Hochfrequenzbaugruppen
Fernsteuerkomponenten
Fernsteueranlagen
FUNKTECHNIK GMBH
Sales office/Beratung & Vertrieb · Design & Production/Entwicklung & Produktion

SPI-interface of the 70RX-S2 receiver module

1 System description

D0	-	SDO
D1	-	SCL
D2	-	SDI
D3	-	$\overline{CS}$
D4	-	
D5	-	
D6	-	
D7	-	
GND	-	
LD	-	
MOD	-	
Ub	-	
N.C.	-	
Audio	-	
RSSI	-	
Mute	-	

Our new reciver 70RX-S2 combines the approved 8bit wide parallel interface with a 4bit wide SPI interface. You can use only one of the two interface types at the same time due to the fact that they share four wires. The 16-pin connector follows the pin description next to the text. In the „parallel“ mode the bit combination at port D0-D7 („0-active“) is interpreted as channel number (0...255). The internal microcontroller of the receiver calculates the needed parameters for the synthesizer.

The SPI interface at port D0-D3 is active at the same time. If you send the ASCII character 'H' via SPI (D0-D3), the module switches to the serial mode and D0 is activated as SPI output for the feedback. The module keeps in serial mode until the power is switched off. You can now modify the frequency, read mute status and RSSI level and access the parameters of the module.

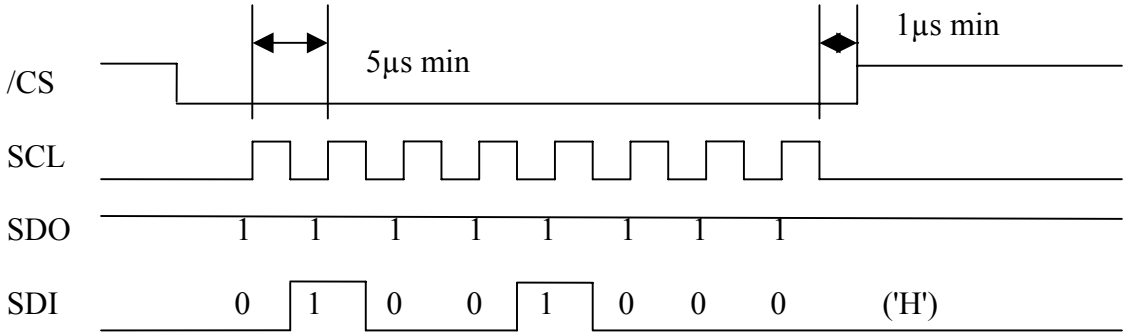
If you want to start in serial mode you can set up a flag in the CPU (as from now the serial mode is directly active after power-on and D0 is set as data output, the 8bit parallel interface is disabled). If you want to start the module in „parallel“ mode again you have to disable this flag and the 8 wires D0..D7 are set as input with the next power-on condition. In SPI mode the channel number stored in the EEPROM is used at startup.

The access to the special parameters of the module are only available in SPI mode.

The SPI interface uses the four standard signals

- /CS chip select (low aktive)
- SCL serial clock (data transfer with rising edge!)
- SDI data in (module) MOSI
- SDO data out (module) MISO

The data must be transmitted as ASCII character. Every byte is followed by a CS =“high“ (interrupt trigger).



To illustrate the communication flow please check the time response in the following samples.
D1= CLK; D2=SDI; D3=/CS ; D0= SDO

The first diagram shows the transmission of the sign 'H' (between the two time markings).

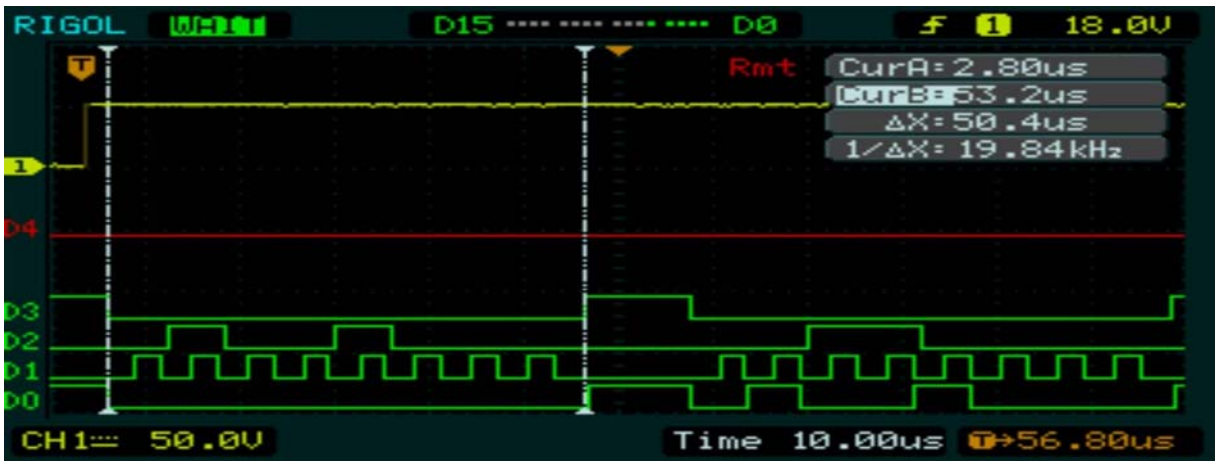


Figure 1

The next diagram figure 2 shows the transmission of „H01“ with the end string 0. You can see the characters '0' = 0x30 and '1' = 0x31 between the two time markings.
The signal D0 shows at the same time 'H' followed by 'R' (read).

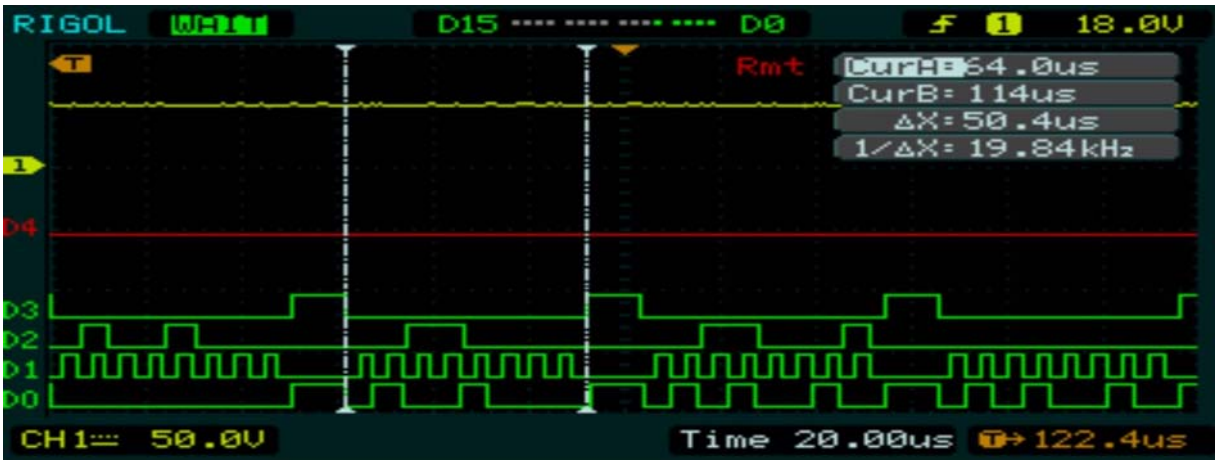


Figure 2

If you poll the SPI interface with '0xFF' (line D2) the modul will answer with '0xFF' (line D0), if the needed information is unavailable.

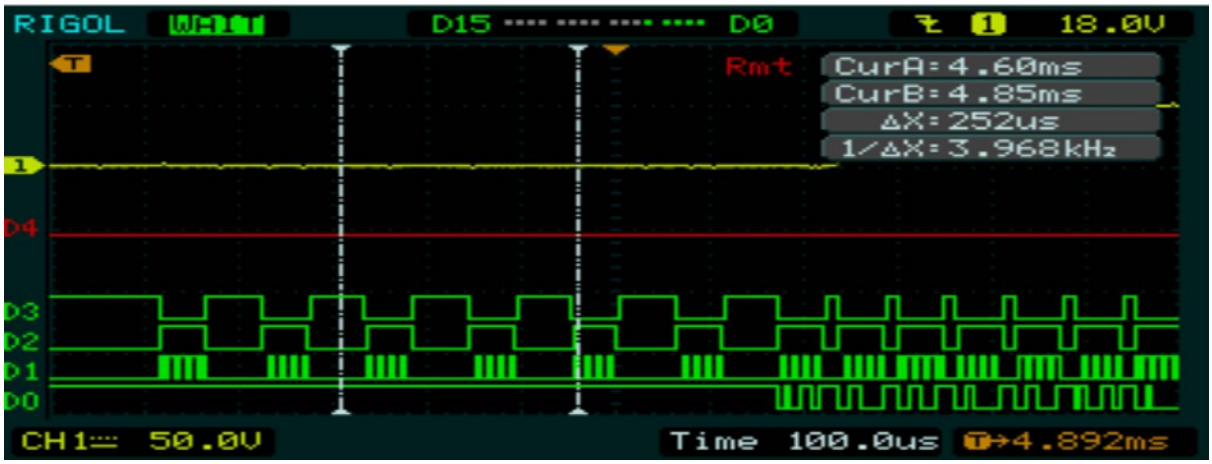


Figure 3

The data between the two markings in the figure 4 shows a posibble answer of the module: 0x4D = 'M' followed by 0x30 = '0' and 0x31 = '1' („M01...“)

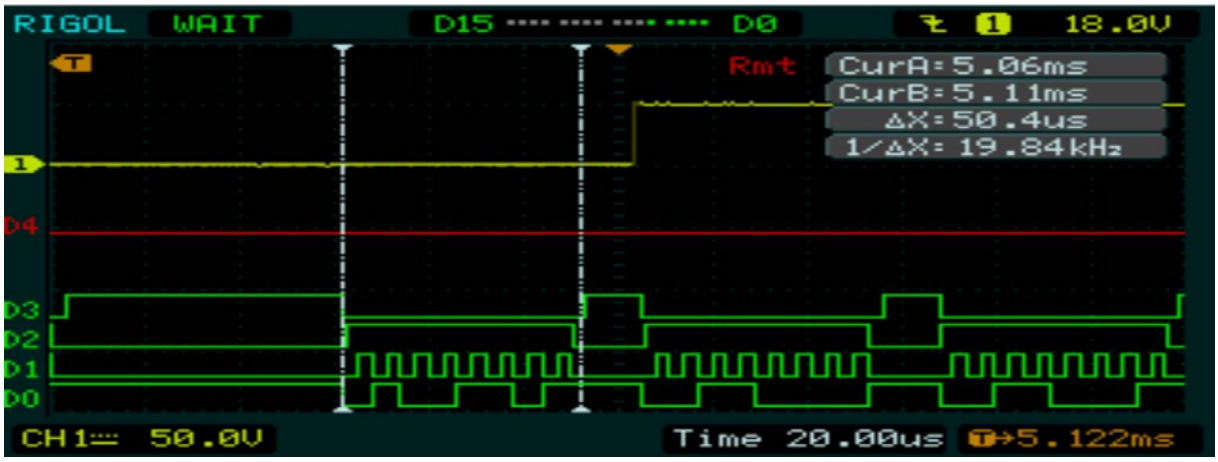


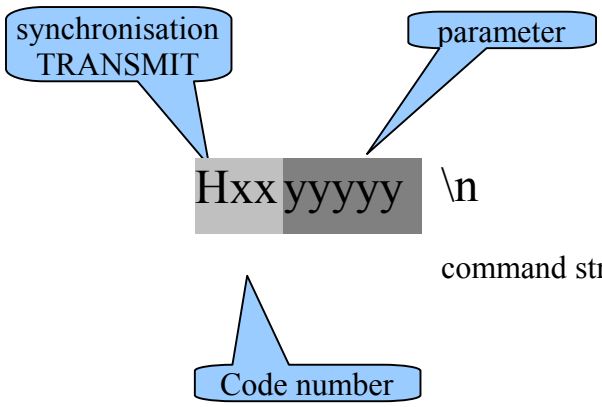
Figure 4

If you transmit the very first character the SPI output is maybe not active (startup in parallel-mode). In this case the first received character is '0xFF'!

Important hint: There should be no communication if there is no need for changements at parameters or monitoring. Especially the signals SCL and SDI should be calm, else every edge change of the SPI communication may influence the receiving signal.

2 Command structure

The communication starts with a command string that is finalized with 0H. Any command string begins with the character 'H' followed by blank characters and HEX characters. The character 0Hex at the end of the transmission will start the decoding of the command string. The input buffer allows maximum 30 characters. In case of an overflow the data aquisition will be stopeed. It depends on the command string if you have to transmit single characters or complete data blocks. The code number (xx) MUST BE USED with 2 HEX characters.



command structure „Transmit to module“

The receive command structure is conform to the transmit command structure. The only difference is that the leading synchronization character is a 'M' if the transmitted command was successful and an 'E' in case of error.

There are 4 states during the decoding of the transmitted command:

- data transfer until terminating character (string end = 0)
- command decoding (master waiting time)
- command execution (master waiting time)
- transmit answer (with 0xFF)

If you transmit commands to the module, the module will return 'R' as long as the end string 0h is recognized. The following transmission break (“pause”) allows the controller of the radio module to decode the command and to prepare the answer in the data buffer.

The master can now poll the information by transmitting 0xFF to the module. If the module is not ready preparing the informations it will return 0xFF. If the answer is available, the module answer starts with 'M' (for message) or 'E' (for error). The following data (blank 0x20 and Hex characters) contains the answer of the radio module (slave) und are finalized by the end string (0x00). If the master recognize this end string (0x00) he must confirm it by transmitting 0x00 (synchronisation)!

The Hex characters can be up to 32Bit long (4Byte) and are separated by the blank character 0x20. The data conversion ((HEX)ASCII--> value) is quiet easy. Each Hex character between the synchronisations character 'M' (normal answer) or 'E' (error) and the next blank character and between all following blank characters can be assigned to value[0],value[1],value[2]...

Value[0] is the receipt for the transmitted command string. Value[1],value[2]... contains the needed data.

2.1 Command strings

Based on the command structure that is described in 1.2. strings will be transmitted from the master to the module. The data maybe generated with the following C-instruction:

```
sprintf(txt,"H%02X %lX",cmd,wert);
sende_SPI(txt);
```

Important hind : **ALL characters are ASCII-characters !!!!!**

2.1.1 GET_STATUS (01) Read status of the module

Action: The module returns the RSSI-Signal (radio signal strength indicator at the antenna input) in 10mV units and the actual working parameter of the frontend amplification. The working parameter of the frontend can only varies if the frontend control is not blocked in the setup of the module (see 2.1.7).

Sample: command string: „H01“
answer: „M01 2F 1B“
value[0] = 01 command string
value[1] = 2F RSSI signal (here 2Fh = 47, field strength = 47*10mV = 470mV)
value[2] = 1B working point of the preamplifier (here 1Bh = 27)

2.1.2 GET_INFO (02) Read module infomations

Action: Returns module type, software version and crystal frequency.

Sample: command string: „H02“
answer: „M02 02 6E C35000“
value[0] = 02 command string
value[1] = 02 module type
value[2] = 6E software version (6E = 110, so the software version is 1.10)
value[3] = C35000 crystal frequency in Hz (here 12800000Hz = 12,8MHz)

2.1.3 GET_CHANNEL (03) Read actual channel number

Action: Returns the actual channel number of the module.

Sample: command string: „H03“
answer: „M03 85“
value[0] = 03 command string
value[1] = 85 channel number (85h = 133, channel 133 is set)

2.1.4 SET_CHANNEL (04) Set a channel number

Action: Switching to another channel number.

Sample: command string: „H04 61“
answer: „M04 61“
value[0] = 04 command string
value[1] = 61 channel number (61h = 97, confirms new channel 97)

2.1.5 SET_MUTE (05) Defines a new mute-threshold

Action: Defines a new theshold for the mute signal. The mute signal shows the receipt of an external radio signal and depends on the RSSI level (we calibrate the threshold to a SINAD value of 20dB). The unit is 10mV.
Special cases: The character 0h calibrates the threshold to the actual RSSI value, 255h restores the factory calibration.

Sample: command string: „H04 61“ (set threshold to 97*10mV=970mV)
answer: „M04 61“
value[0] = 05 command string
value[1] = 61 confirms the new threshold for the mute signal at 970mV

2.1.6 SET_SPIFLAG (06) Define/clear SPI- startflag

Action: The factory preset offers equal rights to the 8bit parallel and the SPI interface. The user can deactivate the 8bit parallel interface by setting the SPI Flag. In this case the module starts with the startup channel that was defined before (see 2.1.8) and the module remains in the SPI-mode. The channel input at the parallel interface is now completely locked and all 8bit frequency words are ignored by the controller of the radio module.

The flag is set by any value > 0 . If the value is 0 the flag is deleted again. The module returns the status of **ALL Flags** (16bit). You can find a list of the flags in the appendix 3.3

Sample: command string: „H06 01“ (set the SPI- flag)
„H06 00“ (delete the SPI- flag)
Answer: „M06 C001“
value[0] = 06 command string
value[1] = C001 flags 1100 0000 0000 0001
└ SPI-flag
└ adaptive control lock

2.1.7 SET_V_DISABLE (07) Locking of the adaptive control of the frontend

Action: Setting of this flag will lock the adaptive control of the frontend and the receiver keeps on maximum sensitivity under all circumstances.

The flag is set by any value > 0 . If the value is 0 the flag is deleted again. The module returns the status of **ALL Flags** (16bit). You can find a list of the flags in the appendix 3.3

Sample: command string: „H07 01“ (set the adaptive control lock)
„H07 00“ (enables the adaptive control)
answer: „M07 C003“
value[0] = 07 command string
value[1] = C003 flags 1100 0000 0000 0011
└ SPI-flag
└ adaptive control lock

2.1.8 SET_START_CH (08) Define start channel of the module

Action: You can define the startup channel (means the start channel when the power supply of the module is switched on) of the module if you set the SPI-flag (see 2.1.6). The module works like a single channel module.

Sample: command string: „H08 67“ (67H = 103, set channel 103 as startup channel)
answer: „M08 67“
value[0] = 08 command string
value[1] = 67 confirms the new startup channel

2.1.9 SET_MAX_CH (09) Define the channel limit

Action: You can define a new supreme channel of the module. All incoming channel numbers will be checked and limited to this supreme channel if the limit is exceeded! The value may consist of 2Byte!

Sample: command string: „H09 78“ (78H = 120, set channel limit to 120)
answer: „M09 78“
value[0] = 09 command string
value[1] = 78 confirms the supreme channel (limit)

2.1.10 RESET_MODUL (10) Restart the module

Action: Restarts the radio module (without answer!)

Sample: command string: „H0A“

2.2 Error codes

If the module returns the string 'E' instead of 'M' , the command code was not executed.

The reason is a wrong command code.

The command code that was not executed will follow the character 'E'.

Sample: command string: H13 20
answer: E13 20 (this command code does not exist)

3 Appendix

3.1 Codes summary

Code	Number	Operation
GET_STATUS	01	RSSI and adaptive frontend control
GET_INFO	02	module typ(16bit), version of the software(16bit), frequency of the crystal (32bit) of the module
GET_CHANNEL	03	read channel
SET_CHANNEL	04	set channel
SET_MUTE	05	defines the mute threshold (0 = to actual RSSI data; 255 = factory setting)
SET_SPIFLAG	06	set of the SPI – startflag
SET_V_DISABLE	07	blocking of the adaptive frontend control
SET_START_CH	08	defines the startup channel (EEPROM)
SET_MAX_CH	09	defines the supreme channel
RESET_MODUL	10	restart of the module

Table 1: Codes summary of the radio module

3.2 Execution time of the command codes

Command string	Transmit	Execution	Receive	Sum
GET_STATUS	250µs	5,2ms	650µs	
GET_INFO	250µs	5,4ms	1,1ms	
GET_CHANNEL	250µs	5,25ms	580µs	
SET_CHANNEL	480µs	5,3ms	420µs	
SET_MUTE	460µs	8,7ms	420µs	
SET_SPIFLAG	420µs	12,3ms	600µs	
SET_V_DISABLE	380µs	12,3ms	580µs	
SET_START_CH	440µs	12,2ms	470µs	
SET_MAX_CH	460µs	12,3ms	470µs	
RESET_MODUL				

Table 2: Execution time of the codes

3.3 Definition of the flags

Description	Bit	Operation
SPI_FLAG	0	operation mode under startup conditions
VORSTUFE_FLAG	1	blocking of the adaptive frontend control
TX_PWR_ON	2	startup with full transmit power (TX-module)
POTI_VORSTUFE	14	potentiometer frontend available (RX-module)
POTI_TCXO	15	potentiometer TCXO available

Table 3: flag definitions