/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Program Operation - This is a simple LoRa test transmitter. A packet containing ASCII text is sent

according to the frequency and LoRa settings specified in the 'Settings.h' file. The pins to access

the lora device need to be defined in the 'Settings.h' file also.

The details of the packet sent and any errors are shown on the Serial Monitor, together with the transmit

power used, the packet length and the CRC of the packet. The matching receive program, '4\_LoRa\_Receive'

can be used to check the packets are being sent correctly, the frequency and LoRa settings (in Settings.h)

must be the same for the Transmit and Receive program. Sample Serial Monitor output;

10dBm Packet> {packet contents\*} BytesSent,23 CRC,DAAB TransmitTime,54mS PacketsSent,1

Serial monitor baud rate is set at 9600

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define Program\_Version "V1.0"

#include <SPI.h> //the SX128X device is SPI based so load the SPI library

#include <SX128XLT.h> //include the appropriate library

#include "Settings.h" //include the setiings file, frequencies, LoRa settings etc

SX128XLT LT; //create a library class instance called LT

uint8\_t TXPacketL;

uint32\_t TXPacketCount, startmS, endmS;

uint8\_t buff[] = "Hello World";

void loop()

{

Serial.print(TXpower); //print the transmit power defined

Serial.print(F("dBm "));

Serial.print(F("Packet> "));

Serial.flush();

TXPacketL = sizeof(buff); //set TXPacketL to length of array

buff[TXPacketL - 1] = '\*'; //replace null character at buffer end so its visible on reciver

LT.printASCIIPacket(buff, TXPacketL); //print the buffer (the sent packet) as ASCII

digitalWrite(LED1, HIGH);

startmS = millis(); //start transmit timer

if (LT.transmit(buff, TXPacketL, 10000, TXpower, WAIT\_TX)) //will return packet length sent if OK, otherwise 0 if transmit, timeout 10 seconds

{

endmS = millis(); //packet sent, note end time

TXPacketCount++;

packet\_is\_OK();

}

else

{

packet\_is\_Error(); //transmit packet returned 0, there was an error

}

digitalWrite(LED1, LOW);

Serial.println();

delay(packet\_delay); //have a delay between packets

}

void packet\_is\_OK()

{

//if here packet has been sent OK

uint16\_t localCRC;

Serial.print(F(" BytesSent,"));

Serial.print(TXPacketL); //print transmitted packet length

localCRC = LT.CRCCCITT(buff, TXPacketL, 0xFFFF);

Serial.print(F(" CRC,"));

Serial.print(localCRC, HEX); //print CRC of sent packet

Serial.print(F(" TransmitTime,"));

Serial.print(endmS - startmS); //print transmit time of packet

Serial.print(F("mS"));

Serial.print(F(" PacketsSent,"));

Serial.print(TXPacketCount); //print total of packets sent OK

}

void packet\_is\_Error()

{

//if here there was an error transmitting packet

uint16\_t IRQStatus;

IRQStatus = LT.readIrqStatus(); //read the the interrupt register

Serial.print(F(" SendError,"));

Serial.print(F("Length,"));

Serial.print(TXPacketL); //print transmitted packet length

Serial.print(F(",IRQreg,"));

Serial.print(IRQStatus, HEX); //print IRQ status

LT.printIrqStatus(); //prints the text of which IRQs set

}

void led\_Flash(uint16\_t flashes, uint16\_t delaymS)

{

uint16\_t index;

for (index = 1; index <= flashes; index++)

{

digitalWrite(LED1, HIGH);

delay(delaymS);

digitalWrite(LED1, LOW);

delay(delaymS);

}

}

void setup()

{

pinMode(LED1, OUTPUT); //setup pin as output for indicator LED

led\_Flash(2, 125); //two quick LED flashes to indicate program start

Serial.begin(9600);

Serial.println();

Serial.print(F(\_\_TIME\_\_));

Serial.print(F(" "));

Serial.println(F(\_\_DATE\_\_));

Serial.println(F(Program\_Version));

Serial.println();

Serial.println(F("3\_LoRa\_Transmitter Starting"));

SPI.begin();

//SPI beginTranscation is normally part of library routines, but if it is disabled in library

//a single instance is needed here, so uncomment the program line below

//SPI.beginTransaction(SPISettings(8000000, MSBFIRST, SPI\_MODE0));

//setup hardware pins used by device, then check if device is found

if (LT.begin(NSS, NRESET, RFBUSY, DIO1, DIO2, DIO3, RX\_EN, TX\_EN, LORA\_DEVICE))

{

Serial.println(F("LoRa Device found"));

led\_Flash(2, 125); //two further quick LED flashes to indicate device found

delay(1000);

}

else

{

Serial.println(F("No device responding"));

while (1)

{

led\_Flash(50, 50); //long fast speed LED flash indicates device error

}

}

//The function call list below shows the complete setup for the LoRa device using the information defined in the

//Settings.h file.

//The 'Setup LoRa device' list below can be replaced with a single function call;

//LT.setupLoRa(Frequency, Offset, SpreadingFactor, Bandwidth, CodeRate);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Setup LoRa device

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LT.setMode(MODE\_STDBY\_RC);

LT.setRegulatorMode(USE\_LDO);

LT.setPacketType(PACKET\_TYPE\_LORA);

LT.setRfFrequency(Frequency, Offset);

LT.setBufferBaseAddress(0, 0);

LT.setModulationParams(SpreadingFactor, Bandwidth, CodeRate);

LT.setPacketParams(12, LORA\_PACKET\_VARIABLE\_LENGTH, 255, LORA\_CRC\_ON, LORA\_IQ\_NORMAL, 0, 0);

LT.setDioIrqParams(IRQ\_RADIO\_ALL, (IRQ\_TX\_DONE + IRQ\_RX\_TX\_TIMEOUT), 0, 0);

LT.setHighSensitivity();

//LT.setLowPowerRX();

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Serial.println();

LT.printModemSettings(); //reads and prints the configured LoRa settings, useful check

Serial.println();

LT.printOperatingSettings(); //reads and prints the configured operating settings, useful check

Serial.println();

Serial.println();

LT.printRegisters(0x900, 0x9FF); //print contents of device registers

Serial.println();

Serial.println();

Serial.print(F("Transmitter ready"));

Serial.println();

}