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

Programs for Arduino - Copyright of the author Stuart Robinson - 05/11/21

This program is supplied as is, it is up to the user of the program to decide if the program is

suitable for the intended purpose and free from errors.

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

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

Program Operation - This is a minimum setup LoRa test receiver. The program listens for incoming packets

using the frequency and LoRa settings in the LT.setupLoRa() command. The pins to access the lora device

need to be defined at the top of the program also.

This program does not need the DIO1 pin on the LoRa device connected.

There is a printout on the Arduino IDE serial monitor of the valid packets received, the packet is assumed

to be in ASCII printable text, if it's not ASCII text characters from 0x20 to 0x7F, expect weird things to

happen on the Serial Monitor. Sample serial monitor output;

8s Hello World 1234567890\*,RSSI,-44dBm,SNR,9dB,Length,23,Packets,7,Errors,0,IRQreg,50

If there is a packet error it might look like this, which is showing a CRC error;

137s PacketError,RSSI,-89dBm,SNR,-8dB,Length,23,Packets,37,Errors,2,IRQreg,70,IRQ\_HEADER\_VALID,IRQ\_CRC\_ERROR,IRQ\_RX\_DONE

If there are no packets received in a 10 second period then you should see a message like this;

112s RXTimeout

For an example of a more detailed configuration for a receiver, see program 104\_LoRa\_Receiver.

Serial monitor baud rate is set at 9600.

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

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

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

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

#define NSS PA4 //select pin on LoRa device

#define NRESET PB0 //reset pin on LoRa device

#define RFBUSY PB1 //busy pin on LoRa device

#define LORA\_DEVICE DEVICE\_SX1280 //we need to define the device we are using

#define RXBUFFER\_SIZE 255 //RX buffer size

uint32\_t RXpacketCount;

uint32\_t errors;

uint8\_t RXBUFFER[RXBUFFER\_SIZE]; //create the buffer that received packets are copied into

uint8\_t RXPacketL; //stores length of packet received

int16\_t PacketRSSI; //stores RSSI of received packet

int8\_t PacketSNR; //stores signal to noise ratio (SNR) of received packet

int ledState = LOW; // ledState used to set the LED

void loop()

{

RXPacketL = LT.receiveIRQ(RXBUFFER, RXBUFFER\_SIZE, 100000, WAIT\_RX); //wait for a packet to arrive with 60seconds (60000mS) timeout

PacketRSSI = LT.readPacketRSSI(); //read the received packets RSSI value

PacketSNR = LT.readPacketSNR(); //read the received packets SNR value

if (RXPacketL == 0) //if the LT.receive() function detects an error RXpacketL is 0

{

packet\_is\_Error();

}

else

{

packet\_is\_OK();

}

Serial.println();

}

void packet\_is\_OK()

{

//digitalWrite(PC13, HIGH); // turn the LED on by making the voltage HIGH

// if the LED is off turn it on and vice-versa:

if (ledState == LOW) {

ledState = HIGH;

} else {

ledState = LOW;

}

// set the LED with the ledState of the variable:

digitalWrite(PC13, ledState);

uint16\_t IRQStatus;

RXpacketCount++;

IRQStatus = LT.readIrqStatus(); //read the LoRa device IRQ status register

printElapsedTime(); //print elapsed time to Serial Monitor

Serial.print(F(" "));

LT.printASCIIPacket(RXBUFFER, RXPacketL); //print the packet as ASCII characters

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

Serial.print(PacketRSSI);

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

Serial.print(PacketSNR);

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

Serial.print(RXPacketL);

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

Serial.print(RXpacketCount);

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

Serial.print(errors);

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

//Serial.print(IRQStatus, HEX);

// digitalWrite(PC13, LOW); // turn the LED off by making the voltage LOW

}

void packet\_is\_Error()

{

uint16\_t IRQStatus;

IRQStatus = LT.readIrqStatus(); //read the LoRa device IRQ status register

printElapsedTime(); //print elapsed time to Serial Monitor

if (IRQStatus & IRQ\_RX\_TIMEOUT) //check for an RX timeout

{

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

}

else

{

errors++;

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

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

Serial.print(PacketRSSI);

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

Serial.print(PacketSNR);

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

Serial.print(LT.readRXPacketL()); //get the real packet length

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

Serial.print(RXpacketCount);

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

Serial.print(errors);

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

//Serial.print(IRQStatus, HEX);

LT.printIrqStatus(); //print the names of the IRQ registers set

}

}

void printElapsedTime()

{

float seconds;

seconds = millis() / 1000;

Serial.print(seconds, 0);

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

}

void setup()

{

delay(5000);

// initialize digital pin PB1 as an output.

pinMode(PC13, OUTPUT);

Serial.begin(9600);

Serial.println();

Serial.println(F("4\_LoRa\_ReceiverIRQ Starting"));

Serial.println();

SPI.begin();

if (LT.begin(NSS, NRESET, RFBUSY, LORA\_DEVICE))

{

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

delay(1000);

}

else

{

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

while (1);

}

LT.setupGFSK(2424500000, 0, GFS\_BLE\_BR\_0\_500\_BW\_1\_2, GFS\_BLE\_MOD\_IND\_0\_50, 0); //configure frequency and LoRa settings

Serial.print(F("Receiver ready - RXBUFFER\_SIZE "));

Serial.println(RXBUFFER\_SIZE);

Serial.println();

}