```
//  2023.11.19 Add Si5351a with library
//   "p""q" works for Si5351a frequency setting
//  2023.11.01 R909 PANEL & RF PCB OK  FM received
//   "c" sets for 21.4MHz RX 2023.11.03
//
//
//   by JA3KPA, nobcha, Nov 2023
/*
   Test and validation of the SI4735 Arduino Library.
   It is a FM, MW and SW (1700kHz to 30000kHz)

   ATTENTION:  Please, avoid using the computer connected to the mains during
testing.

   The main advantages of using this sketch are:
    1) It is a easy way to check if your circuit is working;
    2) You do not need to connect any display device to make your radio works;
    3) You do not need connect any push buttons or encoders to change volume and
frequency;
    4) The Arduino IDE is all you need to control the radio.

   This sketch has been successfully tested on:
    1) Pro Mini 3.3V;
    2) UNO (by using a voltage converter);
    3) Arduino Yún;
    4) Arduino Mega (by using a voltage converter); and
    5) ESP32 (LOLIN32 WEMOS)


    The table below shows the Si4735 and Arduino Pro Mini pin connections

    | Si4735 pin      |  Arduino Pin  |
    | ----------------| ------------  |
    | RESET (pin 15)  |     12        |
    | SDIO (pin 18)   |     A4        |
    | CLK (pin 17)    |     A5        |


  I strongly recommend starting with this sketch.

  Schematic:
https://github.com/pu2clr/SI4735/blob/master/extras/images/basic_schematic.png

  Prototype documentation : https://pu2clr.github.io/SI4735/
  PU2CLR Si47XX API documentation:
https://pu2clr.github.io/SI4735/extras/apidoc/html/

   By Ricardo Lima Caratti, Nov 2019.
*/

#include <SI4735.h>

#include <si5351JBH.h>
```

```
#define RESET_PIN 7  // change

#define AM_FUNCTION 1
#define FM_FUNCTION 0

uint16_t currentFrequency;
uint16_t previousFrequency;
uint8_t bandwidthIdx = 0;
const char *bandwidth[] = {"6", "4", "3", "2", "1", "1.8", "2.5"};
unsigned long int startF = 118100000; //

SI4735 rx;

//initialise object SI5351 pll
Si5351 si5351; // start the object for the si5351
#define XT_CAL_F 37000  //Si5351 calibration factor, adjust to get exatcly
10MHz.
                        //Increasing this value will decreases the frequency and
vice versa.


void setup()
{
  Serial.begin(9600);
  while(!Serial);

  digitalWrite(RESET_PIN, HIGH);

  Serial.println("AM and FM station tuning test.");

  showHelp();

  // Look for the Si47XX I2C bus address
  int16_t si4735Addr = rx.getDeviceI2CAddress(RESET_PIN);
  if ( si4735Addr == 0 ) {
    Serial.println("Si473X not found!");
    Serial.flush();
    while (1);
  } else {
    Serial.print("The Si473X I2C address is 0x");
    Serial.println(si4735Addr, HEX);
  }

  si5351_init();   // initialise the si5351

  delay(500);
  rx.setup(RESET_PIN, FM_FUNCTION);
  // rx.setup(RESET_PIN, -1, 1, SI473X_ANALOG_AUDIO);
  // Starts defaul radio function and band (FM; from 70 to 108 MHz; 103.9 MHz;
step 100kHz)
  rx.setFM(7000, 10800, 80200, 10);
  delay(500);
  currentFrequency = previousFrequency = rx.getFrequency();
  rx.setVolume(15);   // changed
```

```
  showStatus();
}

void showHelp()
{

  Serial.println("Type F to FM; A to MW; 1 to All Band (100kHz to 30MHz)");
  Serial.println("Type U to increase and D to decrease the frequency");
  Serial.println("Type S or s to seek station Up or Down");
  Serial.println("Type C or c to set AM 21.4MHz RX");
  Serial.println("Type P or p to set 80.0MHz on Si5351aq");
  Serial.println("Type Q or q to set 139.5MHz on Si5351aq");
  Serial.println("Type + or - to volume Up or Down");
  Serial.println("Type 0 to show current status");
  Serial.println("Type B to change Bandwidth filter");
  Serial.println("Type 4 to 8 (4 to step 1; 5 to step 5kHz; 6 to 10kHz; 7 to
100kHz; 8 to 1000kHz)");
  Serial.println("Type ? to this help.");
  Serial.println("================================================");
  delay(1000);
}

// Show current frequency
void showStatus()
{
  // rx.getStatus();
  previousFrequency = currentFrequency = rx.getFrequency();
  rx.getCurrentReceivedSignalQuality();
  Serial.print("You are tuned on ");
  if (rx.isCurrentTuneFM())
  {
    Serial.print(String(currentFrequency / 100.0, 2));
    Serial.print("MHz ");
    Serial.print((rx.getCurrentPilot()) ? "STEREO" : "MONO");
  }
  else
  {
    Serial.print(currentFrequency);
    Serial.print("kHz");
  }
  Serial.print(" [SNR:");
  Serial.print(rx.getCurrentSNR());
  Serial.print("dB");

  Serial.print(" Signal:");
  Serial.print(rx.getCurrentRSSI());
  Serial.println("dBuV]");
}

void showFrequency( uint16_t freq ) {

  if (rx.isCurrentTuneFM())
  {
    Serial.print(String(freq / 100.0, 2));
```

```
    Serial.println("MHz ");
  }
  else
  {
    Serial.print(freq);
    Serial.println("kHz");
  }

}

void si5351_init() {
  // Initialize the Si5351
  si5351.init(SI5351_CRYSTAL_LOAD_8PF , 0 , 0);


  // Set CLK0 to output vfo frequency with a fixed PLL frequency
  si5351.set_pll(SI5351_PLL_FIXED, SI5351_PLLA);

  // Initialize each ports frequency
  si5351.set_freq(startF, SI5351_CLK0);

  // Set each ports power
  si5351.output_enable(SI5351_CLK0, 1);
  si5351.output_driver(SI5351_CLK0, 1);
  si5351.drive_strength(SI5351_CLK0, SI5351_DRIVE_2MA);

  // in case phase needs to be shifted
 si5351.set_phase(SI5351_CLK2, 50); //90 DEGREE SHIFT
 si5351.pll_reset(SI5351_PLLA); // RESET THE PLL

// enable the output and driver strength
//  si5351.output_enable(SI5351_CLK1, 1);
//  si5351.output_driver(SI5351_CLK1, 1);
//  si5351.drive_strength(SI5351_CLK1, SI5351_DRIVE_8MA);

}




// Main
void loop()
{
  if (Serial.available() > 0)
  {
    char key = Serial.read();
    switch (key)
    {
    case '+':
      rx.volumeUp();
      break;
    case '-':
      rx.volumeDown();
      break;
```

```
case 'a':
case 'A':
  rx.setAM(520, 1750, 810, 10);
  rx.setSeekAmLimits(520, 1750);
  rx.setSeekAmSpacing(10); // spacing 10kHz
  break;
case 'c':
case 'C':                     // airband IF
  rx.setAM(21200, 21600, 21400, 4);
  rx.setSeekAmLimits(21200, 21600);
  rx.setSeekAmSpacing(4);     // spacing 1kHz
  break;

case 'p':
case 'P':                     // Si5351a clk0 set
  si5351.set_freq(8000000000, SI5351_CLK0);
  Serial.println("\nSet 80.0f
  MHz on 5351");
  break;
case 'q':
case 'Q':                     // Si5351a clk0 set
  si5351.set_freq(13950000000, SI5351_CLK0);
  Serial.println("\nSet 139.5MHz on 5351");
  break;

case 'f':
case 'F':
  rx.setFM(7000, 10800, 8020, 50);
  rx.setSeekAmRssiThreshold(0);
  rx.setSeekAmSrnThreshold(10);
  break;
case '1':
  rx.setAM(100, 30000, 7200, 5);
  rx.setSeekAmLimits(100, 30000);   // Range for seeking.
  rx.setSeekAmSpacing(1); // spacing 1kHz
  Serial.println("\nALL - LW/MW/SW");
  break;
case 'U':
case 'u':
  rx.frequencyUp();
  break;
case 'D':
case 'd':
  rx.frequencyDown();
  break;
case 'b':
case 'B':
  if (rx.isCurrentTuneFM())
  {
    Serial.println("Not valid for FM");
  }
  else
  {
    if (bandwidthIdx > 6)
```

```
        bandwidthIdx = 0;
      rx.setBandwidth(bandwidthIdx, 1);
      Serial.print("Filter - Bandwidth: ");
      Serial.print(String(bandwidth[bandwidthIdx]));
      Serial.println(" kHz");
      bandwidthIdx++;
    }
    break;
  case 'S':
    rx.seekStationProgress(showFrequency,1);
    // rx.seekStationUp();
    break;
  case 's':
    rx.seekStationProgress(showFrequency,0);
    // rx.seekStationDown();
    break;
  case '0':
    showStatus();
    break;
  case '4':
    rx.setFrequencyStep(1);
    Serial.println("\nStep 1");
    break;
  case '5':
    rx.setFrequencyStep(5);
    Serial.println("\nStep 5");
    break;
  case '6':
    rx.setFrequencyStep(10);
    Serial.println("\nStep 10");
    break;
  case '7':
    rx.setFrequencyStep(100);
    Serial.println("\nStep 100");
    break;
  case '8':
    rx.setFrequencyStep(1000);
    Serial.println("\nStep 1000");
    break;
  case '?':
    showHelp();
    break;
  default:
    break;
  }
}
delay(100);
currentFrequency = rx.getCurrentFrequency();
if (currentFrequency != previousFrequency)
{
  previousFrequency = currentFrequency;
  showStatus();
  delay(300);
}
```

```
}
```