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

Programs for Arduino - Copyright of the author Stuart Robinson - 11/02/20

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 program is stand alone, it is not necessary to install the SX12XX-LoRa library

to use it. This test program is for the SX128X LoRa devices.

The program checks that a SX128X LoRa device can be accessed by doing a test register write and read.

If there is no device found a message is printed on the serial monitor. The contents of the registers

from 0x00 to 0x7F are printed, there is a copy of a typical printout below. Note that the read back

changed frequency may be slightly different to the programmed frequency, there is a rounding error due

to the use of floats to calculate the frequency.

The Arduino pin numbers that the NSS and NRESET pins on the LoRa device are connected to must be

specified in the hardware definitions section below. The LoRa device type in use, SX1280 or SX1281

must be specified also.

Typical printout;

2\_Register\_Test Starting

Reset device

LoRa Device found

Reset device

Registers at reset

Reg 0 1 2 3 4 5 6 7 8 9 A B C D E F

0x900 80 0C 7B 02 20 FA C0 00 00 80 00 00 00 00 00 FF

0x910 FF FF 00 00 00 19 00 00 00 19 87 65 43 21 7F FF

0x920 FF FF FF 0C 70 37 0A 50 D0 80 00 C0 5F D2 8F 0A

0x930 00 C0 00 00 00 24 00 21 28 B0 30 09 1A 59 70 08

0x940 58 0B 32 0A 14 24 6A 96 00 18 00 00 00 00 00 00

0x950 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x960 00 00 00 00 00 00 00 00 00 00 FF FF FF FF FF FF

0x970 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF 04

0x980 00 0B 18 70 00 00 00 4C 00 F0 64 00 00 00 00 00

0x990 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x9A0 00 08 EC B8 9D 8A E6 66 06 00 00 00 00 00 00 00

0x9B0 00 08 EC B8 9D 8A E6 66 06 00 00 00 00 00 00 00

0x9C0 00 16 00 3F E8 01 FF FF FF FF 5E 4D 25 10 55 55

0x9D0 55 55 55 55 55 55 55 55 55 55 55 55 55 00 00 00

0x9E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x9F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Frequency at reset 2495996672hz

Change Frequency to 2445000000hz

Frequency now 2444999936hz

Reg 0 1 2 3 4 5 6 7 8 9 A B C D E F

0x900 80 0C 7B 02 20 FA BC 13 C1 80 00 00 00 00 00 61

Serial monitor baud rate is set at 9600.

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

const uint16\_t REG\_RFFrequency23\_16 = 0x906;

const uint16\_t REG\_RFFrequency15\_8 = 0x907;

const uint16\_t REG\_RFFrequency7\_0 = 0x908;

const uint8\_t RADIO\_WRITE\_REGISTER = 0x18;

const uint8\_t RADIO\_READ\_REGISTER = 0x19;

const uint8\_t RADIO\_SET\_RFFREQUENCY = 0x86; //commnad to change frequency

const uint8\_t RADIO\_SET\_PACKETTYPE = 0x8A; //commnad to set packet mode

const float FREQ\_STEP = 198.364;

const uint8\_t PACKET\_TYPE\_LORA = 0x01;

//\*\*\*\*\*\*\*\*\* Setup hardware definitions here ! \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//These are the pin definitions for one of the Tracker boards, be sure to change them to match your

//own setup. You will also need to connect up the pins for the SPI bus, which on an Arduino Pro Mini are

//SCK pin 13, MISO pin 12, and MOSI pin 11.

#define NSS 10 //SX128X device select

#define NRESET 9 //SX128X reset pin

#define RFBUSY 7 //SX128X busy pin

#define LED1 8 //for on board LED, put high for on

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

#include <SPI.h>

uint8\_t saveddevice;

void setup()

{

Serial.begin(9600);

Serial.println(F("2\_Register\_Test Starting"));

SPI.begin();

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

//The begin function setups the hardware pins used by device and then checks if device is found

//the DIO1, DIO2 and DIO3 pins are not used in this example so are set to -1

if (begin(NSS, NRESET, RFBUSY, -1, -1, -1, 0))

{

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

}

else

{

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

}

}

void loop()

{

uint32\_t frequency;

resetDevice(); //reset the device

Serial.println(F("Registers at reset")); //show the all registers following a reset

printRegisters(0x0900, 0x09FF);

Serial.println();

Serial.println();

frequency = getFreqInt(); //read the set frequency following a reset

Serial.print(F(" Frequency at reset "));

Serial.print(frequency);

Serial.println(F("hz"));

Serial.print(F("Change Frequency to 2445000000hz"));

setPacketType(PACKET\_TYPE\_LORA); //this is needed to ensure frequency change is reflected in register print

setRfFrequency(2445000000, 0); //change the frequency to 2445000000hertz

frequency = getFreqInt(); //read back the changed frequency

Serial.println();

Serial.print(F(" Frequency now "));

Serial.print(frequency); //print the changed frequency, did the write work (allow for rounding errors) ?

Serial.println(F("hz"));

Serial.println();

printRegisters(0x0900, 0x090F); //show the registers after frequency change

Serial.println();

Serial.println();

delay(5000);

}

void readRegisters(uint16\_t address, uint8\_t \*buffer, uint16\_t size)

{

uint16\_t index;

uint8\_t addr\_l, addr\_h;

addr\_h = address >> 8;

addr\_l = address & 0x00FF;

checkBusy();

digitalWrite(NSS, LOW);

SPI.transfer(RADIO\_READ\_REGISTER);

SPI.transfer(addr\_h); //MSB

SPI.transfer(addr\_l); //LSB

SPI.transfer(0xFF);

for (index = 0; index < size; index++)

{

\*(buffer + index) = SPI.transfer(0xFF);

}

digitalWrite(NSS, HIGH);

checkBusy();

}

uint8\_t readRegister(uint16\_t address)

{

uint8\_t data;

readRegisters(address, &data, 1);

return data;

}

void writeRegisters(uint16\_t address, uint8\_t \*buffer, uint16\_t size)

{

uint8\_t addr\_l, addr\_h;

uint8\_t i;

addr\_l = address & 0xff;

addr\_h = address >> 8;

checkBusy();

digitalWrite(NSS, LOW);

SPI.transfer(RADIO\_WRITE\_REGISTER);

SPI.transfer(addr\_h); //MSB

SPI.transfer(addr\_l); //LSB

for (i = 0; i < size; i++)

{

SPI.transfer(buffer[i]);

}

digitalWrite(NSS, HIGH);

checkBusy();

}

void writeRegister(uint16\_t address, uint8\_t value)

{

writeRegisters(address, &value, 1 );

}

uint32\_t getFreqInt()

{

//get the current set device frequency, return as long integer

uint8\_t Msb, Mid, Lsb;

uint32\_t uinttemp;

float floattemp;

Msb = readRegister(REG\_RFFrequency23\_16);

Mid = readRegister(REG\_RFFrequency15\_8);

Lsb = readRegister(REG\_RFFrequency7\_0);

floattemp = ((Msb \* 0x10000ul) + (Mid \* 0x100ul) + Lsb);

floattemp = ((floattemp \* FREQ\_STEP) / 1000000ul);

uinttemp = (uint32\_t)(floattemp \* 1000000);

return uinttemp;

}

void printRegisters(uint16\_t Start, uint16\_t End)

{

//prints the contents of SX128x registers to serial monitor

uint16\_t Loopv1, Loopv2, RegData;

Serial.print(F("Reg 0 1 2 3 4 5 6 7 8 9 A B C D E F"));

Serial.println();

for (Loopv1 = Start; Loopv1 <= End;) //32 lines

{

Serial.print(F("0x"));

Serial.print((Loopv1), HEX); //print the register number

Serial.print(F(" "));

for (Loopv2 = 0; Loopv2 <= 15; Loopv2++)

{

RegData = readRegister(Loopv1);

if (RegData < 0x10)

{

Serial.print(F("0"));

}

Serial.print(RegData, HEX); //print the register number

Serial.print(F(" "));

Loopv1++;

}

Serial.println();

}

}

void setRfFrequency(uint32\_t frequency, int32\_t offset)

{

frequency = frequency + offset;

uint8\_t buffer[3];

uint32\_t freqtemp = 0;

freqtemp = ( uint32\_t )( (float) frequency / (float) FREQ\_STEP);

buffer[0] = ( uint8\_t )( ( freqtemp >> 16 ) & 0xFF );

buffer[1] = ( uint8\_t )( ( freqtemp >> 8 ) & 0xFF );

buffer[2] = ( uint8\_t )( freqtemp & 0xFF );

writeCommand(RADIO\_SET\_RFFREQUENCY, buffer, 3);

writeCommand(RADIO\_SET\_RFFREQUENCY, buffer, 3);

}

void checkBusy()

{

uint8\_t busy\_timeout\_cnt;

busy\_timeout\_cnt = 0;

while (digitalRead(RFBUSY))

{

delay(1);

busy\_timeout\_cnt++;

if (busy\_timeout\_cnt > 10) //wait 10mS for busy to complete

{

busy\_timeout\_cnt = 0;

Serial.println(F("ERROR - Busy Timeout!"));

break;

}

}

}

void resetDevice()

{

Serial.println(F("Reset device"));

delay(10);

digitalWrite(NRESET, LOW);

delay(2);

digitalWrite(NRESET, HIGH);

delay(25);

checkBusy();

}

bool begin(int8\_t pinNSS, int8\_t pinNRESET, int8\_t pinRFBUSY, int8\_t pinDIO1, int8\_t pinDIO2, int8\_t pinDIO3, uint8\_t device)

{

saveddevice = device;

pinMode(pinNSS, OUTPUT);

digitalWrite(pinNSS, HIGH);

pinMode(pinNRESET, OUTPUT);

digitalWrite(pinNRESET, HIGH);

pinMode(pinRFBUSY, INPUT);

if (pinDIO1 >= 0)

{

pinMode( pinDIO1, INPUT);

}

if (pinDIO2 >= 0)

{

pinMode(pinDIO2, INPUT);

}

if (pinDIO3 >= 0)

{

pinMode(pinDIO3, INPUT);

}

resetDevice();

if (checkDevice())

{

return true;

}

return false;

}

bool checkDevice()

{

//check there is a device out there, writes a register and reads back

uint8\_t Regdata1, Regdata2;

Regdata1 = readRegister(0x0908); //low byte of frequency setting

writeRegister(0x0908, (Regdata1 + 1));

Regdata2 = readRegister(0x0908); //read changed value back

writeRegister(0x0908, Regdata1); //restore register to original value

if (Regdata2 == (Regdata1 + 1))

{

return true;

}

else

{

return false;

}

}

void writeCommand(uint8\_t Opcode, uint8\_t \*buffer, uint16\_t size)

{

uint8\_t index;

checkBusy();

digitalWrite(NSS, LOW);

SPI.transfer(Opcode);

for (index = 0; index < size; index++)

{

SPI.transfer(buffer[index]);

//Serial.println(buffer[index], HEX);

}

digitalWrite(NSS, HIGH);

checkBusy();

}

void setPacketType(uint8\_t packettype)

{

writeCommand(RADIO\_SET\_PACKETTYPE, &packettype, 1);

}