**(ulPort,ucPin);  **IntEnable**(ulInterrupt);  } |

|  |
| --- |
| **gpio\_if.h** |
| /\*  \* gpio\_if.h  \*  \* Created on: 2016. 9. 7.  \* Author: Sokwhan  \*/  **#ifndef** \_\_GPIOIF\_H\_\_  **#define** \_\_GPIOIF\_H\_\_  **typedef** **enum**  {  *NO\_LED*,  *LED1* = 0x1, /\* RED LED D7/GP9/Pin64 \*/  *LED2* = 0x2, /\* ORANGE LED D6/GP10/Pin1 \*/  *LED3* = 0x4 /\* GREEN LED D5/GP11/Pin2 \*/  } ledEnum;  **typedef** **enum**  {  *MCU\_RED\_LED\_GPIO* = 9, /\* GP09 for LED RED as per LP 3.0 \*/  *MCU\_ORANGE\_LED\_GPIO* = 10,/\* GP10 for LED ORANGE as per LP 3.0 \*/  *MCU\_GREEN\_LED\_GPIO* = 11, /\* GP11 for LED GREEN as per LP 3.0 \*/  *MCU\_ALL\_LED\_IND* = 12  } ledNames;  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // API Function prototypes  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **extern** **void** **GPIO\_IF\_GetPortNPin**(**unsigned** **char** ucPin,  **unsigned** **int** \*puiGPIOPort,  **unsigned** **char** \*pucGPIOPin);  **extern** **void** **GPIO\_IF\_ConfigureNIntEnable**(**unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **int** uiIntType,  **void** (\*pfnIntHandler)(**void**));  **extern** **void** **GPIO\_IF\_Set**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **char** ucGPIOValue);  **extern** **unsigned** **char** **GPIO\_IF\_Get**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin);  **extern** **void** **GPIO\_IF\_LedConfigure**(**unsigned** **char** ucPins);  **extern** **void** **GPIO\_IF\_LedOn**(**char** ledNum);  **extern** **void** **GPIO\_IF\_LedOff**(**char** ledNum);  **extern** **unsigned** **char** **GPIO\_IF\_LedStatus**(**unsigned** **char** ucGPIONum);  **extern** **void** **GPIO\_IF\_LedToggle**(**unsigned** **char** ucLedNum);  **extern** **void** **GPIO\_Set**(**unsigned** **char** ucGPIONum);  **extern** **void** **GPIO\_Clear**(**unsigned** **char** ucGPIONum);  **extern** **void** **GPIO\_Toggle**(**unsigned** **char** ucGPIONum);  **void** **GPIOIntInit**( **unsigned** **long** ulPort,  **unsigned** **char** ucPin,  **unsigned** **long** ulInterrupt,  **void** (\*pfnHandler)(**void**),  **unsigned** **long** ulIntType,  **unsigned** **char** ucPriority);  **void** **GPIOs3IntHandler**(**void**);  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  // Mark the end of the C bindings section for C++ compilers.  //  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **#ifdef** \_\_cplusplus  }  **#endif**  **#endif** // \_\_GPIOIF\_H\_\_ |

|  |
| --- |
| **pinmux.c** |
| /\*  \* Smart Doorlock  \*  \* pinmux.c  \*  \* Created on: 2016. 8. 27.  \* Author: Sokwhan  \*/  **#include** "pinmux.h"  **#include** "hw\_types.h"  **#include** "hw\_memmap.h"  **#include** "hw\_gpio.h"  **#include** "pin.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "gpio.h"  **#include** "prcm.h"  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void**  **PinMuxConfig**(**void**)  {  // Enable Peripheral Clocks  MAP\_PRCMPeripheralClkEnable(PRCM\_UARTA0, PRCM\_RUN\_MODE\_CLK);  // Reserved for Future Use (Probably IRQ)  MAP\_PRCMPeripheralClkEnable(PRCM\_GPIOA0, PRCM\_RUN\_MODE\_CLK);  // Used for LED1,2,3  MAP\_PRCMPeripheralClkEnable(PRCM\_GPIOA1, PRCM\_RUN\_MODE\_CLK);  // Configure PIN\_55 for UART0 UART0\_TX  MAP\_PinTypeUART(PIN\_55, PIN\_MODE\_3);  // Configure PIN\_57 for UART0 UART0\_RX  MAP\_PinTypeUART(PIN\_57, PIN\_MODE\_3);  // Configure PIN\_64 for LED1 (GPIO9)  MAP\_PinTypeGPIO(PIN\_64, PIN\_MODE\_0, false);  MAP\_GPIODirModeSet(GPIOA1\_BASE, 0x2, GPIO\_DIR\_MODE\_OUT);  // Configure PIN\_01 for LED2  MAP\_PinTypeGPIO(PIN\_01, PIN\_MODE\_0, false);  MAP\_GPIODirModeSet(GPIOA1\_BASE, 0x4, GPIO\_DIR\_MODE\_OUT);  // Configure PIN\_02 for LED3  MAP\_PinTypeGPIO(PIN\_02, PIN\_MODE\_0, false);  MAP\_GPIODirModeSet(GPIOA1\_BASE, 0x8, GPIO\_DIR\_MODE\_OUT);  } |

# September 22nd, 2016

Over the week, the portion of LCD and keypad interfacing was worked on. Unfortunately, for LCD, the hardware needs a specific socket for it to be able to connect to a breadboard (or even jumper cables in this case), so I had to order those. The keypad interfacing worked flawlessly with a simple test code. Substantial changes were made to GPIO interfacing code in general.

|  |
| --- |
| **main.c (only test code portion is shown)** |
| **void** **SmartDoorlockApp**(**void** **\***pvParameters) {  **long** lRetVal **=** **-**1;  **unsigned** **int** uiConnectTimeoutCnt **=** 0;    *//GPIO\_IF\_LedConfigure(LED1|LED2|LED3);*    GPIO\_IF\_Set(PIN\_KEYPAD\_R1,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R2,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R3,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R4,1);    **unsigned** **int** rowEnum **=** 0;    **for** (;;) {  **unsigned** **int** test **=** 0;  **if** (rowEnum **==** 0) {  GPIO\_IF\_Set(PIN\_KEYPAD\_R1,0);  GPIO\_IF\_Set(PIN\_KEYPAD\_R2,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R3,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R4,1);  }  **else** **if** (rowEnum **==** 1) {  GPIO\_IF\_Set(PIN\_KEYPAD\_R1,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R2,0);  GPIO\_IF\_Set(PIN\_KEYPAD\_R3,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R4,1);  }  **else** **if** (rowEnum **==** 2) {  GPIO\_IF\_Set(PIN\_KEYPAD\_R1,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R2,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R3,0);  GPIO\_IF\_Set(PIN\_KEYPAD\_R4,1);  }  **else** **if** (rowEnum **==** 3) {  GPIO\_IF\_Set(PIN\_KEYPAD\_R1,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R2,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R3,1);  GPIO\_IF\_Set(PIN\_KEYPAD\_R4,0);  }  Report("------------------------");  Report("Row: %d", rowEnum **+** 1);  test **=** GPIO\_IF\_Get(PIN\_KEYPAD\_C1);  Report("C1: %d ",test);  test **=** GPIO\_IF\_Get(PIN\_KEYPAD\_C2);  Report("C2: %d ",test);  test **=** GPIO\_IF\_Get(PIN\_KEYPAD\_C3);  Report("C3: %d\n\r",test);  rowEnum**++**;  **if** (rowEnum **>=** 4)  rowEnum **=** 0;  osi\_Sleep(500);    } |

|  |
| --- |
| **gpio.h** |
| */\**  *\* gpio\_if.h*  *\**  *\* Created on: 2016. 9. 7.*  *\* Author: Sokwhan*  *\*/*    **#ifndef \_\_GPIOIF\_H\_\_**  **#define \_\_GPIOIF\_H\_\_**    **typedef** **enum**  {  PIN\_LCD\_RS **=** 0, *//4 RS (CS) H / L H=Data, L=Command*  PIN\_LCD\_RW **=** 3, *//5 R/W (SID) H / L H=Read, L=Write*  PIN\_LCD\_E **=** 4, *//6 E (SCLK) H Enable (falling edge)*  PIN\_LCD\_D0 **=** 5, *//7 D0 (SOD) H / L Display Data, LSB*  PIN\_LCD\_D1 **=** 6, *//8 D1 H / L Display Data*  PIN\_LCD\_D2 **=** 7, *//9 D2 H / L Display Data*  PIN\_LCD\_D3 **=** 8, *//10 D3 H / L Display Data*  PIN\_LCD\_D4 **=** 9, *//11 D4 (D0) H / L Display Data*  PIN\_LCD\_D5 **=** 10, *//12 D5 (D1) H / L Display Data*  PIN\_LCD\_D6 **=** 11, *//13 D6 (D2) H / L Display Data*  PIN\_LCD\_D7 **=** 12 *//14 D7 (D3) H / L Display Data, MSB*  } lcdPinEnum;    **typedef** **enum**  {  PIN\_KEYPAD\_C1 **=** 13,  PIN\_KEYPAD\_C2 **=** 14,  PIN\_KEYPAD\_C3 **=** 15,  PIN\_KEYPAD\_R1 **=** 17,  PIN\_KEYPAD\_R2 **=** 9,  PIN\_KEYPAD\_R3 **=** 30,  PIN\_KEYPAD\_R4 **=** 28  } keypadPinEnum;      *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *// API Function prototypes*  *//*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **extern** **void** **GPIO\_IF\_GetPortNPin**(**unsigned** **char** ucPin,  **unsigned** **int** **\***puiGPIOPort,  **unsigned** **char** **\***pucGPIOPin);    **extern** **void** **GPIO\_IF\_ConfigureNIntEnable**(**unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **int** uiIntType,  **void** (**\***pfnIntHandler)(**void**));  **extern** **void** **GPIO\_IF\_Set**(**unsigned** **int** gpioNum, **unsigned** **int** state);  **extern** **void** **GPIO\_IF\_Toggle**(**unsigned** **int** gpioNum);    **extern** **unsigned** **char** **GPIO\_IF\_GetVal**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin);  **static** **void** **GPIO\_IF\_SetVal**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **char** ucGPIOValue);  **extern** **void** **GPIO\_Set**(**unsigned** **char** ucGPIONum);  **extern** **void** **GPIO\_Clear**(**unsigned** **char** ucGPIONum);  **extern** **void** **GPIO\_Toggle**(**unsigned** **char** ucGPIONum);  **extern** **unsigned** **char** **GPIO\_IF\_Get**(**unsigned** **int** gpioNum);  **void** **GPIOIntInit**( **unsigned** **long** ulPort,  **unsigned** **char** ucPin,  **unsigned** **long** ulInterrupt,  **void** (**\***pfnHandler)(**void**),  **unsigned** **long** ulIntType,  **unsigned** **char** ucPriority);  **void** **GPIOs3IntHandler**(**void**);        *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *// Mark the end of the C bindings section for C++ compilers.*  *//*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **#ifdef \_\_cplusplus**  }  **#endif**    **#endif** *// \_\_GPIOIF\_H\_\_* |

|  |
| --- |
| **gpio.c** |
| */\**  *\* gpio\_if.c*  *\**  *\* Created on: 2016. 9. 7.*  *\* Author: Sokwhan*  *\*/*        **#include** <stdio.h>  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "hw\_memmap.h"  **#include** "interrupt.h"  **#include** "pin.h"  **#include** "gpio.h"  **#include** "hw\_apps\_rcm.h"  **#include** "prcm.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "gpio\_if.h"    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// GLOBAL VARIABLES*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**      **static** **unsigned** **long** ulReg[]**=**  {  GPIOA0\_BASE,  GPIOA1\_BASE,  GPIOA2\_BASE,  GPIOA3\_BASE  };    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// Variables to store TIMER Port,Pin values*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**      *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// LOCAL FUNCTION DEFINITIONS*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**    **void**  **GPIO\_IF\_Set**(**unsigned** **int** gpioNum, **unsigned** **int** state) {  **unsigned** **int** portNum **=** 0;  **unsigned** **char** pinNum;    GPIO\_IF\_GetPortNPin(gpioNum,  **&**portNum,  **&**pinNum);  GPIO\_IF\_SetVal(gpioNum, portNum, pinNum, state);  }    **void**  **GPIO\_IF\_Toggle**(**unsigned** **int** gpioNum) {  **unsigned** **int** portNum **=** 0;  **unsigned** **char** pinNum;  **unsigned** **int** portStatus;    GPIO\_IF\_GetPortNPin(gpioNum,  **&**portNum,  **&**pinNum);    portStatus **=** **!**GPIO\_IF\_GetVal(gpioNum, portNum, pinNum);  GPIO\_IF\_SetVal(gpioNum, portNum, pinNum, portStatus);  }    **unsigned** **char**  **GPIO\_IF\_Get**(**unsigned** **int** gpioNum) {  **unsigned** **int** portNum **=** 0;  **unsigned** **char** pinNum;    GPIO\_IF\_GetPortNPin(gpioNum,  **&**portNum,  **&**pinNum);    **return** GPIO\_IF\_GetVal(gpioNum, portNum, pinNum);  }      *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *//! Get the port and pin of a given GPIO*  *//!*  *//! \param ucPin is the pin to be set-up as a GPIO (0:39)*  *//! \param puiGPIOPort is the pointer to store GPIO port address return value*  *//! \param pucGPIOPin is the pointer to store GPIO pin return value*  *//!*  *//! This function*  *//! 1. Return the GPIO port address and pin for a given external pin number*  *//!*  *//! \return None.*  *//*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **void**  **GPIO\_IF\_GetPortNPin**(**unsigned** **char** ucPin,  **unsigned** **int** **\***puiGPIOPort,  **unsigned** **char** **\***pucGPIOPin)  {  *//*  *// Get the GPIO pin from the external Pin number*  *//*  **\***pucGPIOPin **=** 1 **<<** (ucPin **%** 8);    *//*  *// Get the GPIO port from the external Pin number*  *//*  **\***puiGPIOPort **=** (ucPin **/** 8);  **\***puiGPIOPort **=** ulReg[**\***puiGPIOPort];  }    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *//! Configures the GPIO selected as input to generate interrupt on activity*  *//!*  *//! \param uiGPIOPort is the GPIO port address*  *//! \param ucGPIOPin is the GPIO pin of the specified port*  *//! \param uiIntType is the type of the interrupt (refer gpio.h)*  *//! \param pfnIntHandler is the interrupt handler to register*  *//!*  *//! This function*  *//! 1. Sets GPIO interrupt type*  *//! 2. Registers Interrupt handler*  *//! 3. Enables Interrupt*  *//!*  *//! \return None*  *//*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **void**  **GPIO\_IF\_ConfigureNIntEnable**(**unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **int** uiIntType,  **void** (**\***pfnIntHandler)(**void**))  {  *//*  *// Set GPIO interrupt type*  *//*  MAP\_GPIOIntTypeSet(uiGPIOPort,ucGPIOPin,uiIntType);    *//*  *// Register Interrupt handler*  *//*    MAP\_GPIOIntRegister(uiGPIOPort,pfnIntHandler);      *//*  *// Enable Interrupt*  *//*  MAP\_GPIOIntClear(uiGPIOPort,ucGPIOPin);  MAP\_GPIOIntEnable(uiGPIOPort,ucGPIOPin);  }    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *//! Set a value to the specified GPIO pin*  *//!*  *//! \param ucPin is the GPIO pin to be set (0:39)*  *//! \param uiGPIOPort is the GPIO port address*  *//! \param ucGPIOPin is the GPIO pin of the specified port*  *//! \param ucGPIOValue is the value to be set*  *//!*  *//! This function*  *//! 1. Sets a value to the specified GPIO pin*  *//!*  *//! \return None.*  *//*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **static** **void**  **GPIO\_IF\_SetVal**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin,  **unsigned** **char** ucGPIOValue)  {  *//*  *// Set the corresponding bit in the bitmask*  *//*  ucGPIOValue **=** ucGPIOValue **<<** (ucPin **%** 8);    *//*  *// Invoke the API to set the value*  *//*  MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,ucGPIOValue);  }    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *//! Set a value to the specified GPIO pin*  *//!*  *//! \param ucPin is the GPIO pin to be set (0:39)*  *//! \param uiGPIOPort is the GPIO port address*  *//! \param ucGPIOPin is the GPIO pin of the specified port*  *//!*  *//! This function*  *//! 1. Gets a value of the specified GPIO pin*  *//!*  *//! \return value of the GPIO pin*  *//*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **unsigned** **char**  **GPIO\_IF\_GetVal**(**unsigned** **char** ucPin,  **unsigned** **int** uiGPIOPort,  **unsigned** **char** ucGPIOPin)  {  **unsigned** **char** ucGPIOValue;  **long** lGPIOStatus;    *//*  *// Invoke the API to Get the value*  *//*    lGPIOStatus **=** MAP\_GPIOPinRead(uiGPIOPort,ucGPIOPin);    *//*  *// Set the corresponding bit in the bitmask*  *//*  ucGPIOValue **=** lGPIOStatus **>>** (ucPin **%** 8);  **return** ucGPIOValue;  }        */\**  *void GPIO\_Set(unsigned char ucGPIONum){*    *int ucGPIOPin, uiGPIOPort;*  *ucGPIOPin = 1 << (ucGPIONum % 8);*  *uiGPIOPort = ulReg[ucGPIONum / 8];*    *MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,0xff);*  *}*    *void GPIO\_Clear(unsigned char ucGPIONum){*    *int ucGPIOPin, uiGPIOPort;*  *ucGPIOPin = 1 << (ucGPIONum % 8);*  *uiGPIOPort = ulReg[ucGPIONum / 8];*    *MAP\_GPIOPinWrite(uiGPIOPort,ucGPIOPin,0x0);*  *}*  *\*/*    **void** **GPIOIntInit**(**unsigned** **long** ulPort, **unsigned** **char** ucPin, **unsigned** **long** ulInterrupt, **void** (**\***pfnHandler)(**void**), **unsigned** **long** ulIntType, **unsigned** **char** ucPriority) {    IntRegister(ulInterrupt, pfnHandler);  IntPrioritySet(ulInterrupt, INT\_PRIORITY\_LVL\_1);      GPIOIntTypeSet(ulPort,ucPin,ulIntType);  GPIOIntClear(ulPort,0xFF);  GPIOIntEnable(ulPort,ucPin);  IntEnable(ulInterrupt);  } |

# September 26th, 2016

Commenced work on MQTT client code. The TI example is mostly straight forward, so implementing the protocol on the project was not troublesome. Tested the implementation on cloud server and results show that both publishing & subscribing on topics works pretty well.

|  |
| --- |
| **mqtt\_client.c** |
| */\**  *\* mqtt\_client.c*  *\**  *\* Created on: Sep 26, 2016*  *\* Author: shuh*  *\*/*      *// Standard includes*  **#include** <stdlib.h>    *// Common Interface Includes*  **#include** "common.h"  **#include** "sl\_mqtt\_client.h"  **#include** "uart\_if.h"  **#include** "network.h"    *// Project Includes*  **#include** "mqtt\_client.h"      */\*Operate Lib in MQTT 3.1 mode.\*/*  **#define MQTT\_3\_1\_1 false** */\*MQTT 3.1.1 \*/*  **#define MQTT\_3\_1 true** */\*MQTT 3.1\*/*    **#define WILL\_TOPIC "Client"**  **#define WILL\_MSG "Client Stopped"**  **#define WILL\_QOS QOS2**  **#define WILL\_RETAIN false**    */\*Background receive task priority\*/*  **#define TASK\_PRIORITY 3**    **#define UART\_PRINT Report**    */\*Defining Number of topics\*/*  **#define TOPIC\_COUNT 3**    */\* Keep Alive Timer value\*/*  **#define KEEP\_ALIVE\_TIMER 25**    */\*Defining Broker IP address and port Number\*/*  **#define SERVER\_ADDRESS "messagesight.demos.ibm.com"**  **#define PORT\_NUMBER 1883**    **#define SERVER\_MODE MQTT\_3\_1**  */\*Specifying Receive time out for the Receive task\*/*  **#define RCV\_TIMEOUT 30**    */\*Defining QOS levels\*/*  **#define QOS0 0**  **#define QOS1 1**  **#define QOS2 2**    */\*Defining Subscription Topic Values\*/*  **#define TOPIC1 "/cc3200/ToggleLEDCmdL1"**  **#define TOPIC2 "/cc3200/ToggleLEDCmdL2"**  **#define TOPIC3 "/cc3200/ToggleLEDCmdL3"**    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// LOCAL FUNCTION PROTOTYPES*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **static** **void**  **Mqtt\_Recv**(**void** **\***app\_hndl, **const** **char** **\***topstr, **long** top\_len, **const** **void** **\***payload,  **long** pay\_len, **bool** dup,**unsigned** **char** qos, **bool** retain);  **static** **void** **sl\_MqttEvt**(**void** **\***app\_hndl,**long** evt, **const** **void** **\***buf,  **unsigned** **long** len);  **static** **void** **sl\_MqttDisconnect**(**void** **\***app\_hndl);  **int** **initMqtt**();    */\* library configuration \*/*  SlMqttClientLibCfg\_t Mqtt\_Client**=**{  1882,  TASK\_PRIORITY,  30,  true,  (**long**(**\***)(**const** **char** **\***, ...))UART\_PRINT  };    **typedef** **struct** connection\_config{  SlMqttClientCtxCfg\_t broker\_config;  **void** **\***clt\_ctx;  **unsigned** **char** **\***client\_id;  **unsigned** **char** **\***usr\_name;  **unsigned** **char** **\***usr\_pwd;  **bool** is\_clean;  **unsigned** **int** keep\_alive\_time;  SlMqttClientCbs\_t CallBAcks;  **int** num\_topics;  **char** **\***topic[TOPIC\_COUNT];  **unsigned** **char** qos[TOPIC\_COUNT];  SlMqttWill\_t will\_params;  **bool** is\_connected;  }connect\_config;      */\* connection configuration \*/*  connect\_config usr\_connect\_config[] **=**  {  {  {  {  SL\_MQTT\_NETCONN\_URL,  SERVER\_ADDRESS,  PORT\_NUMBER,  0,  0,  0,  NULL  },  SERVER\_MODE,  true,  },  NULL,  "user1",  NULL,  NULL,  true,  KEEP\_ALIVE\_TIMER,  {Mqtt\_Recv, sl\_MqttEvt, sl\_MqttDisconnect},  TOPIC\_COUNT,  {TOPIC1, TOPIC2, TOPIC3},  {QOS2, QOS2, QOS2},  {WILL\_TOPIC,WILL\_MSG,WILL\_QOS,WILL\_RETAIN},  false  }  };      */\*Message Queue\*/*  OsiMsgQ\_t g\_PBQueue;    **void** **\***app\_hndl **=** (**void\***)usr\_connect\_config;      **int** **initMqtt**() {  **long** lRetVal **=** **-**1;  lRetVal **=** sl\_ExtLib\_MqttClientInit(**&**Mqtt\_Client);  **if**(lRetVal **!=** 0)  {  *// lib initialization failed*  UART\_PRINT("MQTT Client lib initialization failed\n\r");  **return** **-**1;  }    **int** iCount **=** 0;  event\_msg RecvQue;    connect\_config **\***local\_con\_conf **=** (connect\_config **\***)app\_hndl;    connect\_to\_broker:  *//create client context*  local\_con\_conf[iCount].clt\_ctx **=**  sl\_ExtLib\_MqttClientCtxCreate(**&**local\_con\_conf[iCount].broker\_config,  **&**local\_con\_conf[iCount].CallBAcks,  **&**(local\_con\_conf[iCount]));    *//*  *// Set Client ID*  *//*  sl\_ExtLib\_MqttClientSet((**void\***)local\_con\_conf[iCount].clt\_ctx,  SL\_MQTT\_PARAM\_CLIENT\_ID,  local\_con\_conf[iCount].client\_id,  strlen((**char\***)(local\_con\_conf[iCount].client\_id)));    *//*  *// Set will Params*  *//*  **if**(local\_con\_conf[iCount].will\_params.will\_topic **!=** NULL)  {  sl\_ExtLib\_MqttClientSet((**void\***)local\_con\_conf[iCount].clt\_ctx,  SL\_MQTT\_PARAM\_WILL\_PARAM,  **&**(local\_con\_conf[iCount].will\_params),  **sizeof**(SlMqttWill\_t));  }    *//*  *// setting username and password*  *//*  **if**(local\_con\_conf[iCount].usr\_name **!=** NULL)  {  sl\_ExtLib\_MqttClientSet((**void\***)local\_con\_conf[iCount].clt\_ctx,  SL\_MQTT\_PARAM\_USER\_NAME,  local\_con\_conf[iCount].usr\_name,  strlen((**char\***)local\_con\_conf[iCount].usr\_name));    **if**(local\_con\_conf[iCount].usr\_pwd **!=** NULL)  {  sl\_ExtLib\_MqttClientSet((**void\***)local\_con\_conf[iCount].clt\_ctx,  SL\_MQTT\_PARAM\_PASS\_WORD,  local\_con\_conf[iCount].usr\_pwd,  strlen((**char\***)local\_con\_conf[iCount].usr\_pwd));  }  }    *//*  *// connectin to the broker*  *//*  **if**((sl\_ExtLib\_MqttClientConnect((**void\***)local\_con\_conf[iCount].clt\_ctx,  local\_con\_conf[iCount].is\_clean,  local\_con\_conf[iCount].keep\_alive\_time) **&** 0xFF) **!=** 0)  {  UART\_PRINT("\n\rBroker connect fail for conn no. %d \n\r",iCount**+**1);    *//delete the context for this connection*  sl\_ExtLib\_MqttClientCtxDelete(local\_con\_conf[iCount].clt\_ctx);    **return** **-**1;  }  **else**  {  UART\_PRINT("\n\rSuccess: conn to Broker no. %d\n\r ", iCount**+**1);  local\_con\_conf[iCount].is\_connected **=** true;  }    *//*  *// Subscribe to topics*  *//*    **if**(sl\_ExtLib\_MqttClientSub((**void\***)local\_con\_conf[iCount].clt\_ctx,  local\_con\_conf[iCount].topic,  local\_con\_conf[iCount].qos, TOPIC\_COUNT) **<** 0)  {  UART\_PRINT("\n\r Subscription Error for conn no. %d\n\r", iCount**+**1);  UART\_PRINT("Disconnecting from the broker\r\n");  sl\_ExtLib\_MqttClientDisconnect(local\_con\_conf[iCount].clt\_ctx);  local\_con\_conf[iCount].is\_connected **=** false;    *//delete the context for this connection*  sl\_ExtLib\_MqttClientCtxDelete(local\_con\_conf[iCount].clt\_ctx);  **return** **-**1;  }  **else**  {  **int** iSub;  UART\_PRINT("Client subscribed on following topics:\n\r");  **for**(iSub **=** 0; iSub **<** local\_con\_conf[iCount].num\_topics; iSub**++**)  {  UART\_PRINT("%s\n\r", local\_con\_conf[iCount].topic[iSub]);  }  }    **for**(;;)  {  osi\_MsgQRead( **&**g\_PBQueue, **&**RecvQue, OSI\_WAIT\_FOREVER);    **if**(BROKER\_DISCONNECTION **==** RecvQue.event)  {  */\* Derive the value of the local\_con\_conf or clt\_ctx from the message \*/*  sl\_ExtLib\_MqttClientCtxDelete(((connect\_config**\***)(RecvQue.hndl))**->**clt\_ctx);    **if**(**!**IS\_CONNECTED(g\_ulStatus))  {  UART\_PRINT("device has disconnected from AP \n\r");    UART\_PRINT("retry connection to the AP\n\r");    **while**(**!**(IS\_CONNECTED(g\_ulStatus)) **||** **!**(IS\_IP\_ACQUIRED(g\_ulStatus)))  {  osi\_Sleep(10);  }  **goto** connect\_to\_broker;    }  *//*  *// device not connected to any broker*  *//*  **goto** end;    }  }  end:  *//*  *// Deinitializating the client library*  *//*  sl\_ExtLib\_MqttClientExit();  UART\_PRINT("\n\r Exiting the Application\n\r");    **return** 0;  }      *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//! Defines Mqtt\_Pub\_Message\_Receive event handler.*  *//! Client App needs to register this event handler with sl\_ExtLib\_mqtt\_Init*  *//! API. Background receive task invokes this handler whenever MQTT Client*  *//! receives a Publish Message from the broker.*  *//!*  *//!\param[out] topstr => pointer to topic of the message*  *//!\param[out] top\_len => topic length*  *//!\param[out] payload => pointer to payload*  *//!\param[out] pay\_len => payload length*  *//!\param[out] retain => Tells whether its a Retained message or not*  *//!\param[out] dup => Tells whether its a duplicate message or not*  *//!\param[out] qos => Tells the Qos level*  *//!*  *//!\return none*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **static** **void**  **Mqtt\_Recv**(**void** **\***app\_hndl, **const** **char** **\***topstr, **long** top\_len, **const** **void** **\***payload,  **long** pay\_len, **bool** dup,**unsigned** **char** qos, **bool** retain)  {    **char** **\***output\_str**=**(**char\***)malloc(top\_len**+**1);  memset(output\_str,'\0',top\_len**+**1);  strncpy(output\_str, (**char\***)topstr, top\_len);  output\_str[top\_len]**=**'\0';      **if**(strncmp(output\_str,TOPIC1, top\_len) **==** 0)  {  *//ToggleLedState(LED1);*  }  **else** **if**(strncmp(output\_str,TOPIC2, top\_len) **==** 0)  {  *//ToggleLedState(LED2);*  }  **else** **if**(strncmp(output\_str,TOPIC3, top\_len) **==** 0)  {  *//ToggleLedState(LED3);*  }    UART\_PRINT("\n\rPublish Message Received");  UART\_PRINT("\n\rTopic: ");  UART\_PRINT("%s",output\_str);  free(output\_str);  UART\_PRINT(" [Qos: %d] ",qos);  **if**(retain)  UART\_PRINT(" [Retained]");  **if**(dup)  UART\_PRINT(" [Duplicate]");    output\_str**=**(**char\***)malloc(pay\_len**+**1);  memset(output\_str,'\0',pay\_len**+**1);  strncpy(output\_str, (**char\***)payload, pay\_len);  output\_str[pay\_len]**=**'\0';  UART\_PRINT("\n\rData is: ");  UART\_PRINT("%s",(**char\***)output\_str);  UART\_PRINT("\n\r");  free(output\_str);    **return**;  }      *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//! Defines sl\_MqttEvt event handler.*  *//! Client App needs to register this event handler with sl\_ExtLib\_mqtt\_Init*  *//! API. Background receive task invokes this handler whenever MQTT Client*  *//! receives an ack(whenever user is in non-blocking mode) or encounters an error.*  *//!*  *//! param[out] evt => Event that invokes the handler. Event can be of the*  *//! following types:*  *//! MQTT\_ACK - Ack Received*  *//! MQTT\_ERROR - unknown error*  *//!*  *//!*  *//! \param[out] buf => points to buffer*  *//! \param[out] len => buffer length*  *//!*  *//! \return none*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **static** **void**  **sl\_MqttEvt**(**void** **\***app\_hndl, **long** evt, **const** **void** **\***buf,**unsigned** **long** len)  {  **int** i;  **switch**(evt)  {  **case** SL\_MQTT\_CL\_EVT\_PUBACK:  UART\_PRINT("PubAck:\n\r");  UART\_PRINT("%s\n\r",buf);  **break**;    **case** SL\_MQTT\_CL\_EVT\_SUBACK:  UART\_PRINT("\n\rGranted QoS Levels are:\n\r");    **for**(i**=**0;i**<**len;i**++**)  {  UART\_PRINT("QoS %d\n\r",((**unsigned** **char\***)buf)[i]);  }  **break**;    **case** SL\_MQTT\_CL\_EVT\_UNSUBACK:  UART\_PRINT("UnSub Ack \n\r");  UART\_PRINT("%s\n\r",buf);  **break**;    **default:**  **break**;    }  }    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *//! callback event in case of MQTT disconnection*  *//!*  *//! \param app\_hndl is the handle for the disconnected connection*  *//!*  *//! return none*  *//*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **static** **void**  **sl\_MqttDisconnect**(**void** **\***app\_hndl)  {  connect\_config **\***local\_con\_conf;  event\_msg msg;  local\_con\_conf **=** app\_hndl;  msg.hndl **=** app\_hndl;  msg.event **=** BROKER\_DISCONNECTION;    UART\_PRINT("disconnect from broker %s\r\n",  (local\_con\_conf**->**broker\_config).server\_info.server\_addr);  local\_con\_conf**->**is\_connected **=** false;  *//*  *// write message indicating publish message*  *//*  osi\_MsgQWrite(**&**g\_PBQueue,**&**msg,OSI\_NO\_WAIT);    } |

|  |
| --- |
| **mqtt\_client.h** |
| */\**  *\* mqtt\_client.h*  *\**  *\* Created on: Sep 26, 2016*  *\* Author: shuh*  *\*/*    **#ifndef MQTT\_CLIENT\_H\_**  **#define MQTT\_CLIENT\_H\_**    **typedef** **enum**  {  PUSH\_BUTTON\_SW2\_PRESSED,  PUSH\_BUTTON\_SW3\_PRESSED,  BROKER\_DISCONNECTION  }events;    **typedef** **struct**  {  **void** **\*** hndl;  events event;  }event\_msg;    **extern** **int** **initMqtt**();    **extern** OsiMsgQ\_t g\_PBQueue;      **#endif** */\* MQTT\_CLIENT\_H\_ \*/* |

# September 27th, 2016

**Source Control (Mercurial)**: https://bitbucket.org/l46kok/smart-doorlock-web-server

Created a repository for Smart Doorlock web server. Originally, I was planning to use ASP.NET MVC for web server, but the framework is a bit too heavy for the cloud server (Amazon EC2). I’ve decided to use NancyFX instead - testing it on my local machine showed that it has a very low memory consumption, making it pretty ideal for this project. For MQTT server, there is an open source .NET project just for this purpose known as GnatMQ. (<https://github.com/ppatierno/gnatmq>). The documentation is lacking (actually I’d say it’s practically non-existent), but the source code itself is nicely written, so determining its functionalities by reading over the source code was not difficult at all.

As mentioned in the project requirement section, the language for the webserver is C#. The following files are for the back-end of the server

|  |
| --- |
| **Program.cs** |
| **using** Nancy.Hosting.Self;  **using** System;  **using** System.Collections.Generic;  **using** System.Linq;  **using** System.Text;  **using** System.Threading.Tasks;    **namespace** smart\_doorlock\_web\_server  {  **class** **Program**  {  **private** **const** **string** WEB\_SERVER\_PORT = "1234";  **private** **const** **string** TERMINATE\_STRING = "/Terminate";    **static** **void** **Main**(**string**[] args)  {  MqttBroker mqttBroker = MqttBroker.Instance;  **#if TRACE**  *//MqttUtility.Trace.TraceLevel = MqttUtility.TraceLevel.Verbose | MqttUtility.TraceLevel.Frame;*  *//MqttUtility.Trace.TraceListener = (f, a) => System.Diagnostics.Trace.WriteLine(System.String.Format(f, a));*  **#endif**  mqttBroker.Start();    **var** uri = **new** Uri("http://localhost:" + WEB\_SERVER\_PORT + "/");  **var** config = **new** HostConfiguration  {  UrlReservations = { CreateAutomatically = **true** },  AllowChunkedEncoding = **false**  };    **var** host = **new** NancyHost(config, uri);    **try**  {  host.Start();    Console.Write("Smart Doorlock Web Server\n" +  "\t\"" + uri + "\"\n" +  "To quit, input \"" + TERMINATE\_STRING + "\".\n\n");  **do** Console.Write("> "); **while** (Console.ReadLine() != TERMINATE\_STRING);  }  **catch** (Exception e)  {  Console.WriteLine("Unhandled exception has been occured!\n"  + e.Message);  Console.ReadKey(**true**);  }  **finally**  {  host.Stop();  mqttBroker.Stop();  }  }  }  } |

|  |
| --- |
| **WebServerMainModule.cs** |
| **using** Nancy;  **using** System;  **using** System.Collections.Generic;  **using** System.Linq;  **using** System.Text;  **using** System.Threading.Tasks;    **namespace** smart\_doorlock\_web\_server  {  **public** **class** **WebServerMainModule** : NancyModule  {  MainPageSL mainPageSl = MainPageSL.Instance;  MqttBroker mqttBroker = MqttBroker.Instance;  **public** **WebServerMainModule**()  {  Get["/"] = \_ =>  {  *//MqttBroker.Instance.Publish("/cc3200/ToggleLEDCmdL1", "Test2");*  **return** View["Views/MainPage.sshtml",mainPageSl.GetMainPageData()];  };    Post["/Subscribe"] = x =>  {  **string** clientId = Request.Form.clientId;  **string** topic = Request.Form.topic;  **if** (mqttBroker.Subscribe(clientId, topic))  **return** "OK";  **return** "Fail";  };    Post["/Publish"] = x =>  {  **string** clientId = Request.Form.clientId;  **string** topic = Request.Form.topic;  **string** message = Request.Form.message;  mqttBroker.Publish(topic, message);    **return** "OK";  };  }  }  } |

|  |
| --- |
| **MqttBroker.cs** |
| **using** System;  **using** System.Collections.Generic;  **using** System.Linq;  **using** System.Text;  **using** uPLibrary.Networking.M2Mqtt.Messages;  **using** uPLibrary.Networking.M2Mqtt.Exceptions;  **using** uPLibrary.Networking.M2Mqtt.Managers;  **using** uPLibrary.Networking.M2Mqtt.Communication;  **using** uPLibrary.Networking.M2Mqtt.Session;  **using** System.Security.Cryptography.X509Certificates;  **using** System.Net.Security;  **using** uPLibrary.Networking.M2Mqtt;    **namespace** smart\_doorlock\_web\_server  {  */// <summary>*  */// MQTT broker business logic*  */// </summary>*  **public** **class** **MqttBroker**  {  **internal** **const** **byte** PROTOCOL\_VERSION\_V3\_1 = 0x03;  **internal** **const** **byte** PROTOCOL\_VERSION\_V3\_1\_1 = 0x04; *// [v.3.1.1]*    *// max length for client id (removed in 3.1.1)*  **internal** **const** **int** CLIENT\_ID\_MAX\_LENGTH = 23;    *// MQTT broker settings*  **private** MqttSettings settings;    *// clients connected list*  **private** MqttClientCollection clients;    *// reference to publisher manager*  **private** MqttPublisherManager publisherManager;    *// reference to subscriber manager*  **private** MqttSubscriberManager subscriberManager;    *// reference to session manager*  **private** MqttSessionManager sessionManager;    *// reference to User Access Control manager*  **private** MqttUacManager uacManager;    *// MQTT communication layer*  **private** IMqttCommunicationLayer commLayer;    **private** **static** **readonly** MqttBroker \_instance = **new** MqttBroker();  **public** **static** MqttBroker Instance  {  **get** {  **return** \_instance;  }  }    */// <summary>*  */// User authentication method*  */// </summary>*  **public** MqttUserAuthenticationDelegate UserAuth  {  **get** { **return** **this**.uacManager.UserAuth; }  **set** { **this**.uacManager.UserAuth = **value**; }  }    */// <summary>*  */// Constructor (TCP/IP communication layer on port 1883 and default settings)*  */// </summary>*  **private** **MqttBroker**()  : **this**(**new** MqttTcpCommunicationLayer(MqttSettings.MQTT\_BROKER\_DEFAULT\_PORT), MqttSettings.Instance)  {  }    */// <summary>*  */// Constructor*  */// </summary>*  */// <param name="commLayer">Communication layer to use (TCP)</param>*  */// <param name="settings">Broker settings</param>*  **private** **MqttBroker**(IMqttCommunicationLayer commLayer, MqttSettings settings)  {  *// MQTT broker settings*  **this**.settings = settings;    *// MQTT communication layer*  **this**.commLayer = commLayer;  **this**.commLayer.ClientConnected += commLayer\_ClientConnected;    *// create managers (publisher, subscriber, session and UAC)*  **this**.subscriberManager = **new** MqttSubscriberManager();  **this**.sessionManager = **new** MqttSessionManager();  **this**.publisherManager = **new** MqttPublisherManager(**this**.subscriberManager, **this**.sessionManager);  **this**.uacManager = **new** MqttUacManager();    **this**.clients = **new** MqttClientCollection();  }    */// <summary>*  */// Start broker*  */// </summary>*  **public** **void** **Start**()  {  **this**.commLayer.Start();  **this**.publisherManager.Start();  }    */// <summary>*  */// Stop broker*  */// </summary>*  **public** **void** **Stop**()  {  **this**.commLayer.Stop();  **this**.publisherManager.Stop();    *// close connection with all clients*  **foreach** (MqttClient client **in** **this**.clients)  {  client.Close();  }  }    **public** **void** **Publish**(**string** topic, **string** message)  {  MqttMsgPublish publishMsg = **new** MqttMsgPublish(topic, Encoding.UTF8.GetBytes(message),**false**,1,**false**);  publisherManager.Publish(publishMsg);  }  **public** MqttClientCollection **GetClientList**()  {  **return** clients;  }    **public** **bool** **Subscribe**(**string** clientId, **string** topic)  {  MqttClient client = GetClient(clientId);  **if** (client == **null**)  **return** **false**;  **byte**[] qosLevel = **new** **byte**[] {MqttMsgBase.QOS\_LEVEL\_AT\_LEAST\_ONCE};  **string**[] topicParam = **new** **string**[] { topic };  client.Subscribe(topicParam,qosLevel);  **return** **true**;  }    */// <summary>*  */// Close a client*  */// </summary>*  */// <param name="client">Client to close</param>*  **private** **void** **CloseClient**(MqttClient client)  {  **if** (**this**.clients.Contains(client))  {  *// if client is connected and it has a will message*  **if** (!client.IsConnected && client.WillFlag)  {  *// create the will PUBLISH message*  MqttMsgPublish publish =  **new** **MqttMsgPublish**(client.WillTopic, Encoding.UTF8.GetBytes(client.WillMessage), **false**, client.WillQosLevel, **false**);    *// publish message through publisher manager*  **this**.publisherManager.Publish(publish);  }    *// if not clean session*  **if** (!client.CleanSession)  {  List<MqttSubscription> subscriptions = **this**.subscriberManager.GetSubscriptionsByClient(client.ClientId);    **if** ((subscriptions != **null**) && (subscriptions.Count > 0))  {  **this**.sessionManager.SaveSession(client.ClientId, client.Session, subscriptions);    *// TODO : persist client session if broker close*  }  }    *// delete client from runtime subscription*  **this**.subscriberManager.Unsubscribe(client);    *// close the client*  client.Close();    *// remove client from the collection*  **this**.clients.Remove(client);  }  }    **void** **commLayer\_ClientConnected**(**object** sender, MqttClientConnectedEventArgs e)  {  *// register event handlers from client*  e.Client.MqttMsgDisconnected += Client\_MqttMsgDisconnected;  e.Client.MqttMsgPublishReceived += Client\_MqttMsgPublishReceived;  e.Client.MqttMsgConnected += Client\_MqttMsgConnected;  e.Client.MqttMsgSubscribeReceived += Client\_MqttMsgSubscribeReceived;  e.Client.MqttMsgUnsubscribeReceived += Client\_MqttMsgUnsubscribeReceived;  e.Client.ConnectionClosed += Client\_ConnectionClosed;    *// add client to the collection*  **this**.clients.Add(e.Client);    *// start client threads*  e.Client.Open();  }    **void** **Client\_MqttMsgPublishReceived**(**object** sender, MqttMsgPublishEventArgs e)  {  MqttClient client = (MqttClient)sender;    *// create PUBLISH message to publish*  *// [v3.1.1] DUP flag from an incoming PUBLISH message is not propagated to subscribers*  *// It should be set in the outgoing PUBLISH message based on transmission for each subscriber*  MqttMsgPublish publish = **new** MqttMsgPublish(e.Topic, e.Message, **false**, e.QosLevel, e.Retain);    Console.WriteLine("Publish Received: [{0},{1}]",e.Topic,e.Message);    *// publish message through publisher manager*  **this**.publisherManager.Publish(publish);  }    **void** **Client\_MqttMsgUnsubscribeReceived**(**object** sender, MqttMsgUnsubscribeEventArgs e)  {  MqttClient client = (MqttClient)sender;    **for** (**int** i = 0; i < e.Topics.Length; i++)  {  *// unsubscribe client for each topic requested*  **this**.subscriberManager.Unsubscribe(e.Topics[i], client);  }    **try**  {  *// send UNSUBACK message to the client*  client.Unsuback(e.MessageId);  }  **catch** (MqttCommunicationException)  {  **this**.CloseClient(client);  }  }    **void** **Client\_MqttMsgSubscribeReceived**(**object** sender, MqttMsgSubscribeEventArgs e)  {  MqttClient client = (MqttClient)sender;  Console.WriteLine("Subscribe Received");  **for** (**int** i = 0; i < e.Topics.Length; i++)  {  *// TODO : business logic to grant QoS levels based on some conditions ?*  *// now the broker granted the QoS levels requested by client*    *// subscribe client for each topic and QoS level requested*  **this**.subscriberManager.Subscribe(e.Topics[i], e.QoSLevels[i], client);    Console.WriteLine("Topic: " + e.Topics[i]);  }    **try**  {  *// send SUBACK message to the client*  client.Suback(e.MessageId, e.QoSLevels);    **for** (**int** i = 0; i < e.Topics.Length; i++)  {  *// publish retained message on the current subscription*  **this**.publisherManager.PublishRetaind(e.Topics[i], client.ClientId);  }  }  **catch** (MqttCommunicationException)  {  **this**.CloseClient(client);  }  }    **void** **Client\_MqttMsgConnected**(**object** sender, MqttMsgConnectEventArgs e)  {  *// [v3.1.1] session present flag*  **bool** sessionPresent = **false**;  *// [v3.1.1] generated client id for client who provides client id zero bytes length*  **string** clientId = **null**;    MqttClient client = (MqttClient)sender;    *// verify message to determine CONNACK message return code to the client*  **byte** returnCode = **this**.MqttConnectVerify(e.Message);    *// [v3.1.1] if client id is zero length, the broker assigns a unique identifier to it*  clientId = (e.Message.ClientId.Length != 0) ? e.Message.ClientId : Guid.NewGuid().ToString();    *// connection "could" be accepted*  **if** (returnCode == MqttMsgConnack.CONN\_ACCEPTED)  {  *// check if there is a client already connected with same client Id*  MqttClient clientConnected = **this**.GetClient(clientId);    *// force connection close to the existing client (MQTT protocol)*  **if** (clientConnected != **null**)  {  **this**.CloseClient(clientConnected);  }  }    **try**  {  *// connection accepted, load (if exists) client session*  **if** (returnCode == MqttMsgConnack.CONN\_ACCEPTED)  {  *// check if not clean session and try to recovery a session*  **if** (!e.Message.CleanSession)  {  *// create session for the client*  MqttClientSession clientSession = **new** MqttClientSession(clientId);    *// get session for the connected client*  MqttBrokerSession session = **this**.sessionManager.GetSession(clientId);    *// set inflight queue into the client session*  **if** (session != **null**)  {  clientSession.InflightMessages = session.InflightMessages;  *// [v3.1.1] session present flag*  **if** (client.ProtocolVersion == MqttProtocolVersion.Version\_3\_1\_1)  sessionPresent = **true**;  }    *// send CONNACK message to the client*  client.Connack(e.Message, returnCode, clientId, sessionPresent);    *// load/inject session to the client*  client.LoadSession(clientSession);    **if** (session != **null**)  {  *// set reference to connected client into the session*  session.Client = client;    *// there are saved subscriptions*  **if** (session.Subscriptions != **null**)  {  *// register all subscriptions for the connected client*  **foreach** (MqttSubscription subscription **in** session.Subscriptions)  {  **this**.subscriberManager.Subscribe(subscription.Topic, subscription.QosLevel, client);    *// publish retained message on the current subscription*  **this**.publisherManager.PublishRetaind(subscription.Topic, clientId);  }  }    *// there are saved outgoing messages*  **if** (session.OutgoingMessages.Count > 0)  {  *// publish outgoing messages for the session*  **this**.publisherManager.PublishSession(session.ClientId);  }  }  }  *// requested clean session*  **else**  {  *// send CONNACK message to the client*  client.Connack(e.Message, returnCode, clientId, sessionPresent);    **this**.sessionManager.ClearSession(clientId);  }  }  **else**  {  *// send CONNACK message to the client*  client.Connack(e.Message, returnCode, clientId, sessionPresent);  }  }  **catch** (MqttCommunicationException)  {  **this**.CloseClient(client);  }  }    **void** **Client\_MqttMsgDisconnected**(**object** sender, EventArgs e)  {  MqttClient client = (MqttClient)sender;    *// close the client*  **this**.CloseClient(client);  }    **void** **Client\_ConnectionClosed**(**object** sender, EventArgs e)  {  MqttClient client = (MqttClient)sender;    *// close the client*  **this**.CloseClient(client);  }    */// <summary>*  */// Check CONNECT message to accept or not the connection request*  */// </summary>*  */// <param name="connect">CONNECT message received from client</param>*  */// <returns>Return code for CONNACK message</returns>*  **private** **byte** **MqttConnectVerify**(MqttMsgConnect connect)  {  **byte** returnCode = MqttMsgConnack.CONN\_ACCEPTED;    *// unacceptable protocol version*  **if** ((connect.ProtocolVersion != PROTOCOL\_VERSION\_V3\_1) &&  (connect.ProtocolVersion != PROTOCOL\_VERSION\_V3\_1\_1))  returnCode = MqttMsgConnack.CONN\_REFUSED\_PROT\_VERS;  **else**  {  *// client id length exceeded (only for old MQTT 3.1)*  **if** ((connect.ProtocolVersion == PROTOCOL\_VERSION\_V3\_1) &&  (connect.ClientId.Length > CLIENT\_ID\_MAX\_LENGTH))  returnCode = MqttMsgConnack.CONN\_REFUSED\_IDENT\_REJECTED;  **else**  {  *// [v.3.1.1] client id zero length is allowed but clean session must be true*  **if** ((connect.ClientId.Length == 0) && (!connect.CleanSession))  returnCode = MqttMsgConnack.CONN\_REFUSED\_IDENT\_REJECTED;  **else**  {  *// check user authentication*  **if** (!**this**.uacManager.UserAuthentication(connect.Username, connect.Password))  returnCode = MqttMsgConnack.CONN\_REFUSED\_USERNAME\_PASSWORD;  *// server unavailable and not authorized ?*  **else**  {  *// TODO : other checks on CONNECT message*  }  }  }  }    **return** returnCode;  }    */// <summary>*  */// Return reference to a client with a specified Id is already connected*  */// </summary>*  */// <param name="clientId">Client Id to verify</param>*  */// <returns>Reference to client</returns>*  **private** MqttClient **GetClient**(**string** clientId)  {  **var** query = **from** c **in** **this**.clients  **where** c.ClientId == clientId  **select** c;    **return** query.FirstOrDefault();  }  }  } |

|  |
| --- |
| **MainPageSL.cs** |
| **using** System;  **using** System.Collections.Generic;  **using** System.Linq;  **using** System.Text;  **using** System.Threading.Tasks;  **using** uPLibrary.Networking.M2Mqtt;    **namespace** smart\_doorlock\_web\_server  {  **public** **class** **MainPageSL**  {  MqttBroker mqttBroker = MqttBroker.Instance;  **private** **static** **readonly** MainPageSL \_instance = **new** MainPageSL();  **public** **static** MainPageSL Instance  {  **get**  {  **return** \_instance;  }  }    **private** **MainPageSL**() { }  **public** List<MainPageData> GetMainPageData()  {  List<MainPageData> dataList = **new** List<MainPageData>();  MqttClientCollection clientList = mqttBroker.GetClientList();  **int** doorlockNo = 1;  **foreach** (MqttClient client **in** clientList) {  MainPageData data = **new** MainPageData();  data.DoorlockNo = doorlockNo;  data.ClientId = client.ClientId;  data.ProtocolVer = client.ProtocolVersion.ToString();  dataList.Add(data);  doorlockNo++;  }    **return** dataList;  }  }  } |

|  |
| --- |
| **MainPageData.cs** |
| **using** System;  **using** System.Collections.Generic;  **using** System.Linq;  **using** System.Text;  **using** System.Threading.Tasks;    **namespace** smart\_doorlock\_web\_server  {  **public** **class** **MainPageData**  {  **public** **int** DoorlockNo {**get**; **set**;}  **public** **string** ClientId { **get**; **set**; }  **public** **string** ProtocolVer { **get**; **set**; }  }  } |

The following portions are the front-end portion of the server. SuperSimpleViewEngine was used to render the pages. Due to the verbose nature of HTML and excessive tabbing required, the front-end code is a bit difficult to read on this notebook. Referencing the front-end portion of the server code from the source control is recommended.

|  |
| --- |
| **MainPage.sshtml** |
| @Master['MasterPage']    @Section['Content']  <link href="./Content/css/MainPage.css" rel="stylesheet">  *<!-- Begin page content -->*  <div class="container">  <div class="page-header">  <h1>Doorlock Status</h1>  </div>    <table id="myTable" class="tablesorter">  <thead>  <tr>  <th>Doorlock No.</th>  <th>Client Id</th>  <th>Protocol Ver.</th>  <th>Status</th>  </tr>  </thead>  <tbody class="centerText">  @Each.Model  <tr>  <td rowspan="3">@Current.DoorlockNo</td>  <td>@Current.ClientId</td>  <td>@Current.ProtocolVer</td>  <td>Active</td>  </tr>  <tr>  <td colspan="2" class="txtInput"><input type="text" id="inputSubscribeTopic@Current.ClientId" class="form-control" id="usr" placeholder="Topic"></td>  <td><button id="btnSubscribe" data-cid="@Current.ClientId" type="button" class="btn btn-success btnStyle">Subscribe</button></td>  </tr>  <tr>  <td class="txtInput"><input id="inputPublishTopic@Current.ClientId" type="text" class="form-control" id="usr" placeholder="Topic" value="/SmartDoorlock/DoorControl"></td>  <td class="txtInput"><input id="inputPublishMessage@Current.ClientId" type="text" class="form-control" id="usr" placeholder="Message" value="Open"></td>  <td><button id="btnPublish" data-cid="@Current.ClientId" type="button" class="btn btn-info btnStyle">Publish</button></td>  </tr>  @EndEach    </tbody>  </table>    <h1>MQTT Log</h1>  <textarea id="mqttLog" readonly class="form-control" rows="15"></textarea>  </div>    <script language="JavaScript" src="/Content/scripts/jquery.tablesorter.js"></script>  <script language="JavaScript" src="/Content/scripts/MainPage.js"></script>  @EndSection |
| **MainPage.css** |
| **@media** **(**min-width**:** 1200px**)** {  **.container**{  max-width**:** 1000px;  }  }      */\* tables \*/*  table**.tablesorter** {  font-family**:**inherit;  border**:** 1px solid #CDD2D2;  margin**:**10px 0pt 15px;  width**:** 100%;  text-align**:** center;  }  table**.tablesorter** thead tr th**,** table**.tablesorter** tfoot tr th {  background-color**:** #e6EEEE;  border**:** 1px solid #CDD2D2;  padding**:** 4px;  white-space**:**nowrap;  width**:**1%;  }  table**.tablesorter** thead tr **.header** {  background-image**:** url(tblColArrow.gif);  background-repeat**:** no-repeat;  background-position**:** center right;  cursor**:** pointer;  }  table**.tablesorter** tbody td {  color**:** #3D3D3D;  padding**:** 4px;  background-color**:** #FFF;  vertical-align**:** middle;  border**:** 1px solid #CDD2D2;  white-space**:**nowrap;  width**:**0;  }    table**.tablesorter** thead tr **.headerSortUp** {  background-image**:** url(tblColAsc.gif);  }  table**.tablesorter** thead tr **.headerSortDown** {  background-image**:** url(tblColDesc.gif);  }  table**.tablesorter** thead tr **.headerSortDown,** table**.tablesorter** thead tr **.headerSortUp** {  background-color**:** #8dbdd8;  }    table**.tablesorter** **.winrateheader** {  min-width**:** 180px;  }    **.txtInput** {  min-width**:** 180px;  }    **.btnStyle** {  width**:** 100px;  }    **.leftText** {  text-align**:** left **!important**;  } |

|  |
| --- |
| **MasterPage.sshtml** |
| **<!DOCTYPE html>**  <html xmlns**=**"http://www.w3.org/1999/xhtml">  <head>  <title>Smart Doorlock</title>  <link href**=**"/Content/css/MasterPage.css" rel**=**"stylesheet">  <link href**=**"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css" rel**=**"stylesheet">  <script src**=**"https://code.jquery.com/jquery-2.2.4.min.js" integrity**=**"sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44=" crossorigin**=**"anonymous"></script>  <script src**=**"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js"></script>  </head>  <body>  <nav class**=**"navbar navbar-default navbar-fixed-top">  <div class**=**"container">  <div class**=**"navbar-header">  <a class**=**"navbar-brand" href**=**"/">Smart Doorlock</a>  </div>  <div id**=**"navbar" class**=**"navbar-collapse collapse">  <ul class**=**"nav navbar-nav">  *<!--<li class="active"><a href="/">Home</a></li>-->*  <li><a href**=**"/">Home</a></li>  <li><a href**=**"/About">About</a></li>  <li><a href**=**"/Log">Log</a></li>  </ul>  </div>  </div>  </nav>    *<!-- Wrap all page content here -->*  <div id**=**"wrap">  @Section['Content']  </div>    <div id**=**"footer">  <div class**=**"container">  <p class**=**"text-muted">(C) 2016 Sokwhan Huh.</p>  </div>  </div>  </body>  </html> |

|  |
| --- |
| **MasterPage.css** |
| body {  margin-bottom**:** 60px;  }  body **>** **.container** {  padding**:** 60px 15px 0;  }  **.container** **.text-muted** {  margin**:** 20px 0;  }    */\* Sticky footer styles*  *-------------------------------------------------- \*/*    html**,**  body {  height**:** 100%;  overflow**:** hidden;  */\* The html and body elements cannot have any padding or margin. \*/*  }    */\* Wrapper for page content to push down footer \*/*  #wrap {  height**:** calc(100% **-** 120px);  overflow-y**:** scroll;  padding**:** 0px 10%;  margin-top**:** 60px;  }    */\* Set the fixed height of the footer here \*/*  #footer {  height**:** 60px;  background-color**:** #f5f5f5;  position**:** fixed;  bottom**:** 0;  width**:** 100%;  } |

|  |
| --- |
| **MainPage.js** |
| **function** appendToLog(txt) {  **var** log **=** $("#mqttLog");  log.val(log.val() **+** txt **+** "\n");  }    $(document).ready(**function** () {  $("#myTable").tablesorter();  $("#btnSubscribe").click(**function** () {  **var** btnElement **=** $(**this**);  **var** clientId **=** btnElement.attr("data-cid");  **var** txtElement **=** $("#inputSubscribe" **+** clientId);  **var** data **=** { "clientId"**:** clientId, "topic"**:** txtElement.val() }  $.ajax({  type**:** "POST",  url**:** "/Subscribe",  data**:** data,  success**:** **function** (data) { alert(data); },  dataType**:** "json"  });  });    $("#btnPublish").click(**function** () {  **var** btnElement **=** $(**this**);  **var** clientId **=** btnElement.attr("data-cid");  **var** txtTopic **=** $("#inputPublishTopic" **+** clientId);  **var** txtMessage **=** $("#inputPublishMessage" **+** clientId);  **var** data **=** { "clientId"**:** clientId, "topic"**:** txtTopic.val(), "message"**:** txtMessage.val() }  $.ajax({  type**:** "POST",  url**:** "/Publish",  data**:** data,  dataType**:** "json"  });    appendToLog("Published to topic: " **+** txtTopic.val() **+** ", message: " **+** txtMessage.val())  });  }); |

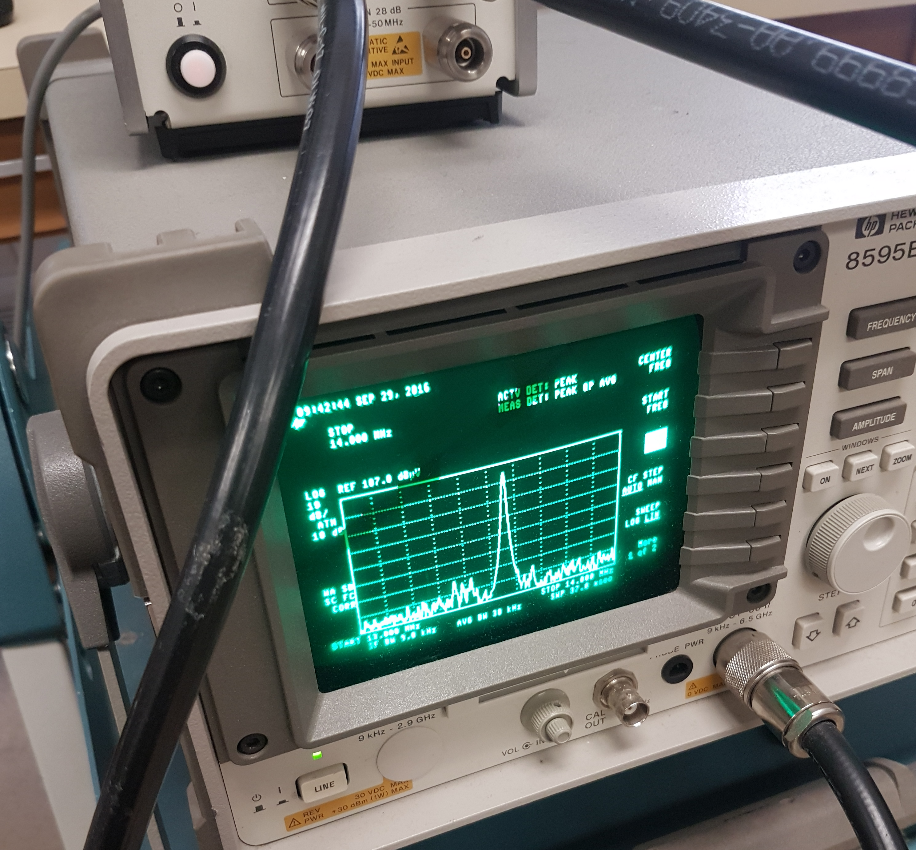
# September 28th, 2016

The solenoid motor that will be used for this project requires 12V @ 550mA upon activation. This is slightly problematic for two reasons. 1) Rest of the circuitry in this project uses 5V, 3.3V. 2) Without the solenoid, the expected average current draw is about 400~500mA. With the solenoid, this adds up to approximately 1A of maximum current draw. This means that I will have to find a good power supply that not only provides voltage for all 3.3V, 5V and 12V, but the power supply will also have to supply a fair amount of current (say, 2A). After inquiring Dr. Barrett about this problem, he recommended some very good power supplies with dual voltage outputs (5V, 12V). This only requires me to use a single 3.3V linear regulator to step the 5V down to 3.3V. Dr. Barrett also recommended me some very good components to go along with the power supply, such as AC receptacle, chrome bezel LED indicator, fuse holder etc. and a method to mount the power supply to my chassis – something I haven’t given too much thought about. There is one more concern I have regarding the project – that is, it has a requirement of operating under a backup battery. I still haven’t found a good solution for this, mostly due to the high voltage requirement and a large current consumption. Car batteries for 12V @ 2A do exist, but these are way too big for this project (After all, this is a doorlock. It does have a practical size limitation).

The following is a list of items I was planning to place on order: Lock-style Solenoid - 12VDC, 1N4001 Diode - 10 pack, TIP120 Power Darlington Transistors - 3 pack, Molex 2695 Series 10-Position Mating Connector, Molex 6373 Series 10-Position Male-Pin Connector, AC/DC CONVERTER 5V 12V 30W, Fuseholder Panel Mount D Type .25" X1.25" Fuse Solder Terminal, 8-Foot 3-Conductor SJT Detachable AC Power Cord, Toggle Switch Full Single Pole Single Throw (On-Off) 6.0 Inch Wire 125VAC @ 6 Amp 1/2 Inch Mounting, Receptacle Male 10A@250VoltFused Fast-On UL/CSA/Vde, Led Indicator Chrome Bezel 2.0Vf@20mA 0.24" Diameter. After inquiring George – our master technician about these components, it turned out that the shop already had majority of these components at hand so I ended up only placing order on solenoid, molex connectors and AC/DC power supply.

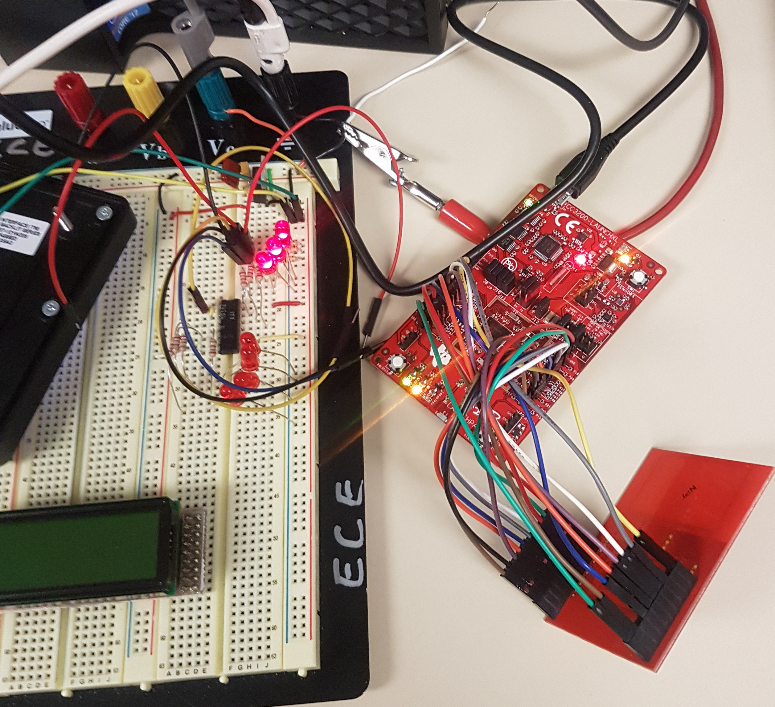
# September 30th, 2016

I’ve finally borrowed a spectrum analyzer from Dr. Kubichek to confirm whether my NFC reader (TRF7970A) actually works or not.

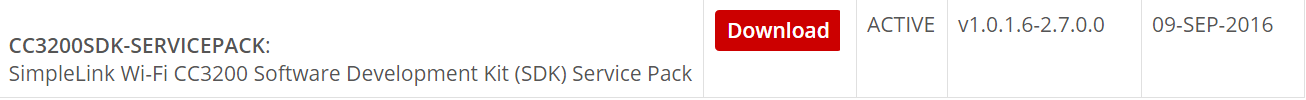
Much to my surprise, when I’ve powered on the spectrum analyzer and brought the antenna close to TRF7970A, I was getting a clear reading – indicating that the NFC reader is actually transmitting something.

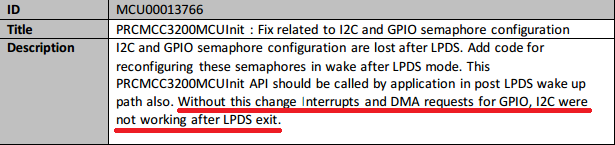
This shows that the NFC reader is likely not defective and the code, at least for generating RF field, is correct. This definitely narrowed down places to look on getting this reader to work.

I’ve decided to test every single pin of the NFC reader (at least for those pins that are possible) by hooking them up into GPIO pins.



I was able to determine that due to the hardware update of TRF7970A, the boosterpack can no longer be just inserted on top of the CC3200 launchpad – one of the pin is no longer compatible. But there was one more bizzare issue – CC3200 would not be handling interrupts coming from TRF7970A. Even manually toggling the interrupt by short circuiting between 3.3V and ground would not toggle the interrupt. After investigating this issue, it turned out that there was a critical issue with the latest CC3200 Launchpad (Rev 4.5) that would cause the microcontroller to not handle interrupts under a very specific circumstance. Using certain semaphore commands with TI-RTOS, causes all interrupt handling to halt on the CC3200. A hotfix was released for this on September 9th, 2016 with a service pack, of which the MCU image needed flashing in order to rectify this issue.





After flashing the MCU, the NFC reader started to work properly. This probably caused me to waste a good 3 months in trying to interface the NFC reader.

I’m genuinely shocked that TI would release an untested software (hardware?) on the market. Had TI not release this service pack soon enough, it definitely jeopardized my entire project, likely forcing me spend a great deal of time trying to figure out what’s going on, until I actually start looking for a different NFC reader. Part of this ordeal is from my own fault – I didn’t spend enough time investigating whether there has been a major hardware update, with completely different pin configurations. But to my knowledge, there’s literally no way a typical consumer can figure out in advance whether a required component would work or not.

Another work that was done today is with regards to the LCD. Because the LCD pins were too close to each other (0.7mm apart) for it to be connected on the breadboard, and pins were also too thin for jumper cables, I’ve requested for an adapter board to be made. Vic and George were very helpful (in fact, I couldn’t have done this without them) in creating this adapter board. Interfacing the LCD (with slight modification to the earlier LCD code) worked well.

|  |
| --- |
| **lcd.c** |
| */\**  *\* lcd.c*  *\**  *\* Created on: Sep 23, 2016*  *\* Author: shuh*  *\*/*  **#include** "lcd.h"    *// Driverlib includes*      **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_types.h"  **#include** "prcm.h"  **#include** "simplelink.h"    *// Common interface include*  **#include** "spi.h"    *// Project Includes*  **#include** "spi\_l.h"    **#define CMD\_START\_BIT 0xF8** *//5 1s for start bit*  **#define CMD\_RW 2**  **#define CMD\_RS 1**  **#define CMD\_DATA\_SHIFT\_BIT 4**    **#define LCD\_HOME\_L1 0x80**    **unsigned** **long** ulDummy;    **static** **unsigned** **char** **reverse**(**unsigned** **char** b) {  b **=** (b **&** 0xF0) **>>** 4 **|** (b **&** 0x0F) **<<** 4;  b **=** (b **&** 0xCC) **>>** 2 **|** (b **&** 0x33) **<<** 2;  b **=** (b **&** 0xAA) **>>** 1 **|** (b **&** 0x55) **<<** 1;  **return** b;  }    **void** **lcdCheckBusy**(**void**) {  **unsigned** **char** command **=** CMD\_START\_BIT;  **unsigned** **char** isBusy **=** 1;  command **|=** (1 **<<** CMD\_RW) **|** (0 **<<** CMD\_RS);    **do** {  SPI\_LCD\_CS\_ON;  MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  isBusy **=** (**char**)ulDummy;  SPI\_LCD\_CS\_OFF;  }  **while** (isBusy **&** 0x01 **>** 0);  }    **static** **void** **lcdSPIPutData**(**unsigned** **char** spiData) {  *//SPI Data Protocol (Weirdest crap I've ever seen in my life when it comes to SPI)*  *//First, 5 start bytes are transferred (Handled in lcdputcommand).*  *//Then, a byte of data is sent into two chunks. To be specific,*  *//(4 Lower Bit + 0000) then (4 Upper Bit + 0000), LSB first*  *//http://www.lcd-module.de/fileadmin/eng/pdf/zubehoer/ssd1803\_2.0.pdf*  *//The level of confusion is LEGENDARY*  **unsigned** **char** reversedData **=** reverse(spiData);  *//Get lower 4 bits (Since bit was reversed, we actually do and operation on upper 4 bit)*  **unsigned** **char** lowerData **=** reversedData **&** 0xF0;  MAP\_SPIDataPut(GSPI\_BASE,lowerData);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  *//Get upper 4 bits*  **unsigned** **char** upperData **=** (reversedData **&** 0x0F) **<<** CMD\_DATA\_SHIFT\_BIT;  MAP\_SPIDataPut(GSPI\_BASE,upperData);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  }    **static** **void** **lcdPutCommand**(lcdCommandEnum cmdType) {  **unsigned** **char** command **=** CMD\_START\_BIT;  command **|=** (0 **<<** CMD\_RW) **|** (0 **<<** CMD\_RS);  **switch** (cmdType) {  **case** CLEAR\_SCREEN:  SPI\_LCD\_CS\_ON;  MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  lcdSPIPutData(0x01);  SPI\_LCD\_CS\_OFF;  **break**;  **case** DISPLAY\_ON:  SPI\_LCD\_CS\_ON;  MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  *//Display on command*  lcdSPIPutData(0x0F);  *//lcdSPIPutData(0x0C);*  SPI\_LCD\_CS\_OFF;  **break**;  **case** LCD\_INIT:  SPI\_LCD\_CS\_ON;  MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  *//8-Bit, RE=1*  lcdSPIPutData(0x34);  SPI\_LCD\_CS\_OFF;  osi\_Sleep(2);    SPI\_LCD\_CS\_ON;  MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  *//4 Line Mode*  lcdSPIPutData(0x09);  SPI\_LCD\_CS\_OFF;  osi\_Sleep(2);      SPI\_LCD\_CS\_ON;  MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  *//8-Bit, RE=0*  lcdSPIPutData(0x30);  SPI\_LCD\_CS\_OFF;  **break**;  **case** RETURN\_HOME:  SPI\_LCD\_CS\_ON;  MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  *//Return home*  lcdSPIPutData(0x06);  SPI\_LCD\_CS\_OFF;  **break**;  }  }    **void** **lcdSetPosition**(**unsigned** **char** position) {  lcdCheckBusy();  SPI\_LCD\_CS\_ON;  **unsigned** **char** command **=** CMD\_START\_BIT;    MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  lcdSPIPutData(LCD\_HOME\_L1 **+** position);  SPI\_LCD\_CS\_OFF;  }    **void** **lcdClearScreen**(**void**) {  lcdPutCommand(CLEAR\_SCREEN);  lcdPutCommand(RETURN\_HOME);  osi\_Sleep(2);  }    **void** **lcdInit**(**void**) {  lcdPutCommand(LCD\_INIT);  osi\_Sleep(2);  }    **void** **lcdReset**(**void**) {  *//Reset LCD*  GPIO\_IF\_Set(13,0);  osi\_Sleep(50);  GPIO\_IF\_Set(13,1);  }    **void** **lcdDisplayOn**(**void**) {  lcdPutCommand(DISPLAY\_ON);  osi\_Sleep(2);  }    **void** **lcdPutChar**(**unsigned** **char** lcdChar) {  *//RW = 0*  *//RS = 1*  SPI\_LCD\_CS\_ON;  **unsigned** **char** command **=** CMD\_START\_BIT;  command **|=** (0 **<<** CMD\_RW) **|** (1 **<<** CMD\_RS);  MAP\_SPIDataPut(GSPI\_BASE,command);  MAP\_SPIDataGet(GSPI\_BASE,**&**ulDummy);  lcdSPIPutData(lcdChar);  SPI\_LCD\_CS\_OFF;  }    **void** **lcdPutString**(**unsigned** **char\*** str) {  **do**  {  lcdCheckBusy();  lcdPutChar(**\***str**++**);  }  **while**(**\***str);  } |

|  |
| --- |
| **lcd.h** |
| */\**  *\* lcd.h*  *\**  *\* Created on: Sep 23, 2016*  *\* Author: shuh*  *\*/*    **#ifndef LCD\_H\_**  **#define LCD\_H\_**    **#define LCD\_LINE1 0**  **#define LCD\_LINE2 LCD\_LINE1+0x20**  **#define LCD\_LINE3 LCD\_LINE1+0x40**  **#define LCD\_LINE4 LCD\_LINE1+0x60**    **typedef** **enum**  {  LCD\_INIT,  DISPLAY\_ON,  CLEAR\_SCREEN,  RETURN\_HOME  } lcdCommandEnum;    **extern** **void** **lcdInit**();  **extern** **void** **lcdReset**();  **extern** **void** **lcdDisplayOn**();  **extern** **void** **lcdClearScreen**(**void**);  **extern** **void** **lcdPutString**(**unsigned** **char\*** str);  **extern** **void** **lcdPutChar**(**unsigned** **char** lcdChar);  **extern** **void** **lcdSetPosition**(**unsigned** **char** position);  **#endif** */\* LCD\_H\_ \*/* |

|  |
| --- |
| **spi\_l.c** |
| */\**  *\* spi.c*  *\**  *\* Created on: Sep 30, 2016*  *\* Author: shuh*  *\*/*    *//Driver Lib Includes*  **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_types.h"  **#include** "prcm.h"    *// Project Includes*  **#include** "spi\_l.h"    **void** **SPIInit**(**void**) {  MAP\_PRCMPeripheralReset(PRCM\_GSPI);    *// Reset SPI*  MAP\_SPIReset(GSPI\_BASE);    *// Configure SPI interface*  *//SPI\_SUB\_MODE\_1 = [POL = 0, PHA = 1]*  MAP\_SPIConfigSetExpClk(GSPI\_BASE,MAP\_PRCMPeripheralClockGet(PRCM\_GSPI),  SPI\_IF\_BIT\_RATE,SPI\_MODE\_MASTER,SPI\_SUB\_MODE\_1,  (SPI\_SW\_CTRL\_CS **|**  SPI\_4PIN\_MODE **|**  SPI\_TURBO\_OFF **|**  SPI\_CS\_ACTIVELOW **|**  SPI\_WL\_8));    *// Enable SPI for communication*  MAP\_SPIEnable(GSPI\_BASE);    } |

|  |
| --- |
| **spi\_l.h** |
| */\**  *\* spi.h*  *\**  *\* Created on: Sep 30, 2016*  *\* Author: shuh*  *\*/*    **#ifndef SPI\_L\_H\_**  **#define SPI\_L\_H\_**    *//Common Interface Includes*  **#include** "spi.h"  **#include** "gpio\_if.h"    *//#define SPI\_IF\_BIT\_RATE 2000000*  **#define SPI\_IF\_BIT\_RATE 2000000**    **#define SPI\_TRF\_CS\_ON MAP\_SPICSEnable(GSPI\_BASE);**  **#define SPI\_TRF\_CS\_OFF MAP\_SPICSDisable(GSPI\_BASE);**  **#define SPI\_LCD\_CS\_ON GPIO\_IF\_Set(10,0);**  **#define SPI\_LCD\_CS\_OFF GPIO\_IF\_Set(10,1);**    **extern** **void** **SPIInit**(**void**);    **#endif** */\* SPI\_L\_H\_ \*/* |

|  |
| --- |
| **NDEF.c** |
| */\* {NDEF.c}*  *\**  *\* {NDEF specific functions file}*  *\***\**  *\*/*  **#include** "hw\_types.h"  **#include** "NDEF.h"  **#include** "timer.h"  **#include** "timer\_if.h"  **#include** "trf7970.h"  **#include** "trf7970BoosterPack.h"  **#include** "hw\_types.h"  **#include** "uart\_if.h"    **#include** "utils.h"  **#include** "gpio.h"  **#include** "gpio\_if.h"  **#include** "hw\_memmap.h"  **#include** "uart\_if.h"  **#include** "utils.h"  *//===============================================================*    **extern** u08\_t buf[140];  **extern** u08\_t i\_reg;  **extern** u08\_t irq\_flag;  **extern** u08\_t stand\_alone\_flag;  **extern** s08\_t rxtx\_state;  **extern** u08\_t rx\_error\_flag;    *//===============================================================*    **unsigned** **char** **NDEFApplicationSelect**(**void**)  {  u08\_t NDEF\_Support **=** 0;    rx\_error\_flag **=** 0x00;    buf[0] **=** 0x8f;  buf[1] **=** 0x91;  buf[2] **=** 0x3d;  buf[3] **=** 0x00;  buf[4] **=** 0xE0; *// length*  buf[5] **=** 0x02;  buf[6] **=** 0x00;  buf[7] **=** 0xA4;  buf[8] **=** 0x04;  buf[9] **=** 0x00;  buf[10] **=** 0x07;  buf[11] **=** 0xD2;  buf[12] **=** 0x76;  buf[13] **=** 0x00;  buf[14] **=** 0x00;  buf[15] **=** 0x85;  buf[16] **=** 0x01;  buf[17] **=** 0x01;  buf[18] **=** 0x00;      Trf7970RawWrite(**&**buf[0], 19); *//writing to FIFO*    *//Trf7970ResetIrqStatus();*  IRQ\_ON;    i\_reg **=** 0x01;  rxtx\_state **=** 0; */\* the response will be stored in buf[0] upwards \*/*      *// COUNT\_VALUE = COUNT\_1ms \* 20; /\* 10ms for TIMEOUT \*/*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); *// Wait for end of TX*    i\_reg **=** 0x01;    RESET\_COUNTER;      *// COUNT\_VALUE = COUNT\_1ms \* 20; /\* 10ms for TIMEOUT \*/*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); */\* wait for RX complete \*/*    **if**( (buf[1] **==** 0x90) **&&** (buf[2] **==** 0x00)){  NDEF\_Support **=** 1;  }    RESET\_COUNTER;    Trf7970ResetIrqStatus();    McuDelayMillisecond(1);    **return** NDEF\_Support;  }    **void** **CapabilityContainerSelect**(**void**)  {  rx\_error\_flag **=** 0x00;    buf[0] **=** 0x8f;  buf[1] **=** 0x91;  buf[2] **=** 0x3d;  buf[3] **=** 0x00;  buf[4] **=** 0x80; *// length*  buf[5] **=** 0x03;  buf[6] **=** 0x00;  buf[7] **=** 0xA4;  buf[8] **=** 0x00;  buf[9] **=** 0x0C;  buf[10] **=** 0x02;  buf[11] **=** 0xE1;  buf[12] **=** 0x03;    Trf7970RawWrite(**&**buf[0], 13);    IRQ\_CLR; *// PORT2 interrupt flag clear (inside MSP430)*  IRQ\_ON;    i\_reg **=** 0x01;  rxtx\_state **=** 0; */\* the response will be stored in buf[0] upwards \*/*      *//COUNT\_VALUE = COUNT\_1ms \* 20; /\* 10ms for TIMEOUT \*/*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); *// Wait for end of TX*    i\_reg **=** 0x01;    RESET\_COUNTER;      A2CounterLoad(COUNT\_1ms **\*** 20);  *// COUNT\_VALUE = COUNT\_1ms \* 20; /\* 10ms for TIMEOUT \*/*  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); */\* wait for RX complete \*/*    Trf7970ResetIrqStatus();    *//stopCounter;*    McuDelayMillisecond(1);  }    **unsigned** **char** **ReadBinary**(**unsigned** **char** Offset, **unsigned** **char** Read\_Length)  {  */\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/*  **unsigned** **char** Nlen**=**0;  *// extern unsigned char Tag\_found;*  */\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/*    rx\_error\_flag **=** 0x00;    buf[0] **=** 0x8f;  buf[1] **=** 0x91;  buf[2] **=** 0x3d;  buf[3] **=** 0x00;  buf[4] **=** 0x60; *// length*  buf[5] **=** 0x02;  buf[6] **=** 0x00;  buf[7] **=** 0xB0;  buf[8] **=** 0x00;  buf[9] **=** Offset; *// offset*  buf[10] **=** Read\_Length; *// Read Length*    Trf7970RawWrite(**&**buf[0], 11);    Trf7970ResetIrqStatus();  IRQ\_ON;    i\_reg **=** 0x01;  rxtx\_state **=** 0; */\* the response will be stored in buf[0] upwards \*/*      *//COUNT\_VALUE = COUNT\_1ms \* 20;*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); *// Wait for end of TX*      i\_reg **=** 0x01;    RESET\_COUNTER;      A2CounterLoad(COUNT\_1ms **\*** 20);  *// COUNT\_VALUE = COUNT\_1ms \* 20; /\* 10ms for TIMEOUT \*/*  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); */\* wait for RX complete \*/*    RESET\_COUNTER;    *//Trf7970ResetIrqStatus();*    *// might need test case here, like if(i\_reg == 0xFF)*  Nlen **=** buf[2];  McuDelayMillisecond(1);    **return** Nlen;  }    **unsigned** **char** **ReadBinary2**(**unsigned** **char** Offset, **unsigned** **char** Read\_Length)  {  */\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/*  **unsigned** **char** Nlen**=**0;  *// extern unsigned char Tag\_found;*  */\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/*    rx\_error\_flag **=** 0x00;    buf[0] **=** 0x8f;  buf[1] **=** 0x91;  buf[2] **=** 0x3d;  buf[3] **=** 0x00;  buf[4] **=** 0x60; *// length*  buf[5] **=** 0x03;  buf[6] **=** 0x00;  buf[7] **=** 0xB0;  buf[8] **=** 0x00;  buf[9] **=** Offset; *// offset*  buf[10] **=** Read\_Length; *// Read Length*    Trf7970RawWrite(**&**buf[0], 11);    Trf7970ResetIrqStatus();  IRQ\_ON;    i\_reg **=** 0x01;  rxtx\_state **=** 0; */\* the response will be stored in buf[0] upwards \*/*      *//COUNT\_VALUE = COUNT\_1ms \* 20;*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); *// Wait for end of TX*      i\_reg **=** 0x01;    RESET\_COUNTER;      *// COUNT\_VALUE = COUNT\_1ms \* 20; /\* 10ms for TIMEOUT \*/*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); */\* wait for RX complete \*/*    RESET\_COUNTER;    *//Trf7970ResetIrqStatus();*    *// might need test case here, like if(i\_reg == 0xFF)*  Nlen **=** buf[2];  McuDelayMillisecond(1);    **return** Nlen;  }      **void** **SelectNDEF**(**void**)  {  rx\_error\_flag **=** 0x00;    buf[0] **=** 0x8f;  buf[1] **=** 0x91;  buf[2] **=** 0x3d;  buf[3] **=** 0x00;  buf[4] **=** 0x80; *// length*  buf[5] **=** 0x03;  buf[6] **=** 0x00;  buf[7] **=** 0xA4;  buf[8] **=** 0x00;  buf[9] **=** 0x0C;  buf[10] **=** 0x02;  buf[11] **=** 0xE1;  buf[12] **=** 0x04;    Trf7970RawWrite(**&**buf[0], 13);    Trf7970ResetIrqStatus();  IRQ\_ON;    i\_reg **=** 0x01;  rxtx\_state **=** 0; */\* the response will be stored in buf[0] upwards \*/*      *// COUNT\_VALUE = COUNT\_1ms \* 20;*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); *// Wait for end of TX*    i\_reg **=** 0x01;    RESET\_COUNTER;      *// COUNT\_VALUE = COUNT\_1ms \* 20; /\* 10ms for TIMEOUT \*/*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); */\* wait for RX complete \*/*    Trf7970ResetIrqStatus();    RESET\_COUNTER;    McuDelayMillisecond(1);  }    **void** **RATS**(**void**)  {  u08\_t write[4];  rx\_error\_flag **=** 0x00;    write[0] **=** ISO\_CONTROL;  write[1] **=** 0x08;  Trf7970WriteSingle(write, 2);    buf[0] **=** 0x8f;  buf[1] **=** 0x91;  buf[2] **=** 0x3d;  buf[3] **=** 0x00;  buf[4] **=** 0x20; *// length*  buf[5] **=** 0xE0;  buf[6] **=** 0x80;    Trf7970RawWrite(**&**buf[0], 7);    Trf7970ResetIrqStatus();  IRQ\_ON;    i\_reg **=** 0x01;  rxtx\_state **=** 0; */\* the response will be stored in buf[0] upwards \*/*      *// COUNT\_VALUE = COUNT\_1ms \* 20;*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); *// Wait for end of TX*    i\_reg **=** 0x01;    RESET\_COUNTER;      *// COUNT\_VALUE = COUNT\_1ms \* 20; /\* 10ms for TIMEOUT \*/*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; */\* start timer up mode \*/*    **while**(i\_reg **==** 0x01); */\* wait for RX complete \*/*    Trf7970ResetIrqStatus();    RESET\_COUNTER;    McuDelayMillisecond(1);  } |

|  |
| --- |
| **NDEF.h** |
| */\**  *\* {NDEF.h}*  *\**  *\* {NDEF header file}*  *\**  *\*/*    **#ifndef \_NDEF\_H\_**  **#define \_NDEF\_H\_**    *//===============================================================*    *//#include <MSP430x23x0.h> // processor specific header*  *//#include <stdio.h> // standard input/output header*    **#include** "trf7970.h"  **#include** "types.h"      *//===============================================================*  *//void Attrib(void);*  **void** **RATS**(**void**);  **unsigned** **char** **NDEFApplicationSelect**(**void**);  **void** **CapabilityContainerSelect**(**void**);  **unsigned** **char** **ReadBinary**(**unsigned** **char** Offset, **unsigned** **char** Read\_Length);  **unsigned** **char** **ReadBinary2**(**unsigned** **char** Offset, **unsigned** **char** Read\_Length);  **void** **SelectNDEF**(**void**);    *//===============================================================*    **#endif** |

|  |
| --- |
| **iso15693.c** |
| */\**  *\* {iso15693.c}*  *\**  *\* {ISO15693 Specific Functions & Anti-collision}*  *\**  *\*/*    **#include** "hw\_types.h"  **#include** <string.h>  **#include** <stdio.h>  **#include** "iso15693.h"  **#include** "uart\_if.h"  **#include** "timer.h"  **#include** "timer\_if.h"  **#include** "trf7970BoosterPack.h"  **#include** "utils.h"  **#include** "gpio.h"  **#include** "gpio\_if.h"  **#include** "hw\_memmap.h"  **#include** "uart\_if.h"      **#define UART\_PRINT Report**    *//===============================================================*    u08\_t afi **=** 0;  **extern** u08\_t flags **=** 0; *// request flags used in command, based on ISO15693-3, tables 3, 4 and 5*  **extern** u08\_t blocknum **=** 0;*// for making block number dynamic (somewhat at this time :))*  **extern** u08\_t buf[300];  **extern** u08\_t i\_reg;  **extern** u08\_t irq\_flag;  **extern** s08\_t rxtx\_state;  **extern** u08\_t rx\_error\_flag;  **extern** u08\_t stand\_alone\_flag;  **extern** u08\_t remote\_flag;  **extern** u08\_t Tag\_Count;  **extern** u08\_t g\_uid[300];  **extern** **char** g\_block\_content[200];  **extern** **char** g\_tag\_content[600];  **extern** u08\_t g\_rssi[10];  **extern** u08\_t g\_tag\_found;        *//===============================================================*  *// NAME: void Iso15693FindTag(void)*  *//*  *// BRIEF: Is used to detect ISO15693 conform tags in stand alone*  *// mode.*  *//*  *// INPUTS:*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] turn on RF driver*  *// [2] do a complete anti-collision sequence*  *// [3] turn off RF driver*  *//*  *// NOTE: If ISO15693 conform Tag is detected, ISO15693 LED will*  *// be turned on.*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 23Nov2010 RP Original Code*  *//===============================================================*    **void** **Iso15693FindTag**(**void**) {  Trf7970TurnRfOn();    Trf7970WriteIsoControl(0x02);    *// The VCD should wait at least 1 ms after it activated the*  *// powering field before sending the first request, to*  *// ensure that the VICCs are ready to receive it. (ISO15693-3)*  McuDelayMillisecond(6);    flags **=** SIXTEEN\_SLOTS;  *//flags = ONE\_SLOT;*    buf[20] **=** 0x00;  Iso15693Anticollision(**&**buf[20], 0x00); *// send Inventory request*    Trf7970TurnRfOff();    *// clear any IRQs*  Trf7970ResetIrqStatus();  }    *//===============================================================*  *// NAME: void NFC\_TYPEV\_READ\_SINGLE\_BLOCK(void)*  *//*  *// BRIEF: Is used to issue optional ISO15693 command: Read Single Block*  *// Function: Reads a single block of ISO15693 tag memory and loops, incrementing the block number each time.*  *//*  *// INPUTS: request flags, block # (in this case default request flags are 0x02, block # starts a 0x00)*  *//*  *// OUTPUTS: Block Data, from 0x00 to 0x3F (256 bytes of data) or to the point where error flags are returned (for tags smaller than 256 bytes, i.e. HF-I Pro or Standard)*  *//*  *// PROCESS: [1] turn on RF driver*  *// [2] do a complete read single block sequence (Command Code 0x20)*  *// [3] turn off RF driver*  *// [4] loop until block 0x3F is read or error flags are returned in response (see note in INPUTS section)*  *////*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 01/17/2014 JDW / JC Original Code*  *//===============================================================*    **void** **NFC\_TYPEV\_READ\_SINGLE\_BLOCK**(u08\_t ui8StartBlock) {    u16\_t k;    Trf7970TurnRfOn();    Trf7970WriteIsoControl(0x02);    *// The VCD should wait at least 2mSec after activating the*  *// magnetic field before sending the first request, to*  *// ensure that the VICC is ready to receive it. (ISO15693-3)*    rxtx\_state **=** 1; *//resetting buffer pointer*  McuDelayMillisecond(6); *//plenty of time :)*    flags **=** SSC\_HTDR\_NA; *//non-addressed request flags*    *// flags = SSC\_HTDR\_ADDR; //addressed (requires getting and storing UID first)*    buf[0] **=** 0x8F; *//reset FIFO*  buf[1] **=** 0x91; *// sending with CRC, means TRF79xx will append CRC to string going out over the air*  buf[2] **=** 0x3D; *// write continuous from 1D*  buf[3] **=** 0x00; *//upper and middle nibbles of transmit byte length*  buf[4] **=** 0x30; *//lower and broken nibbles of transmit byte length*  buf[5] **=** flags; *// ISO15693 flags*  buf[6] **=** 0x20; *// Read Single Block command code*  buf[7] **=** ui8StartBlock; *// Block # (variable, for HF-I Plus device can go to 0x3F, Pro and Standard handled with "error" response flags)*    Trf7970ResetIrqStatus(); *//clearing IRQ (just in case)*    A2CounterLoad(COUNT\_1ms **\*** 30); *// TimerA set 30ms, not 20ms*  IRQ\_CLR; *// PORT2 interrupt flag clear (inside MSP430)*  IRQ\_ON;          Trf7970RawWrite(**&**buf[0], 8); *//issuing the Read Single Block command*    i\_reg **=** 0x01;  irq\_flag **=** 0x00;    START\_COUNTER; *// Starting Timeout*    **while** (irq\_flag **==** 0x00) {  } *// wait for end of TX interrupt*  RESET\_COUNTER;  A2CounterLoad(COUNT\_1ms **\*** 20); *// TimerA set*  START\_COUNTER; *// start timer up mode*    irq\_flag **=** 0x00;    **while** (irq\_flag **==** 0x00) {  } *// wait for interrupt*  RESET\_COUNTER;  **while** (i\_reg **==** 0x01) *// wait for RX complete*  {  k**++**;    **if** (k **==** 0xFFF0) {  i\_reg **=** 0x00;  rx\_error\_flag **=** 0x00;  }  }    **if** (i\_reg **==** 0xFF) { *// if received block data in buffer*  **if** ((stand\_alone\_flag **==** 1) **&&** (buf[1] **==** 0x00)) *// Confirming "no error" in response flags byte // Confirming "no error" in response flags byte*  {    *//Sergey, printing the content of vlovk into global block\_content variable*  **int** cx;    cx **=** snprintf(g\_block\_content, 300, "%s", "NFC-V Block ");  cx **=** cx **+** snprintf(g\_block\_content**+**cx, **sizeof**(g\_block\_content)**-**cx, "%x", (ui8StartBlock **>>** 4) **&** 0x0F );  cx **=** cx **+** snprintf(g\_block\_content**+**cx, **sizeof**(g\_block\_content)**-**cx, "%x", ui8StartBlock **&** 0x0F );  cx **=** cx **+** snprintf(g\_block\_content**+**cx, **sizeof**(g\_block\_content)**-**cx, "%s", " Data: [" );  **int** i;  **for** (i **=** 2; i **<** 6; i**++**) {  cx **=** cx **+** snprintf(g\_block\_content**+**cx, **sizeof**(g\_block\_content)**-**cx, "%x", (buf[i] **>>** 4) **&** 0x0F );  cx **=** cx **+** snprintf(g\_block\_content**+**cx, **sizeof**(g\_block\_content)**-**cx, "%x", buf[i] **&** 0x0F );  }  snprintf(g\_block\_content**+**cx, **sizeof**(g\_block\_content)**-**cx, "%s", "]\n" );    rxtx\_state **=** 1; *//resetting buffer pointer*    }  }    Trf7970TurnRfOff();    *// clear any IRQs*  Trf7970ResetIrqStatus();  }    *//===============================================================*  *// NAME: void Iso15693ReadMultipleBlocks(void)*  *//*  *// BRIEF: Is used to issue optional ISO15693 command: Read Multiple Blocks*  *// Function: Reads multiple blocks of ISO15693 tag memory.*  *//*  *// INPUTS: request flags, block # (in this case default request flags are 0x02, starting block # is 0x00, reading 8 blocks of data)*  *//*  *// OUTPUTS: Block Data*  *//*  *// PROCESS: [1] turn on RF driver*  *// [2] do a complete read multiple blocks sequence (Command Code 0x23)*  *// [3] turn off RF driver*  *//*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 01/17/2014 JDW Original Code*  *//===============================================================*  **void** **Iso15693ReadMultipleBlocks\_0\_18**(**void**) *//only use for 2k bit tag*  {    u16\_t k;    Trf7970TurnRfOn();    Trf7970WriteIsoControl(0x02);    *// The VCD should wait at least 2mSec after activating the*  *// magnetic field before sending the first request, to*  *// ensure that the VICC is ready to receive it. (ISO15693-3)*  McuDelayMillisecond(6); *//plenty of time :)*    flags **=** SSC\_HTDR\_NA; *//non-addressed*  *// flags = SSC\_HTDR\_ADDR; //addressed (requires getting and storing UID first)*    buf[0] **=** 0x8F; *//reset FIFO*  buf[1] **=** 0x91;*// sending with CRC, means TRF79xx will append CRC to string going out over the air*  buf[2] **=** 0x3D; *// write continuous from 1D*  buf[3] **=** 0x00; *// upper and middle nibbles of transmit byte length*  buf[4] **=** 0x40; *// lower and broken nibbles of transmit byte length*  buf[5] **=** flags; *// ISO15693 flags*  buf[6] **=** 0x23; *// Read Multiple Blocks command code*  buf[7] **=** 0x00; *// starting block # (hardcoded for now)*  buf[8] **=** 0x18;*// # of blocks (0x18 = 25 blocks, hardcoded for now, see comment on 26th block )*    Trf7970ResetIrqStatus(); *//clearing IRQ (just in case)*  A2CounterLoad(COUNT\_1ms **\*** 30); *// TimerA set 30ms, not 20ms*    IRQ\_CLR; *// PORT2 interrupt flag clear (inside MSP430)*  IRQ\_ON;    Trf7970RawWrite(**&**buf[0], 9); *//issuing the Read Multiple Blocks command*    i\_reg **=** 0x01;  irq\_flag **=** 0x00;  START\_COUNTER; *//Starting Timeout*    **while** (irq\_flag **==** 0x00) {  } *// wait for end of TX interrupt*  RESET\_COUNTER;  *// COUNT\_VALUE = COUNT\_1ms \* 250;*  A2CounterLoad(COUNT\_1ms **\*** 250); *// TimerA set*    START\_COUNTER; *// start timer up mode*    irq\_flag **=** 0x00;    **while** (irq\_flag **==** 0x00) {  } *// wait for interrupt*  RESET\_COUNTER;    **while** (i\_reg **==** 0x01) *// wait for RX complete*  {  k**++**;    **if** (k **==** 0xFFF0) {  i\_reg **=** 0x00;  rx\_error\_flag **=** 0x00;  }  }    **if** (i\_reg **==** 0xFF) { *// if received block data in buffer*  **if** (stand\_alone\_flag **==** 1) {  *//found = 1;*  **#ifdef ENABLE\_HOST**  */\**  *//Tag\_Count++;*  *UartPutChar('[');*  *UartSendCString("Now reading 9 data blocks, data is rotated");*  *UartPutChar(']');*  *UartPutCrlf();*  *UartSendCString("BLOCK 00: ");*  *//for(i = 5; i > 1; i--) //is 4 bytes of block data as it comes in*  *for (i = 2; i < 6; i++) //rotates the 4 bytes of block data*  *{*  *UartPutByte(buf[i]); // send block 0 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 01: ");*  *for (i = 6; i < 10; i++) {*  *UartPutByte(buf[i]); // send block 1 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 02: ");*  *for (i = 10; i < 14; i++) {*  *UartPutByte(buf[i]); // send block 2 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 03: ");*  *for (i = 14; i < 18; i++) {*  *UartPutByte(buf[i]); // send block 3 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 04: ");*  *for (i = 18; i < 22; i++) {*  *UartPutByte(buf[i]); // send block 4 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 05: ");*  *for (i = 22; i < 26; i++) {*  *UartPutByte(buf[i]); // send block 5 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 06: ");*  *for (i = 26; i < 30; i++) {*  *UartPutByte(buf[i]); // send block 6 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 07: ");*  *for (i = 30; i < 34; i++) {*  *UartPutByte(buf[i]); // send block 7 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 08: ");*  *for (i = 34; i < 38; i++) {*  *UartPutByte(buf[i]); // send block 8 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 09: ");*  *for (i = 38; i < 42; i++) {*  *UartPutByte(buf[i]); // send block 9 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 0A: ");*  *for (i = 42; i < 46; i++) {*  *UartPutByte(buf[i]); // send block 10 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 0B: ");*  *for (i = 46; i < 50; i++) {*  *UartPutByte(buf[i]); // send block 11 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 0C: ");*  *for (i = 50; i < 54; i++) {*  *UartPutByte(buf[i]); // send block 12 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 0D: ");*  *for (i = 54; i < 58; i++) {*  *UartPutByte(buf[i]); // send block 13 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 0E: ");*  *for (i = 58; i < 62; i++) {*  *UartPutByte(buf[i]); // send block 14 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 0F: ");*  *for (i = 62; i < 66; i++) {*  *UartPutByte(buf[i]); // send block 15 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 10: ");*  *for (i = 66; i < 70; i++) {*  *UartPutByte(buf[i]); // send block 16 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 11: ");*  *for (i = 70; i < 74; i++) {*  *UartPutByte(buf[i]); // send block 17 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 12: ");*  *for (i = 74; i < 78; i++) {*  *UartPutByte(buf[i]); // send block 18 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 13: ");*  *for (i = 78; i < 82; i++) {*  *UartPutByte(buf[i]); // send block 19 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 14: ");*  *for (i = 82; i < 86; i++) {*  *UartPutByte(buf[i]); // send block 20 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 15: ");*  *for (i = 86; i < 90; i++) {*  *UartPutByte(buf[i]); // send block 21 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 16: ");*  *for (i = 90; i < 94; i++) {*  *UartPutByte(buf[i]); // send block 22 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 17: ");*  *for (i = 94; i < 98; i++) {*  *UartPutByte(buf[i]); // send block 23 data to host*  *}*  *UartPutCrlf();*  *UartSendCString("BLOCK 18: ");*  *for (i = 98; i < 102; i++) {*  *UartPutByte(buf[i]); // send block 24 data to host*  *}*    *UartPutCrlf();*  *UartPutCrlf();*    *McuDelayMillisecond(250);*  *\*/*    **#endif**  }  }    Trf7970TurnRfOff();    Trf7970ResetIrqStatus(); *// clear any IRQs*  }    *//===============================================================*  *// NAME: void Iso15693Anticollision(u08\_t \*mask, u08\_t length)*  *//*  *// BRIEF: Is used to perform a inventory cycle of 1 or 16 time slots.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*mask mask value*  *// u08\_t length number of significant bits of*  *// mask value*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] send command*  *// [2] receive respond*  *// [3] send respond to host*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 23Nov2010 RP Original Code*  *//===============================================================*    **void** **Iso15693Anticollision**(u08\_t **\***mask, u08\_t length) *// host command 0x14*  {  u08\_t i **=** 1, j **=** 1, command[2], no\_slots, found **=** 0;    u08\_t **\***p\_slot\_no;  u08\_t slot\_no[17];  u08\_t new\_mask[8], new\_length, mask\_size;  u32\_t size;  u08\_t ui8BlockNumber;  u08\_t fifo\_length **=** 0;    u16\_t k;    slot\_no[0] **=** 0x00;    buf[0] **=** ISO\_CONTROL;  buf[1] **=** 0x02;*// Receive with no CRC, means the TRF79xx will check CRC and strip out of FIFO*  Trf7970WriteIsoControl(buf[1]);  Trf7970ReadSingle(buf, 1);  McuDelayMillisecond(2);      **if** ((flags **&** BIT5)**==** 0x00) *// flag bit5 is the number of slots indicator*  {  no\_slots **=** 16; *// 16 slots if bit is cleared*  } **else** {  no\_slots **=** 1; *// 1 slot if bit is set*  }    p\_slot\_no **=** **&**slot\_no[0]; *// slot number pointer*    mask\_size **=** (((length **>>** 2) **+** 1) **>>** 1); *// mask\_size is 1 for length = 4 or 8*    buf[0] **=** 0x8F;  buf[1] **=** 0x91; *// send with CRC*  buf[2] **=** 0x3D; *// write continuous from 1D*  buf[5] **=** flags; *// ISO15693 flags*  buf[6] **=** 0x01; *// anti-collision command code*    *//optional AFI should be here*  **if** (flags **&** 0x10) {  *// mask\_size is 2 for length = 12 or 16 ;*  *// and so on*    size **=** mask\_size **+** 4;*// mask value + mask length + AFI + command code + flags*    buf[7] **=** afi;  buf[8] **=** length; *// mask length*  **if** (length **>** 0) {  **for** (i **=** 0; i **<** mask\_size; i**++**) {  buf[9 **+** i] **=** **\***(mask **+** i);  }  }  fifo\_length **=** 9;  } **else** {  *// mask\_size is 2 for length = 12 or 16*  *// and so on*    size **=** mask\_size **+** 3;*// mask value + mask length + command code + flags*    buf[7] **=** length; *// mask length*  **if** (length **>** 0) {  **for** (i **=** 0; i **<** mask\_size; i**++**) {  buf[8 **+** i] **=** **\***(mask **+** i);  }  }  fifo\_length **=** 8;  }    buf[3] **=** (**char**) (size **>>** 8);  buf[4] **=** (**char**) (size **<<** 4);    Trf7970ResetIrqStatus();    RESET\_COUNTER; *// TimerA set*  A2CounterLoad(COUNT\_1ms **\*** 30);  *// COUNT\_VALUE = COUNT\_1ms \* 30; // 30ms, not 20ms*  IRQ\_CLR; *// PORT2 interrupt flag clear*  IRQ\_ON;    Trf7970RawWrite(**&**buf[0], mask\_size **+** fifo\_length);*// Transmitting 15693 Inventory Command.*    i\_reg **=** 0x01;  irq\_flag **=** 0x00;  START\_COUNTER; *// Starting Timeout*        *// while(TA0R < 1000) {*  *// i\_reg = TA0R;*  *// }*    **while** (irq\_flag **==** 0x00) {  } *// wait for end of TX interrupt*  RESET\_COUNTER;    **for** (j **=** 1; j **<=** no\_slots; j**++**) *// 1 or 16 available time slots*  {  rxtx\_state **=** 1; *// prepare the external counter*    *// the first UID will be stored from buf[1] upwards*  *// TimerA set*  *// COUNT\_VALUE = COUNT\_1ms \* 20;*  A2CounterLoad(COUNT\_1ms **\*** 20);  START\_COUNTER; *// start timer up mode*    irq\_flag **=** 0x00;    **while** (irq\_flag **==** 0x00) {    } *// wait for interrupt*  RESET\_COUNTER;    **while** (i\_reg **==** 0x01) *// wait for RX complete*  {  k**++**;    **if** (k **==** 0xFFF0) {  i\_reg **=** 0x00;  rx\_error\_flag **=** 0x00;  }  }    command[0] **=** RSSI\_LEVELS; *// read RSSI levels*  Trf7970ReadSingle(command, 1);  **switch** (i\_reg) {  **case** 0xFF**:** *// if received UID in buffer*  **if** (stand\_alone\_flag **==** 1) {  found **=** 1;  g\_tag\_found **=** ISO15693;  *// build uid string*  **char** **\*** cptr;  cptr **=** (**char** **\***) g\_uid;  **for** (i **=** 10; i **>** 2; i**--**) {  cptr **=** cptr **+** sprintf(cptr, "%x", (buf[i] **>>** 4) **&** 0x0F);  cptr **=** cptr **+** sprintf(cptr, "%x", buf[i] **&** 0x0F);  }  cptr **=** (**char** **\***) g\_rssi;  cptr **=** cptr **+** sprintf(cptr, "%x", (command[0] **>>** 4) **&** 0x0F);  cptr **=** cptr **+** sprintf(cptr, "%x", command[0] **&** 0x0F);    **#ifdef ENABLE\_HOST**    Tag\_Count**++**;    *// UartPutCrlf();*    *// UartSendCString("ISO15693/NFC-V UID: ");*  *// UartPutChar('[');*    **for** (i **=** 10; i **>** 2; i**--**) {  *// UartPutByte(buf[i]); // send UID to host*    }    *// UartPutChar(']');*  *// Report("\r\n");*  *// UartSendCString("RSSI LEVEL: ");*  *// UartPutChar('[');*  *// UartPutByte(command[0]); // RSSI levels*  *// UartPutChar(']');*  *// UartPutCrlf();*  *// Report("\r\n ");*  *// Report((char \*) g\_uid);*  *// Report("\r\n ");*    **#endif**  *// build\_message(my\_message, buf, 10, 3, "ISO15693/NFC-V UID: ");*      }  **break**;    **case** 0x02**:** *// collision occurred*  p\_slot\_no**++**; *// remember a collision was detected*  **\***p\_slot\_no **=** j;  **break**;    **case** 0x00**:** *// timer interrupt*  **if** (stand\_alone\_flag **==** 1) {  }  **break**;    **default:**  **break**;  }    Trf7970Reset(); *// FIFO has to be reset before receiving the next response*    **if** ((no\_slots **==** 16) **&&** (j **<** 16))*// if 16 slots used send EOF(next slot)*  {  Trf7970StopDecoders();  Trf7970RunDecoders();  Trf7970TransmitNextSlot();  } **else** **if** ((no\_slots **==** 16) **&&** (j **==** 16))*// at the end of slot 16 stop the slot counter*  {  Trf7970StopDecoders();  Trf7970DisableSlotCounter();  } **else** **if** (no\_slots **==** 1) *// 1 slot is used*  {  **break**;  }  } *// for*    **if** (found **==** 1) *// LED on?*  {  *//LED\_15693\_ON; // LEDs indicate detected ISO15693 tag*  **int** cx **=** 0;      **for** (ui8BlockNumber **=** 0; ui8BlockNumber **<** 0x4; ui8BlockNumber**++**) *//0x7F support includes TI Tag-It HF-I Plus, Pro, standard and STM M24LR04E tags)*  {  NFC\_TYPEV\_READ\_SINGLE\_BLOCK(ui8BlockNumber);  cx **=** cx **+** snprintf(g\_tag\_content**+**cx, **sizeof**(g\_tag\_content)**-**cx, "%s", g\_block\_content);  }    *// UART\_PRINT(g\_tag\_content);*    } **else** {  *//LED\_15693\_OFF;*  }    new\_length **=** length **+** 4; *// the mask length is a multiple of 4 bits*    mask\_size **=** (((new\_length **>>** 2) **+** 1) **>>** 1);    **while** ((**\***p\_slot\_no **!=** 0x00) **&&** (no\_slots **==** 16) **&&** (new\_length **<** 61)  **&&** (slot\_no[16] **!=** 16)) {  **\***p\_slot\_no **=** **\***p\_slot\_no **-** 1;    **for** (i **=** 0; i **<** 8; i**++**) {  new\_mask[i] **=** **\***(mask **+** i); *//first the whole mask is copied*  }    **if** ((new\_length **&** BIT2)**==** 0x00){  **\***p\_slot\_no **=** **\***p\_slot\_no **<<** 4;  } **else** {  **for** (i **=** 7; i **>** 0; i**--**) {  new\_mask[i] **=** new\_mask[i **-** 1];  }  new\_mask[0] **&=** 0x00;  }  new\_mask[0] **|=** **\***p\_slot\_no; *// the mask is changed*  McuDelayMillisecond(2);    Iso15693Anticollision(**&**new\_mask[0], new\_length);*// recursive call with new Mask*    p\_slot\_no**--**;  }    IRQ\_OFF;  } |

|  |
| --- |
| **iso15693.h** |
| */\**  *\* {iso15693.h}*  *\**  *\* {ISO15693 header file}*  *\**  *\**  *\*/*    **#ifndef \_ISO15693\_H\_**  **#define \_ISO15693\_H\_**    *//================================================================*      **#include** "trf7970.h"  **#include** "types.h"  **#include** "uart.h"    *//===============================================================*  *//Mandatory Command Request Flags*  **#define SIXTEEN\_SLOTS 0x06** *//tables 3 & 5 in ISO15693-3; single-subcarrier, high tag data rate, sixteen slots*  **#define ONE\_SLOT 0x26** *//tables 3 & 5 in ISO15693-3; single-subcarrier, high tag data rate, one slot*    *//Optional Command Request Flags*  **#define SSC\_HTDR\_NA 0x02** *//tables 3 & 4 in ISO15693-3; single-subcarrier, high tag data rate, non-addressed*  **#define SSC\_HTDR\_ADDR 0x22** *//tables 3 & 4 in ISO15693-3; single-subcarrier, high tag data rate, addressed*    **#define ISO15693 1**      *//===============================================================*    *// if disabled file ISO15693.c may be excluded from build*  *//#define ENABLE15693 // delete to disable standard*    *//===============================================================*    **void** **Iso15693FindTag**(**void**);  **void** **NFC\_TYPEV\_READ\_SINGLE\_BLOCK**(u08\_t StartBlock);  **void** **Iso15693Anticollision**(u08\_t **\***mask, u08\_t length);  *//void Iso15693ReadMultipleBlocks\_0\_18(void);*    *//===============================================================*    **#endif** |

|  |
| --- |
| **spi\_for\_trf.c** |
| */\**  *\* {spi\_for\_trf.c}*  *\**  *\* {SPI Interface Functions}*  *\**  *\**  *\*/*  **#include** "hw\_types.h"  **#include** "spi.h"  **#include** "spi\_for\_trf.h"  **#include** "gpio\_if.h"  **#include** "trf7970.h"  **#include** "hw\_memmap.h"  **#include** "hw\_common\_reg.h"  **#include** "hw\_ints.h"  **#include** "pin.h"  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "utils.h"  **#include** "prcm.h"  **#include** "interrupt.h"  **#include** "pinmux.h"  **#include** "uart.h"  **#include** "uart\_if.h"  **#include** "types.h"  **#include** "trf7970BoosterPack.h"        **#define ASSERT\_CS() (SPI\_CS\_ON)**  **#define DEASSERT\_CS() (SPI\_CS\_OFF)**    **typedef** **unsigned** **int** Fd\_t;    *//===============================================================*    u08\_t temp **=** 0;    **extern** u08\_t direct\_mode;    *//===============================================================*    **void** **SpiStartCondition**(**void**);  **void** **SpiStopCondition**(**void**);  **void** **SpiUsciSet**(**void**);  **void** **SpiUsciDisable**(**void**);    *//===============================================================*  *// NAME: void SpiDirectCommand (u08\_t \*pbuf)*  *//*  *// BRIEF: Is used in SPI mode to transmit a Direct Command to*  *// reader chip.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*pbuf Direct Command*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] transmit Direct Command*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *// 07Dec2010 RP integrated wait while busy loops*  *//===============================================================*    **void**  **SpiDirectCommand**(u08\_t **\***pbuf)  {  **unsigned** **char** ucDummy;  *// set Address/Command Word Bit Distribution to command*  **\***pbuf **=** (0x80 **|** **\***pbuf); *// command*  **\***pbuf **=** (0x9f **&** **\***pbuf); *// command code*  SPI\_CS\_OFF;  SPITransfer(GSPI\_BASE, pbuf, **&**ucDummy, 1, SPI\_CS\_ENABLE);  SPI\_CS\_ON;  }      *//===============================================================*  *// NAME: void SpiDirectMode (void)*  *//*  *// BRIEF: Is used in SPI mode to start Direct Mode.*  *//*  *// INPUTS:*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] start Direct Mode*  *//*  *// NOTE: No stop condition*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *// 07Dec2010 RP integrated wait while busy loops*  *//===============================================================*    **void**  **SpiDirectMode**(**void**)  {  u08\_t command [2];    command[0] **=** CHIP\_STATE\_CONTROL;  command[1] **=** CHIP\_STATE\_CONTROL;  *// read byte to command[1]*  SPI\_CS\_OFF;  SPITransfer(GSPI\_BASE, 0, **&**command[1], 1, SPI\_CS\_ENABLE);      command[1] **|=** 0x60; *// RF on and BIT 6 in Chip Status Control Register set*  *// write command[0] byte*  SPITransfer(GSPI\_BASE, command, 0, 1, SPI\_CS\_ENABLE);  SpiWriteSingle(command, 2);  SPI\_CS\_ON;  }    *//===============================================================*  *// NAME: void SpiRawWrite (u08\_t \*pbuf, u08\_t length)*  *//*  *// BRIEF: Is used in SPI mode to write direct to the reader chip.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*pbuf raw data*  *// u08\_t length number of data bytes*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] send raw data to reader chip*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *// 07Dec2010 RP integrated wait while busy loops*  *//===============================================================*    **void**  **SpiRawWrite**(u08\_t **\***pbuf, u08\_t length)  {  SPI\_CS\_OFF;  SPITransfer(GSPI\_BASE, pbuf, 0, length, SPI\_CS\_ENABLE);  SPI\_CS\_ON;  }    *//===============================================================*  *// NAME: void SpiReadCont (u08\_t \*pbuf, u08\_t length)*  *//*  *// BRIEF: Is used in SPI mode to read a specified number of*  *// reader chip registers from a specified address upwards.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*pbuf address of first register*  *// u08\_t length number of registers*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] read registers*  *// [2] write contents to \*pbuf*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *// 07Dec2010 RP integrated wait while busy loops*  *//===============================================================*    **void**  **SpiReadCont**(u08\_t **\***pbuf, u08\_t length)  {    *// Address/Command Word Bit Distribution*  **\***pbuf **=** (0x60 **|** **\***pbuf); *// address, read, continuous*  **\***pbuf **=** (0x7f **&\***pbuf); *// register address*    *// write command address/word*  SPI\_CS\_OFF;  SPITransfer(GSPI\_BASE, pbuf, 0, 1, SPI\_CS\_ENABLE);    *// read length bytes into pbuf[length]*  SPITransfer(GSPI\_BASE, 0, pbuf, length, SPI\_CS\_ENABLE);  SPI\_CS\_ON;    }    *//===============================================================*  *// NAME: void SpiReadSingle (u08\_t \*pbuf, u08\_t number)*  *//*  *// BRIEF: Is used in SPI mode to read specified reader chip*  *// registers.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*pbuf addresses of the registers*  *// u08\_t number number of the registers*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] read registers*  *// [2] write contents to \*pbuf*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *// 07Dec2010 RP integrated wait while busy loops*  *//===============================================================*    **void**  **SpiReadSingle**(u08\_t **\***pbuf, u08\_t number)  {    SPI\_CS\_OFF;    **while**(number **>** 0)  {  *// Address/Command Word Bit Distribution*  **\***pbuf **=** (0x40 **|** **\***pbuf); *// address, read, single*  **\***pbuf **=** (0x5f **&** **\***pbuf); *// register address*      *// write address/command byte into pbuf*  SPITransfer(GSPI\_BASE, pbuf, 0, 1, SPI\_CS\_ENABLE);    *// read single byte into pbuf*  SPITransfer(GSPI\_BASE, 0, pbuf, 1, SPI\_CS\_ENABLE);    pbuf**++**;  number**--**;    }  SPI\_CS\_ON;      }              *//===============================================================*  *// NAME: void SpiWriteCont (u08\_t \*pbuf, u08\_t length)*  *//*  *// BRIEF: Is used in SPI mode to write to a specific number of*  *// reader chip registers from a specific address upwards.*  *//*  *// INPUTS:*  *// u08\_t \*pbuf address of first register followed by the*  *// contents to write*  *// u08\_t length number of registers + 1*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] write to the registers*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *// 07Dec2010 RP integrated wait while busy loops*  *//===============================================================*    **void**  **SpiWriteCont**(u08\_t **\***pbuf, u08\_t length)  {      SPI\_CS\_OFF;    *// Address/Command Word Bit Distribution*  **\***pbuf **=** (0x20 **|** **\***pbuf); *// address, write, continuous*  **\***pbuf **=** (0x3f **&\***pbuf); *// register address*    *// write command address/word + number of registers*  SPITransfer(GSPI\_BASE, pbuf, 0, length, SPI\_CS\_ENABLE);    SPI\_CS\_ON;    }    *//===============================================================*  *// NAME: void SpiWriteSingle (u08\_t \*pbuf, u08\_t length)*  *//*  *// BRIEF: Is used in SPI mode to write to a specified reader chip*  *// registers.*  *//*  *// INPUTS:*  *// u08\_t \*pbuf addresses of the registers followed by the*  *// contends to write*  *// u08\_t length number of registers \* 2*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] write to the registers*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *// 07Dec2010 RP integrated wait while busy loops*  *//===============================================================*    **void**  **SpiWriteSingle**(u08\_t **\***pbuf, u08\_t length)  {    SPI\_CS\_OFF;  **while**(length **>** 0)  {  *// Address/Command Word Bit Distribution*  *// address, write, single (fist 3 bits = 0)*  **\***pbuf **=** (0x1f **&\***pbuf); *// register address*  SPITransfer(GSPI\_BASE, pbuf, 0, 2, SPI\_CS\_ENABLE);  pbuf**=**pbuf **+** 2;  length**=**length **-** 2;  }  SPI\_CS\_ON;    } |

|  |
| --- |
| **spi\_for\_trf.h** |
| */\**  *\* {spi\_for\_trf.h}*  *\**  *\* {Header File}*  *\**  *\*/*  **#ifndef \_SPI\_TRF\_H\_**  **#define \_SPI\_TRF\_H\_**    **#include** "types.h"  **#include** "hw\_types.h"  *//================================================================*            *//===============================================================*      **extern** **void** **SpiDirectCommand**(u08\_t **\***pbuf);  **extern** **void** **SpiDirectMode**(**void**);  **extern** **void** **SpiRawWrite**(u08\_t **\***pbuf, u08\_t length);  **extern** **void** **SpiReadCont**(u08\_t **\***pbuf, u08\_t length);  **extern** **void** **SpiReadSingle**(u08\_t **\***pbuf, u08\_t number);  **extern** **void** **SpiWriteCont**(u08\_t **\***pbuf, u08\_t length);  **extern** **void** **SpiWriteSingle**(u08\_t **\***pbuf, u08\_t length);    *//===============================================================*    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// Values that can be passed to SPIConfigSetExpClk() as ulMode parameter*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **#define SPI\_MODE\_MASTER 0x00000000**  **#define SPI\_MODE\_SLAVE 0x00000004**    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// Values that can be passed to SPIConfigSetExpClk() as ulSubMode parameter*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **#define SPI\_SUB\_MODE\_0 0x00000000**  **#define SPI\_SUB\_MODE\_1 0x00000001**  **#define SPI\_SUB\_MODE\_2 0x00000002**  **#define SPI\_SUB\_MODE\_3 0x00000003**      *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// Values that can be passed to SPIConfigSetExpClk() as ulConfigFlags parameter*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **#define SPI\_SW\_CTRL\_CS 0x01000000**  **#define SPI\_HW\_CTRL\_CS 0x00000000**  **#define SPI\_3PIN\_MODE 0x02000000**  **#define SPI\_4PIN\_MODE 0x00000000**  **#define SPI\_TURBO\_ON 0x00080000**  **#define SPI\_TURBO\_OFF 0x00000000**  **#define SPI\_CS\_ACTIVEHIGH 0x00000000**  **#define SPI\_CS\_ACTIVELOW 0x00000040**  **#define SPI\_WL\_8 0x00000380**  **#define SPI\_WL\_16 0x00000780**  **#define SPI\_WL\_32 0x00000F80**    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// Values that can be passed to SPIFIFOEnable() and SPIFIFODisable()*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **#define SPI\_TX\_FIFO 0x08000000**  **#define SPI\_RX\_FIFO 0x10000000**    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// Values that can be passed to SPIDMAEnable() and SPIDMADisable()*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **#define SPI\_RX\_DMA 0x00008000**  **#define SPI\_TX\_DMA 0x00004000**    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// Values that can be passed to SPIIntEnable(), SPIIntDiasble(),*  *// SPIIntClear() or returned from SPIStatus()*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **#define SPI\_INT\_DMATX 0x20000000**  **#define SPI\_INT\_DMARX 0x10000000**  **#define SPI\_INT\_EOW 0x00020000**  **#define SPI\_INT\_WKS 0x00010000**  **#define SPI\_INT\_RX\_OVRFLOW 0x00000008**  **#define SPI\_INT\_RX\_FULL 0x00000004**  **#define SPI\_INT\_TX\_UDRFLOW 0x00000002**  **#define SPI\_INT\_TX\_EMPTY 0x00000001**    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *// Values that can be passed to SPITransfer()*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **#define SPI\_CS\_ENABLE 0x00000001**  **#define SPI\_CS\_DISABLE 0x00000002**      **#endif** |

|  |
| --- |
| **trf7970.c** |
| */\**  *\* {trf7970.c}*  *\**  *\* {TRF7970 Communication functions}*  *\**  *\*/*    **#include** "trf7970.h"  **#include** "spi\_for\_trf.h"  **#include** "hw\_memmap.h"  **#include** "hw\_gpio.h"  **#include** "gpio.h"  **#include** "timer.h"  **#include** "timer\_if.h"  **#include** "trf7970BoosterPack.h"  *//===============================================================*    u08\_t command[2];  u08\_t direct\_mode **=** 0;  **extern** u08\_t buf[140];  **extern** u08\_t i\_reg;  **#ifdef ENABLE14443A**  **extern** u08\_t coll\_poss;  **#endif**  **extern** u08\_t irq\_flag;  **extern** u08\_t rx\_error\_flag;  **extern** s08\_t rxtx\_state;  **extern** u08\_t stand\_alone\_flag;    *//P\_EVENT\_HANDLER pIraEventHandler = 0;*    *//===============================================================*    **void** **Trf7970ISR**(u08\_t **\***irq\_status);  *//===============================================================*  *// ;*  *//===============================================================*  */\**  *void*  *Trf7970CommunicationSetup(void)*  *{*  *if(SPIMODE)*  *{*  *SpiSetup();*  *}*    *}*  *\*/*  *//===============================================================*  *// NAME: void Trf7970DirectCommand (u08\_t \*pbuf)*  *//*  *// BRIEF: Is used to transmit a Direct Command to Trf7970.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*pbuf Direct Command*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] transmit Direct Command*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970DirectCommand**(u08\_t **\***pbuf)  {  **if**(SPIMODE) *// SPI mode given by jumper setting*  {  SpiDirectCommand(pbuf);  }    }    *//===============================================================*  *// NAME: void Trf7970DirectMode (void)*  *//*  *// BRIEF: Is used to start Direct Mode.*  *//*  *// INPUTS:*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] start Direct Mode*  *//*  *// NOTE: No stop condition*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970DirectMode**(**void**)  {  direct\_mode **=** 1;    **if**(SPIMODE) *// SPI mode given by jumper setting*  {  SpiDirectMode();  }  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970DisableSlotCounter**(**void**)  {  buf[40] **=** IRQ\_MASK;  buf[41] **=** IRQ\_MASK; *// next slot counter*  Trf7970ReadSingle(**&**buf[41], 1);  buf[41] **&=** 0xFE; *// clear BIT0 in register 0x01*  Trf7970WriteSingle(**&**buf[40], 2);  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970EnableSlotCounter**(**void**)  {  buf[40] **=** IRQ\_MASK;  buf[41] **=** IRQ\_MASK; *// next slot counter*  Trf7970ReadSingle (**&**buf[41], 1);  buf[41] **|=** BIT0; *// set BIT0 in register 0x01*  Trf7970WriteSingle(**&**buf[40], 2);  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970InitialSettings**(**void**)  {  command[0] **=** SOFT\_INIT;  Trf7970DirectCommand(command);    command[0] **=** IDLE;  Trf7970DirectCommand(command);      **#ifdef TRF7970A**  command[0] **=** MODULATOR\_CONTROL;  command[1] **=** 0x01; *// ASK 100%, no SYS\_CLK output*  Trf7970WriteSingle(command, 2);  **#endif**  }    *//===============================================================*  *// The Interrupt Service Routine determines how the IRQ should ;*  *// be handled. The Trf7970 IRQ status register is read to ;*  *// determine the cause of the IRQ. Conditions are checked and ;*  *// appropriate actions taken. ;*  *//===============================================================*    **void**  **Trf7970ISR**(u08\_t **\***irq\_status)  {  **if**(**\***irq\_status **==** 0xA0) *// BIT5 and BIT7*  { *// TX active and only 3 bytes left in FIFO*  *// i\_reg = 0x00;*  }    **else** **if**(**\***irq\_status **==** BIT7)  { *// TX complete*  i\_reg **=** 0x00;  Trf7970Reset(); *// reset the FIFO after TX*  }    **else** **if**((**\***irq\_status **&** BIT1) **==** BIT1)  { *// collision error*  i\_reg **=** 0x02; *// RX complete*    **#ifdef ENABLE14443A**  coll\_poss **=** COLLISION\_POSITION;  Trf7970ReadSingle(**&**coll\_poss, 1);  buf[rxtx\_state] **=** FIFO; *// write the recieved bytes to the correct place of the*  *// buffer;*  **if** (coll\_poss **>=** 0x20)  {  Trf7970ReadCont(**&**buf[rxtx\_state], 5);  }  **#endif**    Trf7970StopDecoders(); *// reset the FIFO after TX*    Trf7970Reset();    Trf7970ResetIrqStatus();    GPIOIntClear(GPIOA1\_BASE, GPIO\_PIN\_3);  }  **else** **if**(**\***irq\_status **==** BIT6)  { *// RX flag means that EOF has been recieved*  *// and the number of unread bytes is in FIFOstatus regiter*  **if**(rx\_error\_flag **==** 0x02)  {  i\_reg **=** 0x02;  **return**;  }    **\***irq\_status **=** FIFO\_STATUS;  Trf7970ReadSingle(irq\_status, 1); *// determine the number of bytes left in FIFO*    **\***irq\_status **=** 0x7F **&** **\***irq\_status;  buf[rxtx\_state] **=** FIFO; *// write the recieved bytes to the correct place of the buffer*    Trf7970ReadCont(**&**buf[rxtx\_state], **\***irq\_status);  rxtx\_state **=** rxtx\_state **+** **\***irq\_status;    Trf7970Reset(); *// reset the FIFO after last byte has been read out*    i\_reg **=** 0xFF; *// signal to the recieve funnction that this are the last bytes*  }  **else** **if**(**\***irq\_status **==** 0x60)  { *// RX active and 9 bytes allready in FIFO*  i\_reg **=** 0x01;  buf[rxtx\_state] **=** FIFO;  Trf7970ReadCont(**&**buf[rxtx\_state], 9); *// read 9 bytes from FIFO*  rxtx\_state **=** rxtx\_state **+** 9;    **if**((IRQ\_PORT **&** IRQ\_PIN) **==** IRQ\_PIN) *// if IRQ pin high*  {  Trf7970ReadIrqStatus(irq\_status);    GPIOIntClear(GPIOA1\_BASE, GPIO\_PIN\_3);    **if**(**\***irq\_status **==** 0x40) *// end of recieve*  {  **\***irq\_status **=** FIFO\_STATUS;  Trf7970ReadSingle(irq\_status, 1); *// determine the number of bytes left in FIFO*    **\***irq\_status **=** 0x7F **&** **\***irq\_status;  buf[rxtx\_state] **=** FIFO; *// write the recieved bytes to the correct place of the buffer*      Trf7970ReadCont(**&**buf[rxtx\_state], **\***irq\_status);  rxtx\_state **=** rxtx\_state **+** **\***irq\_status;    i\_reg **=** 0xFF; *// signal to the recieve funnction that this are the last bytes*  Trf7970Reset(); *// reset the FIFO after last byte has been read out*  }  **else** **if**(**\***irq\_status **==** 0x50) *// end of recieve and error*  {  i\_reg **=** 0x02;  }  }  **else**  {  Trf7970ReadIrqStatus(irq\_status);    **if**(irq\_status[0] **==** 0x00)  {  i\_reg **=** 0xFF;  }  }  }  **else** **if**((**\***irq\_status **&** BIT4) **==** BIT4) *// CRC error*  {  **if**((**\***irq\_status **&** BIT5) **==** BIT5)  {  i\_reg **=** 0x01; *// RX active*  rx\_error\_flag **=** 0x02;  }  **if**((**\***irq\_status **&** BIT6) **==** BIT6) *// 4 Bit receive*  {  buf[100] **=** FIFO; *// write the received bytes to the correct place of the buffer*    Trf7970ReadCont(**&**buf[100], 1);    Trf7970Reset();    i\_reg **=** 0x02; *// end of RX*  rx\_error\_flag **=** 0x02;  }  **else**  {  i\_reg **=** 0x02; *// end of RX*  }  }  **else** **if**((**\***irq\_status **&** BIT2) **==** BIT2) *// byte framing error*  {  **if**((**\***irq\_status **&** BIT5) **==** BIT5)  {  i\_reg **=** 0x01; *// RX active*  rx\_error\_flag **=** 0x02;  }  **else**  i\_reg **=** 0x02; *// end of RX*  }  **else** **if**((**\***irq\_status **==** BIT0))  { *// No response interrupt*  i\_reg **=** 0x00;  }  **else**  { *// Interrupt register not properly set*  i\_reg **=** 0x02;    Trf7970StopDecoders(); *// reset the FIFO after TX*    Trf7970Reset();    Trf7970ResetIrqStatus();    GPIOIntClear(GPIOA1\_BASE, GPIO\_PIN\_4);  }  } *// Interrupt Service Routine*    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void** **Trf7970PortB**(**void**) *// interrupt handler*  {  u08\_t irq\_status[4], iso\_control;    irq\_flag **=** 0x02;    *// Sergey to fix STOP\_COUNTER; // stop timer mode*  RESET\_COUNTER;    **do**  {  GPIOIntClear(GPIOA1\_BASE, GPIO\_PIN\_4); *// interrupt flag clear*    iso\_control **=** ISO\_CONTROL;  Trf7970ReadSingle(**&**iso\_control, 1);  Trf7970ReadIrqStatus(irq\_status);    **if**((iso\_control **&** BIT5) **!=** BIT5) *// RFID mode*  {  Trf7970ISR(irq\_status);  }    } **while**(GPIOIntStatus(GPIOA1\_BASE,1) **&** GPIO\_PIN\_4);  *// \_\_bic\_SR\_register\_on\_exit(LPM0\_bits);*  }    *//===============================================================*  *// NAME: void Trf7970RawWrite (u08\_t \*pbuf, u08\_t length)*  *//*  *// BRIEF: Is used to write direct to the Trf7970.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*pbuf raw data*  *// u08\_t length number of data bytes*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] send raw data to Trf7970*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970RawWrite**(u08\_t **\***pbuf, u08\_t length)  {  **if**(SPIMODE) *// SPI mode given by jumper setting*  {  SpiRawWrite(pbuf, length);  }  }        *//===============================================================*  *// NAME: void Trf7970ReadCont (u08\_t \*pbuf, u08\_t length)*  *//*  *// BRIEF: Is used to read a specified number of Trf7970 registers*  *// from a specified address upwards.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*pbuf address of first register*  *// u08\_t length number of registers*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] read registers*  *// [2] write contents to \*pbuf*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970ReadCont**(u08\_t **\***pbuf, u08\_t length)  {  **if**(SPIMODE) *// SPI mode given by jumper setting*  {  SpiReadCont(pbuf, length);  }  **else** *// parallel mode*  {    }  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970ReadIrqStatus**(u08\_t **\***pbuf)  {  **\***pbuf **=** IRQ\_STATUS;  **\***(pbuf **+** 1) **=** IRQ\_MASK;  Trf7970ReadCont(pbuf, 2); *// read second reg. as dummy read*    }    *//===============================================================*  *// NAME: void Trf7970ReadSingle (u08\_t \*pbuf, u08\_t number)*  *//*  *// BRIEF: Is used to read specified Trf7970 registers.*  *//*  *// INPUTS:*  *// Parameters:*  *// u08\_t \*pbuf addresses of the registers*  *// u08\_t number number of the registers*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] read registers*  *// [2] write contents to \*pbuf*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970ReadSingle**(u08\_t **\***pbuf, u08\_t number)  {  **if**(SPIMODE) *// SPI mode given by jumper setting*  {  SpiReadSingle(pbuf, number);  }    }    *//===============================================================*  *// resets FIFO ;*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970Reset**(**void**)    {  command[0] **=** RESET;  Trf7970DirectCommand(command);    }    *//===============================================================*  *// resets IRQ Status ;*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970ResetIrqStatus**(**void**)  {  u08\_t irq\_status[4];  irq\_status[0] **=** IRQ\_STATUS;  irq\_status[1] **=** IRQ\_MASK;    **if**(SPIMODE) *// SPI mode given by jumper setting*  {  Trf7970ReadCont(irq\_status, 2); *// read second reg. as dummy read*  }    }    *//===============================================================*  *// ;*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970RunDecoders**(**void**)  {  command[0] **=** RUN\_DECODERS;  Trf7970DirectCommand(command);  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970StopDecoders**(**void**)  {  command[0] **=** STOP\_DECODERS;  Trf7970DirectCommand(command);  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970TransmitNextSlot**(**void**)  {  command[0] **=** TRANSMIT\_NEXT\_SLOT;  Trf7970DirectCommand(command);  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970TurnRfOff**(**void**)  {  command[0] **=** CHIP\_STATE\_CONTROL;  command[1] **=** CHIP\_STATE\_CONTROL;  Trf7970ReadSingle(**&**command[1], 1);  command[1] **&=** 0x1F;  Trf7970WriteSingle(command, 2);  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970TurnRfOn**(**void**)  {  u08\_t write[4];  write[0] **=** CHIP\_STATE\_CONTROL;  write[1] **=** 0x20; *// 3.3VDC, Full power out*  Trf7970WriteSingle(write, 2);    }    *//===============================================================*  *// NAME: void SpiWriteCont (u08\_t \*pbuf, u08\_t length)*  *//*  *// BRIEF: Is used to write to a specific number of reader chip*  *// registers from a specific address upwards.*  *//*  *// INPUTS:*  *// u08\_t \*pbuf address of first register followed by the*  *// contents to write*  *// u08\_t length number of registers + 1*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] write to the registers*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *//===============================================================*    **void**  **Trf7970WriteCont**(u08\_t **\***pbuf, u08\_t length)  {  **if**(SPIMODE) *// SPI mode given by jumper setting*  {  SpiWriteCont(pbuf, length);  }  **else** *// parallel mode*  {    }  }    *//===============================================================*  *// 02DEC2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970WriteIsoControl**(u08\_t iso\_control)  {  u08\_t write[16];  */\**  *write[0] = ISO\_CONTROL;*  *write[1] = iso\_control;*  *write[1] &= ~BIT5;*  *Trf7970WriteSingle(write, 2);*    *iso\_control &= 0x1F;*  *write[0] = IRQ\_MASK;*  *write[1] = 0x3F;*  *Trf7970WriteSingle(write, 2);*  *\*/*  *//#ifdef ENABLE15693 // this standard can be disabled in ISO15693.h*  **if** (iso\_control **==** 0x02)  {  write[0] **=** 0x20; *//Continuous Write, starting with register 0x00*  write[1] **=** 0x20; *//Value for Chip Status Control Register 0x00, 0x20 = +3.3VDC, full power, etc.*  write[2] **=** 0x02; *//Value for ISO Control Register 0x01, 0x02 = high tag data rate, etc.*  write[3] **=** 0x00;  write[4] **=** 0x00;  write[5] **=** 0xC1;  write[6] **=** 0xBB;  write[7] **=** 0x00;  write[8] **=** 0x30;  write[9] **=** 0x1F;  write[10] **=** 0x01;  write[11] **=** 0x40;  write[12] **=** 0x03;    Trf7970WriteCont(write, 13); *//writes registers 0x00:0x0B*  }  *//#endif*    *//#ifdef ENABLE14443A // this standard can be disabled in ISO14443A.h*  **if** (iso\_control **==** 0x88)    {  */\**  *write[0] = 0x20;*  *write[1] = 0x20; //full power out, 3.3VDC*  *write[2] = 0x88;*  *write[3] = 0x00;*  *write[4] = 0x00;*  *write[5] = 0xC1;*  *write[6] = 0xBB;*  *write[7] = 0x20;*  *write[8] = 0x0E;*  *write[9] = 0x07;*  *write[10] = 0x21;*  *write[11] = 0x20;*  *write[12] = 0x87;*    *Trf7970WriteCont(write, 13);*  *\*/*      *//write[0] = IRQ\_MASK;*  *//write[1] = 0x3E;*  *//Trf7970WriteSingle(write, 2);*  write[0] **=** ISO\_CONTROL;  write[1] **=** 0x88;  Trf7970WriteSingle(write, 2);    write[0] **=** MODULATOR\_CONTROL;  write[1] **=** 0x01;  Trf7970WriteSingle(write, 2);    write[0] **=** ADJUSTABLE\_FIFO\_LEVEL; *//this is register 0x14*  write[1] **=** 0x0F;  Trf7970WriteSingle(write, 2);    }  */\**  *write[0] = 0x20;*  *write[1] = 0x20; //full power out, 3.3VDC*  *write[2] = 0x88;*  *write[3] = 0x00;*  *write[4] = 0x00;*  *write[5] = 0xC1;*  *write[6] = 0xBB;*  *write[7] = 0x20;*  *write[8] = 0x0E;*  *write[9] = 0x07;*  *write[10] = 0x21;*  *write[11] = 0x20;*  *write[12] = 0x87;*    *Trf7970WriteCont(write, 13);*  *\*/*  *//#endif*    *//#ifdef ENABLE14443B // this standard can be disabled in ISO14443B.h*  **if** (iso\_control **==** 0x0C)  {  write[0] **=** 0x20;  write[1] **=** 0x20; *//full power out, 3.3VDC*  write[2] **=** 0x0C;  write[3] **=** 0x00;  write[4] **=** 0x00;  write[5] **=** 0xC1;  write[6] **=** 0xBB;  write[7] **=** 0x00;  write[8] **=** 0x0D;  write[9] **=** 0x07;  write[10] **=** 0x03;  write[11] **=** 0x00;  write[12] **=** 0x87;    Trf7970WriteCont(write, 13);    write[0] **=** IRQ\_MASK;  write[1] **=** 0x3E;  Trf7970WriteSingle(write, 2);    write[0] **=** ADJUSTABLE\_FIFO\_LEVEL;  write[1] **=** 0x0F;  Trf7970WriteSingle(write, 2);  }  *//#endif*    *//#ifdef ENABLEFELICA // this standard can be disabled in felica.h*  **if** (iso\_control **==** 0x1A)  {  write[0] **=** 0x20;  write[1] **=** 0x20; *//full power out, 3.3VDC*  write[2] **=** 0x1A;  *//write[3] = 0x00;*  *//write[4] = 0x00;*  *//write[5] = 0xC1;*  *//write[6] = 0xBB;*  *//write[7] = 0x00;*  *//write[8] = 0x14;*  *//write[9] = 0x00;*  *//write[10] = 0x00;*  *//write[11] = 0x00;*  *//write[12] = 0x87;*    Trf7970WriteCont(write, 03);    write[0] **=** IRQ\_MASK;  write[1] **=** 0x3E;  Trf7970WriteSingle(write, 2);    write[0] **=** ADJUSTABLE\_FIFO\_LEVEL;  write[1] **=** 0x0F;  Trf7970WriteSingle(write, 2);  }  *//#endif*    }    *//===============================================================*  *// NAME: void Trf7970WriteSingle (u08\_t \*pbuf, u08\_t length)*  *//*  *// BRIEF: Is used to write to specified reader chip registers.*  *//*  *// INPUTS:*  *// u08\_t \*pbuf addresses of the registers followed by the*  *// contents to write*  *// u08\_t length number of registers \* 2*  *//*  *// OUTPUTS:*  *//*  *// PROCESS: [1] write to the registers*  *//*  *// CHANGE:*  *// DATE WHO DETAIL*  *// 24Nov2010 RP Original Code*  *//===============================================================*    **void**  **Trf7970WriteSingle**(u08\_t **\***pbuf, u08\_t length)  {  **if**(SPIMODE) *// SPI mode given by jumper setting*  {  SpiWriteSingle(pbuf, length);  }  **else** *// parallel mode*  {    }  } |

|  |
| --- |
| **trf7970.h** |
| */\**  *\* {trf7970.h}*  *\**  *\* {Header File}*  *\**  *\*/*  **#ifndef \_Trf7970\_H\_**  **#define \_Trf7970\_H\_**    *//===============================================================*    **#include** <stdlib.h> // general purpose standard library    **#include** "types.h"      *//===============================================================*    **#define DBG 0** *// if DBG 1 interrupts are display*    *//==== Trf7970 definitions ======================================*    *//---- Direct commands ------------------------------------------*    **#define BIT0 0x01**  **#define BIT1 0x02**  **#define BIT2 0x04**  **#define BIT3 0x08**  **#define BIT4 0x10**  **#define BIT5 0x20**  **#define BIT6 0x40**  **#define BIT7 0x80**    **#define IDLE 0x00**  **#define SOFT\_INIT 0x03**  **#define INITIAL\_RF\_COLLISION 0x04**  **#define RESPONSE\_RF\_COLLISION\_N 0x05**  **#define RESPONSE\_RF\_COLLISION\_0 0x06**  **#define RESET 0x0F**  **#define TRANSMIT\_NO\_CRC 0x10**  **#define TRANSMIT\_CRC 0x11**  **#define DELAY\_TRANSMIT\_NO\_CRC 0x12**  **#define DELAY\_TRANSMIT\_CRC 0x13**  **#define TRANSMIT\_NEXT\_SLOT 0x14**  **#define CLOSE\_SLOT\_SEQUENCE 0x15**  **#define STOP\_DECODERS 0x16**  **#define RUN\_DECODERS 0x17**  **#define CHECK\_INTERNAL\_RF 0x18**  **#define CHECK\_EXTERNAL\_RF 0x19**  **#define ADJUST\_GAIN 0x1A**    *//---- Reader register ------------------------------------------*    **#define CHIP\_STATE\_CONTROL 0x00**  **#define ISO\_CONTROL 0x01**  **#define ISO\_14443B\_OPTIONS 0x02**  **#define ISO\_14443A\_OPTIONS 0x03**  **#define TX\_TIMER\_EPC\_HIGH 0x04**  **#define TX\_TIMER\_EPC\_LOW 0x05**  **#define TX\_PULSE\_LENGTH\_CONTROL 0x06**  **#define RX\_NO\_RESPONSE\_WAIT\_TIME 0x07**  **#define RX\_WAIT\_TIME 0x08**  **#define MODULATOR\_CONTROL 0x09**  **#define RX\_SPECIAL\_SETTINGS 0x0A**  **#define REGULATOR\_CONTROL 0x0B**  **#define IRQ\_STATUS 0x0C** *// IRQ Status Register*  **#define IRQ\_MASK 0x0D** *// Collision Position and Interrupt Mask Register*  **#define COLLISION\_POSITION 0x0E**  **#define RSSI\_LEVELS 0x0F**  **#define SPECIAL\_FUNCTION 0x10**  **#define RAM\_START\_ADDRESS 0x11** *//RAM is 6 bytes long (0x11 - 0x16)*  **#define ADJUSTABLE\_FIFO\_LEVEL 0x14**  **#define NFCID 0x17**  **#define NFC\_TARGET\_LEVEL 0x18**  **#define NFC\_TARGET\_PROTOCOL 0x19**  **#define TEST\_SETTINGS\_1 0x1A**  **#define TEST\_SETTINGS\_2 0x1B**  **#define FIFO\_STATUS 0x1C**  **#define TX\_LENGTH\_BYTE\_1 0x1D**  **#define TX\_LENGTH\_BYTE\_2 0x1E**  **#define FIFO 0x1F**    *//===============================================================*    **void** **Trf7970CommunicationSetup**(**void**);  **void** **Trf7970DirectCommand**(u08\_t **\***pbuf);  **void** **Trf7970DirectMode**(**void**);  **void** **Trf7970DisableSlotCounter**(**void**);  **void** **Trf7970EnableSlotCounter**(**void**);  **void** **Trf7970InitialSettings**(**void**);  **void** **Trf7970RawWrite**(u08\_t **\***pbuf, u08\_t length);  **void** **Trf7970ReConfig**(**void**);  **void** **Trf7970ReadCont**(u08\_t **\***pbuf, u08\_t length);  **void** **Trf7970ReadIrqStatus**(u08\_t **\***pbuf);  **void** **Trf7970ReadSingle**(u08\_t **\***pbuf, u08\_t length);  **void** **Trf7970Reset**(**void**);  **void** **Trf7970ResetIrqStatus**(**void**);  **void** **Trf7970RunDecoders**(**void**);  **void** **Trf7970StopDecoders**(**void**);  **void** **Trf7970TransmitNextSlot**(**void**);  **void** **Trf7970TurnRfOff**(**void**);  **void** **Trf7970TurnRfOn**(**void**);  **void** **Trf7970WriteCont**(u08\_t **\***pbuf, u08\_t length);  **void** **Trf7970WriteIsoControl**(u08\_t iso\_control);  **void** **Trf7970WriteSingle**(u08\_t **\***pbuf, u08\_t length);  **void** **Trf7970PortB**(**void**) ;  *//===============================================================*    **#endif** |

|  |
| --- |
| **trf7970BoosterPack.c** |
| */\**  *\* {trf7970BoosterPack.c}*  *\**  *\* {MSP430G2553 specific functions}*  *\**  *\*/*      **#include** "trf7970.h"  **#include** "trf7970BoosterPack.h"  **#include** "utils.h"      *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *//! Delay*  *//!*  *//! \param interval is time in milli second*  *//!*  *//! \return none*  *//!*  *//! \brief*  *//*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **void** **Delay**(**unsigned** **long** interval)  {  **while**(interval **>** 0)  {  UtilsDelay(80000);  interval**--**;  }  }    **void** **McuDelayMillisecond**(**unsigned** **int** ui\_delay\_in\_msec){  UtilsDelay(80000**\***ui\_delay\_in\_msec);    } |

|  |
| --- |
| **trf7970BoosterPack.h** |
| */\**  *\* {trf7970BoosterPack.h}*  *\**  *\* {Header File}*  *\**  *\**  *\*/*    **#ifndef \_Trf7970BOOSTERPACK\_H\_**  **#define \_Trf7970BOOSTERPACK\_H\_**    *//================================================================*      **#include** <stdint.h>  **#include** "types.h"      **#define \_BOARD\_H**        */\* This macro is for use by other macros to form a fully valid C statement. \*/*  **#define st(x) do { x } while (\_\_LINE\_\_ == -1)**      **#define COUNT\_1ms 80000**  **#define IRQ\_CLR GPIOIntClear(GPIOA1\_BASE, GPIO\_PIN\_4);** *//GPIO 7*  **#define IRQ\_ON GPIOIntEnable(GPIOA1\_BASE,GPIO\_PIN\_4);**  **#define IRQ\_OFF GPIOIntDisable(GPIOA1\_BASE,GPIO\_PIN\_4);**  **#define START\_COUNTER A2CounterEnable();**  **#define RESET\_COUNTER A2CounterDisable();**  **#define TRF\_OFF GPIO\_IF\_Set(11,0);**  **#define TRF\_ON GPIO\_IF\_Set(11,1);**  **#define SPI\_CS\_ON MAP\_SPICSEnable(GSPI\_BASE);**  **#define SPI\_CS\_OFF MAP\_SPICSDisable(GSPI\_BASE);**      **void** **Delay**(**unsigned** **long** interval);  **void** **McuDelayMillisecond**(**unsigned** **int** ui\_delay\_in\_msec);              *//===============================================================*    **#define TRIGGER 0** *// if TRIGGER 1 trigger-point at LED 5*    *//=====MCU constants=============================================*      *// IRQ on P2.0*  *// rising edge interrupt*  **#define IRQ\_PIN BIT7**  **#define IRQ\_PORT HWREG(GPIOA1\_BASE + GPIO\_O\_GPIO\_RIS)**            *// Hard code to only SPI mode*  **#define SPIMODE 1**    *// CSn*    *//-----Counter-timer constants-----------------------------------*      *//---------------------------------------------------------------*  *// Set timer capture register value based on clock source*      *//---------------------------------------------------------------*    *// 25MHz clock*      *//===============================================================*    **#endif** |

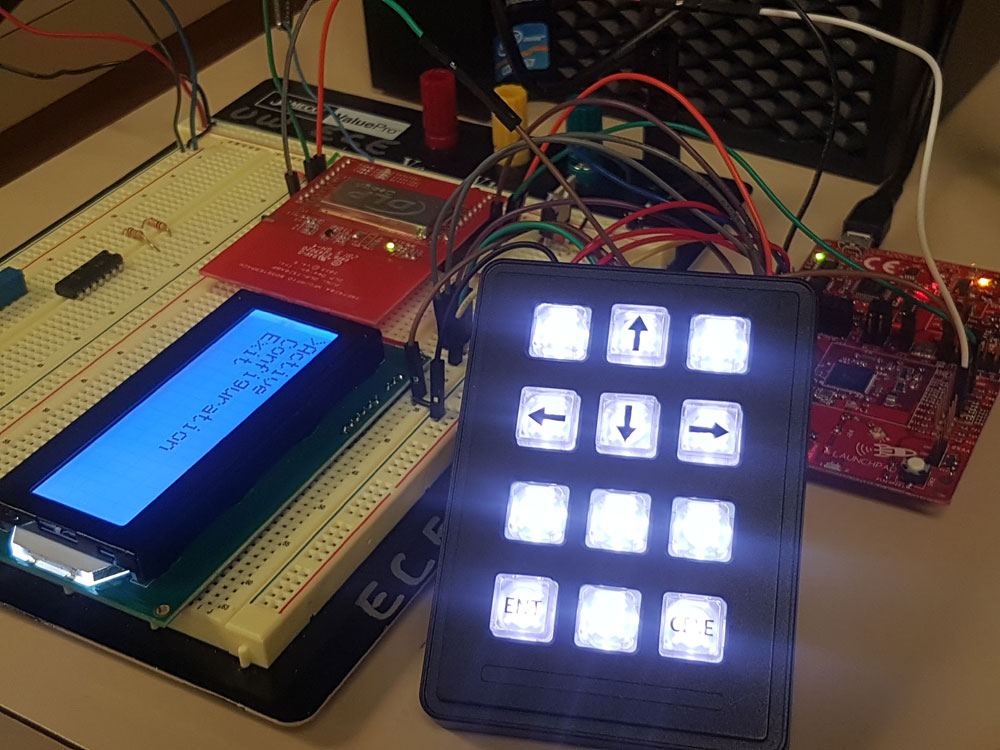
|  |
| --- |
| **main.c** |
| */\**  *\* Smart Doorlock*  *\* main.c*  *\**  *\* Created on: 2016. 8. 27.*  *\**  *\* Author: Sokwhan Huh*  *\*/*    *// C-Library includes*  **#include** <string.h>    *// Driverlib includes*  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_common\_reg.h"  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "uart.h"  **#include** "interrupt.h"  **#include** "pinmux.h"  **#include** "utils.h"  **#include** "prcm.h"  **#include** "simplelink.h"  **#include** "gpio.h"  **#include** "timer.h"    *// Common interface include*  **#include** "common.h"  **#include** "uart\_if.h"  **#include** "gpio\_if.h"  **#include** "timer\_if.h"    *// Project includes*  **#include** "network.h"  **#include** "keypad.h"  **#include** "lcd.h"  **#include** "mqtt\_client.h"  **#include** "spi\_l.h"  **#include** "trf7970.h"  **#include** "trf7970BoosterPack.h"  **#include** "iso15693.h"    **#define APP\_NAME "Smart Doorlock"**    *//RTOS Related Defines*  **#define OSI\_STACK\_SIZE 4096** */\* 2048 \*/*  **#define SPAWN\_TASK\_PRIORITY 9**  **#define CONNECTION\_TIMEOUT\_COUNT 20** */\* 10sec \*/*    *//Globals*  **unsigned** **int** g\_appReady **=** 0;  **unsigned** **int** g\_activeMode **=** 0;    *//===============================================================*  */\*\*\*\*\*\*\*\*\*\* GLOBAL VARIABLES TRF7970A \*\*\*\*\*\*\*\*\*\*/*  *//===============================================================*  u08\_t buf[300]; *// TX/RX BUFFER FOR TRF7970A*  u08\_t g\_uid[300] **=** "none"; *// used for coping card ID*  **char** g\_tag\_content[600]; *// used for saving a content of TAG buffer*  **char** g\_block\_content[200]; *// used for saving a content of single/multiple block(s)*    u08\_t g\_rssi[10];  u08\_t g\_tag\_found **=** 0; *// 0->no tag found*  *// 1- ISO15693 tag found*  *// 2- ISO14443A tag found*  *// 8 - MASTER*  u08\_t Tag\_Count;  u08\_t i\_reg **=** 0x01; *// INTERRUPT REGISTER*  u08\_t irq\_flag **=** 0x00;  u08\_t rx\_error\_flag **=** 0x00;  s08\_t rxtx\_state **=** 1; *// USED FOR TRANSMIT RECEIVE BYTE COUNT*  u08\_t host\_control\_flag **=** 0;  u08\_t stand\_alone\_flag **=** 1;  **int** g\_tag\_count; *// Tag counter*  **char** g\_tag\_count\_str[10]; *// string representation of tag counter*  **static** **void** **DisplayBanner**(**char** **\*** AppName)  {  Report("\n\n\n\r");  Report("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  Report("\t\t %s \n\r", AppName);  Report("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  Report("\n\n\n\r");  }  **static** **void** **SmartDoorlockLCDBanner**() {  lcdReset();  lcdInit();  lcdDisplayOn();  lcdClearScreen();  lcdPutString("Smart Doorlock");  lcdSetPosition(LCD\_LINE2);  lcdPutString("Initializing");  }  **static** **void** **BoardInit**(**void**)  {  *// Enable Processor*  MAP\_IntMasterEnable();  MAP\_IntEnable(FAULT\_SYSTICK);  PRCMCC3200MCUInit();  }  **static** **void** **SmartDoorlockMenuTask**(**void** **\***pvParameters) {  SmartDoorlockLCDBanner();  **for** (;;) {  **if** (g\_appReady) {  buttonEnum pressedBtn **=** getPressedButton();  **char** **\***btnType;*//*  **switch** (pressedBtn) {  **case** UP\_ARROW:  btnType **=** "UP";  **break**;  **case** LEFT\_ARROW:  btnType **=** "LEFT";  **break**;  **case** DOWN\_ARROW:  btnType **=** "DOWN";  **break**;  **case** RIGHT\_ARROW:  btnType **=** "RIGHT";  **break**;  **case** ENTER:  btnType **=** "ENTER";  **case** CANCEL:  Report("Disconnecting from MQTT/AP\n\r");  Mqtt\_ClientExit();  Network\_IF\_DisconnectFromAP();  Network\_IF\_DeInitDriver();  Report("Exiting");  **return**;  }  **if** (pressedBtn **!=** NONE) {  Report("Pressed: %s \n\r",btnType);  }  }  osi\_Sleep(50);  }  }    **static** **void** **SmartDoorlockNFCTask**(**void** **\***pvParameters) {  Report("Entering NFC tag read mode\n\r");    g\_tag\_found **=** 0;  *// TRF IRQ disable and clear*  IRQ\_OFF;  *// TRF disable*  TRF\_OFF;  *// delay at least 10 ms*  osi\_Sleep(100);    *// Enter LPM3*  TRF\_ON;  *// Must wait at least 4.8 mSec to allow TRF7970A to initialize.*  osi\_Sleep(5);    **for** (;;) {  g\_tag\_found **=** 0;  Iso15693FindTag(); *// Scan for 15693 tags*    **if**(g\_tag\_found) {  UART\_PRINT("Tag Found \n\r");  }  osi\_Sleep(10);  }  }      **static** **void** **SmartDoorlockIoTTask**(**void** **\***pvParameters) {  **int** retVal **=** ConnectAP("SW\_Private", "smartdoorlock");  **if** (retVal **!=** 0) {  Report("Connection to AP failed!\n\r");  **return**;  }    Report("Connection Successful!\n\r");  g\_appReady **=** 1;  retVal **=** initMqtt();  **if** (retVal **!=** 0)  **return**;    mqttConnect();  event\_msg RecvQue;  **for**(;;)  {  osi\_MsgQRead( **&**g\_PBQueue, **&**RecvQue, OSI\_WAIT\_FOREVER);  **if**(BROKER\_DISCONNECTION **==** RecvQue.event)  {  attemptReconnect();  }  **if**(DOORLOCK\_OPEN **==** RecvQue.event)  {  Report("Queue: Opening Doorlock\n\r");  }  */\* const char \*pub\_topic\_sw3 = "/cc3200/ButtonPressEvtSw3";*  *unsigned char \*data\_sw2={"Push button sw2 is pressed on CC32XX device"};*  *sl\_ExtLib\_MqttClientSend((void\*)local\_con\_conf[0].clt\_ctx,//*  *pub\_topic\_sw3,data\_sw2,strlen((char\*)data\_sw2),QOS2,RETAIN);*  *UART\_PRINT("\n\r CC3200 Publishes the following message \n\r");*  *UART\_PRINT("Topic: %s\n\r","TEST");*  *UART\_PRINT("Data: %s\n\r","TEST");\*/*  }    }    *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  *//*  *//! This function gets triggered when A2CNT interrupt occures*  *//!*  *//! \param none*  *//!*  *//! \return None*  *//!*  *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**  **void** **A2CNTIntHandler** (**void**){  u08\_t irq\_status[4];  TimerIntClear(TIMERA2\_BASE, TIMER\_A); *// STOP\_COUNTER;*    irq\_flag **=** 0x02;    Trf7970ReadIrqStatus(irq\_status);    **\***irq\_status **=** **\***irq\_status **&** 0xF7; *// set the parity flag to 0*    **if**(**\***irq\_status **==** 0x00 **||** **\***irq\_status **==** 0x80)  {  i\_reg **=** 0x00; *// timer interrupt*  }  **else**  {  i\_reg **=** 0x01;  }  }    **int** **main**(**void**) {  *// Initailizing the board*  BoardInit();  *// Muxing for Enabling GPIO, UART\_TX and UART\_RX.*  PinMuxConfig();  *//Init SPI*  SPIInit();  *//Turn off TRF7970A*  SPI\_TRF\_CS\_OFF;  SPI\_LCD\_CS\_OFF;    *// GPIO interrupt setting*  *// TRF7970 IRQ*  GPIOIntInit(GPIOA1\_BASE, GPIO\_PIN\_4, INT\_GPIOA1, Trf7970PortB, GPIO\_RISING\_EDGE, INT\_PRIORITY\_LVL\_1);    *// Set Clock Frequency and Modulation*  Trf7970InitialSettings();    A2CounterInit(A2CNTIntHandler);    *// Init Terminal*  InitTerm();  ClearTerm();  DisplayBanner(APP\_NAME);    *//Start the simplelink host*  VStartSimpleLinkSpawnTask(SPAWN\_TASK\_PRIORITY);        */\**    *// Start the SmartDoorlock task*  *osi\_TaskCreate( SmartDoorlockNFCTask,*  *(const signed char\*)"Smart Doorlock NFCTask",*  *OSI\_STACK\_SIZE, NULL, 1, NULL );*  *\*/*      *// Start the SmartDoorlock task*  osi\_MsgQCreate(**&**g\_PBQueue,"PBQueue",**sizeof**(event\_msg),10);  osi\_TaskCreate( SmartDoorlockIoTTask,  (**const** **signed** **char\***)"Smart Doorlock IoTTask",  OSI\_STACK\_SIZE, NULL, 1, NULL );    *// Start the Keypad task*  osi\_TaskCreate( SmartDoorlockMenuTask,  (**const** **signed** **char\***)"MenuTask",  OSI\_STACK\_SIZE, NULL, 1, NULL );  osi\_start();  **return** 0;  } |

# October 7th, 2016

I’ve made a critical mistake of burning out my MCU, LCD and NFC reader. What happened was I left one of the jumper wire unconnected and while trying to rearrange some connections on the breadboard, that wire left unconnected somehow made its way to the 9V power line, immediately destroying everything that was connected to it. Unfortunately, those parts were unsalvageable and I had to reorder the CC3200 launchpad, TRF7970A NFC reader and LCD.

I decided to change the LCD to a different one. The reason for this was that I wanted a display that’s slightly bigger and since I was experiencing issues when TRF7970A and LCD were connected at the same time (strangely, they work perfectly when connected independently. When connected together, TRF7970A simply stops reading and LCD starts to display broken characters), I wanted a display that runs on UART, instead of SPI. I decided to purchase Newhaven’s 20x4 display (NHD-0420D3Z-NSW-BBW-V3-ND), as its size was slightly larger than EA’s display and it supports four different interfacing modes: parallel, UART, SPI and I2C.

When I’ve reconnected everything with new parts, everything worked perfectly. Only issue I’ve encountered was that the NFC reading would slow down my entire program, so I had to optimize the program a bit by adjusting the NFC reading interval.

*Full view of prototype setup*

*Smart doorlock in active mode*

Because the LCD was changed, its code was updated accordingly.

|  |
| --- |
| **lcd.c** |
| /\*  \* lcd.c  \*  \* Created on: Sep 23, 2016  \* Author: shuh  \*/  **#include** "lcd.h"  // Driverlib includes  **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_types.h"  **#include** "prcm.h"  **#include** "simplelink.h"  **#include** "uart.h"  // Common Interface Includes  **#include** "uart\_if.h"  **#define** LCD\_LINE1 0  **#define** LCD\_LINE2 0x40  **#define** LCD\_LINE3 0x14  **#define** LCD\_LINE4 0x54  **typedef** **enum**  {  *LCD\_INIT*,  *CLEAR\_SCREEN*  } lcdCommandEnum;  **static** **void** **lcdPutCommand**(lcdCommandEnum cmdType) {  **switch** (cmdType) {  **case** *CLEAR\_SCREEN*:  MAP\_UARTCharPut(UARTA1\_BASE,0xFE);  MAP\_UARTCharPut(UARTA1\_BASE,0x51);  **break**;  **case** *LCD\_INIT*:  MAP\_UARTConfigSetExpClk(UARTA1\_BASE,MAP\_PRCMPeripheralClockGet(CONSOLE\_PERIPH),  9600, (UART\_CONFIG\_WLEN\_8 | UART\_CONFIG\_STOP\_ONE |  UART\_CONFIG\_PAR\_NONE));  **osi\_Sleep**(5);  MAP\_UARTCharPut(UARTA1\_BASE,0xFE);  MAP\_UARTCharPut(UARTA1\_BASE,0x41);  **break**;  }  }  **void** **lcdSetPosition**(**unsigned** **int** position) {  **if** (position < 1 || position > 4)  **return**;  MAP\_UARTCharPut(UARTA1\_BASE,0xFE);  MAP\_UARTCharPut(UARTA1\_BASE,0x45);  **switch** (position) {  **case** 1:  MAP\_UARTCharPut(UARTA1\_BASE,LCD\_LINE1);  **break**;  **case** 2:  MAP\_UARTCharPut(UARTA1\_BASE,LCD\_LINE2);  **break**;  **case** 3:  MAP\_UARTCharPut(UARTA1\_BASE,LCD\_LINE3);  **break**;  **case** 4:  MAP\_UARTCharPut(UARTA1\_BASE,LCD\_LINE4);  **break**;  }  }  **void** **lcdClearScreen**(**void**) {  lcdPutCommand(*CLEAR\_SCREEN*);  **osi\_Sleep**(3);  }  **void** **lcdInit**(**void**) {  lcdPutCommand(*LCD\_INIT*);  **osi\_Sleep**(1);  }  **void** **lcdPutChar**(**unsigned** **char** lcdChar) {  MAP\_UARTCharPut(UARTA1\_BASE,lcdChar);  }  **void** **lcdPutString**(**unsigned** **char**\* str) {  **do**  {  lcdPutChar(\*str++);  }  **while**(\*str);  } |

|  |
| --- |
| **lcd.h** |
| /\*  \* lcd.h  \*  \* Created on: Sep 23, 2016  \* Author: shuh  \*/  **#ifndef** LCD\_H\_  **#define** LCD\_H\_  **extern** **void** **lcdInit**();  **extern** **void** **lcdClearScreen**(**void**);  **extern** **void** **lcdPutString**(**unsigned** **char**\* str);  **extern** **void** **lcdPutChar**(**unsigned** **char** lcdChar);  **extern** **void** **lcdSetPosition**(**unsigned** **int** position);  **#endif** /\* LCD\_H\_ \*/ |

Slight modifications were made to main.c as well (lcd related changes & refactoring)

|  |
| --- |
| **main.c** |
| /\*  \* Smart Doorlock  \* main.c  \*  \* Created on: 2016. 8. 27.  \*  \* Author: Sokwhan Huh  \*/  // C-Library includes  **#include** <string.h>  // Driverlib includes  **#include** "rom.h"  **#include** "rom\_map.h"  **#include** "hw\_memmap.h"  **#include** "hw\_common\_reg.h"  **#include** "hw\_types.h"  **#include** "hw\_ints.h"  **#include** "uart.h"  **#include** "interrupt.h"  **#include** "pinmux.h"  **#include** "utils.h"  **#include** "prcm.h"  **#include** "simplelink.h"  **#include** "gpio.h"  **#include** "timer.h"  // Common interface include  **#include** "common.h"  **#include** "uart\_if.h"  **#include** "gpio\_if.h"  **#include** "timer\_if.h"  // Project includes  **#include** "network.h"  **#include** "keypad.h"  **#include** "lcd.h"  **#include** "mqtt\_client.h"  **#include** "spi\_l.h"  **#include** "trf7970.h"  **#include** "trf7970BoosterPack.h"  **#include** "iso15693.h"  **#define** APP\_NAME "Smart Doorlock"  //RTOS Related Defines  **#define** OSI\_STACK\_SIZE 4096 /\* 2048 \*/  **#define** SPAWN\_TASK\_PRIORITY 9  **#define** CONNECTION\_TIMEOUT\_COUNT 20 /\* 10sec \*/  //Globals  **unsigned** **int** g\_appReady;  **unsigned** **int** g\_openingDoor;  **unsigned** **int** g\_appMode;  **unsigned** **int** g\_currMenuOption;  //===============================================================  /\*\*\*\*\*\*\*\*\*\* GLOBAL VARIABLES TRF7970A \*\*\*\*\*\*\*\*\*\*/  //===============================================================  u08\_t buf[300]; // TX/RX BUFFER FOR TRF7970A  u08\_t g\_uid[300] = "none"; // used for coping card ID  **char** g\_tag\_content[600]; // used for saving a content of TAG buffer  **char** g\_block\_content[200]; // used for saving a content of single/multiple block(s)  u08\_t g\_rssi[10];  u08\_t g\_tag\_found = 0; // 0->no tag found  // 1- ISO15693 tag found  // 2- ISO14443A tag found  // 8 - MASTER  u08\_t Tag\_Count;  u08\_t i\_reg = 0x01; // INTERRUPT REGISTER  u08\_t irq\_flag = 0x00;  u08\_t rx\_error\_flag = 0x00;  s08\_t rxtx\_state = 1; // USED FOR TRANSMIT RECEIVE BYTE COUNT  u08\_t host\_control\_flag = 0;  u08\_t stand\_alone\_flag = 1;  **int** g\_tag\_count; // Tag counter  **char** g\_tag\_count\_str[10]; // string representation of tag counter  **typedef** **enum**  {  *LCD\_DISP\_INIT*,  *LCD\_DISP\_CONNECT\_AP*,  *LCD\_DISP\_CONNECT\_MQTT*,  *LCD\_DISP\_ACTIVE*,  *LCD\_DISP\_OPENING\_DOOR*,  *LCD\_DISP\_EXITING\_APP*  } sdLcdEnum;  **typedef** **enum**  {  *MODE\_MENU*,  *MODE\_ACTIVE*,  *MODE\_CONFIG*  } appModeEnum;  **typedef** **enum**  {  *MENU\_ACTIVE*,  *MENU\_CONFIG*,  *MENU\_EXIT*  } appMenuEnum;  **#define** MENU\_COUNT 3  **const** **unsigned** **char** \*menuList[3];  **static** **void** **DisplayBanner**(**char** \* AppName)  {  **Report**("\n\n\n\r");  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\t\t %s \n\r", AppName);  **Report**("\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\r");  **Report**("\n\n\n\r");  }  **static** **void** **BoardInit**(**void**)  {  // Enable Processor  MAP\_IntMasterEnable();  MAP\_IntEnable(FAULT\_SYSTICK);  **PRCMCC3200MCUInit**();  }  **static** **void** **SmartDoorlockLCDDisplay**(sdLcdEnum lcdEnum) {  lcdClearScreen();  **switch** (lcdEnum) {  **case** *LCD\_DISP\_INIT*:  lcdPutString("Smart Doorlock");  lcdSetPosition(2);  lcdPutString("Initializing");  **break**;  **case** *LCD\_DISP\_CONNECT\_AP*:  lcdPutString("Smart Doorlock");  lcdSetPosition(2);  lcdPutString("Connecting to AP...");  lcdSetPosition(3);  lcdPutString("SSID: SW\_Private");  **break**;  **case** *LCD\_DISP\_CONNECT\_MQTT*:  lcdPutString("Smart Doorlock");  lcdSetPosition(2);  lcdPutString("Connecting to");  lcdSetPosition(3);  lcdPutString("MQTT Broker...");  **break**;  **case** *LCD\_DISP\_ACTIVE*:  lcdPutString("Smart Doorlock");  lcdSetPosition(2);  lcdPutString("NFC / IoT Ready");  **break**;  **case** *LCD\_DISP\_OPENING\_DOOR*:  lcdPutString("Smart Doorlock");  lcdSetPosition(2);  lcdPutString("Opening Door...");  **break**;  **case** *LCD\_DISP\_EXITING\_APP*:  lcdPutString("Smart Doorlock");  lcdSetPosition(2);  lcdPutString("Exiting App.");  **break**;  }  }  **static** **void** **MoveMenu**(**int** menuOption) {  lcdClearScreen();  **int** i = 0;  **for** (i = 0; i < MENU\_COUNT; i++) {  lcdSetPosition(i+1);  i == menuOption ? lcdPutChar('>') : lcdPutChar(' ');  lcdPutString((**unsigned** **char**\*)menuList[i]);  }  }  **static** **void** **OpenDoor**() {  **Report**("Opening Doorlock\n\r");  SmartDoorlockLCDDisplay(*LCD\_DISP\_OPENING\_DOOR*);  g\_openingDoor = 1;  GPIO\_IF\_Set(22,1);  **osi\_Sleep**(3000);  GPIO\_IF\_Set(22,0);  g\_openingDoor = 0;  SmartDoorlockLCDDisplay(*LCD\_DISP\_ACTIVE*);  **Report**("Closing Doorlock\n\r");  }  **static** **void** **SmartDoorlockMenuTask**(**void** \*pvParameters) {  lcdInit();  lcdClearScreen();  SmartDoorlockLCDDisplay(*LCD\_DISP\_INIT*);  menuList[0] = "Active";  menuList[1] = "Configuration";  menuList[2] = "Exit";  g\_appMode = *MODE\_MENU*;  g\_currMenuOption = 0;  **while** (!g\_appReady) {  **osi\_Sleep**(1);  }  MoveMenu(g\_currMenuOption);  **for** (;;) {  buttonEnum pressedBtn = getPressedButton();  **if** (g\_appMode == *MODE\_MENU*) {  **if** (pressedBtn == *UP\_ARROW* && g\_currMenuOption > 0) {  g\_currMenuOption--;  MoveMenu(g\_currMenuOption);  }  **else** **if** (pressedBtn == *DOWN\_ARROW* && g\_currMenuOption < MENU\_COUNT - 1) {  g\_currMenuOption++;  MoveMenu(g\_currMenuOption);  }  **else** **if** (pressedBtn == *ENTER*) {  **if** (g\_currMenuOption == *MENU\_ACTIVE*) {  g\_appMode = *MODE\_ACTIVE*;  SmartDoorlockLCDDisplay(*LCD\_DISP\_ACTIVE*);  }  **else** **if** (g\_currMenuOption == *MENU\_EXIT*) {  SmartDoorlockLCDDisplay(*LCD\_DISP\_EXITING\_APP*);  **Report**("Disconnecting from MQTT/AP\n\r");  Mqtt\_ClientExit();  Network\_IF\_DisconnectFromAP();  Network\_IF\_DeInitDriver();  **Report**("Exiting");  **return**;  }  }  }  **else** **if** (g\_appMode == *MODE\_ACTIVE* && g\_openingDoor == 0) {  **if** (pressedBtn == *CANCEL*) {  g\_appMode = *MODE\_MENU*;  MoveMenu(g\_currMenuOption);  }  }  **osi\_Sleep**(40);  }  }  **static** **void** **SmartDoorlockNFCTask**(**void** \*pvParameters) {  **Report**("Entering NFC tag read mode\n\r");  **for** (;;) {  **if** (g\_appMode != *MODE\_ACTIVE* || g\_openingDoor == 1) {  **osi\_Sleep**(1);  **continue**;  }  g\_tag\_found = 0;  // TRF IRQ disable and clear  IRQ\_OFF;  // TRF disable  TRF\_OFF;  // delay at least 10 ms  **osi\_Sleep**(100);  // Enter LPM3  TRF\_ON;  // Must wait at least 4.8 mSec to allow TRF7970A to initialize.  **osi\_Sleep**(5);  Iso15693FindTag(); // Scan for 15693 tags  **if**(g\_tag\_found) {  UART\_PRINT("Tag Found \n\r");  OpenDoor();  }  **osi\_Sleep**(300);  }  }  **static** **void** **SmartDoorlockIoTTask**(**void** \*pvParameters) {  **osi\_Sleep**(500);  SmartDoorlockLCDDisplay(*LCD\_DISP\_CONNECT\_AP*);  **int** retVal = ConnectAP("SW\_Private", "smartdoorlock");  **if** (retVal != 0) {  lcdClearScreen();  lcdPutString("Connection to AP failed!");  **Report**("Connection to AP failed!\n\r");  **return**;  }  **Report**("Connection Successful!\n\r");  retVal = initMqtt();  **if** (retVal != 0)  **return**;  **osi\_Sleep**(100);  SmartDoorlockLCDDisplay(*LCD\_DISP\_CONNECT\_MQTT*);  retVal = mqttConnect();  **osi\_Sleep**(500);  **if** (retVal != 0) {  lcdClearScreen();  lcdPutString("Connection to MQTT failed!");  **return**;  }  g\_appReady = 1;  event\_msg RecvQue;  **for**(;;)  {  **osi\_MsgQRead**( &g\_PBQueue, &RecvQue, OSI\_WAIT\_FOREVER);  **if** (g\_appMode != *MODE\_ACTIVE*) {  **Report**("IoT Task: Msg received but not in active mode\n\r");  **osi\_Sleep**(1);  **continue**;  }  **if**(*BROKER\_DISCONNECTION* == RecvQue.event)  {  attemptReconnect();  }  **if**(*DOORLOCK\_OPEN* == RecvQue.event)  {  **if** (g\_openingDoor == 1) {  **Report**("IoT Task: Doorlock is already being opened\n\r");  **continue**;  }  OpenDoor();  }  }  }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  //  //! This function gets triggered when A2CNT interrupt occures  //!  //! \param none  //!  //! \return None  //!  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  **void** **A2CNTIntHandler** (**void**){  u08\_t irq\_status[4];  **TimerIntClear**(TIMERA2\_BASE, TIMER\_A); // STOP\_COUNTER;  irq\_flag = 0x02;  Trf7970ReadIrqStatus(irq\_status);  \*irq\_status = \*irq\_status & 0xF7; // set the parity flag to 0  **if**(\*irq\_status == 0x00 || \*irq\_status == 0x80)  {  i\_reg = 0x00; // timer interrupt  }  **else**  {  i\_reg = 0x01;  }  }  **int** **main**(**void**) {  // Initailizing the board  BoardInit();  // Muxing for Enabling GPIO, UART\_TX and UART\_RX.  PinMuxConfig();  //Init SPI  SPIInit();  //Turn off TRF7970A  SPI\_TRF\_CS\_OFF;  // GPIO interrupt setting  // TRF7970 IRQ  GPIOIntInit(GPIOA1\_BASE, GPIO\_PIN\_4, INT\_GPIOA1, Trf7970PortB, GPIO\_RISING\_EDGE, INT\_PRIORITY\_LVL\_1);  // Set Clock Frequency and Modulation  Trf7970InitialSettings();  A2CounterInit(A2CNTIntHandler);  // Init Terminal  **InitTerm**();  **ClearTerm**();  DisplayBanner(APP\_NAME);  //Start the simplelink host  **VStartSimpleLinkSpawnTask**(SPAWN\_TASK\_PRIORITY);  // Start the Keypad task  **osi\_TaskCreate**( SmartDoorlockMenuTask,  (**const** **signed** **char**\*)"MenuTask",  OSI\_STACK\_SIZE, NULL, 1, NULL );  // Start the SmartDoorlock NFC task  **osi\_TaskCreate**( SmartDoorlockNFCTask,  (**const** **signed** **char**\*)"Smart Doorlock NFCTask",  OSI\_STACK\_SIZE, NULL, 1, NULL );  // Start the SmartDoorlock IoT task  **osi\_MsgQCreate**(&g\_PBQueue,"PBQueue",**sizeof**(event\_msg),10);  **osi\_TaskCreate**( SmartDoorlockIoTTask,  (**const** **signed** **char**\*)"Smart Doorlock IoTTask",  OSI\_STACK\_SIZE, NULL, 1, NULL );  **osi\_start**();  **return** 0;  } |

# October 10th, 2016

The 5V/12V dual output power supply & solenoid motor finally came. George, our master technician has helped me solder the ends to the AC receptacle. Testing the 5V line with a multimeter yielded correct voltage, so I’ve went ahead and connected the 5V line to the breadboard. The 12V line however, showed fluctuating voltage between 14.1V to 14.6V. To determine the cause for this, I’ve relooked at the datasheet. It turned out that the 12V line requires a minimum load of 20% for proper operation. The 12V line is rated at 1.3A max current, so a simple calculation shows

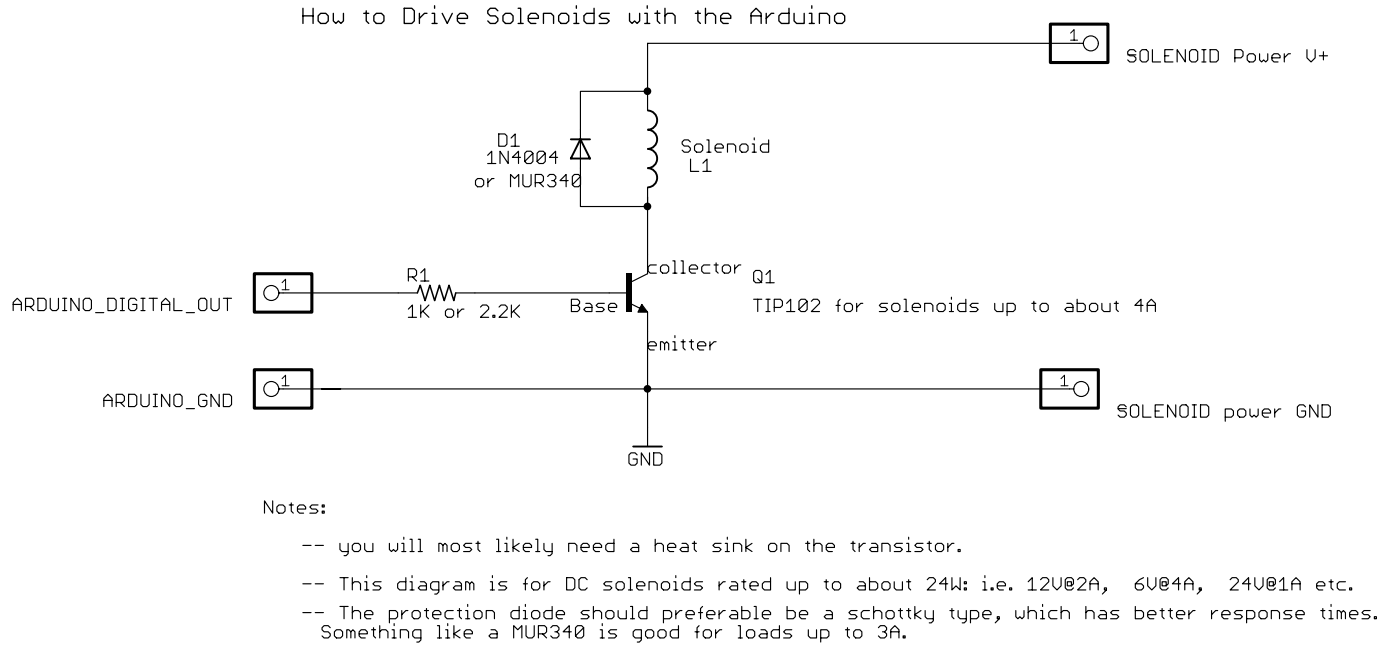
1.3A \* 0.20 = 0.26A minimum load required

or lower value resistor required

or greater watt threshold resistor required

I’m not too happy about the excessive power wasted just to power the solenoid, but I suppose this is the price of buying a really small-sized power supply.

The following circuit diagram, taken from Adafruit’s website was used to quickly test the solenoid.



*Adafruit’s example on how to drive a solenoid with Arduino*

Feeding 5V to the base of the transistor demonstrated that the solenoid works correctly. The solenoid will be connected later when I acquire a 5W, ~40 ohm power resistor.