ORIGINAL PORTABLE SERIAL AND TIMER WRAPPER LIBRARY – C++ ARDUINO CPP FILE

```
/***************************************************************************//**
* @file
* Portable serial and timer wrapper library.
*
* @version @n 1.1
* @date @n 2/7/2013
*
* @authors @n Kwabena W. Agyeman & Christopher J. Leaf
* @copyright @n (c) 2013 Kwabena W. Agyeman & Christopher J. Leaf
* @n All rights reserved - Please see the end of the file for the terms of use
*
* @par Update History:
* @n v0.1 - Beta code - 3/20/2012
* @n v0.9 - Original release - 4/18/2012
* @n v1.0 - Documented and updated release - 8/3/2012
* @n v1.1 - Added support for the Arduino Due, fixed the send frame command,
		and fixed a number of compile time warnings - 2/7/2013.
*******************************************************************************/

#include "CMUcom4.h"

/*******************************************************************************
* Constructor Functions
*******************************************************************************/

CMUcom4::CMUcom4()
{
		_port = CMUCOM4_SERIAL;
}

CMUcom4::CMUcom4(int port)
{
		_port = port;
}

/*******************************************************************************
* Public Functions
*******************************************************************************/

void CMUcom4::begin(unsigned long baud)
{
		delayMilliseconds(CMUCOM4_BEGIN_DELAY);

#if defined(__AVR_ATmega1280__) || \
		defined(__AVR_ATmega2560__) || \
		defined(__SAM3X8E__)
		switch(_port)
		{
		case CMUCOM4_SERIAL1: Serial1.begin(baud); break;
		case CMUCOM4_SERIAL2: Serial2.begin(baud); break;
		case CMUCOM4_SERIAL3: Serial3.begin(baud); break;
		default: Serial.begin(baud); break;
		}
#else
		Serial.begin(baud);
#endif

		delayMilliseconds(CMUCOM4_BEGIN_DELAY);
}

void CMUcom4::end()
{
		delayMilliseconds(CMUCOM4_END_DELAY);
#if defined(__AVR_ATmega1280__) || \
	defined(__AVR_ATmega2560__) || \
	defined(__SAM3X8E__)
	switch(_port)
	{
	case CMUCOM4_SERIAL1: Serial1.end(); break;
	case CMUCOM4_SERIAL2: Serial2.end(); break;
	case CMUCOM4_SERIAL3: Serial3.end(); break;
	default: Serial.end(); break;
	}
#else
	Serial.end();
#endif

	delayMilliseconds(CMUCOM4_END_DELAY);
}

int CMUcom4::read()
{
#if defined(__AVR_ATmega1280__) || \
	defined(__AVR_ATmega2560__) || \
	defined(__SAM3X8E__)
	switch(_port)
	{
	case CMUCOM4_SERIAL1: return Serial1.read(); break;
	case CMUCOM4_SERIAL2: return Serial2.read(); break;
	case CMUCOM4_SERIAL3: return Serial3.read(); break;
	default: return Serial.read(); break;
	}
#else
	return Serial.read();
#endif
}

size_t CMUcom4::write(uint8_t c)
{
#if defined(__AVR_ATmega1280__) || \
	defined(__AVR_ATmega2560__) || \
	defined(__SAM3X8E__)
	switch(_port)
	{
	case CMUCOM4_SERIAL1: return Serial1.write(c); break;
	case CMUCOM4_SERIAL2: return Serial2.write(c); break;
	case CMUCOM4_SERIAL3: return Serial3.write(c); break;
	default: return Serial.write(c); break;
	}
#else
	return Serial.write(c);
#endif
}

size_t CMUcom4::write(const char * str)
{
#if defined(__AVR_ATmega1280__) || \
	defined(__AVR_ATmega2560__) || \
	defined(__SAM3X8E__)
	switch(_port)
	{
	case CMUCOM4_SERIAL1: return Serial1.write(str); break;
	case CMUCOM4_SERIAL2: return Serial2.write(str); break;
	case CMUCOM4_SERIAL3: return Serial3.write(str); break;
	default: return Serial.write(str); break;
	}
#else
	return Serial.write(str);
#endif
}
```

```
size_t CMUcom4::write(const uint8_t * buffer, size_t size)
{
#if defined(__AVR_ATmega1280__) || \
	defined(__AVR_ATmega2560__) || \
	defined(__SAM3X8E__)
	switch(_port)
	{
	case CMUCOM4_SERIAL1: return Serial1.write(buffer, size); break;
	case CMUCOM4_SERIAL2: return Serial2.write(buffer, size); break;
	case CMUCOM4_SERIAL3: return Serial3.write(buffer, size); break;
	default: return Serial.write(buffer, size); break;
	}
#else
	return Serial.write(buffer, size);
#endif
}

int CMUcom4::available()
{
#if defined(__AVR_ATmega1280__) || \
	defined(__AVR_ATmega2560__) || \
	defined(__SAM3X8E__)
	switch(_port)
	{
	case CMUCOM4_SERIAL1: return Serial1.available(); break;
	case CMUCOM4_SERIAL2: return Serial2.available(); break;
	case CMUCOM4_SERIAL3: return Serial3.available(); break;
	default: return Serial.available(); break;
	}
#else
	return Serial.available();
#endif
}

void CMUcom4::flush()
{
#if defined(__AVR_ATmega1280__) || \
	defined(__AVR_ATmega2560__) || \
	defined(__SAM3X8E__)
	switch(_port)
	{
	case CMUCOM4_SERIAL1: Serial1.flush(); break;
	case CMUCOM4_SERIAL2: Serial2.flush(); break;
	case CMUCOM4_SERIAL3: Serial3.flush(); break;
	default: Serial.flush(); break;
	}
#else
	Serial.flush();
#endif
}

int CMUcom4::peek()
{
#if defined(__AVR_ATmega1280__) || \
	defined(__AVR_ATmega2560__) || \
	defined(__SAM3X8E__)
	switch(_port)
	{
	case CMUCOM4_SERIAL1: return Serial1.peek(); break;
	case CMUCOM4_SERIAL2: return Serial2.peek(); break;
	case CMUCOM4_SERIAL3: return Serial3.peek(); break;
	default: return Serial.peek(); break;
	}
#else
	return Serial.peek();
#endif
}
```

```
void CMUcom4::delayMilliseconds(unsigned long ms)
{
		return delay(ms);
}

unsigned long CMUcom4::milliseconds()
{
		return millis();
}

/***************************************************************************//**
* @file
* @par MIT License - TERMS OF USE:
* @n Permission is hereby granted, free of charge, to any person obtaining a
* copy of this software and associated documentation files (the "Software"), to
* deal in the Software without restriction, including without limitation the
* rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
* sell copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
* @n
* @n The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
* @n
* @n THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*******************************************************************************/
```

MODIFIED PORTABLE SERIAL AND TIMER WRAPPER LIBRARY – C++ PIC32 FILE

```
/***************************************************************************//**
* @file
* Portable serial and timer wrapper library.
*
* @version @n 1.1
* @date @n 2/7/2013
*
* @authors @n Kwabena W. Agyeman & Christopher J. Leaf
* @copyright @n (c) 2013 Kwabena W. Agyeman & Christopher J. Leaf
* @n All rights reserved - Please see the end of the file for the terms of use
*
* @par Update History:
* @n v0.1 - Beta code - 3/20/2012
* @n v0.9 - Original release - 4/18/2012
* @n v1.0 - Documented and updated release - 8/3/2012
* @n v1.1 - Added support for the Arduino Due, fixed the send frame command,
	and fixed a number of compile time warnings - 2/7/2013.
*******************************************************************************/

#include "CMUcom4.h"
#include <stdbool.h>
#include <xc.h>
#include <stdlib.h>
#include <stdio.h>



/*******************************************************************************
* Constructor Functions
*******************************************************************************/
```

```
CMUcom4::CMUcom4()
{
  _port = CMUCOM4_SERIAL;
}

CMUcom4::CMUcom4(int port)
{
  _port = port;
}

/*********************************************************************************
* Public Functions
*********************************************************************************/

void begin(unsigned long baud)
{
   //code need to open the PICs serial port to the CMUcam4
          U3MODEbits.BRGH=1;
          U3BRG = 128; // Set Baud rate
          U3MODEbits.PDSEL=0;
          U3MODEbits.STSEL=0;
          U3STAbits.UTXEN=1;
          U3STAbits.URXEN=1;
          U3MODEbits.ON=1;
}



void end()
{
  //code need to close the PIC?s serial port to the CMUcam4
}



int read()
{
          while (U3STAbits.URXDA == 0)
          {
          }

          return(U3RXREG);

  //code needed to read a byte from the PIC?s serial port... should return ?1 if no byte
}



int write(uint8_t c) {
          while (U3STAbits.UTXBF == 1){
        TXSTAbits.TXEN=0;// disable transmission
        TXREG=c;         // load txreg with data
         TXSTAbits.TXEN=1;    // enable transmission
 while(TXSTAbits.TRMT==0) // wait here till transmit complete
 {
  Nop();
 }
          }

          U3TXREG = c;

  //code needed to write a character to PIC32
}
```

```
int write(const char * str)
{

    //code needed to write a string to PIC32



}

void write(const uint8_t * buffer, size_t size)
{

                   //code needed to write a buffer to PIC32

    void SendD(const char *buffer, UINT32 size)

    while(size)
    {
        while(!UARTTransmitterIsReady(UART_MODULE_ID))
            ;

        UARTSendDataByte(UART_MODULE_ID, *buffer);

        buffer++;
        size--;
    }

    while(!UARTTransmissionHasCompleted(UART_MODULE_ID))
        ;

}

/*int available()
{
    int incomingByte = 0;              // for incoming serial data

    void setup() {

                   void begin();              // opens serial port
        }

    void loop() {

                   // send data only when you receive data:
                   if (available() > 0) {
                                      // read the incoming byte:
                                      incomingByte = int read();


                   }
} */



 //Get the number of bytes (characters) available for reading from the serial port.
 //This is data that's already arrived and stored in the serial receive buffer
 //(which holds 64 bytes). available() inherits from the Stream utility class.

       //need to check the available function on PIC32 and C++
//}

//int flush()
//{
    //fflush(port)
    // code to waits for the transmission of outgoing serial data to complete.
//}

//int peek()
//{

   // x = peek(a)



  //return 0;


      //Returns the next byte (character) of incoming serial data without
                   //removing it from the internal serial buffer. That is, successive
                   //calls to peek() will return the same character, as will the next
                   //call to read(). peek() inherits from the Stream utility class.
//}

/*void delayMilliseconds(unsigned long ms)
{
    //create a sleep function for the camera
    return delay(ms);
}

unsigned long milliseconds()
{
    //create a timer in millseconds
    return millis();
}
*/

/***************************************************************************//**
* @file
* @par MIT License - TERMS OF USE:
* @n Permission is hereby granted, free of charge, to any person obtaining a
* copy of this software and associated documentation files (the "Software"), to
* deal in the Software without restriction, including without limitation the
* rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
* sell copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
* @n
* @n The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
* @n
* @n THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*******************************************************************************/
```

ORIGINAL PORTABLE SERIAL AND TIMER WRAPPER LIBRARY – C++ ARDUINO HEADER FILE

```
/***************************************************************************//**
* @file
* Portable serial and timer wrapper library.
*
* @version @n 1.1
* @date @n 2/7/2013
*
* @authors @n Kwabena W. Agyeman & Christopher J. Leaf
* @copyright @n (c) 2013 Kwabena W. Agyeman & Christopher J. Leaf
* @n All rights reserved - Please see the end of the file for the terms of use
*
* @par Update History:
* @n v0.1 - Beta code - 3/20/2012
* @n v0.9 - Original release - 4/18/2012
* @n v1.0 - Documented and updated release - 8/3/2012
* @n v1.1 - Added support for the Arduino Due, fixed the send frame command,
            and fixed a number of compile time warnings - 2/7/2013.
*******************************************************************************/

#ifndef _CMUCOM4_H_
#define _CMUCOM4_H_

/**@cond CMUCOM4_PRIVATE*******************************************************/

// Handle Arduino Library renaming.
#if defined(ARDUINO) && (ARDUINO >= 100)
#include "Arduino.h"
#else
#include "WProgram.h"
#endif

// Try to save RAM for non-Mega boards.
#if defined(__AVR_ATmega1280__) || \
            defined(__AVR_ATmega2560__) || \
            defined(__SAM3X8E__)
#define CMUCOM4_INPUT_BUFFER_SIZE   256 ///< Responce input buffer size.
#define CMUCOM4_OUTPUT_BUFFER_SIZE  256 ///< Command output buffer size.
#else
#define CMUCOM4_INPUT_BUFFER_SIZE   160 ///< Responce input buffer size.
#define CMUCOM4_OUTPUT_BUFFER_SIZE  96 ///< Command output buffer size.
#endif

/***************************************************************************//**
* This function macro expands whatever argument name that was passed to this
* function macro into a string. @par For example:
* <tt>@#define ARDUINO 100</tt> @n
* <tt>%CMUCOM4_N_TO_S(ARDUINO)</tt> exapands to @c "ARDUINO"
*******************************************************************************/
#define CMUCOM4_N_TO_S(x)           #x

/***************************************************************************//**
* This function macro expands whatever argument value that was passed to this
* function macro into a string. @par For example:
* <tt>@#define ARDUINO 100</tt> @n
* <tt>%CMUCOM4_V_TO_S(ARDUINO)</tt> exapands to @c "100"
*******************************************************************************/
#define CMUCOM4_V_TO_S(x)      CMUCOM4_N_TO_S(x)

/***************************************************************************//**
* Default firmware startup baud rate number.
*******************************************************************************/
#define CMUCOM4_SLOW_BAUD_RATE          19200

/***************************************************************************//**
* Default firmware startup baud rate string.
*******************************************************************************/
#define CMUCOM4_SLOW_BR_STRING     CMUCOM4_V_TO_S(CMUCOM4_SLOW_BAUD_RATE)

/***************************************************************************//**
* Version 1.01 firmware and below maximum baud rate number.
*******************************************************************************/
#define CMUCOM4_MEDIUM_BAUD_RATE        115200

/***************************************************************************//**
* Version 1.01 firmware and below maximum baud rate string.
```

```
*******************************************************************************/
#define CMUCOM4_MEDIUM_BR_STRING    CMUCOM4_V_TO_S(CMUCOM4_MEDIUM_BAUD_RATE)

/***************************************************************************//**
* Version 1.02 firmware and above maximum baud rate number.
*******************************************************************************/
#define CMUCOM4_FAST_BAUD_RATE                  250000

/***************************************************************************//**
* Version 1.02 firmware and above maximum baud rate string.
*******************************************************************************/
#define CMUCOM4_FAST_BR_STRING      CMUCOM4_V_TO_S(CMUCOM4_FAST_BAUD_RATE)

/***************************************************************************//**
* Default firmware startup stop bits number.
*******************************************************************************/
#define CMUCOM4_SLOW_STOP_BITS   0

/***************************************************************************//**
* Default firmware startup stop bits string.
*******************************************************************************/
#define CMUCOM4_SLOW_SB_STRING      CMUCOM4_V_TO_S(CMUCOM4_SLOW_STOP_BITS)

/***************************************************************************//**
* Version 1.01 firmware and below necessary stop bits number.
*******************************************************************************/
#define CMUCOM4_MEDIUM_STOP_BITS              0

/***************************************************************************//**
* Version 1.01 firmware and below necessary stop bits string.
*******************************************************************************/
#define CMUCOM4_MEDIUM_SB_STRING    CMUCOM4_V_TO_S(CMUCOM4_MEDIUM_STOP_BITS)

/***************************************************************************//**
* Version 1.02 firmware and above necessary stop bits number.
*******************************************************************************/
#define CMUCOM4_FAST_STOP_BITS    0

/***************************************************************************//**
* Version 1.02 firmware and above necessary stop bits string.
*******************************************************************************/
#define CMUCOM4_FAST_SB_STRING      CMUCOM4_V_TO_S(CMUCOM4_FAST_STOP_BITS)

/***************************************************************************//**
* Serial CMUcom4::begin() post delay in milliseconds.
*******************************************************************************/
#define CMUCOM4_BEGIN_DELAY        1

/***************************************************************************//**
* Serial CMUcom4::end() post delay in milliseconds.
*******************************************************************************/
#define CMUCOM4_END_DELAY          1

/**@endcond********************************************************************/

/***************************************************************************//**
* This is a convenient macro for specifying the Serial port when initializing a
* CMUcam4 or CMUcom4 object.
*******************************************************************************/
#define CMUCOM4_SERIAL                     0

/***************************************************************************//**
* This is a convenient macro for specifying the Serial1 port on an Arduino Mega
* when initializing a CMUcam4 or CMUcom4 object.
*******************************************************************************/
```

```
#define CMUCOM4_SERIAL1                 1

/***************************************************************************//**
* This is a convenient macro for specifying the Serial2 port on an Arduino Mega
* when initializing a CMUcam4 or CMUcom4 object.
*******************************************************************************/
#define CMUCOM4_SERIAL2         2

/***************************************************************************//**
* This is a convenient macro for specifying the Serial3 port on an Arduino Mega
* when initializing a CMUcam4 or CMUcom4 object.
*******************************************************************************/
#define CMUCOM4_SERIAL3                 3

/***************************************************************************//**
* This is a hardware abstraction layer for the %CMUcam4 class. The %CMUcom4
* class targets the Ardunio prototyping platform by default.
*******************************************************************************/
class CMUcom4
{

public:

/***************************************************************************//**
* Initialize the %CMUcom4 object to use the default Serial port.
*******************************************************************************/
CMUcom4();

/***************************************************************************//**
* Initialize the %CMUcom4 object to use the @c port Serial port.
* @param [in] port The port.
* @see CMUCOM4_SERIAL
* @see CMUCOM4_SERIAL1
* @see CMUCOM4_SERIAL2
* @see CMUCOM4_SERIAL3
*******************************************************************************/
CMUcom4(int port);

/***************************************************************************//**
* Arduino Serial.begin() wrapper.
* @param [in] baud In bits per second.
* @see http://arduino.cc/en/Serial/Begin
*******************************************************************************/
void begin(unsigned long baud);

/***************************************************************************//**
* Arduino Serial.end() wrapper.
* @see http://arduino.cc/en/Serial/End
*******************************************************************************/
void end();

/***************************************************************************//**
* Arduino Serial.read() wrapper.
* @return The first byte of incoming serial data.
* @see http://arduino.cc/en/Serial/Read
*******************************************************************************/
int read();

/***************************************************************************//**
* Arduino Serial.write() wrapper.
* @param [in] buffer An array to send as a series of bytes.
* @param [in] size The size of the buffer.
* @return The number of bytes written.
* @see http://arduino.cc/en/Serial/Write
*******************************************************************************/
size_t write(const uint8_t * buffer, size_t size);
/******************************************************************************//**
* Arduino Serial.write() wrapper.
* @param [in] str A string to send as a series of bytes.
* @return The number of bytes written.
* @see http://arduino.cc/en/Serial/Write
*********************************************************************************/
size_t write(const char * str);

/******************************************************************************//**
* Arduino Serial.write() wrapper.
* @param [in] c A character to send as a single byte.
* @return The number of bytes written.
* @see http://arduino.cc/en/Serial/Write
*********************************************************************************/
size_t write(uint8_t c);

/******************************************************************************//**
* Arduino Serial.available() wrapper.
* @return The number of bytes available to be read.
* @see http://arduino.cc/en/Serial/Available
*********************************************************************************/
int available();

/******************************************************************************//**
* Arduino Serial.flush() wrapper.
* @see http://arduino.cc/en/Serial/Flush
*********************************************************************************/
void flush();

/******************************************************************************//**
* Arduino Serial.peek() wrapper.
* @return The first byte of incoming serial data available.
* @see http://arduino.cc/en/Serial/Peek
*********************************************************************************/
int peek();

/******************************************************************************//**
* Arduino delay() wrapper.
* @param [in] ms The number of milliseconds to pause for.
* @see http://arduino.cc/en/Reference/Delay
*********************************************************************************/
void delayMilliseconds(unsigned long ms);

/******************************************************************************//**
* Arduino millis() wrapper.
* @return Number of milliseconds since the program started.
* @see http://arduino.cc/en/Reference/Millis
*********************************************************************************/
unsigned long milliseconds();

private:

/******************************************************************************//**
* Selected serial port storage.
* @see CMUCOM4_SERIAL1
* @see CMUCOM4_SERIAL2
* @see CMUCOM4_SERIAL3
*********************************************************************************/
int _port;
};

#endif

/******************************************************************************//**
* @file
* @par MIT License - TERMS OF USE:
* @n Permission is hereby granted, free of charge, to any person obtaining a
* copy of this software and associated documentation files (the "Software"), to
* deal in the Software without restriction, including without limitation the
* rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
* sell copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
* @n
* @n The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
* @n
* @n THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*******************************************************************************/
```

MODIFIED PORTABLE SERIAL AND TIMER WRAPPER LIBRARY – PIC32 HEADER FILE

```
/*****************************************************************************//**
* @file
* Portable serial and timer wrapper library.
*
* @version @n 1.1
* @date @n 2/7/2013
*
* @authors @n Kwabena W. Agyeman & Christopher J. Leaf
* @copyright @n (c) 2013 Kwabena W. Agyeman & Christopher J. Leaf
* @n All rights reserved - Please see the end of the file for the terms of use
*
* @par Update History:
* @n v0.1 - Beta code - 3/20/2012
* @n v0.9 - Original release - 4/18/2012
* @n v1.0 - Documented and updated release - 8/3/2012
* @n v1.1 - Added support for the Arduino Due, fixed the send frame command,
	and fixed a number of compile time warnings - 2/7/2013.
*******************************************************************************/
#include <plib.h>



#ifndef _CMUCOM4_H_
#define _CMUCOM4_H_

#if defined (__32MX695F512H__)

// Configuration Bit settings
// SYSCLK = 80 MHz (8MHz Crystal / FPLLIDIV * FPLLMUL / FPLLODIV)
// PBCLK = 80 MHz (SYSCLK / FPBDIV)



/**@cond CMUCOM4_PRIVATE***************************************************/




#define CMUCOM4_INPUT_BUFFER_SIZE   256 ///< Responce input buffer size.
#define CMUCOM4_OUTPUT_BUFFER_SIZE  256 ///< Command output buffer size.
//#else
//#define CMUCOM4_INPUT_BUFFER_SIZE   160 ///< Responce input buffer size.
//#define CMUCOM4_OUTPUT_BUFFER_SIZE  96 ///< Command output buffer size.
//#endif
```

```
/***************************************************************************//**
* This function macro expands whatever argument name that was passed to this
* function macro into a string. @par For example:
* <tt>@#define ARDUINO 100</tt> @n
* <tt>%CMUCOM4_N_TO_S(ARDUINO)</tt> exapands to @c "ARDUINO"
*******************************************************************************/
#define CMUCOM4_N_TO_S(x)          #x

/***************************************************************************//**
* This function macro expands whatever argument value that was passed to this
* function macro into a string. @par For example:
* <tt>@#define ARDUINO 100</tt> @n
* <tt>%CMUCOM4_V_TO_S(ARDUINO)</tt> exapands to @c "100"
*******************************************************************************/
#define CMUCOM4_V_TO_S(x)          CMUCOM4_N_TO_S(x)

/***************************************************************************//**
* Default firmware startup baud rate number.
*******************************************************************************/
#define CMUCOM4_SLOW_BAUD_RATE      19200

/***************************************************************************//**
* Default firmware startup baud rate string.
*******************************************************************************/
#define CMUCOM4_SLOW_BR_STRING      CMUCOM4_V_TO_S(CMUCOM4_SLOW_BAUD_RATE)

/***************************************************************************//**
* Version 1.01 firmware and below maximum baud rate number.
*******************************************************************************/
#define CMUCOM4_MEDIUM_BAUD_RATE    115200

/***************************************************************************//**
* Version 1.01 firmware and below maximum baud rate string.
*******************************************************************************/
#define CMUCOM4_MEDIUM_BR_STRING    CMUCOM4_V_TO_S(CMUCOM4_MEDIUM_BAUD_RATE)

/***************************************************************************//**
* Version 1.02 firmware and above maximum baud rate number.
*******************************************************************************/
#define CMUCOM4_FAST_BAUD_RATE      250000

/***************************************************************************//**
* Version 1.02 firmware and above maximum baud rate string.
*******************************************************************************/
#define CMUCOM4_FAST_BR_STRING      CMUCOM4_V_TO_S(CMUCOM4_FAST_BAUD_RATE)

/***************************************************************************//**
* Default firmware startup stop bits number.
*******************************************************************************/
#define CMUCOM4_SLOW_STOP_BITS      0

/***************************************************************************//**
* Default firmware startup stop bits string.
*******************************************************************************/
#define CMUCOM4_SLOW_SB_STRING      CMUCOM4_V_TO_S(CMUCOM4_SLOW_STOP_BITS)

/***************************************************************************//**
* Version 1.01 firmware and below necessary stop bits number.
*******************************************************************************/
#define CMUCOM4_MEDIUM_STOP_BITS    0

/***************************************************************************//**
* Version 1.01 firmware and below necessary stop bits string.
*******************************************************************************/
#define CMUCOM4_MEDIUM_SB_STRING    CMUCOM4_V_TO_S(CMUCOM4_MEDIUM_STOP_BITS)
```

```
/***************************************************************************//**
* Version 1.02 firmware and above necessary stop bits number.
*******************************************************************************/
#define CMUCOM4_FAST_STOP_BITS      0

/***************************************************************************//**
* Version 1.02 firmware and above necessary stop bits string.
*******************************************************************************/
#define CMUCOM4_FAST_SB_STRING      CMUCOM4_V_TO_S(CMUCOM4_FAST_STOP_BITS)

/***************************************************************************//**
* Serial CMUcom4::begin() post delay in milliseconds.
*******************************************************************************/
#define CMUCOM4_BEGIN_DELAY         1

/***************************************************************************//**
* Serial CMUcom4::end() post delay in milliseconds.
*******************************************************************************/
#define CMUCOM4_END_DELAY           1

/**@endcond********************************************************************/

/***************************************************************************//**
* This is a convenient macro for specifying the Serial port when initializing a
* CMUcam4 or CMUcom4 object.
*******************************************************************************/
#define CMUCOM4_SERIAL              0

/***************************************************************************//**
* This is a convenient macro for specifying the Serial1 port on an Arduino Mega
* when initializing a CMUcam4 or CMUcom4 object.
*******************************************************************************/
#define CMUCOM4_SERIAL1             1

/***************************************************************************//**
* This is a convenient macro for specifying the Serial2 port on an Arduino Mega
* when initializing a CMUcam4 or CMUcom4 object.
*******************************************************************************/
#define CMUCOM4_SERIAL2             2

/***************************************************************************//**
* This is a convenient macro for specifying the Serial3 port on an Arduino Mega
* when initializing a CMUcam4 or CMUcom4 object.
*******************************************************************************/
#define CMUCOM4_SERIAL3             3

/***************************************************************************//**
* This is a hardware abstraction layer for the %CMUcam4 class. The %CMUcom4
* class targets the Ardunio prototyping platform by default.
*******************************************************************************/
class CMUcom4
{

public:

/***************************************************************************//**
* Initialize the %CMUcom4 object to use the default Serial port.
*******************************************************************************/
CMUcom4();

/***************************************************************************//**
* Initialize the %CMUcom4 object to use the @c port Serial port.
* @param [in] port The port.
* @see CMUCOM4_SERIAL
* @see CMUCOM4_SERIAL1
* @see CMUCOM4_SERIAL2
* @see CMUCOM4_SERIAL3
*****************************************************************************/
CMUcom4(int port);

/**************************************************************************//**
* Arduino Serial.begin() wrapper.
* @param [in] baud In bits per second.
* @see http://arduino.cc/en/Serial/Begin
*****************************************************************************/
void begin(unsigned long baud);

/**************************************************************************//**
* Arduino Serial.end() wrapper.
* @see http://arduino.cc/en/Serial/End
*****************************************************************************/
void end(void);

/**************************************************************************//**
* Arduino Serial.read() wrapper.
* @return The first byte of incoming serial data.
* @see http://arduino.cc/en/Serial/Read
*****************************************************************************/
int read(void);

/**************************************************************************//**
* Arduino Serial.write() wrapper.
* @param [in] buffer An array to send as a series of bytes.
* @param [in] size The size of the buffer.
* @return The number of bytes written.
* @see http://arduino.cc/en/Serial/Write
*****************************************************************************/


int write(const uint8_t *, size_t);

/**************************************************************************//**
* Arduino Serial.write() wrapper.
* @param [in] str A string to send as a series of bytes.
* @return The number of bytes written.
* @see http://arduino.cc/en/Serial/Write
*****************************************************************************/
int write(const char * str);

/**************************************************************************//**
* Arduino Serial.write() wrapper.
* @param [in] c A character to send as a single byte.
* @return The number of bytes written.
* @see http://arduino.cc/en/Serial/Write
*****************************************************************************/


size_t write(uint8_t c);

/**************************************************************************//**
* Arduino Serial.available() wrapper.
* @return The number of bytes available to be read.
* @see http://arduino.cc/en/Serial/Available
*****************************************************************************/
int available();

/**************************************************************************//**
* Arduino Serial.flush() wrapper.
* @see http://arduino.cc/en/Serial/Flush
*****************************************************************************/
//void flush();
```

```
/******************************************************************************//**
* Arduino Serial.peek() wrapper.
* @return The first byte of incoming serial data available.
* @see http://arduino.cc/en/Serial/Peek
*********************************************************************************/
//int peek();

/******************************************************************************//**
* Arduino delay() wrapper.
* @param [in] ms The number of milliseconds to pause for.
* @see http://arduino.cc/en/Reference/Delay
*********************************************************************************/
//void delayMilliseconds(unsigned long ms);

/******************************************************************************//**
* Arduino millis() wrapper.
* @return Number of milliseconds since the program started.
* @see http://arduino.cc/en/Reference/Millis
*********************************************************************************/
//unsigned long milliseconds();

private:

/******************************************************************************//**
* Selected serial port storage.
* @see CMUCOM4_SERIAL1
* @see CMUCOM4_SERIAL2
* @see CMUCOM4_SERIAL3
*********************************************************************************/
int _port;

};

#endif

/******************************************************************************//**
* @file
* @par MIT License - TERMS OF USE:
* @n Permission is hereby granted, free of charge, to any person obtaining a
* copy of this software and associated documentation files (the "Software"), to
* deal in the Software without restriction, including without limitation the
* rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
* sell copies of the Software, and to permit persons to whom the Software is
* furnished to do so, subject to the following conditions:
* @n
* @n The above copyright notice and this permission notice shall be included in
* all copies or substantial portions of the Software.
* @n
* @n THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
********************************************************************************
```

UART PIC32 MAIN FILE

```
/*
 * File:   Assignment8.c
 * Author: nferruol
 *
 * Created on November 26, 2012, 4:33 PM
 */
```

```
#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <sys/attribs.h>
#include <plib.h>
#include "configbits.h"
#include "CMUcom4.h"

/*
char uart3Read(void)
{
  while (U3STAbits.URXDA == 0)
  {
  }

  return(U3RXREG);
}*/

/*void uart3Write(char a)
{
   while (U3STAbits.UTXBF == 1)
   {
   }

   U3TXREG = a;
 }*/

/*void _mon_putc(char c){
    while (U3STAbits.UTXBF);
    U3TXREG = c;
} */

int main(void)
{
        uint8_t c;

        uart_init(BAUD_RATE);
        while (1) {
                if (uart_available()) {
                        c = uart_getchar();
                        uart_print("Byte: ");
                        uart_putchar(c);
                        uart_putchar('\r');
                        uart_putchar('\n');
                }
        }
}
```

Matlab DetectRed.m function

```
 %% Now to track red objects in real time
    % we have to subtract the red component
    % from the grayscale image to extract the red components in the image.
    diff_im = imsubtract(data(:,:,1), rgb2gray(data));
    %Use a median filter to filter out noise
    diff_im = medfilt2(diff_im, [3 3]);
    % Convert the resulting grayscale image into a binary image.
    diff_im = im2bw(diff_im,0.19);
    % Remove all those pixels less than 300px
    diff_im = bwareaopen(diff_im,300);
    % Label all the connected components in the image.
    bw = bwlabel(diff_im, 8);

    % Here we do the image blob analysis.
    % We get a set of properties for each labeled region.
    stats = regionprops(bw, 'BoundingBox', 'Centroid');

    %% Now to track green objects in real time
    % we have to subtract the red component
    % from the grayscale image to extract the red components in the image.
    diff_im2 = imsubtract(data(:,:,2), rgb2gray(data));
    %Use a median filter to filter out noise
    diff_im2 = medfilt2(diff_im2, [3 3]);
    % Convert the resulting grayscale image into a binary image.
    diff_im2 = im2bw(diff_im2,0.05);
    % Remove all those pixels less than 300px
    diff_im2 = bwareaopen(diff_im2,300);
    % Label all the connected components in the image.
    bw2 = bwlabel(diff_im2, 8);

    % Here we do the image blob analysis.
    % We get a set of properties for each labeled region.
    stats2 = regionprops(bw2, 'BoundingBox', 'Centroid');

    %% Display the image
    coder.extrinsic('imshow','hold on','rectangle','plot','text','set','hold off');

    imshow(data)

    hold on

    %This is a loop to bound the red objects in a rectangular box.
    for object = 1:length(stats)
        %for red
        bb = stats(object).BoundingBox;
        bc = stats(object).Centroid;
        rectangle('Position',bb,'EdgeColor','r','LineWidth',2)
        plot(bc(1),bc(2), '-m+')
        a=text(bc(1)+15,bc(2), strcat('X: ', num2str(round(bc(1))), '    Y: ', num2str(round(bc(2)))));
        set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
    end

    for object = 1:length(stats2)
        %for green
        bb2 = stats2(object).BoundingBox;
        bc2 = stats2(object).Centroid;
        rectangle('Position',bb2,'EdgeColor','g','LineWidth',2)
        plot(bc2(1),bc2(2), '-m+')
        a=text(bc2(1)+15,bc2(2), strcat('X: ', num2str(round(bc2(1))), '    Y: ', num2str(round(bc2(2)))));
        set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12, 'Color', 'yellow');
    end

    hold off
```

Matlab RedTrackObject.m Test Program

```
a = imaqhwinfo;
allStats = struct([]);
%[camera_name, camera_id, format] = getCameraInfo(a);



% Capture the video frames using the videoinput function
% You have to replace the resolution & your installed adaptor name.
%vid = videoinput(camera_name, camera_id, format);
vid = videoinput('winvideo',2,'YUY2_1024x768');
%vid = videoinput('matrox',,'Port_#0002.Hub_#0004');
```

```
% Set the properties of the video object
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 8;



%start the video aquisition here
start(vid)

flushdata(vid);

%call function that process data frame by frame

while(vid.FramesAcquired<=400)

    % Get the snapshot of the current frame
    data = getsnapshot(vid);

    [diff_im,bw,stats,diff_im2,bw2,stats2]=DetectRed(data);

    % Create a construct that stores all stats

    allStats([vid.FramesAcquired]).red = stats;
    allStats([vid.FramesAcquired]).green = stats2;

end
% Stop the video aquisition.
stop(vid);

% Flush all the image data stored in the memory buffer.
flushdata(vid);

% Clear all variables
%clear all
sprintf('%s','That was all about Image tracking :) ')
```

Microcontroller initialization configbits_1.h

```
/*
 * File:   configbits_1.h
 * Author: Mike
 *
 * Created on October 9, 2012, 1:50 PM
 */

#ifndef CONFIGBITS_H
#define      CONFIGBITS_H

/* 20 MHz crystal run at 80 mhz

 peripher clock = at 10 MHz (80 MHz/8)

 */

#pragma config FNOSC = FRCPLL // Oscillator selection
#pragma config POSCMOD = OFF // Primary oscillator mode
#pragma config FPLLIDIV = DIV_5 // PLL input divider (20 -> 4)
#pragma config FPLLMUL = MUL_20 // PLL multiplier  ( 4x20 = 80)
#pragma config FPLLODIV = DIV_1 // PLL output divider
```

```
#pragma config FPBDIV = DIV_8 // Peripheral bus clock divider 10 mhz
#pragma config FSOSCEN = OFF // Secondary oscillator enable
/* Clock control settings
*/
#pragma config IESO = OFF // Internal/external clock switchover
#pragma config FCKSM = CSDCMD // Clock switching (CSx)/Clock monitor (CMx)
#pragma config OSCIOFNC = OFF // Clock output on OSCO pin enable
/* USB Settings
*/
#pragma config UPLLEN = ON // USB PLL enable
#pragma config UPLLIDIV = DIV_2 // USB PLL input divider
#pragma config FVBUSONIO = OFF // VBUS pin control
#pragma config FUSBIDIO = OFF // USBID pin control
/* Other Peripheral Device settings
*/
#pragma config FWDTEN = OFF // Watchdog timer enable
#pragma config WDTPS = PS1024 // Watchdog timer post-scaler
#pragma config FSRSSEL = PRIORITY_7 // SRS interrupt priority

#pragma config     ICESEL   = ICS_PGx1        // ICE pin selection
#endif    /* CONFIGBITS_H */
```

Zigbee_1.h

```
/*
 * File:   Zigbee_1.h
 * Author: nferruol
 *
 *
 */

#ifndef ZIGBEE_H
#define   ZIGBEE_H

#ifdef     __cplusplus
extern "C" {
#endif




//Define Registers
#define TRX_STATUS         0x01
#define TRX_STATE          0x02
#define TRX_CTRL_0         0x03
#define PHY_TX_PWR         0x05
#define PHY_RSSI           0x06
#define PHY_ED_LEVEL       0x07
#define PHY_CC_CCA         0x08
#define PHY_CCA_THRES      0x09
#define IRQ_MASK           0x0E
#define IRQ_STATUS         0x0F
#define VREG_CTRL          0x10
#define BATMON             0x11
#define XOSC_CTRL          0x12
#define PLL_CF             0x1A
#define PLL_DCU            0x1B
#define PART_NUM           0x1C
#define VERSION_NUM        0x1D
#define MAN_ID_0           0x1E
#define MAN_ID_1           0x1F
#define SHORT_ADDR_0       0x20
#define SHORT_ADDR_1       0x21
#define PAN_ID_0           0x22
#define PAN_ID_1           0x23
#define IEEE_ADDR_0        0x24
#define IEEE_ADDR_1        0x25
#define IEEE_ADDR_2        0x26
#define IEEE_ADDR_3          0x27
#define IEEE_ADDR_4          0x28
#define IEEE_ADDR_5          0x29
#define IEEE_ADDR_6          0x2A
#define IEEE_ADDR_7          0x2B
#define XAH_CTRL            0x2C
#define CSMA_SEED_0          0x2D
#define CSMA_SEED_1          0x2E

//States written to TRX_STATE
#define TX_START            0x02
#define FORCE_TRX_OFF        0x03
#define RX_ON               0x06
#define TRX_OFF              0x08
#define PLL_ON              0x09

//States read out of TRX_STATUS
#define P_ON                0x00
#define BUSY_RX              0x01
#define BUSY_TX              0x02
#define RX_ON               0x06
#define TRX_OFF              0x08
#define PLL_ON              0x09
#define SLEEP               0x0F
#define STATE_TRANSITION     0x1F

//Define pins
#define sel                 LATFbits.LATF5
#define rst                 LATBbits.LATB5
#define slp                 LATBbits.LATB4

 //Function Prototypes
char check();
void send(char);
void SPI_initialize();
void UART_initialize();
void final(int);
void ZIGBEE_initialize();
void ZIGBEE_pll();
void ZIGBEE_transmit(int, int,int, int,int, int,int, int,int);
void ZIGBEE_receive();
void ZIGBEE_write_register (int, int);
int ZIGBEE_read_register (int);
void ZIGBEE_write_buffer (int, int,int, int,int, int,int, int,int);
void ZIGBEE_read_buffer();
int ZIGBEE_check_interrupt();




#ifdef        __cplusplus
}
#endif

#endif        /* ZIGBEE_H */
```

ZIGBEE_FUNCTIONS.c

```
/*
 * File:   ZIGBEE_FUNCTIONS.c
 * Author: nferruol
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
```

```
#include <plib.h>
#include <sys/attribs.h>
#include "Zigbee_1.h"

void UART_initialize (){
    //Set all bits to digital
    AD1PCFG = 0xFFFF;

    U4MODEbits.BRGH=1;
    U4BRG = 42; // Set Baud rate 9600= 259 57600=42
    U4MODEbits.PDSEL=0;
    U4MODEbits.STSEL=0;
    U4STAbits.UTXEN=1;
    U4STAbits.URXEN=1;
    U4MODEbits.ON=1;
    U4MODEbits.UARTEN = 0x01;

}

char check(void)
{
  while (U4STAbits.URXDA == 0)
  {
  }

  return(U4RXREG);

}

void send(char a)
{
   while (U4STAbits.UTXBF == 1)
   {
   }

   U4TXREG = a;

 }
 void _mon_putc (char c)
 {   while (U4STAbits.UTXBF);
     U4TXREG = c;
 }

void SPI_initialize (){

    //set rst, slp, sel to outputs
    TRISFbits.TRISF5 = 0;       //sel
    TRISBbits.TRISB4 = 0;       //slp
    TRISBbits.TRISB5 = 0;       //rst

    //set rst, sel high  set slp low
    rst = 1;
    sel = 1;
    slp = 0;

    //disable interrupts
    IEC1bits.SPI2RXIE = 0;
    IEC1bits.SPI2EIE  = 0;
    IEC1bits.SPI2TXIE = 0;

    //stops and resets SPI2
    SPI2CONbits.ON = 0;

    //clears the buffer
    int clear = SPI2BUF;

    //standard mode
    SPI2CONbits.ENHBUF = 0;

    //use FPB/4 clock frequency Fsck = FPB/(2*(SPIxBRG+1))    FPB = 40MHz
    SPI2BRG = 42;



    //Turn SPI2 on, 8 bits transfer, master mode
    SPI2CONbits.MODE32 = 0;
    SPI2CONbits.MODE16 = 0;
    SPI2CONbits.MSTEN = 1;      //Master mode
    SPI2CONbits.MSSEN = 0;      //Manually do slave select
    SPI2CONbits.SSEN = 0;       //Not slave mode
    SPI2CONbits.CKP = 0;        //Low idle, high active
    SPI2CONbits.CKE = 1;        //Change output on falling edge
    SPI2CONbits.SMP = 0;        //Sample MISO at middle of clock pulse
    SPI2CONbits.DISSDO = 0;     //MISO is enabled
    //Must be final command
    SPI2CONbits.ON = 1;
    //Ready to transmit or receive via SPI2BUF
}




void ZIGBEE_initialize(){
    rst = 0;
    int k;
    for  (k =0; k<100000;k++)
    {}
    rst = 1;

    //ZIGBEE will start up in P_ON state or has been reset

    //Set up IF from ZIGBEE
    TRISDbits.TRISD8 = 1;

    //Put in TRX_OFF state (clock state)
    ZIGBEE_write_register(TRX_STATE, TRX_OFF);

    //Set channel with PHY_CC_CCA channel #11, 2405MHz 0B on bits 4:0
    ZIGBEE_write_register(PHY_CC_CCA, 0x2B);

    //Enable pertinant interupts TRX_END, PLL_LOCK
    ZIGBEE_write_register(IRQ_MASK, 0b00001001);

    //Turns on automatic FCS appending
    ZIGBEE_write_register(PHY_TX_PWR, 0xC6);

    //Set short address to 0123
    ZIGBEE_write_register(SHORT_ADDR_1, 0x01);
    ZIGBEE_write_register(SHORT_ADDR_0, 0x23);

    //Set pan id to 4567
    ZIGBEE_write_register(PAN_ID_1, 0x45);
    ZIGBEE_write_register(PAN_ID_0, 0x67);
}



void ZIGBEE_pll(){
    //Put in PLL_ON state (ready state)
    ZIGBEE_write_register(TRX_STATE, PLL_ON);
    int k;
    for  (k =0; k<1000;k++)
    {}

    return;
}

void ZIGBEE_transmit(int message1, int message2, int message3, int message4, int message5, int message6, int message7, int message8, int
message9)
{
    //Write info to be sent to buffer
    ZIGBEE_write_buffer(message1, message2, message3,message4,message5,message6,message7,message8,message9);

    //Put in Transmit state
    ZIGBEE_write_register(TRX_STATE, TX_START);

    //Return to PLL_ON
    ZIGBEE_pll();
}

void ZIGBEE_receive (){
    //Put in Receive state
    ZIGBEE_write_register(TRX_STATE, RX_ON);

    //Wait for message to be delivered
    while(!PORTDbits.RD8) {}
    int result = ZIGBEE_check_interrupt();
    if(result == 1) //Wrong interrupt, reset
    {
        ZIGBEE_initialize;
        ZIGBEE_pll;
        ZIGBEE_receive();
    }
    else

    ZIGBEE_read_buffer();

    return;
}
void ZIGBEE_write_register (int address, int message){
    int receive;
    sel = 0;

    //or write command with address
    SPI2BUF = (0b11000000 | address);
    //wait for full recieve buffer so you can write again
    while (!SPI2STATbits.SPIRBF) {}
    receive = SPI2BUF;

    //write message to given register
    SPI2BUF = message;
    //wait for full recieve buffer to confirm transmission
    while (!SPI2STATbits.SPIRBF) {}
    receive = SPI2BUF;

    sel = 1;
}

int ZIGBEE_read_register (int address){
    int receive;
    int message;
    sel = 0;

    //or read command with address
    SPI2BUF = (0b10000000 | address);
   //wait for full recieve buffer so you can write again
    while (!SPI2STATbits.SPIRBF) {}
    receive = SPI2BUF;

    //0x00 to get response
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {}
    message = SPI2BUF;

    sel = 1;
    return message;
}

void ZIGBEE_write_buffer(int message1, int message2, int message3, int message4, int message5, int message6, int message7, int message8, int
message9) {
    //can include longer messages just need to send it as 8 bit packages message, message1, message2 etc. Be sure to change payload bytes
    sel = 0;
    //set number of payload bytes using 8 bits
    int payload = 0x0A;

    //give write to buffer command
    SPI2BUF = 0b01100000;
    //wait for full recieve buffer to confirm transmission
    while (!SPI2STATbits.SPIRBF) {};
    int receive = SPI2BUF;

    //give number of payload bytes
    int numbytes = payload + 13;
    SPI2BUF = numbytes;
    //wait for full recieve buffer to confirm transmission
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    //write MAC Frame Format

    SPI2BUF = 0x01;                 //Frame Control 1st
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

   SPI2BUF = 0x88;                  //Frame Control 2nd
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0x00;                 //Sequence Number
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0xEF;                 //Destination PAN ID 1
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0xEF;                 //Destination PAN ID 2
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0xCD;                 //Destination address 1
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0xAB;                 //Destination address 2
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;
```

```
SPI2BUF = 0x67;                 //Source PAN ID 1
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

SPI2BUF = 0x45;                 //Source PAN ID 2
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

SPI2BUF = 0x23;                 //Source address 1
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

SPI2BUF = 0x01;                 //Source address 2
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//write payload byte 1 to ZIGBEE buffer
SPI2BUF = message1;
//wait for full recieve buffer to confirm transmission
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//write payload byte 2 to ZIGBEE buffer
SPI2BUF = message2;
//wait for full recieve buffer to confirm transmission
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//write payload byte 3 to ZIGBEE buffer
SPI2BUF = message3;
//wait for full recieve buffer to confirm transmission
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//4
SPI2BUF = message4;
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//5
SPI2BUF = message5;
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//6
SPI2BUF = message6;
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//7
SPI2BUF = message7;
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//8
SPI2BUF = message8;
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

//9
SPI2BUF = message9;
while (!SPI2STATbits.SPIRBF) {};
receive = SPI2BUF;

sel = 1;
```

```
}
void ZIGBEE_read_buffer(){
    int numbytes, message1, message2, message3, message4;
    int message5, message6, message7, message8, message9;
    sel = 0;

    //give read buffer command
    SPI2BUF = 0b00100000;
    //wait for full recieve buffer to confirm transmission
    while (!SPI2STATbits.SPIRBF) {};
    int receive = SPI2BUF;

    //Number of payload bytes
    SPI2BUF = 0x00;
    while (!SPI2STATbits.SPIRBF) {};
    numbytes = SPI2BUF;

    //read MAC Frame Format

    SPI2BUF = 0x00;                 //Frame Control 1st
    while (!SPI2STATbits.SPIRBF) {};
    int receive1 = SPI2BUF;

    SPI2BUF = 0x00;             //Frame Control 2nd
    while (!SPI2STATbits.SPIRBF) {};
    int receive2 = SPI2BUF;

    SPI2BUF = 0x00;                 //Sequence Number
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0xEF;                 //Destination PAN ID 1
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0xEF;                 //Destination PAN ID 2
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0xCD;                 //Destination address 1
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0xAB;                 //Destination address 2
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0x67;                 //Source PAN ID 1
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0x45;                 //Source PAN ID 2
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0x23;                 //Source address 1
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    SPI2BUF = 0x01;                 //Source address 2
    while (!SPI2STATbits.SPIRBF) {};
    receive = SPI2BUF;

    //Data 1
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message1 = SPI2BUF;

    //Data 2
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message2 = SPI2BUF;

     //Data 3
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message3 = SPI2BUF;

     //Data 4
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message4 = SPI2BUF;

     //Data 5
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message5 = SPI2BUF;

     //Data 6
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message6 = SPI2BUF;

     //Data 7
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message7 = SPI2BUF;

     //Data 8
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message8 = SPI2BUF;

     //Data 9
    SPI2BUF = 0x00;
    //wait for full recieve buffer
    while (!SPI2STATbits.SPIRBF) {};
    message9 = SPI2BUF;

    sel = 1;
    printf("%7s %19s %18s \n", "Team", "X-Position (in)", "Y-Position (in)");
    printf("%6d %12d %18d \n", message1, message2, message3);
    printf("%6d %12d %18d \n", message1, message4, message5);
    printf("%6d %12d %18d \n", message1, message6, message7);
    printf("%6d %12d %18d \n", message1, message8, message9);

    return;
}

int ZIGBEE_check_interrupt()
{
    int result = 4;
    int problem = ZIGBEE_read_register(IRQ_STATUS);
    //pll locked
    if ((problem & 0x01) == 0x01)
    {
        result = 1;


    }

    //transmission sent, buffer empty OR message received, buffer full
    if ((problem & 0x08) == 0x08)
    {
        result = 2;


    }

    return result;
}
```

ZIGBEE_SEND.c

```
/*
 * File:   ZIGBEE_SEND.c
 * Author: Nick Ferruolo
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <sys/attribs.h>
#include <plib.h>
#include<p32xxxx.h>
#include "Zigbee_1.h"
#include "configbits_1.h"

void main(){
    int k;
    UART_initialize();
    SPI_initialize();
    ZIGBEE_initialize();
    ZIGBEE_pll();

    while (1) {


    ZIGBEE_transmit(1, 1,5, 2,6, 3,7,  4,8);
    ZIGBEE_transmit(2, 9,13, 10,14, 11,15, 12,16);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(1, 17,21, 18,22, 19,23, 20,24);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(2, 25,29, 26,30, 27,31, 28,32);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(1, 33,37, 34,38, 35,39, 36,40);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(2, 41,45, 42,46, 43,47, 44,48);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(1, 49,53, 50,54, 51,55, 52,56);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(2, 57,61, 58,62, 59,63, 60,64);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(1, 65,69, 66,70, 67,71, 68,72);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(2, 73,77, 74,78, 75,79, 76,80);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(1, 81,85, 82,86, 83,87, 84,88);
    for (k=0;k<5000000;k++){}
    ZIGBEE_transmit(2, 89,93, 90,94, 91,95, 92,96);
    for (k=0;k<5000000;k++){}

    }
}
```

ZIGBEE_RECEIVE.c

```
/*
 * File:   ZIGBEE_RECEIVE.c
 * Author: Nick Ferruolo
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>
#include <sys/attribs.h>
#include <plib.h>
#include<p32xxxx.h>
#include "Zigbee_1.h"
#include "configbits_1.h"

void main(){
    UART_initialize();
    SPI_initialize();
    ZIGBEE_initialize();
    ZIGBEE_pll();

    while(1){
        ZIGBEE_receive();
    }
}
```