Si4735 Library for Arduino

This is a library for the SI4735, BROADCAST AM/FM/SW RADIO RECEIVER, IC from Silicon Labs for the Arduino development environment. This library is intended to provide an easier interface for controlling the SI4435.

The SI4835 is a 3.3V part. If you are not using a 3.3V version of Arduino, you have to use a kind of 5V to 3.3V converter.

By Ricardo Lima Caratti, Oct, 2019.

Attention: Documentation under construction.

Summary

Thanks

I would like to thank Mr. Francisco Scaramella for the suggestions and contributions provided in the electronics field as well as for the testing of the functions implemented in this library.

Your support is important.

If you would like to support this library development, consider joining this project via Github. Alternatively, make suggestions on features you would like available in this library. Thank you!

About the SI4735

The Si4735-D60 is DSP radio receiver IC from Silcon Labs. It has great performance on AM (LW/MW/SW) and local FM station. The SI4735 is programmed using commands and responses. You can contron it via a microcontroller like Arduino. To make the SI4735 perform an action, the microccontroller have to send a set of bytes (command and arguments) tha the device interpret it and executes the given command.

The main Si4735-D60 features

- Great Programming Guide and additional documentation to deal with the device
- FM band support (64–108 MHz)
- AM (MW) band support (520–1710 kHz)
- SW band support (2.3–26.1 MHz)
- LW band support (153–279 kHz)
- Allows firmware upgrade. Including the possibility of adjustments to demodulate SSB.
- Advanced AM/FM seek tuning
- Automatic frequency control (AFC)
- Automatic gain control (AGC)
- Digital FM stereo decoder
- AM/FM/SW/LW digital tuning
- RDS/RBDS processor
- Digital audio out
- I2C and SPI interface

SI4735 Terminology

Term	Description
Arduino Libraries	Libraries are files written in C or C++ (.c, .cpp) which provide your sketches with extra functionality. The SI4735 Library provides extra functionalities to make easier the Arduino deal with Si4735 device

Term	Description
IDE	Integrated Development Environment
Sketch	Name that Arduino environment uses for a program
interrupt	In this context, it is a Arduino Resource. Allows important tasks to be performed regardless of the flow of your program
C++	A object-oriented programming (OOP) language. It is a superset of the C language with an additional concept of "classes."
programming guide	In this context it refers to Si47XX PROGRAMMING GUIDE (https://www.silabs.com/documents/public/application-notes/AN332.pdf)
POC	Proof of Concept
SEN	Serial enable pin, active low; used as device select in 3-wire and SPI operation and address selection in 2-wire operation
SDIO	Serial data in/data out pin
SCLK	Serial clock pin
RST	Also RSTb—Reset pin, active low
RCLK	External reference clock
GPO	General purpose output
CTS	Clear to send
STC	Seek/Tune Complete
NVM	Non-volatile internal device memory
CMD	Command byte
COMMANDn	Command register (16-bit) in 3-Wire mode (n = 1 to 4)
ARGn	Argument byte (n = 1 to 7)
STATUS	Status byte
RESP	Response byte (n = 1 to 15)
RESPONSEn	Response register (16-bit) in 3-Wire mode (n = 1 to 8)

SI4735 Arduino Library Features

1. Open Source
2. Built Based on Si47XX PROGRAMMING GUIDE (https://www.silabs.com/documents/public/application-notes/AN332.pdf)
3. C++ Lenguage and Object-oriented programming
4. Available on Arduino IDE (Manage Libraries)
5. Simplifies projects based on SI4735

Library Installation

TO DO....

Arduino 5V and Si4844

TO DO ...

Hardware Requirements and Setup

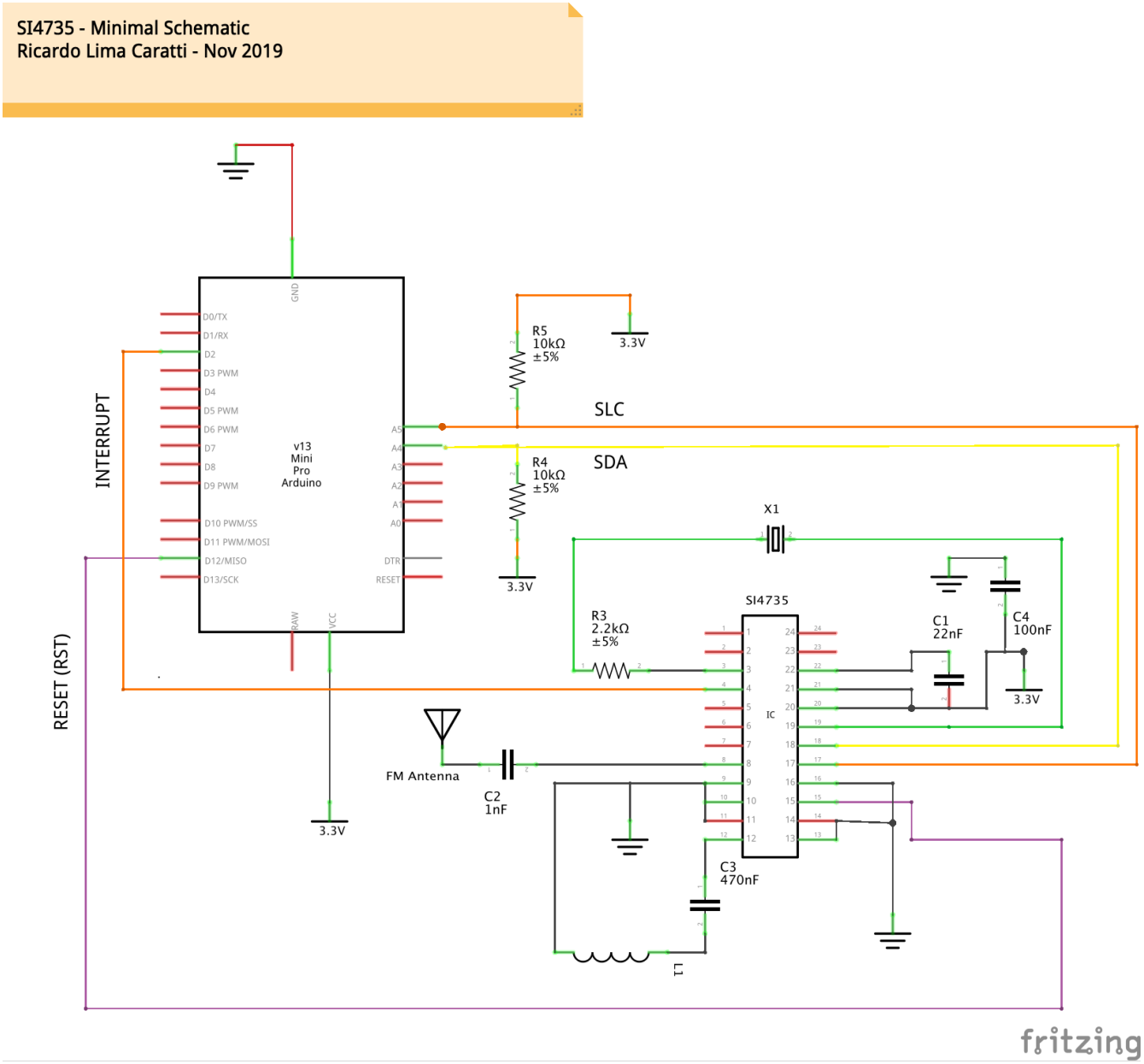
This library has been written for the Arduino platform and has been successfully tested on Pro Mini. I beleave it will work on any other Arduino with I2C support.

Arduino 5V and Si4844

The SI4735 device works with 3.3V only. If you are not using a 3.3V version of Arduino, you have to use a kind of converter.

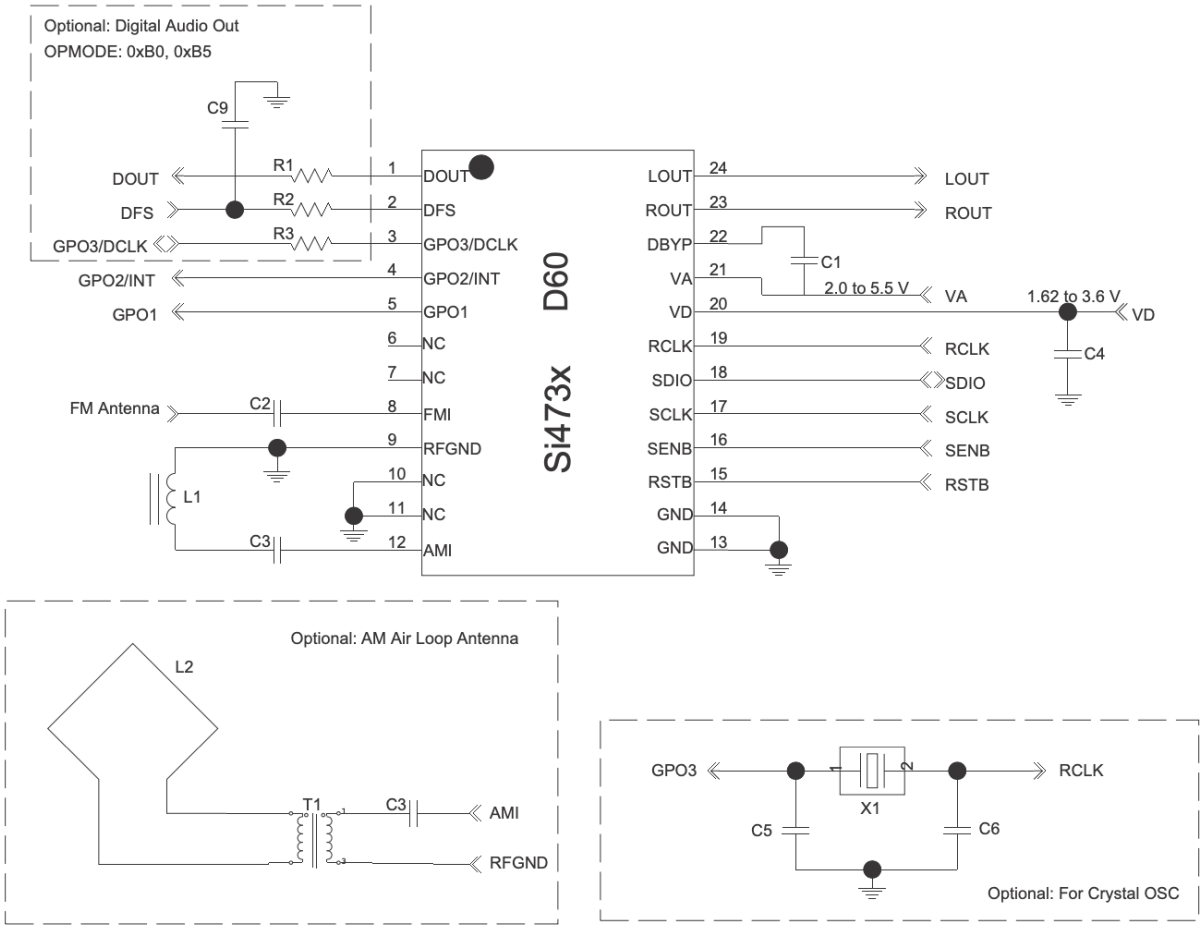
Schematic

The image bellow shows a version of Slicon Labs SSOP Typical Application Schematic. The basic difference are the pull-up resitors on I2C bus.



The image bellow shows the Slicon Labs SSOP Typical Application Schematic.

2.2. SSOP Typical Application Schematic



Notes:

1. Place C1 close to VA and C4 close to VD pin.
2. All grounds connect directly to GND plane on PCB.
3. Pins 6 and 7 are no connects, leave floating.
4. Pins 10 and 11 are unused. Tie these pins to GND.
5. To ensure proper operation and receiver performance, follow the guidelines in “AN383: Si47xx Antenna, Schematic, Layout, and Design Guidelines.” Silicon Laboratories will evaluate schematics and layouts for qualified customers.
6. Pin 8 connects to the FM antenna interface, and pin 12 connects to the AM antenna interface.
7. Place Si473x-D60 as close as possible to antenna and keep the FMI and AMI traces as short as possible.

Parts

Part	Description
C1	22nF (Place C1 close to VA pin)
C2	100pF
C3	470nF
C4	100nF (close to VD pin)
C5 and C6	22pF (Crystal load capacitors)

Part	Description
R3	2.2K
R4 and R5[^1]	10K (pull-up resistors)
L1	Ferrite loop stick (about 500 μH)
X1	32.768 kHz crystal

[^1]: R4 and R5 are pull-up resistor included byte the author of this project. They are not present on original Silicon Labs schematic.

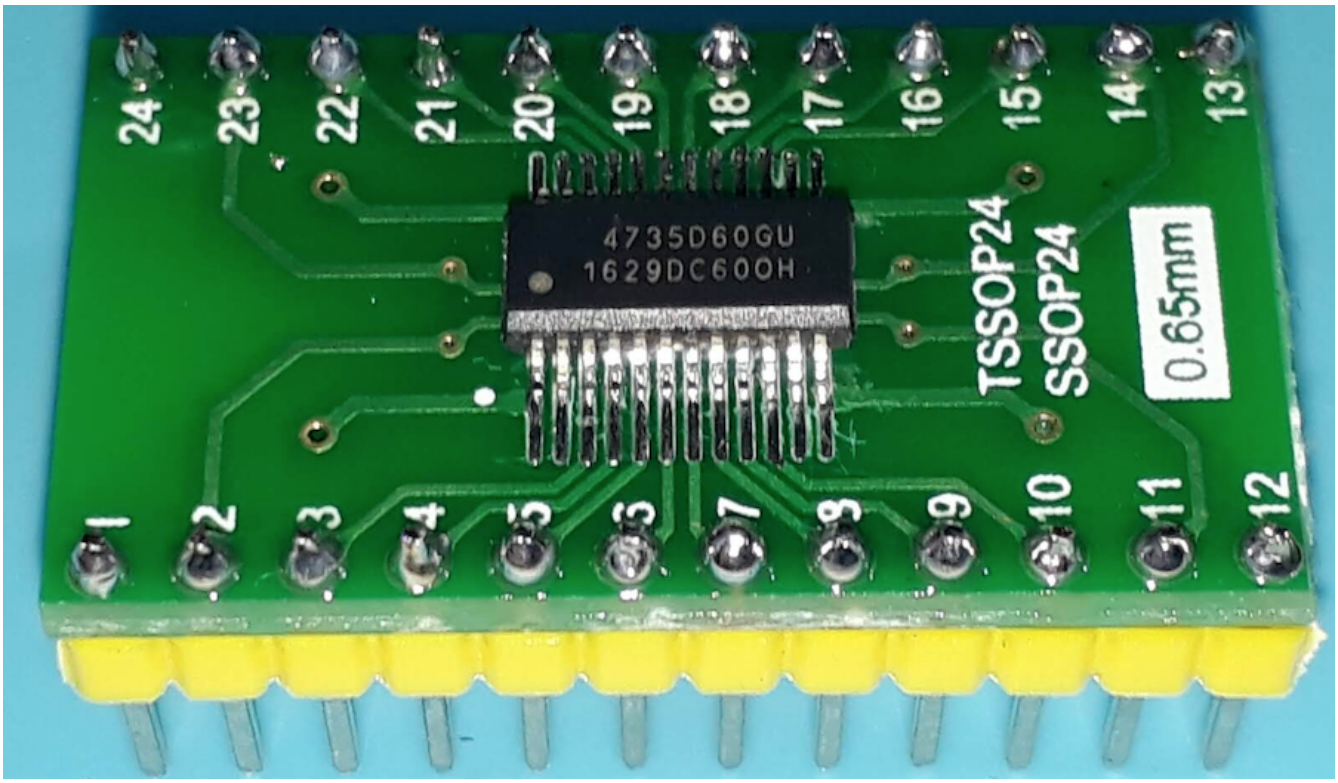
Notes from Silicon Labs Broadcast AM/FM/SW/LW Radio Receiver (page 12):

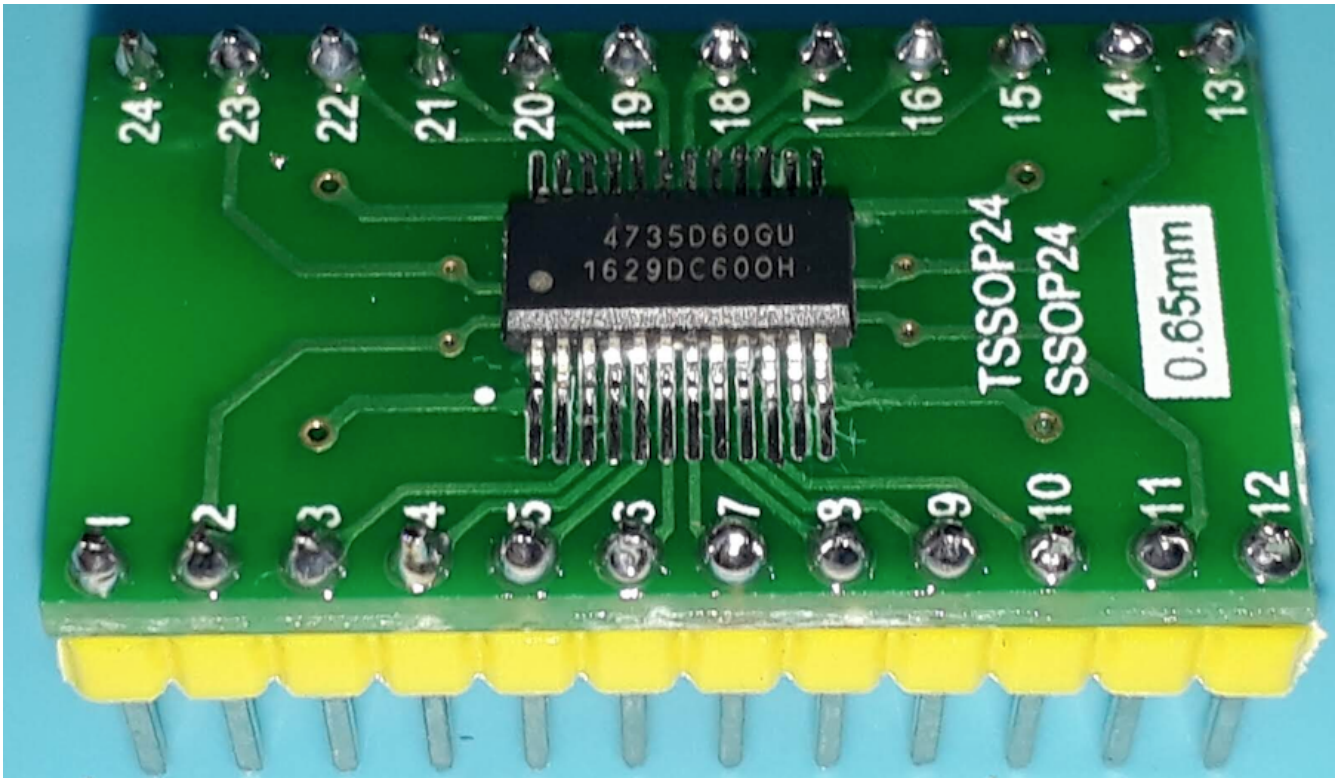
- Place C1 close to VA and C4 close to VD pin.
- All grounds connect directly to GND plane on PCB.
- Pins 6 and 7 are no connects, leave floating.
- Pins 10 and 11 are unused. Tie these pins to GND.
- To ensure proper operation and receiver performance, follow the guidelines in “AN383: Si47xx Antenna, Schematic,
- Layout, and Design Guidelines.” Silicon Laboratories will evaluate schematics and layouts for qualified customers.
- Pin 8 connects to the FM antenna interface, and pin 12 connects to the AM antenna interface.
- Place Si473x-D60 as close as possible to antenna and keep the FMI and AMI traces as short as possible.

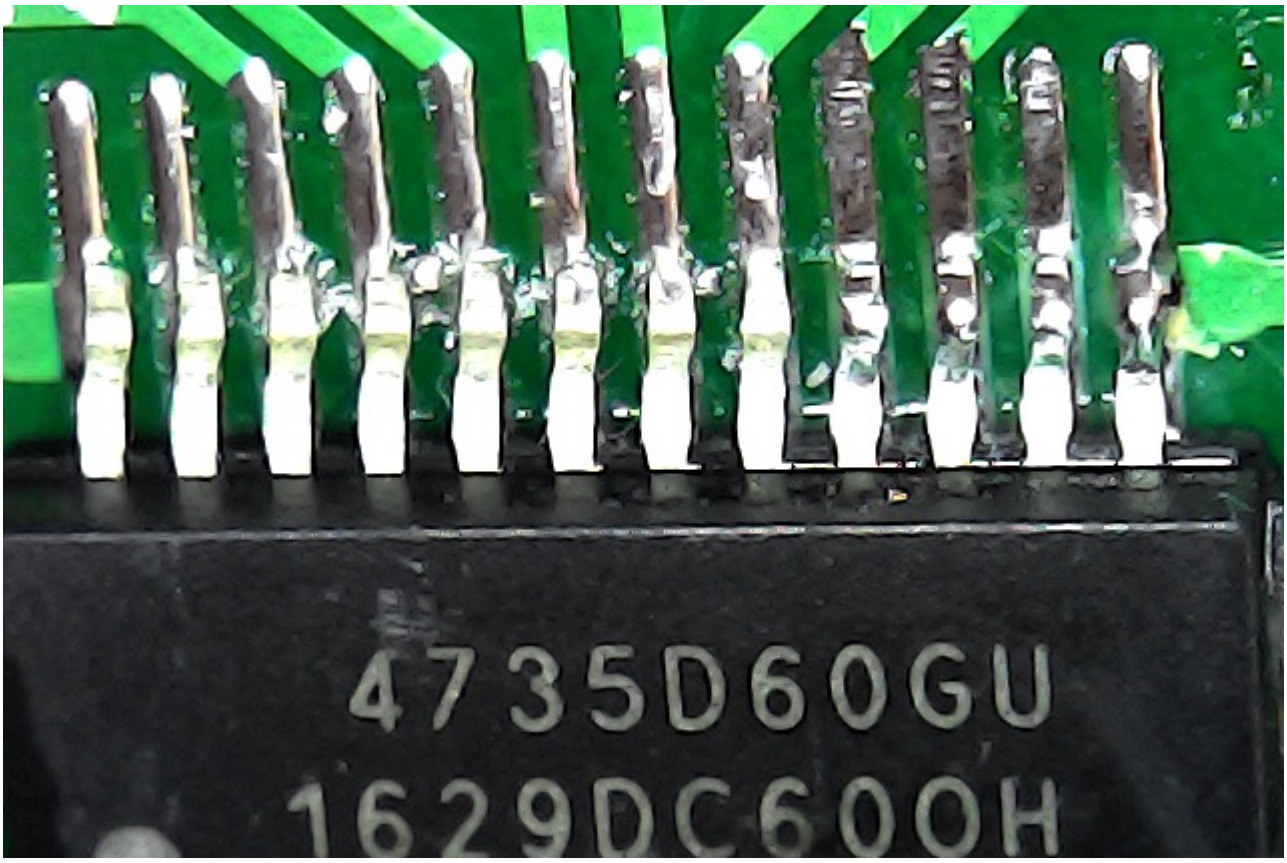
Photos

It was a bit hard to solder the kind of CI on adapter. However, by using a electronic magnifier it was possible.

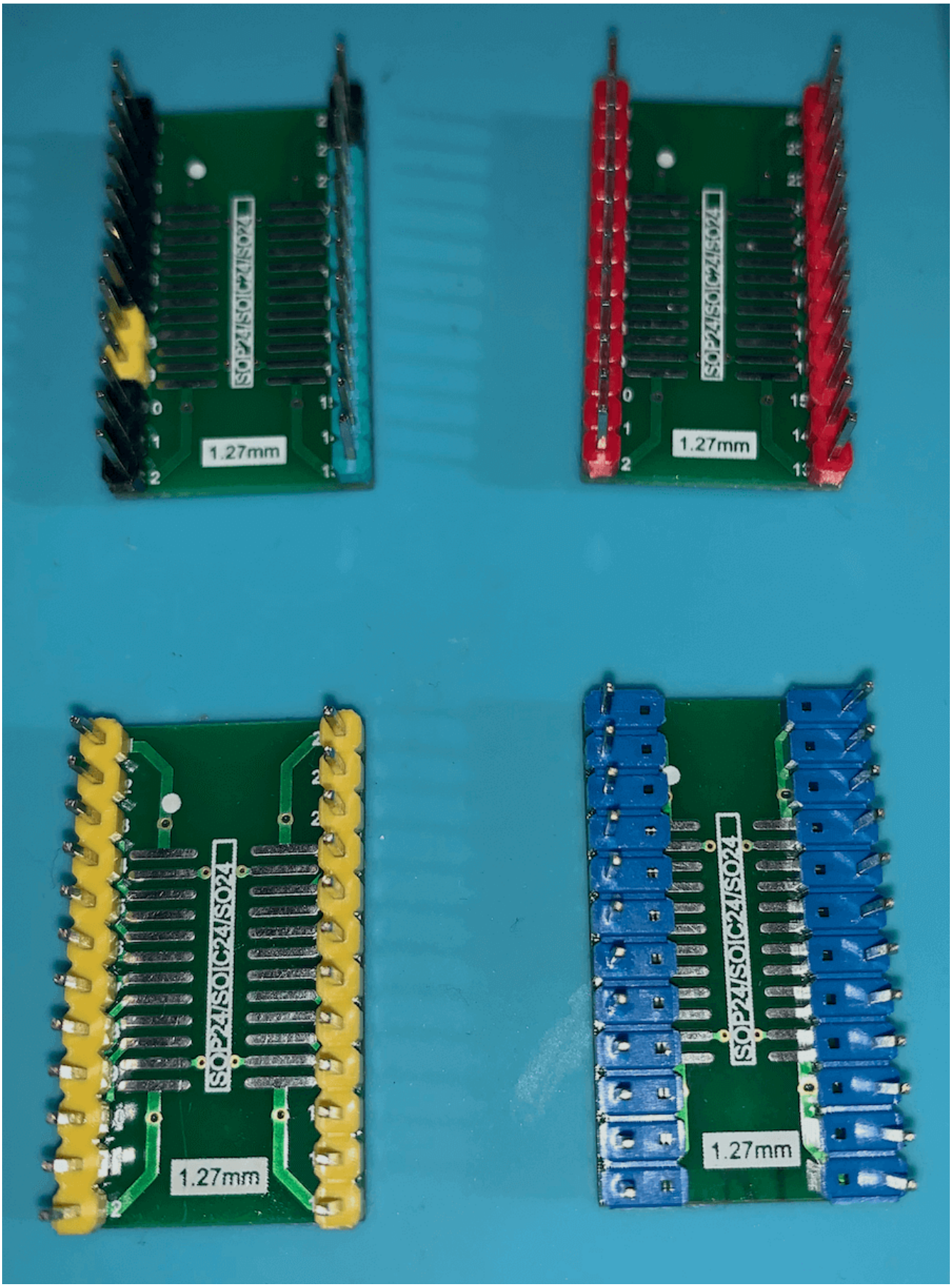
SI4735 soldered on adapter


24 23 22 21 20 19 18 17 16 15 14 13
4735D60GU
1629DC600H
TSSOP24
SSOP24
0.65mm
1 2 3 4 5 6 7 8 9 10 11 12

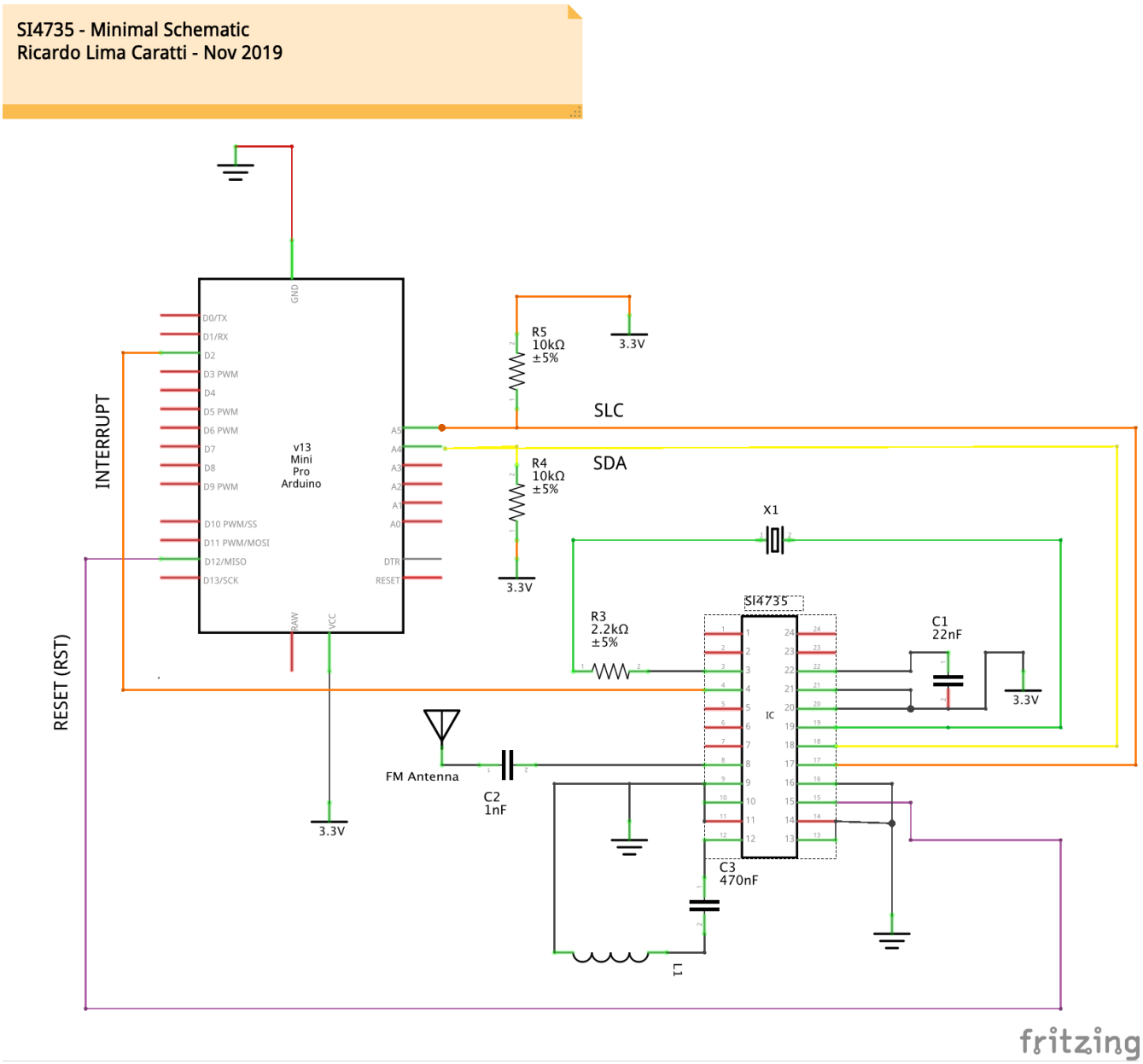

24 23 22 21 20 19 18 17 16 15 14 13
4735D60GU
1629DC600H
TSSOP24
SSOP24
0.65mm
1 2 3 4 5 6 7 8 9 10 11 12

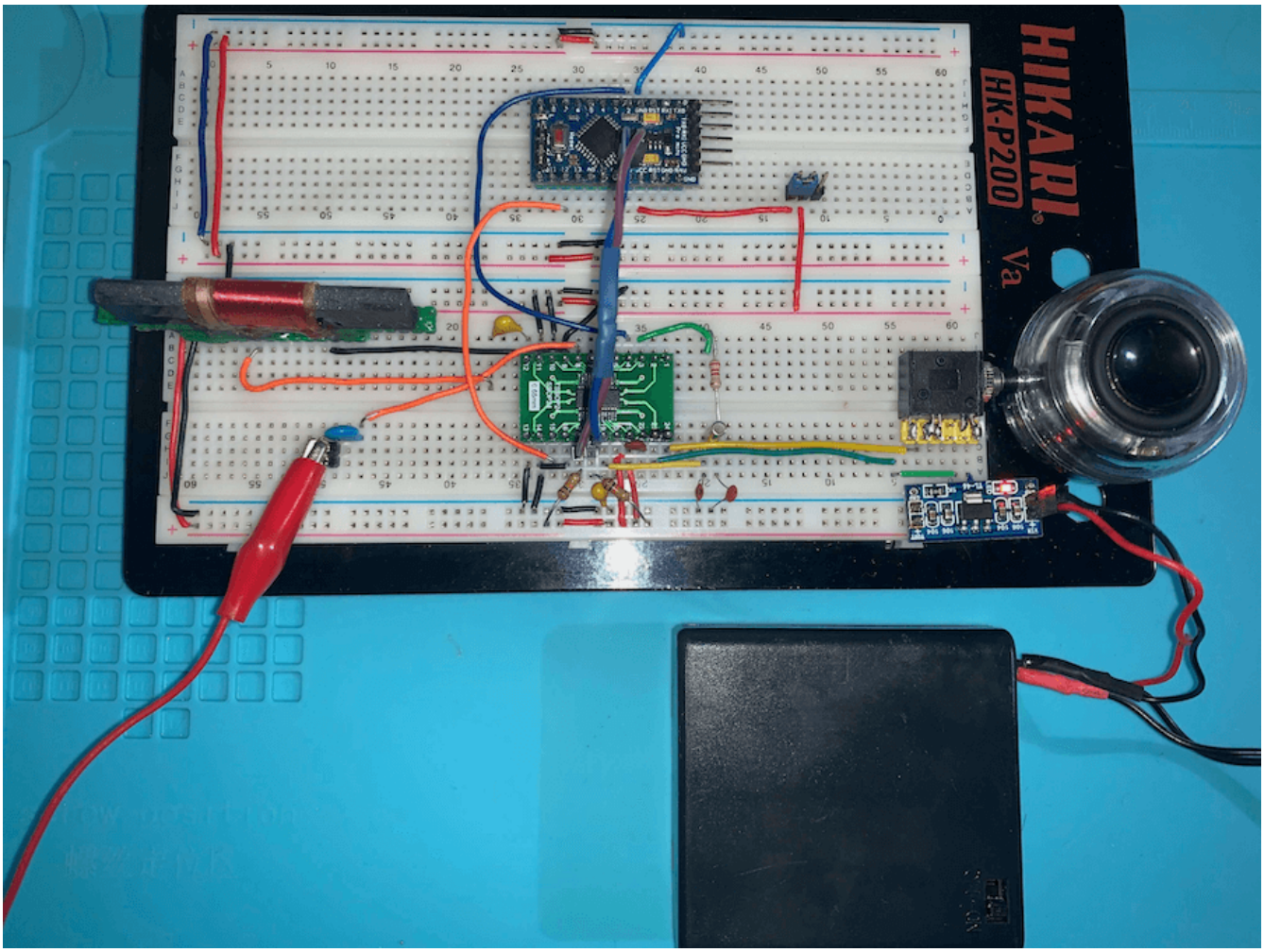


SI4735 on adapter 04


SOP24/SOIC24/SO24
1.27mm
SOP24/SOIC24/SO24
1.27mm
SOP24/SOIC24/SO24
1.27mm
SOP24/SOIC24/SO24
1.27mm

Protoboard


SI4735 - Minimal Schematic
Ricardo Lima Caratti - Nov 2019
INTERRUPT
RESET (RST)
v13
Mini
Pro
Arduino
D0/TX
D1/RX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
GND
RAW
VCC
A5
A4
A3
A2
A1
A0
DTR
RESET
R5
10kΩ
±5%
R4
10kΩ
±5%
R3
2.2kΩ
±5%
3.3V
SLC
SDA
X1
SI4735
IC
C1
22nF
C2
1nF
C3
470nF
FM Antenna
L1
fritzing



API Documentation

Defined Data Types and Structures

To make the SI4735 device easier to deal, some defined data types were built to handle byte and bits responses.

```
/*
 * Power Up arguments data type
 * See Si47XX PROGRAMMING GUIDE; AN332; pages 64 and 65
 */
typedef union {
    struct
    {
        // ARG1
        byte FUNC       : 4;  // Function (0 = FM Receive; 1–14 =
Reserved; 15 = Query Library ID)
        byte XOSCEN     : 1;  // Crystal Oscillator Enable (0 =
crystal oscillator disabled; 1 = Use crystal oscillator and and
OPMODE=ANALOG AUDIO) .
        byte PATCH      : 1;  // Patch Enable (0 = Boot normally; 1
= Copy non-volatile memory to RAM).
        byte GPO2OEN    : 1;  // GPO2 Output Enable (0 = GPO2
output disabled; 1 = GPO2 output enabled).
        byte CTSIEN     : 1;  // CTS Interrupt Enable (0 = CTS
```

```
interrupt disabled; 1 = CTS interrupt enabled).
        // ARG2
        byte OPMODE; // Application Setting. See page 65
    } arg;
    byte raw[2]; // same arg memory position, so same content.
} si473x_powerup;

/*
 * Represents how the frequency is stored in the si4735.
 * It helps to convert frequency in unsigned int to two bytes
(FREQL and FREQH)
 */
typedef union {
    struct
    {
        byte FREQL; // Tune Frequency High Byte.
        byte FREQH; // Tune Frequency Low Byte.
    } raw;
    unsigned value;
} si47x_frequency;

/*
 *  Represents searching for a valid frequency data type.
 */
typedef union {
    struct
    {
        byte RESERVED1 : 2;
        byte WRAP : 1;   // Determines whether the seek should Wrap
= 1, or Halt = 0 when it hits the band limit.
        byte SEEKUP : 1; // Determines the direction of the search,
either UP = 1, or DOWN = 0.
        byte RESERVED2 : 4;
    } arg;
    byte raw;
} si47x_seek;
```

Public Methods

setup

```
/*
 * Starts the Si473X device.
 *
 * @param byte resetPin Digital Arduino Pin used to RESET command
 * @param byte interruptPin interrupt Arduino Pin (see your Arduino
pinout).
 * @param byte defaultFunction
 */
void SI4735::setup(byte resetPin, byte interruptPin, byte
defaultFunction)
```

Example of using setup

```
#include <SI4735.h>

#define INTERRUPT_PIN 2
#define RESET_PIN 12

SI4735 si4735;

void setup()
{
    si4735.setup(RESET_PIN, INTERRUPT_PIN, FM_FUNCTION);
}
```

See full example (https://github.com/pu2clr/SI4735/blob/master/examples/SI4735_POC/SI4735_POC.ino)

setPowerUp

```
/*
 * Set the Power Up parameters for si473X.
 * Use this method to chenge the defaul behavior of the Si473X. Use it before PowerUp()
 * See See Si47XX PROGRAMMING GUIDE; AN332; pages 65 and 129
 *
 * @param byte CTSIEN sets Interrupt anabled or disabled (1 = anabled and 0 = disabled )
 * @param byte GPO2OEN sets GP02 Si473X pin enabled (1 = anabled and 0 = disabled )
 * @param byte PATCH  Used for firmware patch updates. Use it always 0 here.
 * @param byte XOSCEN byte XOSCEN set external Crystal enabled or disabled
 * @param byte FUNC sets the receiver function have to be used (0 = FM Receive; 1 = AM (LW/MW/SW) Receiver)
 * @param byte OPMODE set the kind of audio mode you want to use.
 */
void SI4735::setPowerUp(byte CTSIEN, byte GPO2OEN, byte PATCH, byte XOSCEN, byte FUNC, byte OPMODE)
```

analogPowerUp

```
/*
 * Powerup in Analog Mode
 * You have to call setPowerUp before call analogPowerUp
 */
void SI4735::analogPowerUp(void)
```

Example of using analogPowerUp

```
    // Set the initial SI473X behavior
    // CTSIEN   1 -> Interrupt anabled;
    // GPO2OEN  1 -> GPO2 Output Enable;
    // PATCH    0 -> Boot normally;
```

```
    // XOSCEN   1 -> Use external crystal oscillator;
    // FUNC     defaultFunction = 0 = FM Receive; 1 = AM (LW/MW/SW)
Receiver.
    // OPMODE   SI473X_ANALOG_AUDIO = 00000101 = Analog audio
outputs (LOUT/ROUT).

    setPowerUp(1, 1, 0, 1, defaultFunction, SI473X_ANALOG_AUDIO);
    analogPowerUp();
```

setFrequency

```
/*
 * Set the frequency to the corrent function of the Si4735 (AM or
FM)
 * You have to call setup or setPowerUp before call setFrequency.
 *
 * @param unsigned freq Is the frequency to change. For example, FM
=> 10390 = 103.9 MHz; AM => 810 = 810 KHz.
 */
void SI4735::setFrequency(unsigned freq)
```

Example of using setFrequency

```
    si4735.setFM();
    si4735.setFrequency(fm_freq);
    showStatus(fm_freq,"MHz");
```

See full example (https://github.com/pu2clr/SI4735/blob/master/examples/SI4735_POC/SI4735_POC.ino)

seekStation

```
/*
 * Look for a station
 * See Si47XX PROGRAMMING GUIDE; AN332; page 72
 *
 * @param SEEKUP Seek Up/Down. Determines the direction of the
search, either UP = 1, or DOWN = 0.
 * @param Wrap/Halt. Determines whether the seek should Wrap = 1,
or Halt = 0 when it hits the band limit.
 */
void SI4735::seekStation(byte SEEKUP, byte WRAP)
```

Example of using seekStation

```
    si4735.seekStation(1,1);
```

See full example (https://github.com/pu2clr/SI4735/blob/master/examples/SI4735_POC/SI4735_POC.ino)

setAM

```
/*
 * Set the radio to AM function. It means: LW, MW and SW.
 */
void SI4735::setAM()
```

setFM

```
/*
 * Set the radio to FM function
 */
void SI4735::setFM()
```

Example of using setAM() and setFM()

```
    switch (key)
    {
    case 'A':
      si4735.setAM();
      si4735.setFrequency(am_freq);
      break;
    case 'F':
      si4735.setFM();
      si4735.setFrequency(fm_freq);
      break;
      .
      .
      .
```

See full example (https://github.com/pu2clr/SI4735/blob/master/examples/SI4735_POC/SI4735_POC.ino)

setVolume

```
/*
 * Set the volume level
 * @param byte volume (domain: 0 - 63)
 */
void SI4735::setVolume(byte volume)
```

Example of using setVolume()

```
  si4735.setVolume(45);
```

See full example (https://github.com/pu2clr/SI4735/blob/master/examples/SI4735_POC/SI4735_POC.ino)

volumeUp

```
/*
 *  Set sound volume level Up
```

```
 */
void SI4735::volumeUp()
```

volumeDown

```
/*
 *  Set sound volume level Down
 */
void SI4735::volumeDown()
```

Example of using volumeUp() and volumeDown()

```
    switch (key)
    {
    case '+':
        si4735.volumeUp();
        break;
    case '-':
        si4735.volumeDown();
        break;
    .
    .
    .
```

[See full example (https://github.com/pu2clr/SI4735/blob/master/examples/SI4735_POC/SI4735_POC.ino)](https://github.com/pu2clr/SI4735/blob/master/examples/SI4735_POC/SI4735_POC.ino)

getStatus

```
/*
 * Gets the current status  of the Si4735 (AM or FM)
 * See Si47XX PROGRAMMING GUIDE; AN332; pages 73 (FM) and 139 (AM)
 *
 */
void SI4735::getStatus()


/*
 * Gets the current status  of the Si4735 (AM or FM)
 * See Si47XX PROGRAMMING GUIDE; AN332; pages 73 (FM) and 139 (AM)
 *
 * @param byte INTACK Seek/Tune Interrupt Clear. If set, clears the
seek/tune complete interrupt status indicator;
 * @param byte CANCEL Cancel seek. If set, aborts a seek currently
in progress;
 *
 */
void SI4735::getStatus(byte INTACK, byte CANCEL) {
```

getTuneCompleteTriggered

```
/*
```

```
 * Tune complete has been triggered (STCINT)
 */
inline bool SI4735::getTuneCompleteTriggered()
```

getSignalQualityInterrup

```
/*
 * Gets Received Signal Quality Interrupt(RSQINT)
 *
 */
inline bool SI4735::getSignalQualityInterrup()
```

getRadioDataSystemInterrupt

```
/*
 * Gets Radio Data System (RDS) Interrupt
 */
inline bool SI4735::getRadioDataSystemInterrupt()
```

getStatusError

```
/*
 * Return the Error flag (true or false) of status of the least
Tune or Seek
 * See Si47XX PROGRAMMING GUIDE; AN332; pages 63
 * @return true or false
 */
inline bool SI4735::getStatusError() {
```

getStatusCTS

```
/*
 * Gets the Error flag of status response
 * See Si47XX PROGRAMMING GUIDE; AN332; pages 63
 */
inline bool SI4735::getStatusCTS()
```

getACFIndicator

```
/*
 * Returns true if the AFC rails (AFC Rail Indicator).
 */
inline bool SI4735::getACFIndicator()
```

getBandLimit

```
/*
 * Returns true if a seek hit the band limit (WRAP = 0 in
FM_START_SEEK) or
```

```
 * wrapped to the original frequency (WRAP = 1).
 */
inline bool SI4735::getBandLimit()
```

getReceivedSignalStrengthIndicator

```
/*
 * Received Signal Strength Indicator.
 * This byte contains the receive signal strength when tune is
complete (dBμV).
 */
inline byte SI4735::getReceivedSignalStrengthIndicator()
```

getStatusSNR

```
/*
 * SNR.
 * This byte contains the SNR metric when tune is complete (dB).
 */
inline byte SI4735::getStatusSNR()
```

getStatusMULT

```
/*
 * Multipath.
 * This byte contains the multipath metric when tune is complete.
 * Multipath indi- cator is available only for Si474x, Si4706-C30
and later and
 * Si4704/05/30/31/34/35/84/85-D50 and later.
 */
inline byte SI4735::getStatusMULT()
```

getAntennaTuningCapacitor

```
/*
 * Read Antenna Tuning Capacitor (Si4704/05/06/2x only).
 * Returns a byte that contains the current antenna tuning
capacitor value.
 */
inline byte SI4735::getAntennaTuningCapacitor()
```

getStatusValid

```
/*
 * Returns true if the channel is currently valid as determined by
the seek/tune properties (0x1403, 0x1404, 0x1108)
 * and would have been found during a Seek.
 * See Si47XX PROGRAMMING GUIDE; AN332; pages 63
 */
inline bool SI4735::getStatusValid()
```

getFirmwarePN

```
/*
 * Returns the final 2 digits of Part Number (HEX)
 * See Si47XX PROGRAMMING GUIDE; AN332; page 66
 */
inline byte SI4735::getFirmwarePN()
```

getFirmwareFWMAJOR

```
/*
 * Returns the Firmware Major Revision (ASCII).
 * See Si47XX PROGRAMMING GUIDE; AN332; page 66
 */
inline byte SI4735::getFirmwareFWMAJOR()
```

getFirmwareFWMINOR

```
/*
 * Returns the Firmware Minor Revision (ASCII).
 * See Si47XX PROGRAMMING GUIDE; AN332; page 66
 */
inline byte SI4735::getFirmwareFWMINOR()
```

getFirmwarePATCHH

```
/*
 * Returns the Patch ID High Byte (HEX).
 * See Si47XX PROGRAMMING GUIDE; AN332; page 66
 */
inline byte SI4735::getFirmwarePATCHH()
```

getFirmwarePATCHL

```
/*
 * Returns the Patch ID Low Byte (HEX).
 * See Si47XX PROGRAMMING GUIDE; AN332; page 66
 */
inline byte SI4735::getFirmwarePATCHL()
```

getFirmwareCMPMAJOR

```
/*
 * Returns the Component Major Revision (ASCII).
 * See Si47XX PROGRAMMING GUIDE; AN332; page 66
 */
inline byte SI4735::getFirmwareCMPMAJOR()
```

getFirmwareCMPMINOR

```
/*
 * Returns the Component Minor Revision (ASCII).
 * See Si47XX PROGRAMMING GUIDE; AN332; page 66
 */
inline byte SI4735::getFirmwareCMPMINOR()
```

getFirmwareCHIPREV

```
/*
 * Returns the Chip Revision (ASCII).
 * See Si47XX PROGRAMMING GUIDE; AN332; page 66
 */
inline byte SI4735::getFirmwareCHIPREV()
```

References

1. Silicon Labs Si4737 WB/AM/FM Stereo/RDS single-chip receiver HAL library for Arduino (https://github.com/rickeywang/Si4737_i2c)
2. BROADCAST AM/FM/SW/LW RADIO RECEIVER (https://www.silabs.com/documents/public/data-sheets/Si4730-31-34-35-D60.pdf)
3. SI47XX PROGRAMMING GUIDE (https://www.silabs.com/documents/public/application-notes/AN332.pdf)

Videos

 CLick here (https://youtu.be/i77U8WHXc18)