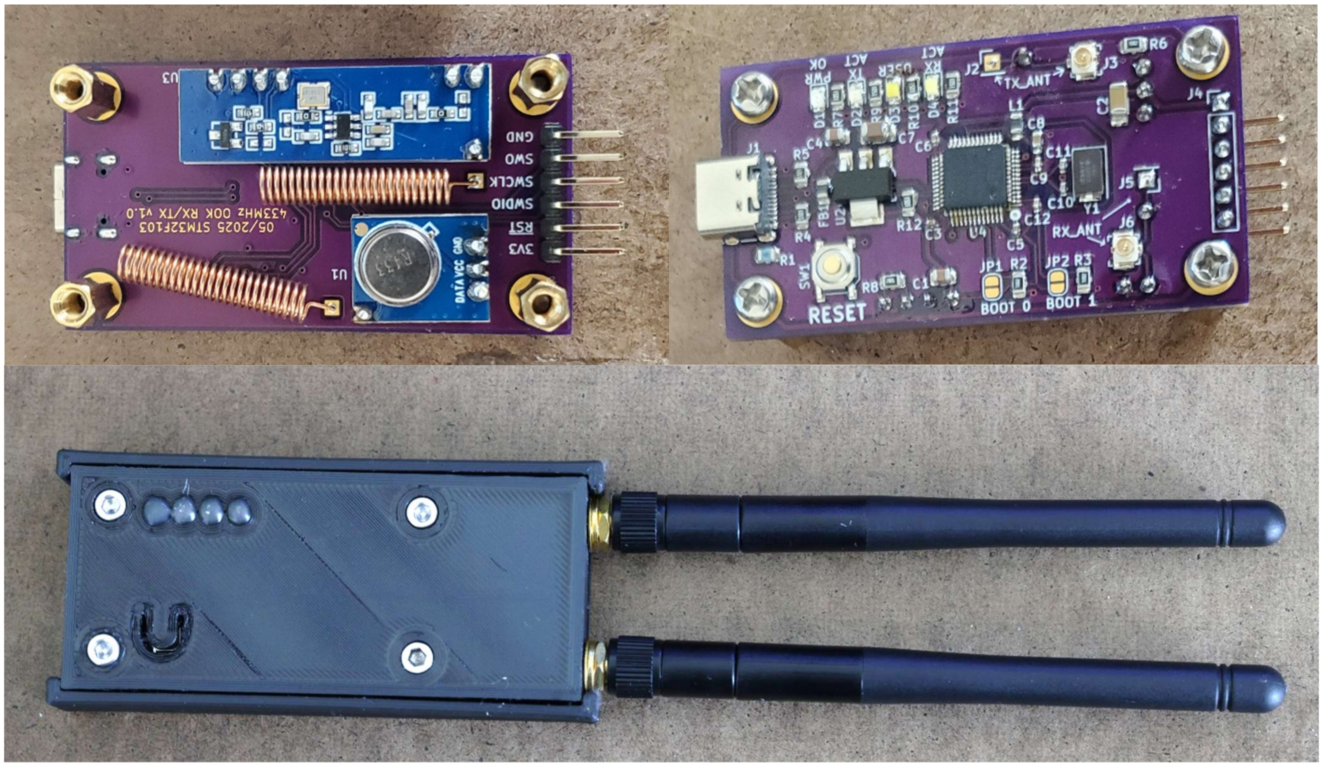
STM32F103 433MHz RX/TX Module

Interface Reference- Firmware v2.1.0

By TimeHack



Summary

This module is a USB-driven RF sniffer and transmitter module based around the SRX882S and STX882 Superheterodyne Receiver/Transmitter pair and an STM32F103C8T6 processor. Both receive and transmit operations are highly configurable by the user via USB serial communications port.

The module is designed to connect to a USB host device with the default Communications Device Class (CDC) driver. A user can then open a serial port to the module to configure receive/transmit parameters, command transmissions, and receive asynchronous messages about sent and received transmissions.

When idle, the module passively listens for changes to the data output from the SRX882S Receiver, collecting information about on/off pulse widths and periods and correlating received data to measure repetitions of bit sequences ("words") to indicate a data transmission. Each correlated word (up to 64 bits in length) then gets communicated to the USB host with additional information regarding word length, waveform timings (high/low/period durations), and receiver parameters (logic assumption, whether an initial sync bit is ignored). The module can be configured to listen for words within a range of lengths, from 1 to 64 bits.

The module can transmit messages up to 64 bits in length with an optional leading synchronizing bit, which is a long-high duration pulse (typically a 0 for standard logic interpretations). That synchronizing bit is automatically added to the transmission payload if the ignoresyncbit flag is set false (0).

The module includes LED's for user feedback. One indicates 3.3V available to the processor core and RX/TX modules; a second indicates USB activity and illuminates upon user commands to the units and asynchronous messages from the module; the third indicates transmission activity, and the fourth represents whether the receiver module is active.

System Status Codes

The internal system status variable consists of a single, 32-bit unsigned integer, divided into four bytes. Values from bits 0-7 are reserved for USB status codes; bits 8-15 represent transmission status codes; bits 16-23 are for receiver status codes; and bits 24-31 are reserved for future expansion.

A complete listing of utilized status codes are shown in Table 1. Status codes beginning with USB_CC are used in USB responses to commands, whereas status codes starting with RX and TX are used in asynchronous messages to the USB host. **The status code will always be the first value in a message string from the module.** See the following subsections for detail on each status code.

Table 1: Hex and Decimal codes for system statuses

Code Identifier	**Byte Value (Hex)**	**Status Value (Decimal)**
USB_CC_OK	0x00	0
USB_CC_BUSY	0x01	1
USB_CC_UNKNOWN	0x10	16
USB_CC_BAD_VALUE	0x20	32
USB_CC_BAD_PARAM	0x30	48
USB_CC_MISSING_PARAM	0x31	49
TX_BUFFER_EMPTY	0x0100	256
TX_PREP_FAILED	0x0200	512
TX_COMPLETE	0x0400	1024
RX_WORD_AVAILABLE	0x010000	65536

USB_CC_OK

Response provided after successful get or set USB operation. When setting parameters, a response of ‘0 OK’ will be sent. For getting parameters, a response ‘0 <parameter> <value>’ will be sent, where ‘parameter’ is the final word of the command (e.g. ‘mode’ for ‘rx mode’; or, ‘short’ for ‘tx time short’) and ‘value’ is the current parameter value. Single-word commands return ‘0 <value>’ for the current parameter value (e.g. ‘version’ returns ‘0 2.1.0’; and, ‘status’ returns ‘0 <status>’).

USB_CC_BUSY

Response provided when the module is busy processing and cannot execute the requested command. As of the current firmware version (2.1.0), this response is only provided if the transmission word buffer is full and cannot accept another call to ‘tx [word]’. Such a response is only the status code.

USB_CC_UNKNOWN

Response provided when the provided command is not recognized – essentially if the first word in the command string is not ‘rx’, ‘tx’, ‘status’, or ‘version’. Response follows structure: ‘<status> <command>’ where ‘command’ is the unrecognized part of the command string.

USB_CC_BAD_VALUE

Response provided when the value given for setting a parameter is invalid for the command. The value may be out of the range limitations internal to the system or the wrong data type. Response format follows ‘<status> <invalid_value>’, where ‘invalid_value’ is the problematic value.

USB_CC_BAD_PARAM

Response provided when parts of the command are recognized, but a subcommand is invalid. Response structure is formatted as ‘<status> <bad_subcommand>’ where ‘bad_subcommand’ is the problematic string. For example, if the command ‘rx data’ is passed, the command ‘rx’ is recognized, but ‘data’ is invalid, so the module will respond with ’48 data’.

USB_CC_MISSING_PARAM

Response to an incomplete command, where all parts are recognized, but it doesn’t correspond to an action for the module to perform due to a missing subcommand or parameter. Response is strictly the status code.

TX_BUFFER_EMPTY

Status indicating there are no pending transmissions in the buffer. Only available upon request by the USB host using command ‘status’. Synonymous with an idle system status during normal operation.

TX_PREP_FAILED

A status which should never occur, provided adequate parsing during command processing. This will asynchronously be sent to the USB host if the transmission word being processed for broadcast contains invalid characters – i.e. values that are not ‘0’ or ‘1’. This state is included as a built-in secondary safety against core faults from processing invalid words. Message structure with this asynchronous message follows ‘<status> <tx_word>’, with ‘tx_word’ being the problematic broadcast word.

TX_COMPLETE

Status message that will be asynchronously sent to the USB host to indicate a transmission has completed successfully. Message structure follows ‘<status> <tx_word>’, with ‘tx_word’ being the binary string successfully broadcast. This string will include any automatically prepended sync bits based on the setting ‘tx ignoresyncbit’.

RX_WORD_AVAILABLE

Status message to be asynchronously sent to the USB host when a transmission is received that matches the settings configuration for the receive channel of the module. Detections / matches will only be triggered if all of the following are met:

- Detected bit sequence longer than ‘rx word minlength’
- Detected bit sequence shorter than ‘rx word maxlength’
- Detected repetition of bit sequence greater than or equal to ‘rx word matchcount’
- Time between last bit sequence received and the newest received code is less than ‘rx word timeout’ microseconds. All sequences received prior to an elapsed word reception gap of this timeout will be used to check for word repetitions.
- A bit sequence repetition for this code has not already been transmitted to the USB host in the last ‘rx word timeout’ microseconds.

These conditions mean that for a broadcast word which might be transmitted 10 times to ensure the receiving device reliably receives the word, the USB host device will not be overwhelmed with messages each time a repetition more than ‘rx word matchcount’ occurs in that repeated burst of word transmissions. For each burst, only one transmission of a repeated word will be sent.

USB Serial Commands

The module accepts a diversity of commands for getting/setting on-board parameters. Each valid command is defined below. Where the command includes a data type in '< >' at the end, the bracketed value is optional. **Including that value uses the write function of that command, and omitting it will execute the read function of that command**.

All successful 'read' operations will return a message structure

<status> <final_subcommand> <value>

where 'status' is USB_CC_OK, 'final_subcommand' is the last non-value word of the command string, and 'value' is the current parameter value.

All 'write' operations, if successful, will return a message structure

<status> OK

with 'status' being the value of USB_CC_OK.

Failure statuses will be returned as applicable as defined in the **System Status Codes** section, and according to limitations described below in the respective commands.

version

(READ ONLY)
Return the installed firmware version string in format <major>:<minor>:<tweak> (present as of 2.1.0). Major changes overhaul major system operations (core behavior of RX/TX, etc.) which are not compatible with older versions of the firmware; minor changes correspond to compatibility with older firmware versions, but with added features; tweaks are small updates which do not impact interface between the USB host and module, but which modify module behaviors not seen by the user.

status

(READ ONLY)
Return the current system 'status' value. Status is a 32-bit (4-byte) code with possible values defined in Table 1 in the section **System Status Codes**.

rx mode <0:1:2>

(READ / WRITE)
Get or set the current receiver mode.

- 0 = always off: do not listen for asynchronous broadcasts
- 1 = always on: always listen for broadcasts, even during this module's broadcasts
- 2 = automatic: disable broadcast detection during this module's broadcasts (DEFAULT)

Default value: 2 (automatic)
Minimum value: 0
Maximum value: 2

rx bitperiod <uint32_t>

(READ / WRITE)
Get or set the maximum bit period, in microseconds. This acts as a timeout during broadcast receptions which allows for processing the data buffer. Due to how OOK broadcasts are structured (with a moderate delay between word transmission repetitions), the most efficient execution for the internal system includes this setting being tuned according to the expected bit period timing of the signal to be received.

Default value: 5000 microseconds (5 ms).
Minimum value: 0 (not recommended)
Maximum value: UINT32_MAX (not recommended)

rx word matchcount <uint8_t>

(READ / WRITE)
Set the minimum number of matching received word transmissions to trigger an asynchronous RX_WORD_RECEIVED message. This number of repetitions indicates the reception as a "valid" word in order to filter erroneous receptions by the system through corruptions or interference with other signals. Maximum value defined by the length of the data correlation buffer.

Default value: 3
Minimum value: 0
Maximum value: 12

rx word minlength <uint8_t>

(READ / WRITE)
Get or set the minimum word length (in bits) to use in correlating received data. Any words shorter than this limit get discarded. Maximum limit defined by 'rx word maxlength'.

Default value: 8
Minimum value: 0
Maximum value: rx word maxlength

rx word maxlength <uint8_t>

(READ / WRITE)
Get or set the maximum word length (in bits) to use in correlating received data. Any words longer than this limit get discarded. Minimum limit defined by 'rx word minlength'. Maximum limit set by maximum word buffer length.

Default value: 64
Minimum value: rx word minlength
Maximum value: 64

rx word timeout <uint32_t>

(READ / WRITE)
Get or set the timeout, in microseconds, for the data correlation buffer which caches the most recently received bit sequences. Once this timeout occurs, the buffer will be cleared. Minimum limit defined by ‘rx bitperiod’.

Default value: 100,000 (100 ms)
Minimum value: rx bitperiod
Maximum value: 5,000,000 (5 s)

rx ignoresyncbit <0:1>

(READ / WRITE)
Get or set whether the synchronizing bit (first bit received in sequence which sets the receiver state) should be included in the bit sequence to be sent to the USB host.

- 0 = no; include sync bit.
- 1 = yes; omit sync bit.

Default value: 1 (omit sync bit)
Minimum value: 0
Maximum value: 1

rx logic <0:1>

(READ / WRITE)
Get or set the logic interpretation by the receiver – i.e. how high-low relative durations should be interpreted as bits.

- 0 = long high with short low corresponds to bit ‘0’
- 1 = long high with short low corresponds to bit ‘1’

Default value: 0
Minimum value: 0
Maximum value: 1

tx time short <uint16_t>

(READ / WRITE)
Get or set the timing, in microseconds, of a short pulse. Used for both logic ‘0’ and ‘1’ short part of bit transmission. Must be less than ‘tx time long’ or UINT16_MAX / 2, whichever is less. The sum of ‘tx time short’ and ‘tx time long’ determines bit period for transmission.

Default value: 300
Minimum value: 0 (not recommended)
Maximum value: tx time long

tx time long <uint16_t>

(READ / WRITE)
Get or set the timing, in microseconds, or a long pulse. Used for both logic ‘0’ and ‘1’ long part of bit transmission. Must be greater than ‘tx time short’ and less than UINT16_MAX / 2. The sum of ‘tx time short’ and ‘tx time long’ determines bit period for transmission.

Default value: 700
Minimum value: tx time short
Maximum value: UINT16_MAX / 2

tx delay frame <uint32_t>

(READ / WRITE)
Get or set the time delay, in microseconds, between sequential transmissions of words within the same frame. Each word transmitted is considered a ‘frame, and is repeated ‘tx repeat’ number of times. This is the delay between each repetition of transmission.

Default value: 6600 (6.6 ms)
Minimum value: [(tx time short) + (tx time long)] (1 bit period)
Maximum value: 50*[(tx time short) + (tx time long)] (50 bit periods)

tx delay burst <uint32_t>

(READ / WRITE)

Get or set the time delay, in microseconds, between sequential broadcasts of buffered words. Each broadcast is considered a ‘burst’ and consists of ‘tx repeat’ number of frames. This is the delay between different broadcast bursts, and controls how fast sequential broadcast requests can be executed.

Default value: 100,000 (100 ms)
Minimum value: [(tx time short) + (tx time long)]
Maximum value: 60,000,000 (60 s)

tx ignoresyncbit <0:1>

(READ / WRITE)
Get or set whether a sync bit should be prepended to the bit sequence to transmit. If this is value 1 (true), the user should compensate for sync bit inclusion in the transmitted bit sequence.

- 0 = prepend a ‘0’ bit to the transmission bit sequence
- 1 = do not prepend a ‘0’ to the transmission bit sequence

Default value: 0
Minimum value: 0
Maximum value: 1

tx repeat <uint8_t>

(READ / WRITE)
Get or set how many times a bit sequence frame gets repeated in a burst. One frame will always be transmitted. This is the number of repetitions of transmissions.

Default value: 7
Minimum value: 0
Maximum value: 100

tx logic <0:1>

(READ / WRITE)
Get or set the logic to be used for interpreting commanded broadcasts.

- 0 = '0' means long high with short low
- 1 = '1' means long high with short low

Default value: 0
Minimum value: 0
Maximum value: 1

tx [sequence of 0:1]

(WRITE ONLY)
Queue the broadcast of a bit sequence, up to 64 bits. Up to 5 messages can be simultaneously queued for transmission and will be processed according to the time separation of 'tx delay burst'.

Asynchronous Messages

Three message types occur asynchronously with user commands. These indicate a broadcast reception, a failure to prepare a transmission burst, or successful completion of a transmission burst. Details on each message structure are below.

Broadcast Received

Message to indicate a valid broadcast was received. See **System Status Codes -> RX_WORD_AVAILABLE** for more information on how received broadcasts are validated. The message format for this follows:

<status> word:[0:1] len:<word_len> long_us:<long> short_us:<short> period_us:<period> logic:<logic> ignoresync:<sync>

where:

- 'status' is RX_WORD_AVAILABLE
- 'word_len' is the length of the received bit sequence
- 'long' is the average of long pulse durations across all received matching bit sequences
- 'short' is the average of short pulse durations across all received matching bit sequences
- 'period' is the average period duration across all received matching bit sequences
- 'logic' is the interpretation used to present '0' and '1' characters. See 'rx logic' for more information
- 'sync' is whether the first received bit (sync bit) is included in the received message. See 'rx ignoresyncbit' for more information.

Transmission Preparation Failed

Message to indicate an error with translating the commanded transmission bit sequence into a sequence of pulses. Namely, a character other than '0' or '1' made it to the buffered bit sequence. Message structure follows:

<status> [word]

where 'status' is TX_PREP_FAILED, and 'word' is the string which failed to translate into pulses.

Message Transmitted

Message ton indicate successful transmission of a bit sequence burst. Message structure follows:

<status> [word]

where 'status' is TX_COMPLETE, and 'word' is the binary string successfully transmitted, including prepended sync bit, as applicable.