Si4735 Arduino Library

AUTHOR
Version 1.1.8
Thu Apr 2 2020

Table of Contents

Table of contents

Module Index

Modules

Here is a list of all modules:

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Module Documentation

Deal with Interrupt

Deal with Interrupt

RDS Data types

Classes

- union **si47x_rqs_status**
- union **si47x_rds_command**
- union **si47x_rds_status**
- union **si47x_rds_int_source**
- union **si47x_rds_config**
- union **si47x_rds_blocka**
- union **si47x_rds_blockb**
- union **si47x_rds_date_time**

Detailed Description

Receiver Status and Setup

Classes

- union **si47x_agc_status**
- union **si47x_agc_overrride**
- union **si47x_bandwidth_config**
- union **si47x_ssb_mode**
- union **si4735_digital_output_format**
- struct **si4735_digital_output_sample_rate**

Detailed Description

SI473X data types

The goal here is separate data from code. The Si47XX family works with many internal data that can be represented by data structure or defined data type in C/C++. These C/C++ resources have been used widely here.

This aproach made the library easier to build and maintain. Each data structure created here has its reference (name of the document and page on which it was based). In other words, to make the SI47XX device easier to deal, some defined data types were created to handle byte and bits to process commands, properties and responses. These data types will be usefull to deal with SI473X.

Classes

- union **si473x_powerup**
- union **si47x_frequency**
- union **si47x_antenna_capacitor**
- union **si47x_set_frequency**
- union **si47x_seek**
- union **si47x_response_status**
- union **si47x_firmware_information**
- union **si47x_firmware_query_library**
- union **si47x_tune_status**
- union **si47x_property**

Detailed Description

The goal here is separate data from code. The Si47XX family works with many internal data that can be represented by data structure or defined data type in C/C++. These C/C++ resources have been used widely here.

This aproach made the library easier to build and maintain. Each data structure created here has its reference (name of the document and page on which it was based). In other words, to make the SI47XX device easier to deal, some defined data types were created to handle byte and bits to process commands, properties and responses. These data types will be usefull to deal with SI473X.

Class Documentation

SI4735 Class Reference

`#include <SI4735.h>`

Public Member Functions

- **SI4735** ()
- void **reset** (void)
- void **waitToSend** (void)
- void **setup** (uint8_t resetPin, uint8_t defaultFunction)
- void **setup** (uint8_t resetPin, int interruptPin, uint8_t defaultFunction, uint8_t audioMode=SI473X_ANALOG_AUDIO)
- void **setPowerUp** (uint8_t CTSIEN, uint8_t GPO2OEN, uint8_t PATCH, uint8_t XOSCEN, uint8_t FUNC, uint8_t OPMODE)
- void **radioPowerUp** (void)
- void **analogPowerUp** (void)
- void **powerDown** (void)
- void **setFrequency** (uint16_t)
- void **getStatus** ()
- void **getStatus** (uint8_t, uint8_t)
- uint16_t **getFrequency** (void)
- uint16_t **getCurrentFrequency** ()
- bool **getSignalQualityInterrupt** ()
- bool **getRadioDataSystemInterrupt** ()
  Gets Received Signal Quality Interrupt(RSQINT)

- bool **getTuneCompleteTriggered** ()
  Gets Radio Data System (RDS) Interrupt.

- bool **getStatusError** ()
  Seek/Tune Complete Interrupt; 1 = Tune complete has been triggered.

- bool **getStatusCTS** ()
  Return the Error flag (true or false) of status of the least Tune or Seek.

- bool **getACFIndicator** ()
  Gets the Error flag of status response.

- bool **getBandLimit** ()
  Returns true if the AFC rails (AFC Rail Indicator).

- bool **getStatusValid** ()
  Returns true if a seek hit the band limit (WRAP = 0 in FM_START_SEEK) or wrapped to the original frequency(WRAP = 1).

- uint8_t **getReceivedSignalStrengthIndicator** ()
  Returns true if the channel is currently valid as determined by the seek/tune properties (0x1403, 0x1404, 0x1108)

- uint8_t **getStatusSNR** ()
  Returns integer Received Signal Strength Indicator (dBÎ¼V).

- uint8_t **getStatusMULT** ()
  Returns integer containing the SNR metric when tune is complete (dB).

- uint8_t **getAntennaTuningCapacitor** ()
  Returns integer containing the multipath metric when tune is complete.

- void **getAutomaticGainControl** ()
  Returns integer containing the current antenna tuning capacitor value.

- void **setAvcAmMaxGain** (uint8_t gain)
- void **setAutomaticGainControl** (uint8_t AGCDIS, uint8_t AGCIDX)
- void **getCurrentReceivedSignalQuality** (uint8_t INTACK)
- void **getCurrentReceivedSignalQuality** (void)
- uint8_t **getCurrentSNR** ()
  current receive signal strength (0â€“127 dBÎ¼V).

- bool **getCurrentRssiDetectLow** ()
  current SNR metric (0–127 dB).

- bool **getCurrentRssiDetectHigh** ()
  RSSI Detect Low.

- bool **getCurrentSnrDetectLow** ()
  RSSI Detect High.

- bool **getCurrentSnrDetectHigh** ()
  SNR Detect Low.

- bool **getCurrentValidChannel** ()
  SNR Detect High.

- bool **getCurrentAfcRailIndicator** ()
  Valid Channel.

- bool **getCurrentSoftMuteIndicator** ()
  AFC Rail Indicator.

- uint8_t **getCurrentStereoBlend** ()
  Soft Mute Indicator. Indicates soft mute is engaged.

- bool **getCurrentPilot** ()
  Indicates amount of stereo blend in % (100 = full stereo, 0 = full mono).

- uint8_t **getCurrentMultipath** ()

Indicates stereo pilot presence.

- uint8_t **getCurrentSignedFrequencyOffset** ()
  Contains the current multipath metric. (0 = no multipath; 100 = full multipath)

- bool **getCurrentMultipathDetectLow** ()
  Signed frequency offset (kHz).

- bool **getCurrentMultipathDetectHigh** ()
  Multipath Detect Low.

- bool **getCurrentBlendDetectInterrupt** ()
  Multipath Detect High.

- uint8_t **getFirmwarePN** ()
  Blend Detect Interrupt.

- uint8_t **getFirmwareFWMAJOR** ()
  RESP1 - Part Number (HEX)

- uint8_t **getFirmwareFWMINOR** ()
  RESP2 - Returns the Firmware Major Revision (ASCII).

- uint8_t **getFirmwarePATCHH** ()
  RESP3 - Returns the Firmware Minor Revision (ASCII).

- uint8_t **getFirmwarePATCHL** ()
  RESP4 - Returns the Patch ID High byte (HEX).

- uint8_t **getFirmwareCMPMAJOR** ()
  RESP5 - Returns the Patch ID Low byte (HEX).

- uint8_t **getFirmwareCMPMINOR** ()
  RESP6 - Returns the Component Major Revision (ASCII).

- uint8_t **getFirmwareCHIPREV** ()
  RESP7 - Returns the Component Minor Revision (ASCII).

- void **setVolume** (uint8_t volume)
  RESP8 - Returns the Chip Revision (ASCII).

- uint8_t **getVolume** ()
- void **volumeDown** ()
- void **volumeUp** ()
- void **setAudioMute** (bool off)
  Returns the current volume level.

- void **digitalOutputFormat** (uint8_t OSIZE, uint8_t OMONO, uint8_t OMODE, uint8_t OFALL)
- void **digitalOutputSampleRate** (uint16_t DOSR)
- void **setAM** ()
- void **setFM** ()
- void **setAM** (uint16_t fromFreq, uint16_t toFreq, uint16_t intialFreq, uint16_t step)
- void **setFM** (uint16_t fromFreq, uint16_t toFreq, uint16_t initialFreq, uint16_t step)
- void **setBandwidth** (uint8_t AMCHFLT, uint8_t AMPLFLT)
- void **setFrequencyStep** (uint16_t step)
- void **setTuneFrequencyFast** (uint8_t FAST)
  Returns the FAST tuning status.

- uint8_t **getTuneFrequencyFreeze** ()
  FAST Tuning. If set, executes fast and invalidated tune. The tune status will not be accurate.

- void **setTuneFrequencyFreeze** (uint8_t FREEZE)
  Returns the FREEZE status.

- void **setTuneFrequencyAntennaCapacitor** (uint16_t capacitor)
  Only FM. Freeze Metrics During Alternate Frequency Jump.

- void **frequencyUp** ()
- void **frequencyDown** ()
- bool **isCurrentTuneFM** ()
- void **getFirmware** (void)
- void **seekStation** (uint8_t SEEKUP, uint8_t WRAP)
- void **seekStationUp** ()
- void **seekStationDown** ()
- void **setSeekAmLimits** (uint16_t bottom, uint16_t top)
- void **setSeekAmSpacing** (uint16_t spacing)
- void **setSeekSrnThreshold** (uint16_t value)
- void **setSeekRssiThreshold** (uint16_t value)
- void **setFmBlendStereoThreshold** (uint8_t parameter)
- void **setFmBlendMonoThreshold** (uint8_t parameter)
- void **setFmBlendRssiStereoThreshold** (uint8_t parameter)
- void **setFmBLendRssiMonoThreshold** (uint8_t parameter)
- void **setFmBlendSnrStereoThreshold** (uint8_t parameter)
- void **setFmBLendSnrMonoThreshold** (uint8_t parameter)
- void **setFmBlendMultiPathStereoThreshold** (uint8_t parameter)
- void **setFmBlendMultiPathMonoThreshold** (uint8_t parameter)
- void **setFmStereoOn** ()
- void **setFmStereoOff** ()
- void **RdsInit** ()
- void **setRdsIntSource** (uint8_t RDSNEWBLOCKB, uint8_t RDSNEWBLOCKA, uint8_t RDSSYNCFOUND, uint8_t RDSSYNCLOST, uint8_t RDSRECV)
- void **getRdsStatus** (uint8_t INTACK, uint8_t MTFIFO, uint8_t STATUSONLY)
- void **getRdsStatus** ()
- bool **getRdsSyncLost** ()
  1 = FIFO filled to minimum number of groups

- bool **getRdsSyncFound** ()
  1 = Lost RDS synchronization

- bool **getRdsNewBlockA** ()

1 = Found RDS synchronization

- bool **getRdsNewBlockB** ()
  1 = Valid Block A data has been received.

- bool **getRdsSync** ()
  1 = Valid Block B data has been received.

- bool **getGroupLost** ()
  1 = RDS currently synchronized.

- uint8_t **getNumRdsFifoUsed** ()
  1 = One or more RDS groups discarded due to FIFO overrun.

- void **setRdsConfig** (uint8_t RDSEN, uint8_t BLETHA, uint8_t BLETHB, uint8_t BLETHC, uint8_t BLETHD)
  RESP3 - RDS FIFO Used; Number of groups remaining in the RDS FIFO (0 if empty).

- uint16_t **getRdsPI** (void)
- uint8_t **getRdsGroupType** (void)
- uint8_t **getRdsFlagAB** (void)
- uint8_t **getRdsVersionCode** (void)
- uint8_t **getRdsProgramType** (void)
- uint8_t **getRdsTextSegmentAddress** (void)
- char * **getRdsText** (void)
- char * **getRdsText0A** (void)
- char * **getRdsText2A** (void)
- char * **getRdsText2B** (void)
- char * **getRdsTime** (void)
- void **getNext2Block** (char *)
- void **getNext4Block** (char *)
- void **ssbSetup** ()
- void **setSSBBfo** (int offset)
- void **setSSBConfig** (uint8_t AUDIOBW, uint8_t SBCUTFLT, uint8_t AVC_DIVIDER, uint8_t AVCEN, uint8_t SMUTESEL, uint8_t DSP_AFCDIS)
- void **setSSB** (uint8_t usblsb)
- void **setSSBAudioBandwidth** (uint8_t AUDIOBW)
- void **setSSBAutomaticVolumeControl** (uint8_t AVCEN)
- void **setSBBSidebandCutoffFilter** (uint8_t SBCUTFLT)
- void **setSSBAvcDivider** (uint8_t AVC_DIVIDER)
- void **setSSBDspAfc** (uint8_t DSP_AFCDIS)
- void **setSSBSoftMute** (uint8_t SMUTESEL)
- **si47x_firmware_query_library queryLibraryId** ()
- void **patchPowerUp** ()
- bool **downloadPatch** (const uint8_t *ssb_patch_content, const uint16_t ssb_patch_content_size)
- bool **downloadPatch** (int eeprom_i2c_address)
- void **ssbPowerUp** ()
- void **setI2CStandardMode** (void)
  Sets I2C buss to 10KHz.

- void **setI2CFastMode** (void)
  Sets I2C buss to 100KHz.

- void **setI2CFastModeCustom** (long value=500000)
  Sets I2C buss to 400KHz.

- void **setDeviceI2CAddress** (uint8_t senPin)
- int16_t **getDeviceI2CAddress** (uint8_t resetPin)
- void **setDeviceOtherI2CAddress** (uint8_t i2cAddr)

Protected Member Functions

- void **waitInterrupr** (void)
- void **sendProperty** (uint16_t propertyValue, uint16_t param)
- void **sendSSBModeProperty** ()
- void **disableFmDebug** ()
- void **clearRdsBuffer2A** ()
- void **clearRdsBuffer2B** ()
- void **clearRdsBuffer0A** ()

Protected Attributes

- char **rds_buffer2B** [33]
  RDS Radio Text buffer - Program Information.

- char **rds_buffer0A** [9]
  RDS Radio Text buffer - Station Informaation.

- char **rds_time** [20]
  RDS Basic tuning and switching information (Type 0 groups)

- uint8_t **currentAvcAmMaxGain** = 48
  Store the last mode used.

Detailed Description

SI4735 Class definition

Definition at line 833 of file SI4735.h.

Constructor & Destructor Documentation

SI4735::SI4735 ()

This is a library for the **SI4735**, BROADCAST AM/FM/SW RADIO RECEIVER, IC from Silicon Labs for the Arduino development environment. It works with I2C protocol. This library is intended to provide an easier interface for controlling the **SI4735**.

See also

documentation on `https://github.com/pu2clr/SI4735`.

also: Si47XX PROGRAMMING GUIDE; AN332 AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; AMENDMENT FOR SI4735-D60 SSB AND NBFM PATCHES

Pay attention: According to Si47XX PROGRAMMING GUIDE; AN332; page 207, "For write operations, the system controller next sends a data byte on SDIO, which is captured

by the device on rising edges of SCLK. The device acknowledges each data byte by driving SDIO low for one cycle on the next falling edge of SCLK. The system controller may write up to 8 data bytes in a single 2-wire transaction. The first byte is a command, and the next seven bytes are arguments. Writing more than 8 bytes results in unpredictable device behavior". So, If you are extending this library, consider that restriction presented earlier.

ATTENTION: Some methods were implemented usin inline resource. Inline methods are implemented in **SI4735.h**

By Ricardo Lima Caratti, Nov 2019. Construct a new **SI4735::SI4735** object

Definition at line 30 of file SI4735.cpp.

Member Function Documentation

void SI4735::analogPowerUp (void)

Powerup in Analog Mode. It will be deprecated. Consider use radioPowerUp instead. Actually this function works fo Digital and Analog modes. You have to call setPowerUp method before.

Definition at line 225 of file SI4735.cpp.

void SI4735::clearRdsBuffer0A () [protected]

Clear RDS buffer 0A (text)

Definition at line 1232 of file SI4735.cpp.

void SI4735::clearRdsBuffer2A () [protected]

Clear RDS buffer 2A (text)

Definition at line 1213 of file SI4735.cpp.

void SI4735::clearRdsBuffer2B () [protected]

Clear RDS buffer 2B (text)

Definition at line 1223 of file SI4735.cpp.

void SI4735::digitalOutputFormat (uint8_t *OSIZE*, uint8_t *OMONO*, uint8_t *OMODE*, uint8_t *OFALL*)

Digital Audio Setup Configures the digital audio output format. Options: DCLK edge, data format, force mono, and sample precision.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 195.

Parameters

uint8_t	OSIZE Digital Output Audio Sample Precision (0=16 bits, 1=20 bits, 2=24 bits, 3=8bits).
uint8_t	OMONO Digital Output Mono Mode (0=Use mono/stereo blend).
uint8_t	OMODE Digital Output Mode (0=I2S, 6 = Left-justified, 8 = MSB at second DCLK after DFS pulse, 12 = MSB at first DCLK after DFS pulse).
uint8_t	OFALL Digital Output DCLK Edge (0 = use DCLK rising edge, 1 = use DCLK falling edge)

Definition at line 777 of file SI4735.cpp.

void SI4735::digitalOutputSampleRate (uint16_t *DOSR*)

Enables digital audio output and configures digital audio output sample rate in samples per second (sps).

See also

Si47XX PROGRAMMING GUIDE; AN332; page 196.

Parameters

uint16_t	DOSR Digital Output Sample Rate(32–48 ksps .0 to disable digital audio output).

Definition at line 794 of file SI4735.cpp.

void SI4735::disableFmDebug () `[protected]`

There is a debug feature that remains active in Si4704/05/3x-D60 firmware which can create periodic noise in audio. Silicon Labs recommends you disable this feature by sending the following bytes (shown here in hexadecimal form): 0x12 0x00 0xFF 0x00 0x00 0x00.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 299.

Definition at line 749 of file SI4735.cpp.

bool SI4735::downloadPatch (const uint8_t * *ssb_patch_content*, const uint16_t *ssb_patch_content_size*)

Transfers the content of a patch stored in a array of bytes to the **SI4735** device. You must mount an array as shown below and know the size of that array as well.

It is importante to say that patches to the **SI4735** are distributed in binary form and have to be transferred to the internal RAM of the device by the host MCU (in this case Arduino). Since the RAM is volatile memory, the patch stored into the device gets lost when you turn off the system. Consequently, the content of the patch has to be transferred again to the device each time after turn on the system or reset the device.

The disadvantage of this approach is the amount of memory used by the patch content. This may limit the use of other radio functions you want implemented in Arduino.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 64 and 215-220.

Example of content: const PROGMEM uint8_t ssb_patch_content_full[] = { // SSB patch for whole SSBRX full download 0x15, 0x00, 0x0F, 0xE0, 0xF2, 0x73, 0x76, 0x2F, 0x16, 0x6F, 0x26, 0x1E, 0x00, 0x4B, 0x2C, 0x58, 0x16, 0xA3, 0x74, 0x0F, 0xE0, 0x4C, 0x36, 0xE4, 0x16, 0x3B, 0x1D, 0x4A, 0xEC, 0x36, 0x28, 0xB7, 0x16, 0x00, 0x3A, 0x47, 0x37, 0x00, 0x00, 0x00, 0x15, 0x00, 0x00, 0x00, 0x00, 0x00, 0x9D, 0x29};

const int size_content_full = sizeof ssb_patch_content_full;

Parameters

ssb_patch_content	point to array of bytes content patch.
ssb_patch_content_size	array size (number of bytes). The maximum size allowed for a patch is 15856 bytes

Returns

false if an error is found.

Definition at line 2169 of file SI4735.cpp.

bool SI4735::downloadPatch (int *eeprom_i2c_address*)

Under construction... Transfers the content of a patch stored in a eeprom to the **SI4735** device.

TO USE THIS METHOD YOU HAVE TO HAVE A EEPROM WRITEN WITH THE PATCH CONTENT

See also

the sketch write_ssb_patch_eeprom.ino (TO DO)

Parameters

eeprom_i2c_address	

Returns

false if an error is found.

Definition at line 2228 of file SI4735.cpp.

void SI4735::frequencyDown ()

Decrements the current frequency on current band/function by using the current step.

See also

setFrequencyStep

Definition at line 442 of file SI4735.cpp.

void SI4735::frequencyUp ()

Increments the current frequency on current band/function by using the current step.

See also

setFrequencyStep()

Definition at line 427 of file SI4735.cpp.

void SI4735::getAutomaticGainControl ()

Returns integer containing the current antenna tuning capacitor value.

Queries AGC STATUS

See also

Si47XX PROGRAMMING GUIDE; AN332; For FM page 80; for AM page 142.

AN332 REV 0.8 Universal Programming Guide Amendment for SI4735-D60 SSB and NBFM patches; page 18.

After call this method, you can call isAgcEnabled to know the AGC status and getAgcGainIndex to know the gain index value.

Definition at line 885 of file SI4735.cpp.

uint16_t SI4735::getCurrentFrequency ()

Gets the current frequency saved in memory. Unlike getFrequency, this method gets the current frequency recorded after the last setFrequency command. This method avoids bus traffic and CI processing. However, you can not get others status information like RSSI.

See also

getFrequency()

Definition at line 829 of file SI4735.cpp.

void SI4735::getCurrentReceivedSignalQuality (uint8_t *INTACK*)

Queries the status of the Received Signal Quality (RSQ) of the current channel. This method sould be called berore call getCurrentRSSI(), **getCurrentSNR()** etc. Command FM_RSQ_STATUS

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 75 and 141

Parameters

INTACK	Interrupt Acknowledge. 0 = Interrupt status preserved; 1 = Clears RSQINT, BLENDINT, SNRHINT, SNRLINT, RSSIHINT, RSSILINT, MULTHINT, MULTLINT.

Definition at line 974 of file SI4735.cpp.

void SI4735::getCurrentReceivedSignalQuality (void)

Queries the status of the Received Signal Quality (RSQ) of the current channel Command FM_RSQ_STATUS

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 75 and 141

Parameters

INTACK	Interrupt Acknowledge. 0 = Interrupt status preserved; 1 = Clears RSQINT, BLENDINT, SNRHINT, SNRLINT, RSSIHINT, RSSILINT, MULTHINT, MULTLINT.

Definition at line 1020 of file SI4735.cpp.

int16_t SI4735::getDeviceI2CAddress (uint8_t *resetPin*)

Scans for two possible addresses for the Si47XX (0x11 or 0x63) This function also sets the system to the found I2C bus address of Si47XX.

You do not need to use this function if the SEN PIN is configured to ground (GND). The default I2C address is 0x11. Use this function if you do not know how the SEN pin is configured.

Parameters

uint8_t	resetPin MCU Mater (Arduino) reset pin

Returns

int16_t 0x11 if the SEN pin of the Si47XX is low or 0x63 if the SEN pin of the Si47XX is HIGH or 0x0 if error.

Definition at line 63 of file SI4735.cpp.

void SI4735::getFirmware (void)

Gets firmware information

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 66, 131

Definition at line 250 of file SI4735.cpp.

uint16_t SI4735::getFrequency (void)

Device Status Information Gets the current frequency of the Si4735 (AM or FM) The method status do it an more. See getStatus below.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 73 (FM) and 139 (AM)

Definition at line 809 of file SI4735.cpp.

void SI4735::getNext2Block (char * *c*)

Process data received from group 2B

Parameters

c	char array reference to the "group 2B" text

Definition at line 1506 of file SI4735.cpp.

void SI4735::getNext4Block (char * *c*)

Process data received from group 2A

Parameters

c	char array reference to the "group 2A" text

Definition at line 1538 of file SI4735.cpp.

uint8_t SI4735::getRdsFlagAB (void)

Returns the current Text Flag A/B

Returns

uint8_t

Definition at line 1440 of file SI4735.cpp.

uint8_t SI4735::getRdsGroupType (void)

Returns the Group Type (extracted from the Block B)

Definition at line 1424 of file SI4735.cpp.

uint16_t SI4735::getRdsPI (void)

Returns the programa type. Read the Block A content

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 77 and 78

Returns

BLOCKAL

Definition at line 1412 of file SI4735.cpp.

uint8_t SI4735::getRdsProgramType (void)

Returns the Program Type (extracted from the Block B)

See also

`https://en.wikipedia.org/wiki/Radio_Data_System`

Returns

program type (an integer betwenn 0 and 31)

Definition at line 1491 of file SI4735.cpp.

void SI4735::getRdsStatus ()

Gets RDS Status. Same result of calling getRdsStatus(0,0,0);

See also

SI4735::getRdsStatus(uint8_t INTACK, uint8_t MTFIFO, uint8_t STATUSONLY)

Please, call **getRdsStatus(uint8_t INTACK, uint8_t MTFIFO, uint8_t STATUSONLY)** instead **getRdsStatus()** if you want other behaviour

Definition at line 1397 of file SI4735.cpp.

void SI4735::getRdsStatus (uint8_t *INTACK*, uint8_t *MTFIFO*, uint8_t *STATUSONLY*)

Gets the RDS status. Store the status in currentRdsStatus member. RDS COMMAND FM_RDS_STATUS

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 55 and 77

Parameters

INTACK	Interrupt Acknowledge; 0 = RDSINT status preserved. 1 = Clears RDSINT.
MTFIFO	0 = If FIFO not empty, read and remove oldest FIFO entry; 1 = Clear RDS Receive FIFO.
STATUSONLY	Determines if data should be removed from the RDS FIFO.

Definition at line 1350 of file SI4735.cpp.

char * SI4735::getRdsText (void)

Gets the RDS Text when the message is of the Group Type 2 version A

Returns

char* The string (char array) with the content (Text) received from group 2A

Definition at line 1572 of file SI4735.cpp.

char * SI4735::getRdsText0A (void)

Gets the station name and other messages.

Returns

char* should return a string with the station name. However, some stations send other kind of messages

Definition at line 1594 of file SI4735.cpp.

char * SI4735::getRdsText2A (void)

Gets the Text processed for the 2A group

Returns

char* string with the Text of the group A2

Definition at line 1625 of file SI4735.cpp.

char * SI4735::getRdsText2B (void)

Gets the Text processed for the 2B group

Returns

char* string with the Text of the group AB

Definition at line 1657 of file SI4735.cpp.

uint8_t SI4735::getRdsTextSegmentAddress (void)

Returns the address of the text segment. 2A - Each text segment in version 2A groups consists of four characters. A messages of this group can be have up to 64 characters. 2B - In version 2B groups, each text segment consists of only two characters. When the current RDS status is using this version, the maximum message length will be 32 characters.

Returns

uint8_t the address of the text segment.

Definition at line 1460 of file SI4735.cpp.

char * SI4735::getRdsTime (void)

Gets the RDS time and date when the Group type is 4

Returns

char* a string with hh:mm +/- offset

Definition at line 1688 of file SI4735.cpp.

uint8_t SI4735::getRdsVersionCode (void)

Gets the version code (extracted from the Block B)

Returns

0=A or 1=B

Definition at line 1474 of file SI4735.cpp.

bool SI4735::getSignalQualityInterrupt () `[inline]`

STATUS RESPONSE Set of methods to get current status information. Call them after getStatus or getFrequency or seekStation See Si47XX PROGRAMMING GUIDE; AN332; pages 63

Definition at line 916 of file SI4735.h.

void SI4735::getStatus ()

Gets the current status of the Si4735 (AM or FM)

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 73 (FM) and 139 (AM)

Definition at line 872 of file SI4735.cpp.

void SI4735::getStatus (uint8_t *INTACK*, uint8_t *CANCEL*)

Gets the current status of the Si4735 (AM or FM)

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 73 (FM) and 139 (AM)

Parameters

uint8_t	INTACK Seek/Tune Interrupt Clear. If set, clears the seek/tune complete interrupt status indicator;
uint8_t	CANCEL Cancel seek. If set, aborts a seek currently in progress;

Definition at line 841 of file SI4735.cpp.

uint8_t SI4735::getVolume ()

Gets the current volume level.

See also

setVolume()

Returns

volume (domain: 0 - 63)

Definition at line 1165 of file SI4735.cpp.

bool SI4735::isCurrentTuneFM ()

Returns true if the current function is FM (FM_TUNE_FREQ).

Returns

true if the current function is FM (FM_TUNE_FREQ).

Definition at line 592 of file SI4735.cpp.

void SI4735::patchPowerUp ()

This method can be used to prepare the device to apply SSBRX patch Call queryLibraryId before call this method. Powerup the device by issuing the POWER_UP command with FUNC = 1 (AM/SW/LW Receive)

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 64 and 215-220 and

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE AMENDMENT FOR SI4735-D60 SSB AND NBFM PATCHES; page 7.

Definition at line 2090 of file SI4735.cpp.

void SI4735::powerDown (void)

Moves the device from powerup to powerdown mode. After Power Down command, only the Power Up command is accepted.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 67, 132

Definition at line 236 of file SI4735.cpp.

si47x_firmware_query_library SI4735::queryLibraryId ()

SI47XX PATCH RESOURCES Call it first if you are applying a patch on **SI4735**. Used to confirm if the patch is compatible with the internal device library revision. See Si47XX PROGRAMMING GUIDE; AN332; pages 64 and 215-220.

Returns

a struct **si47x_firmware_query_library** (see it in **SI4735.h**) Query the library information

You have to call this function if you are applying a patch on SI47XX (SI4735-D60)

The first command that is sent to the device is the POWER_UP command to confirm that the patch is compatible with the internal device library revision. The device moves into the powerup mode, returns the reply, and moves into the powerdown mode. The POWER_UP command is sent to the device again to configure the mode of the device and additionally is used to start the patching process. When applying the patch, the PATCH bit in ARG1 of the POWER_UP command must be set to 1 to begin the patching process. [AN332 page 219].

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 214, 215, 216, 219

si47x_firmware_query_library in **SI4735.h**

Returns

si47x_firmware_query_library Library Identification

Definition at line 2053 of file SI4735.cpp.

void SI4735::radioPowerUp (void)

Powerup the Si47XX Before call this function call the setPowerUp to set up the parameters. Parameters you have to set up with setPowerUp

CTSIEN Interrupt anabled or disabled; GPO2OEN GPO2 Output Enable or disabled; PATCH Boot normally or patch; XOSCEN Use external crystal oscillator; FUNC defaultFunction = 0 = FM Receive; 1 = AM (LW/MW/SW) Receiver. OPMODE SI473X_ANALOG_AUDIO (B00000101) or SI473X_DIGITAL_AUDIO (B00001011)

See also

SI4735::setPowerUp()

Si47XX PROGRAMMING GUIDE; AN332; pages 64, 129

Definition at line 206 of file SI4735.cpp.

void SI4735::RdsInit ()

RDS implementation Starts the control variables for RDS.

Definition at line 1201 of file SI4735.cpp.

void SI4735::reset (void)

Reset the SI473X

See also

Si47XX PROGRAMMING GUIDE; AN332;

Definition at line 126 of file SI4735.cpp.

void SI4735::seekStation (uint8_t *SEEKUP*, uint8_t *WRAP*)

Look for a station

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 55, 72, 125 and 137

Parameters

SEEKUP	Seek Up/Down. Determines the direction of the search, either UP = 1, or DOWN = 0.
Wrap/Halt.	Determines whether the seek should Wrap = 1, or Halt = 0 when it hits the band limit.

Definition at line 1033 of file SI4735.cpp.

void SI4735::seekStationDown ()

Search the previous station

See also

seekStation(uint8_t SEEKUP, uint8_t WRAP)

Definition at line 1078 of file SI4735.cpp.

void SI4735::seekStationUp ()

Search for the next station

See also

seekStation(uint8_t SEEKUP, uint8_t WRAP)

Definition at line 1066 of file SI4735.cpp.

void SI4735::sendProperty (uint16_t *propertyValue*, uint16_t *parameter*) [protected]

Sends (sets) property to the SI47XX This method is used for others to send generic properties and params to SI47XX

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 68, 124 and 133.

Definition at line 603 of file SI4735.cpp.

void SI4735::sendSSBModeProperty () [protected]

Just send the property SSB_MOD to the device. Internal use (privete method).

Definition at line 2006 of file SI4735.cpp.

void SI4735::setAM ()

Sets the radio to AM function. It means: LW MW and SW.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 129.

Definition at line 458 of file SI4735.cpp.

void SI4735::setAM (uint16_t *fromFreq*, uint16_t *toFreq*, uint16_t *initialFreq*, uint16_t *step*)

Sets the radio to AM (LW/MW/SW) function.

See also

setAM()

Parameters

fromFreq	minimum frequency for the band
toFreq	maximum frequency for the band
initialFreq	initial frequency
step	step used to go to the next channel

Definition at line 499 of file SI4735.cpp.

void SI4735::setAudioMute (bool *off*)

Returns the current volume level.

Sets the audio on or off

See also

See Si47XX PROGRAMMING GUIDE; AN332; pages 62, 123, 171

Parameters

value	if true, mute the audio; if false unmute the audio.

Definition at line 1153 of file SI4735.cpp.

void SI4735::setAutomaticGainControl (uint8_t *AGCDIS*, uint8_t *AGCIDX*)

If FM, overrides AGC setting by disabling the AGC and forcing the LNA to have a certain gain that ranges between 0 (minimum attenuation) and 26 (maximum attenuation);

If AM/SSB, Overrides the AM AGC setting by disabling the AGC and forcing the gain index that ranges between 0 (minimum attenuation) and 37+ATTN_BACKUP (maximum attenuation);

See also

Si47XX PROGRAMMING GUIDE; AN332; For FM page 81; for AM page 143

Parameters

uint8_t	AGCDIS This param selects whether the AGC is enabled or disabled (0 = AGC enabled; 1 = AGC disabled);
uint8_t	AGCIDX AGC Index (0 = Minimum attenuation (max gain); 1 – 36 = Intermediate attenuation); if >greater than 36 - Maximum attenuation (min gain)).

Definition at line 926 of file SI4735.cpp.

void SI4735::setAvcAmMaxGain (uint8_t *gain*)

Sets the maximum gain for automatic volume control. If no parameter is sent, it will be consider 48dB.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 152

Parameters

uint8_t	gain Select a value between 12 and 192. Defaul value 48dB.

Definition at line 956 of file SI4735.cpp.

void SI4735::setBandwidth (uint8_t *AMCHFLT*, uint8_t *AMPLFLT*)

Selects the bandwidth of the channel filter for AM reception. The choices are 6, 4, 3, 2, 2.5, 1.8, or 1 (kHz). The default bandwidth is 2 kHz. Works only in AM / SSB (LW/MW/SW)

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 125, 151, 277, 181.

Parameters

AMCHFLT	the choices are: 0 = 6 kHz Bandwidth 1 = 4 kHz Bandwidth 2 = 3 kHz Bandwidth 3 = 2 kHz Bandwidth 4 = 1 kHz Bandwidth 5 = 1.8 kHz Bandwidth 6 = 2.5 kHz Bandwidth, gradual roll off 7–15 = Reserved (Do not use).
AMPLFLT	Enables the AM Power Line Noise Rejection Filter.

Definition at line 557 of file SI4735.cpp.

void SI4735::setDeviceI2CAddress (uint8_t *senPin*)

Sets the I2C Bus Address

ATTENTION: The parameter senPin is not the I2C bus address. It is the SEN pin setup of the schematic (eletronic circuit). If it is connected to the ground, call this function with senPin = 0; else senPin = 1. You do not need to use this function if the SEN PIN configured to ground (GND).

The default value is 0x11 (senPin = 0). In this case you have to ground the pin SEN of the SI473X. If you want to change this address, call this function with senPin = 1

Parameters

senPin	0 - when the pin SEN (16 on SSOP version or pin 6 on QFN version) is set to low (GND - 0V) 1 - when the pin SEN (16 on SSOP version or pin 6 on QFN version) is set to high (+3.3V)

Definition at line 108 of file SI4735.cpp.

void SI4735::setDeviceOtherI2CAddress (uint8_t *i2cAddr*)

Sets the onther I2C Bus Address (for Si470X) You can set another I2C address different of 0x11 and 0x63

Parameters

uint8_t	i2cAddr (example 0x10)

Definition at line 117 of file SI4735.cpp.

void SI4735::setFM ()

Sets the radio to FM function

See also

Si47XX PROGRAMMING GUIDE; AN332; page 64.

Definition at line 478 of file SI4735.cpp.

void SI4735::setFM (uint16_t *fromFreq*, uint16_t *toFreq*, uint16_t *initialFreq*, uint16_t *step*)

Sets the radio to FM function.

See also

setFM()

Parameters

fromFreq	minimum frequency for the band
toFreq	maximum frequency for the band
initialFreq	initial frequency (default frequency)
step	step used to go to the next channel

Definition at line 524 of file SI4735.cpp.

void SI4735::setFmBlendMonoThreshold (uint8_t *parameter*)

Sets RSSI threshold for mono blend (Full mono below threshold, blend above threshold). To force stereo set this to 0. To force mono set this to 127. Default value is 30 dBÎ¼V.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 56.

Parameters

parameter	valid values: 0 to 127

Definition at line 644 of file SI4735.cpp.

void SI4735::setFmBlendMultiPathMonoThreshold (uint8_t *parameter*)

Sets Multipath threshold for mono blend (Full mono above threshold, blend below threshold). To force stereo, set to 100. To force mono, set to 0. The default is 60.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 60.

Parameters

parameter	valid values: 0 to 100

Definition at line 721 of file SI4735.cpp.

void SI4735::setFmBlendMultiPathStereoThreshold (uint8_t *parameter*)

Sets multipath threshold for stereo blend (Full stereo below threshold, blend above threshold). To force stereo, set this to 100. To force mono, set this to 0. Default value is 20.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 60.

Parameters

parameter	valid values: 0 to 100

Definition at line 708 of file SI4735.cpp.

void SI4735::setFmBLendRssiMonoThreshold (uint8_t *parameter*)

Sets RSSI threshold for mono blend (Full mono below threshold, blend above threshold). To force stereo, set this to 0. To force mono, set this to 127. Default value is 30 dBÎ¼V.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 59.

Parameters

parameter	valid values: 0 to 127

Definition at line 669 of file SI4735.cpp.

void SI4735::setFmBlendRssiStereoThreshold (uint8_t *parameter*)

Sets RSSI threshold for stereo blend. (Full stereo above threshold, blend below threshold.) To force stereo, set this to 0. To force mono, set this to 127. Default value is 49 dBÎ¼V.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 59.

Parameters

parameter	valid values: 0 to 127

Definition at line 656 of file SI4735.cpp.

void SI4735::setFmBLendSnrMonoThreshold (uint8_t *parameter*)

Sets SNR threshold for mono blend (Full mono below threshold, blend above threshold). To force stereo, set this to 0. To force mono, set this to 127. Default value is 14 dB.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 59.

Parameters

parameter	valid values: 0 to 127

Definition at line 695 of file SI4735.cpp.

void SI4735::setFmBlendSnrStereoThreshold (uint8_t *parameter*)

Sets SNR threshold for stereo blend (Full stereo above threshold, blend below threshold). To force stereo, set this to 0. To force mono, set this to 127. Default value is 27 dB.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 59.

Parameters

parameter	valid values: 0 to 127

Definition at line 682 of file SI4735.cpp.

void SI4735::setFmBlendStereoThreshold (uint8_t *parameter*)

Sets RSSI threshold for stereo blend (Full stereo above threshold, blend below threshold). To force stereo, set this to 0. To force mono, set this to 127.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 90.

Parameters

parameter	valid values: 0 to 127

Definition at line 631 of file SI4735.cpp.

void SI4735::setFmStereoOff ()

Turn Off Stereo operation.

Definition at line 729 of file SI4735.cpp.

void SI4735::setFmStereoOn ()

Turn Off Stereo operation.

Definition at line 737 of file SI4735.cpp.

void SI4735::setFrequency (uint16_t *freq*)

Set the frequency to the corrent function of the Si4735 (FM, AM or SSB) You have to call setup or setPowerUp before call setFrequency.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 70, 135

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 13

Parameters

uint16_t	freq Is the frequency to change. For example, FM => 10390 = 103.9 MHz; AM => 810 = 810 KHz.

Definition at line 376 of file SI4735.cpp.

void SI4735::setFrequencyStep (uint16_t *step*)

Sets the current step value.

ATTENTION: This function does not check the limits of the current band. Please, don't take a step bigger than your legs.

Parameters

step	if you are using FM, 10 means 100KHz. If you are using AM 10 means 10KHz For AM, 1 (1KHz) to 1000 (1MHz) are valid values. For FM 5 (50KHz) and 10 (100KHz) are valid values.

Definition at line 417 of file SI4735.cpp.

void SI4735::setI2CFastModeCustom (long *value* = `500000`)`[inline]`

Sets I2C buss to 400KHz.

Sets the I2C bus to a given value.

ATTENTION: use this function with cation

Parameters

value	in Hz. For example: The values 500000 sets the bus to 500KHz.

Definition at line 1110 of file SI4735.h.

void SI4735::setPowerUp (uint8_t *CTSIEN*, uint8_t *GPO2OEN*, uint8_t *PATCH*, uint8_t *XOSCEN*, uint8_t *FUNC*, uint8_t *OPMODE*)

Set the Power Up parameters for si473X. Use this method to chenge the defaul behavior of the Si473X. Use it before PowerUp()

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 65 and 129

Parameters

uint8_t	CTSIEN sets Interrupt anabled or disabled (1 = anabled and 0 = disabled)
uint8_t	GPO2OEN sets GP02 Si473X pin enabled (1 = anabled and 0 = disabled)
uint8_t	PATCH Used for firmware patch updates. Use it always 0 here.
uint8_t	XOSCEN sets external Crystal enabled or disabled
uint8_t	FUNC sets the receiver function have to be used [0 = FM Receive; 1 = AM (LW/MW/SW) and SSB (if SSB patch apllied)]
uint8_t	OPMODE set the kind of audio mode you want to use.

Definition at line 163 of file SI4735.cpp.

void SI4735::setRdsConfig (uint8_t *RDSEN*, uint8_t *BLETHA*, uint8_t *BLETHB*, uint8_t *BLETHC*, uint8_t *BLETHD*)

RESP3 - RDS FIFO Used; Number of groups remaining in the RDS FIFO (0 if empty).

Sets RDS property (FM_RDS_CONFIG) Configures RDS settings to enable RDS processing (RDSEN) and set RDS block error thresholds. When a RDS Group is received, all block errors must be less than or equal the associated block error threshold for the group to be stored in the RDS FIFO.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 104

MPORTANT: All block errors must be less than or equal the associated block error threshold for the group to be stored in the RDS FIFO. 0 = No errors. 1 = 1–2 bit errors detected and corrected. 2 = 3–5 bit errors detected and corrected. 3 = Uncorrectable. Recommended Block Error Threshold options: 2,2,2,2 = No group stored if any errors are uncorrected. 3,3,3,3 = Group stored regardless of errors. 0,0,0,0 = No group stored containing corrected or uncorrected errors. 3,2,3,3 = Group stored with corrected errors on B, regardless of errors on A, C, or D.

Parameters

uint8_t	RDSEN RDS Processing Enable; 1 = RDS processing enabled.
uint8_t	BLETHA Block Error Threshold BLOCKA.
uint8_t	BLETHB Block Error Threshold BLOCKB.
uint8_t	BLETHC Block Error Threshold BLOCKC.
uint8_t	BLETHD Block Error Threshold BLOCKD.

Definition at line 1265 of file SI4735.cpp.

void SI4735::setRdsIntSource (uint8_t *RDSNEWBLOCKB*, uint8_t *RDSNEWBLOCKA*, uint8_t *RDSSYNCFOUND*, uint8_t *RDSSYNCLOST*, uint8_t *RDSRECV*)

Configures interrupt related to RDS

Use this method if want to use interrupt

See also

Si47XX PROGRAMMING GUIDE; AN332; page 103

Parameters

RDSRECV	If set, generate RDSINT when RDS FIFO has at least FM_RDS_INT_FIFO_COUNT entries.
RDSSYNCLOST	If set, generate RDSINT when RDS loses synchronization.
RDSSYNCFOUND	set, generate RDSINT when RDS gains synchronization.
RDSNEWBLOCKA	If set, generate an interrupt when Block A data is found or subsequently changed
RDSNEWBLOCKB	If set, generate an interrupt when Block B data is found or subsequently changed

Definition at line 1309 of file SI4735.cpp.

void SI4735::setSBBSidebandCutoffFilter (uint8_t *SBCUTFLT*)

Sets SBB Sideband Cutoff Filter for band pass and low pass filters: 0 = Band pass filter to cutoff both the unwanted side band and high frequency components > 2.0 kHz of the wanted side band. (default) 1 = Low pass filter to cutoff the unwanted side band. Other values = not allowed.

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 24

Parameters

SBCUTFLT	0 or 1; see above

Definition at line 1914 of file SI4735.cpp.

void SI4735::setSeekAmLimits (uint16_t *bottom*, uint16_t *top*)

Sets the bottom frequency and top frequency of the AM band for seek. Default is 520 to 1710.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 127, 161, and 162

Parameters

uint16_t	bottom - the bottom of the AM band for seek
uint16_t	top - the top of the AM band for seek

Definition at line 1093 of file SI4735.cpp.

void SI4735::setSeekAmSpacing (uint16_t *spacing*)

Selects frequency spacingfor AM seek. Default is 10 kHz spacing.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 163, 229 and 283

Parameters

uint16_t	spacing - step in KHz

Definition at line 1106 of file SI4735.cpp.

void SI4735::setSeekRssiThreshold (uint16_t *value*)

Sets the RSSI threshold for a valid AM Seek/Tune. If the value is zero then RSSI threshold is not considered when doing a seek. Default value is 25 dBÎ¼V.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 127

Definition at line 1128 of file SI4735.cpp.

void SI4735::setSeekSrnThreshold (uint16_t *value*)

Sets the SNR threshold for a valid AM Seek/Tune. If the value is zero then SNR threshold is not considered when doing a seek. Default value is 5 dB.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 127

Definition at line 1117 of file SI4735.cpp.

void SI4735::setSSB (uint8_t *usblsb*)

Set the radio to AM function. It means: LW MW and SW.

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; pages 13 and 14

setAM()

void **SI4735::setFrequency(uint16_t freq)**

Parameters

usblsb	upper or lower side band; 1 = LSB; 2 = USB

Definition at line 1960 of file SI4735.cpp.

void SI4735::setSSBAudioBandwidth (uint8_t *AUDIOBW*)

SSB Audio Bandwidth for SSB mode

0 = 1.2 kHz low-pass filter* . (default) 1 = 2.2 kHz low-pass filter* . 2 = 3.0 kHz low-pass filter. 3 = 4.0 kHz low-pass filter. 4 = 500 Hz band-pass filter for receiving CW signal, i.e. [250 Hz, 750 Hz] with center frequency at 500 Hz when USB is selected or [-250 Hz, -750 1Hz] with center frequency at -500Hz when LSB is selected* . 5 = 1 kHz band-pass filter for receiving CW signal, i.e. [500 Hz, 1500 Hz] with center frequency at 1 kHz when USB is selected or [-500 Hz, -1500 1 Hz] with center frequency at -1kHz when LSB is selected* . Other values = reserved. Note: If audio bandwidth selected is about 2 kHz or below, it is recommended to set SBCUTFLT[3:0] to 0 to enable the band pass filter for better high- cut performance on the wanted side band. Otherwise, set it to 1.

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 24

Parameters

AUDIOBW	the valid values are 0, 1, 2, 3, 4 or 5; see description above

Definition at line 1943 of file SI4735.cpp.

void SI4735::setSSBAutomaticVolumeControl (uint8_t *AVCEN*)

Sets SSB Automatic Volume Control (AVC) for SSB mode

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 24

Parameters

AVCEN	0 = Disable AVC; 1 = Enable AVC (default).

Definition at line 1885 of file SI4735.cpp.

void SI4735::setSSBAvcDivider (uint8_t *AVC_DIVIDER*)

Sets AVC Divider

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 24

Parameters

AVC_DIVIDER	SSB mode, set divider = 0; SYNC mode, set divider = 3; Other values = not allowed.

Definition at line 1898 of file SI4735.cpp.

void SI4735::setSSBBfo (int *offset*)

Single Side Band (SSB) implementation

This implementation was tested only on Si4735-D60 device.

SSB modulation is a refinement of amplitude modulation that one of the side band and the carrier are suppressed.

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; pages 3 and 5

First of all, it is important to say that the SSB patch content is not part of this library. The paches used here were made available by Mr. Vadim Afonkin on his Dropbox repository. It is important to note that the author of this library does not encourage anyone to use the SSB patches content for commercial purposes. In other words, this library only supports SSB patches, the patches themselves are not part of this library.

What does SSB patch means? In this context, a patch is a piece of software used to change the behavior of the **SI4735** device. There is little information available about patching the **SI4735**.

The following information is the understanding of the author of this project and it is not necessarily correct.

A patch is executed internally (run by internal MCU) of the device. Usually, patches are used to fixes bugs or add improvements and new features of the firmware installed in the internal ROM of the device. Patches to the **SI4735** are distributed in binary form and have to be transferred to the internal RAM of the device by the host MCU (in this case Arduino boards). Since the RAM is volatile memory, the patch stored into the device gets lost when you turn off the system. Consequently, the content of the patch has to be transferred again to the device each time after turn on the system or reset the device.

I would like to thank Mr Vadim Afonkin for making available the SSBRX patches for SI4735-D60 on his Dropbox repository. On this repository you have two files, amrx_6_0_1_ssbrx_patch_full_0x9D29.csg and amrx_6_0_1_ssbrx_patch_init_0xA902.csg. It is important to know that the patch content of the original files is constant hexadecimal representation used by the language C/C++. Actally, the original files are in ASCII format (not in binary format). If you are not using C/C++ or if you want to load the files directly to the **SI4735**, you must convert the values to numeric value of the hexadecimal constants. For example: 0x15 = 21 (00010101); 0x16 = 22 (00010110); 0x01 = 1 (00000001); 0xFF = 255 (11111111);

ATTENTION: The author of this project does not guarantee that procedures shown here will work in your development environment. Given this, it is at your own risk to continue with the procedures suggested here. This library works with the I²C communication protocol and it is designed to apply a SSB extension PATCH to CI SI4735-D60. Once again, the author disclaims any liability for any damage this procedure may cause to your **SI4735** or other devices that you are using. Sets the SSB Beat Frequency Offset (BFO).

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; pages 5 and 23

Parameters

offset	16-bit signed value (unit in Hz). The valid range is -16383 to +16383 Hz.

Definition at line 1790 of file SI4735.cpp.

void SI4735::setSSBConfig (uint8_t *AUDIOBW*, uint8_t *SBCUTFLT*, uint8_t *AVC_DIVIDER*, uint8_t *AVCEN*, uint8_t *SMUTESEL*, uint8_t *DSP_AFCDIS*)

Set the SSB receiver mode details: 1) Enable or disable AFC track to carrier function for receiving normal AM signals; 2) Set the audio bandwidth; 3) Set the side band cutoff filter; 4) Set soft-mute based on RSSI or SNR; 5) Enable or disbable automatic volume control (AVC) function.

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 24

Parameters

AUDIOBW	SSB Audio bandwidth; 0 = 1.2KHz (default); 1=2.2KHz; 2=3KHz; 3=4KHz; 4=500Hz; 5=1KHz.
SBCUTFLT	SSB side band cutoff filter for band passand low pass filter if 0, the band pass filter to cutoff both the unwanted side band and high frequency component > 2KHz of the wanted side band (default).
AVC_DIVIDER	set 0 for SSB mode; set 3 for SYNC mode.
AVCEN	SSB Automatic Volume Control (AVC) enable; 0=disable; 1=enable (default).
SMUTESEL	SSB Soft-mute Based on RSSI or SNR.
DSP_AFCDIS	DSP AFC Disable or enable; 0=SYNC MODE, AFC enable; 1=SSB MODE, AFC disable.

Definition at line 1835 of file SI4735.cpp.

void SI4735::setSSBDspAfc (uint8_t *DSP_AFCDIS*)

Sets DSP AFC disable or enable

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 24

Parameters

DSP_AFCDIS	0 = SYNC mode, AFC enable; 1 = SSB mode, AFC disable

Definition at line 1858 of file SI4735.cpp.

void SI4735::setSSBSoftMute (uint8_t *SMUTESEL*)

Sets SSB Soft-mute Based on RSSI or SNR Selection:

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 24

Parameters

SMUTESEL	0 = Soft-mute based on RSSI (default); 1 = Soft-mute based on SNR.

Definition at line 1872 of file SI4735.cpp.

void SI4735::setTuneFrequencyAntennaCapacitor (uint16_t *capacitor*)

Only FM. Freeze Metrics During Alternate Frequency Jump.

Selects the tuning capacitor value.

For FM, Antenna Tuning Capacitor is valid only when using TXO/LPI pin as the antenna input.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 71 and 136

Parameters

capacitor	If zero, the tuning capacitor value is selected automatically. If the value is set to anything other than 0: AM - the tuning capacitance is manually set as 95 fF

	x ANTCAP + 7 pF. ANTCAP manual range is 1–6143; FM - the valid range is 0 to 191. According to Silicon Labs, automatic capacitor tuning is recommended (value 0).

Definition at line 343 of file SI4735.cpp.

void SI4735::setup (uint8_t *resetPin*, int *interruptPin*, uint8_t *defaultFunction*, uint8_t *audioMode* = `SI473X_ANALOG_AUDIO`)

Starts the Si473X device.

If the audio mode parameter is not entered, analog mode will be considered.

Parameters

uint8_t	resetPin Digital Arduino Pin used to RESET command
uint8_t	interruptPin interrupt Arduino Pin (see your Arduino pinout). If less than 0, iterrupt disabled
uint8_t	defaultFunction
uint8_t	audioMode default SI473X_ANALOG_AUDIO (Analog Audio). Use SI473X_ANALOG_AUDIO or SI473X_DIGITAL_AUDIO

Definition at line 279 of file SI4735.cpp.

void SI4735::setup (uint8_t *resetPin*, uint8_t *defaultFunction*)

Starts the Si473X device.

Use this setup if you are not using interrupt resource

Parameters

uint8_t	resetPin Digital Arduino Pin used to RESET command
uint8_t	defaultFunction

Definition at line 322 of file SI4735.cpp.

void SI4735::setVolume (uint8_t *volume*)

RESP8 - Returns the Chip Revision (ASCII).

Sets volume level (0 to 63)

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 62, 123, 170, 173 and 204

Parameters

uint8_t	volume (domain: 0 - 63)

Definition at line 1140 of file SI4735.cpp.

void SI4735::ssbPowerUp ()

This function can be useful for debug and teste.

Definition at line 2116 of file SI4735.cpp.

void SI4735::ssbSetup ()

Starts the Si473X device on SSB (same AM Mode). Same **SI4735::setup** optimized to improve loading patch performance

Definition at line 2105 of file SI4735.cpp.

void SI4735::volumeDown ()

Set sound volume level Down

See also

setVolume()

Definition at line 1187 of file SI4735.cpp.

void SI4735::volumeUp ()

Set sound volume level Up

See also

setVolume()

Definition at line 1175 of file SI4735.cpp.

void SI4735::waitInterrupr (void) [protected]

If you setup interrupt, this function will be called whenever the Si4735 changes.

Definition at line 45 of file SI4735.cpp.

void SI4735::waitToSend (void)

Wait for the si473x is ready (Clear to Send (CTS) status bit have to be 1).

This function should be used before sending any command to a SI47XX device.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 63, 128

Definition at line 141 of file SI4735.cpp.

The documentation for this class was generated from the following files:

- SI4735.h
- SI4735.cpp

si4735_digital_output_format Union Reference

`#include <SI4735.h>`

Detailed Description

Digital audio output format data structure (Property 0x0102. DIGITAL_OUTPUT_FORMAT). Useed to configure: DCLK edge, data format, force mono, and sample precision.

See also

Si47XX PROGRAMMING GUIDE; AN332; page 195.

Definition at line 784 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si4735_digital_output_sample_rate Struct Reference

`#include <SI4735.h>`

Detailed Description

Digital audio output sample structure (Property 0x0104. DIGITAL_OUTPUT_SAMPLE_RATE). Used to enable digital audio output and to configure the digital audio output sample rate in samples per second (sps).

See also

Si47XX PROGRAMMING GUIDE; AN332; page 196.

Definition at line 803 of file SI4735.h.

The documentation for this struct was generated from the following file:

- SI4735.h

si473x_powerup Union Reference

```
#include <SI4735.h>
```

Detailed Description

Power Up arguments data type

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 64 and 65

Definition at line 168 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_agc_overrride Union Reference

```
#include <SI4735.h>
```

Detailed Description

If FM, Overrides AGC setting by disabling the AGC and forcing the LNA to have a certain gain that ranges between 0 (minimum attenuation) and 26 (maximum attenuation). If AM, overrides the AGC setting by disabling the AGC and forcing the gain index that ranges between 0

See also

Si47XX PROGRAMMING GUIDE; AN332; For FM page 81; for AM page 143

Definition at line 717 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_agc_status Union Reference

```
#include <SI4735.h>
```

Detailed Description

AGC data types FM / AM and SSB structure to AGC

See also

Si47XX PROGRAMMING GUIDE; AN332; For FM page 80; for AM page 142

AN332 REV 0.8 Universal Programming Guide Amendment for SI4735-D60 SSB and NBFM patches; page 18.

Definition at line 688 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_antenna_capacitor Union Reference

```
#include <SI4735.h>
```

Detailed Description

Antenna Tuning Capacitor data type manupulation

Definition at line 201 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_bandwidth_config Union Reference

`#include <SI4735.h>`

Detailed Description

The bandwidth of the AM channel filter data type AMCHFLT values: 0 = 6 kHz Bandwidth

1 = 4 kHz Bandwidth 2 = 3 kHz Bandwidth 3 = 2 kHz Bandwidth 4 = 1 kHz Bandwidth 5 = 1.8 kHz Bandwidth 6 = 2.5 kHz Bandwidth, gradual roll off 7–15 = Reserved (Do not use)

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 125 and 151

Definition at line 744 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_firmware_information Union Reference

`#include <SI4735.h>`

Detailed Description

Data representation for Firmware Information (GET_REV) The part number, chip revision, firmware revision, patch revision and component revision numbers.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 66 and 131

Definition at line 294 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_firmware_query_library Union Reference

```
#include <SI4735.h>
```

Detailed Description

Firmware Query Library ID response. Used to represent the response of a power up command with FUNC = 15 (patch)

To confirm that the patch is compatible with the internal device library revision, the library revision should be confirmed by issuing the POWER_UP command with Function = 15 (query library ID)

See also

Si47XX PROGRAMMING GUIDE; AN332; page 12

Definition at line 329 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_frequency Union Reference

`#include <SI4735.h>`

Detailed Description

Represents how the frequency is stored in the si4735. It helps to convert frequency in uint16_t to two bytes (uint8_t) (FREQL and FREQH)

Definition at line 188 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_property Union Reference

```
#include <SI4735.h>
```

Detailed Description

Property Data type (help to deal with SET_PROPERTY command on si473X)

Definition at line 375 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_rds_blocka Union Reference

```
#include <SI4735.h>
```

Detailed Description

Block A data type

Definition at line 563 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_rds_blockb Union Reference

`#include <SI4735.h>`

Detailed Description

Block B data type

For GCC on System-V ABI on 386-compatible (32-bit processors), the following stands: 1) Bit-fields are allocated from right to left (least to most significant). 2) A bit-field must entirely reside in a storage unit appropriate for its declared type. Thus a bit-field never crosses its unit boundary. 3) Bit-fields may share a storage unit with other struct/union members, including members that are not bit-fields. Of course, struct members occupy different parts of the storage unit. 4) Unnamed bit-fields' types do not affect the alignment of a structure or union, although individual bit-fields' member offsets obey the alignment constraints.

See also

also Si47XX PROGRAMMING GUIDE; AN332; pages 78 and 79

also `https://en.wikipedia.org/wiki/Radio_Data_System`

Definition at line 592 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_rds_command Union Reference

`#include <SI4735.h>`

Detailed Description

FM_RDS_STATUS (0x24) command Data type for command and response information

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 77 and 78

Also `https://en.wikipedia.org/wiki/Radio_Data_System`

Definition at line 441 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_rds_config Union Reference

```
#include <SI4735.h>
```

Detailed Description

Data type for FM_RDS_CONFIG Property

IMPORTANT: all block errors must be less than or equal the associated block error threshold for the group to be stored in the RDS FIFO. 0 = No errors; 1 = 1–2 bit errors detected and corrected; 2 = 3–5 bit errors detected and corrected; 3 = Uncorrectable. Recommended Block Error Threshold options: 2,2,2,2 = No group stored if any errors are uncorrected. 3,3,3,3 = Group stored regardless of errors. 0,0,0,0 = No group stored containing corrected or uncorrected errors. 3,2,3,3 = Group stored with corrected errors on B, regardless of errors on A, C, or D.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 58 and 104

Definition at line 545 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_rds_date_time Union Reference

`#include <SI4735.h>`

Detailed Description

Group type 4A (RDS Date and Time) When group type 4A is used by the station, it shall be transmitted every minute according to EN 50067. This Structure uses blocks 2,3 and 5 (B,C,D)

ATTENTION: To make it compatible with 8, 16 and 32 bits platforms and avoid Crosses boundary, it was necessary to split minute and hour representation.

Definition at line 663 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_rds_int_source Union Reference

`#include <SI4735.h>`

Detailed Description

FM_RDS_INT_SOURCE property data type

See also

Si47XX PROGRAMMING GUIDE; AN332; page 103

also `https://en.wikipedia.org/wiki/Radio_Data_System`

Definition at line 514 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_rds_status Union Reference

```
#include <SI4735.h>
```

Detailed Description

Response data type for current channel and reads an entry from the RDS FIFO.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 77 and 78

Definition at line 459 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_response_status Union Reference

`#include <SI4735.h>`

Detailed Description

Response status command

See also

Si47XX PROGRAMMING GUIDE; pages 73 and

Definition at line 254 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_rqs_status Union Reference

`#include <SI4735.h>`

Detailed Description

Data type for status information about the received signal quality FM_RSQ_STATUS and AM_RSQ_STATUS

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 75 and

Definition at line 394 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_seek Union Reference

```
#include <SI4735.h>
```

Detailed Description

Represents searching for a valid frequency data type.

Definition at line 236 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_set_frequency Union Reference

```
#include <SI4735.h>
```

Detailed Description

AM_TUNE_FREQ data type command

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 135

Definition at line 216 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_ssb_mode Union Reference

```
#include <SI4735.h>
```

Detailed Description

SSB - datatype for SSB_MODE (property 0x0101)

See also

AN332 REV 0.8 UNIVERSAL PROGRAMMING GUIDE; page 24

Definition at line 762 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

si47x_tune_status Union Reference

`#include <SI4735.h>`

Detailed Description

Status of FM_TUNE_FREQ or FM_SEEK_START commands or Status of AM_TUNE_FREQ or AM_SEEK_START commands.

See also

Si47XX PROGRAMMING GUIDE; AN332; pages 73 and 139

Definition at line 360 of file SI4735.h.

The documentation for this union was generated from the following file:

- SI4735.h

Index

INDEX