SPI

Implementation in the IQRF platform

User's Manual

Simple way to smarter wireless solutions

SPI Overview:

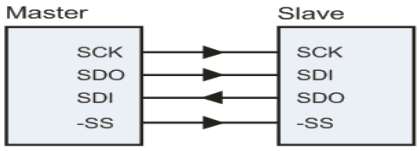
IQRF transceiver modules can communicate with external peripherals via the SPI interface.

SPI™ (Serial Peripheral Interface, introduced by Motorola) is a standard serial four wire synchronous data bus that can operate in full duplex. Devices communicate in master/slave mode with a single master initiating data frames. Multiple slave devices are allowed with individual slave select lines.

The **SPI bus** specifies four logic signals:

SPI signal	TR pin	function	
SCK	C6	Serial Clock	issued by master
SDI	C7	Serial Data In	
SDO	C8	Serial Data Out	
–SS	C5	Slave Select	issued by master, active low

The SPI bus with a single slave:



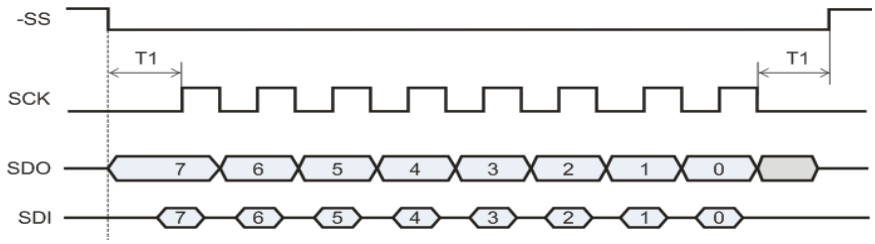
The IQRF transceiver modules can communicate as SPI slaves. Full as well as half duplex is supported. The SPI protocol is implemented in IQRF operating system. Thanks to the state machine architecture the communication is fully synchronous without any timeouts. It is packet oriented and works in OS background. Packets consist of selectable number of bytes (0 to N_{max}). In time constrained cases the communication can be slowed down to work in OS foreground with longer delays between individual bytes. Data stream can even be suspended at all and continue at any time later.

The TR modules operate according the following **slave specification:**

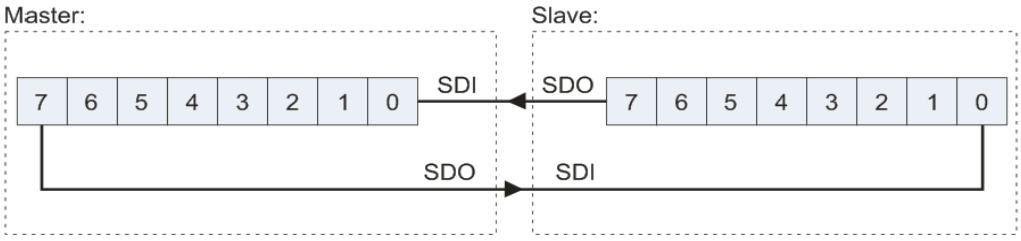
Slave select	Active low
Idle clock polarity	Low
Clock edge	Output data on falling SCK edge

Data transfer (from the slave point of view):

To start the communication, the master pulls the Slave select low. Then the master must wait for at least the T1 period before starting to issue clock cycles. During each SPI clock cycle, a duplex data bit can be transmitted to complete a full duplex byte transmission in 8 clocks.

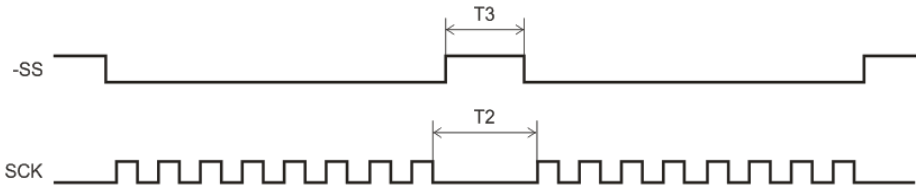


Data is transferred using two internal shift registers to form a circular buffer, MSb first. After the registers have been shifted, the master and slave have swapped register values.



If there is more data to swap, the shift registers are loaded with new data and the process repeats. The master must wait for at least the T2 period before sending next byte. Disactivating the Slave Select for the T3 period between bytes is necessary.

Timing:



SCK	SPI clock	250 kHz	max.
T1	–SS to SCK, SCK to –SS	10 µs	min.
T2	delay between bytes	100 µs 500 µs – in Slow mode	min.
T3	–SS pulse between bytes	20 µs	min.

For detailed information see datasheet of respective microcontroller.

Example: SPI configuration of PIC16F88 used as a master:

	7	6	5	4	3	2	1	0
SSPSTAT	**SMP**	**CKE**	D/-A	P	S	R/-W	UA	**BF**
	0	1	–	–	–	–	–	read only
SSPCON	**WCOL**	**SSPOV**	**SSPEN**	**CKP**	**SSPM**			
	0	0	1	0	0	0	x	x

SMP = 0 Input data sampled at middle of data output time.
CKE = 1 Transmit occurs on transition from active to idle clock state.
CKP = 0 Transmit on rising edge, receive on falling edge. Clock idle state is a low level.
SSPM: SPI Master mode, clock timing (xx) depends on application.

Packet structure:

The master can send two types of packets with the following structure:

Master checks the SPI status of the module:

Master	**SPI_CHECK**
Slave	SPISTAT

Master reads/writes a packet from/to the module:

Master	**SPI_CMD**	PTYPE	DM_1	DM_2	---	$DM_{SPIDLEN}$	CRCM
Slave	SPISTAT	SPISTAT	DS_1	DS_2	---	$DS_{SPIDLEN}$	CRCS

Where:

`SPI_CHECK = 0x00`
`SPI_CMD = 0xF0`

`SPISTAT`: SPI status of the module

hex value	SPI status
00	SPI not working (disabled by the disableSPI() command)
07	SPI suspended by the stopSPI() command
3F	SPI not ready (buffer full, last CRCM O.K.)
3E	SPI not ready (buffer full, last CRCM error)
40 to 63	SPI data ready. Value - 0x40 = number of bytes to be sent from the slave (1 to N_{max})
80	SPI ready (communication mode)
81	SPI ready (programming mode)
82	SPI ready (debugging mode)
83	SPI not working in background (e.g. during receiving of RF packet) – Slow mode Master should prolong the delay between individual bytes when this status is received. See the T2 parameter in the table above.
FF	SPI not working (HW error)

SPI status of the module is indicated by the IQRF IDE when used together with related IQRF development tools (e.g. CK-USB-02):

`PTYPE`:

b7	b6	b5	b4	b3	b2	b1	b0
CTYPE		SPIDLEN					

`CTYPE`: communication type
10: full duplex (the master reads/writes from/to the module, bufferCOM changed)
`00`: half duplex (the master reads from the module, bufferCOM unchanged)

`SPIDLEN`: data length (from 1 to N_{max})
TR-xxx-xxA: $N_{max} = 35$
TR-xxx-xxB: $N_{max} = 35$

`DM`: data from the master
`DS`: data from the slave

`CRCM = SPI_CMD xor PTYPE xor` DM_1 `xor` DM_2 `... xor` $DM_{SPIDLEN}$ `xor 0x5F`
`CRCS = PTYPE xor` DS_1 `xor` DS_2 `... xor` $DS_{SPIDLEN}$ `xor 0x5F`

IQRF OS functions related to SPI:

void ***enableSPI****()*	*Enables SPI communication. Hardware of the microcontroller is configured for SPI, related pins can not be used as general I/Os.*
void ***disableSPI****()*	*Disables SPI communication. SPI HW module of the microcontroller is switched off, related pins are general I/Os.*
void ***startSPI****(length)*	*Starts SPI packet transmission. This operates in OS background except of the Slow mode case.*
void ***stopSPI****()*	*Suspends SPI communication. It can continue at any time later (after another* ***startSPI*** *command)*
bit ***getStatusSPI****()*	*Updates SPI flags and packet length and checks whether SPI is busy*

SPI flags: available in **param2**

bit 7			bit 4	bit 3	bit 2		bit 0
–	–	–	_SPICRCok	_SPIRX	–	–	–

_SPIRX Something received on SPI.
_SPICRCok The last received SPI CRCM was O.K.

SPI data length: available in **param1**

Caution: SPI data length and SPI flags are copied to `param1` and `param2` (parameters used by OS functions) after the `getStatusSPI` command. They are only valid until another OS command using these parameters is called.

Application:

See IQRF OS User's manual, Application examples, www.iqrf.org and www.iq-esupport.com.

Document history:

080516 First release
090427 `_SPIbusy` and `_SPIwrite` flags not documented. Use `getStatusSPI()` instead of `_SPIbusy`.

Sales and Service

Corporate office:

MICRORISC s.r.o., Delnicka 222, 506 01 Jicin, Czech Republic, EU
Tel: +420 493 538 125, Fax: +420 493 538 126, www.microrisc.com

Partners and distribution:

please visit www.iqrf.org/partners

Quality management:

ISO 9001 : 2000 certified

Trademarks:

The IQRF name and logo are registered trademarks of MICRORISC s.r.o.
PIC, SPI, Microchip, RFM and all other trademarks mentioned herein are property of their respective owners.

Legal:

All information contained in this publication is intended through suggestion only and may be superseded by updates without prior notice. No representation or warranty is given and no liability is assumed by MICRORISC s.r.o. with respect to the accuracy or use of such information.

Without written permittion it is not allowed to copy or reproduce this information, even partially.

No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

The IQRF products utilize several patents (CZ, EU, US)

On-line support: http://iq-esupport.com

Simple way to smarter wireless solutions