

IQRF OS

Operating System

**version 2.11
for TR-52B and TR-53B**

Reference Guide



Simple way to smarter wireless solutions

Content

Quick reference.....	4
Parameters.....	4
Functions.....	4
OS functions.....	6
Control.....	6
reset.....	6
iqrSleep.....	7
setRFsleep.....	7
debug.....	8
getSupplyVoltage.....	9
getTemperature.....	9
Active waiting.....	10
waitMS.....	10
waitDelay.....	10
Timing on background.....	11
startCapture.....	11
captureTicks.....	11
startDelay.....	12
isDelay.....	12
LED indication.....	13
setOnPulsingLED.....	13
setOffPulsingLED.....	13
pulsingLEDR.....	14
pulseLEDR.....	14
stopLEDR.....	15
pulsingLEDG.....	15
pulseLEDG.....	16
stopLEDG.....	16
EEPROM.....	17
eeReadByte.....	17
eeReadData.....	17
eeWriteByte.....	18
eeWriteData.....	18
RAM.....	19
readFromRAM.....	19
writeToRAM.....	20
setPIR1.....	21
Buffers, data blocks.....	22
clearBufferINFO.....	22
copyBufferINFO2COM.....	22
copyBufferINFO2RF.....	23
copyBufferRF2COM.....	23
copyBufferRF2INFO.....	23
copyBufferCOM2RF.....	24
copyBufferCOM2INFO.....	24
compareBufferINFO2RF.....	24
copyMemoryBlock.....	25
moduleInfo.....	26
applInfo.....	26
SPI.....	27
enableSPI.....	27
disableSPI.....	27
startSPI.....	28
stopSPI.....	29
getStatusSPI.....	30
RF.....	31
setTXpower.....	31
setRFspeed.....	31
setRFband.....	32
setRFchannel.....	32
setRFmode.....	33

checkRF.....	34
RFTXpacket.....	35
RFRXpacket.....	37
Networking.....	38
setNetworkOne.....	38
setNetworkTwo.....	38
setNetworkingOff.....	39
setCoordinatorMode.....	40
setNodeMode.....	40
setNetworkFilteringOn.....	41
setNetworkFilteringOff.....	41
setLoggingOn.....	42
setLoggingOff.....	42
getNetworkParams.....	43
Routing.....	44
setRoutingOn.....	44
setRoutingOff.....	44
Bonding – Node only.....	45
bondRequest.....	45
amIBonded.....	46
removeBond.....	47
wipeBondNR.....	47
Bonding – Coordinator only.....	48
bondNewNode.....	48
isBondedNode.....	49
removeBondedNode.....	49
rebondNode.....	50
clearAllBonds.....	50
Documentation and Information.....	51
Document revision.....	51
Sales and Service.....	53

Quick reference

Parameters

Values between system functions and superordinate program are passed on via parameters. OS uses 4 parameters in total: `param1` (1 B), `param2` (1 B), `param3` (2 B) and `param4` (2 B). Their location in memory see the RAM map [2]. Individual functions have up to 3 parameters. Some functions use some of these params and W (PIC accumulator) to return output values. Note that they are valid until another function using the same parameter or the `debug` function is called by the user. Additionally, some functions use some params as work variables that is why their previous content can be destroyed.

Additionally, some functions can also use PIC STATUS flags. Refer to the PIC datasheet [8] (`STATUS` register) for details.

Functions

Control		6
<code>reset()</code>	initialization of microcontroller, OS and application	6
<code>iqrfSleep()</code>	set the TR module in power saving mode (Sleep)	7
<code>setRFsleeeep()</code>	set the RF IC in power saving mode (Sleep)	7
<code>debug()</code>	enter the debug mode	8
<code>uns8 getSupplyVoltage()</code>	get voltage level for battery check	9
<code>getTemperature()</code>	temperature measurement	9
Active waiting		10
<code>waitMS(ms)</code>	active waiting (time in ms)	10
<code>waitDelay(ticks)</code>	active waiting (time in ticks)	10
Timing on background		11
<code>startDelay(ticks)</code>	start waiting (time in ticks)	12
<code>bit isDelay()</code>	still waiting	12
<code>startCapture()</code>	resets counter of ticks	11
<code>captureTicks()</code>	get number of ticks counted	11
LED indication		13
<code>setOnPulsingLED(ticks)</code>	LEDR and LEDG On times setting (for blinking)	13
<code>setOffPulsingLED(ticks)</code>	LEDR and LEDG Off times setting (for blinking)	13
<code>pulsingLEDR()</code>	red LED activation (blinking on background)	14
<code>pulseLEDR()</code>	single red LED pulse (one flash on background)	14
<code>stopLEDR()</code>	red LED off, blinking stopped	15
<code>pulsingLEDG()</code>	green LED activation (blinking on background)	15
<code>pulseLEDG()</code>	single green LED pulse (one flash on background)	16
<code>stopLEDG()</code>	green LED off, blinking stopped	16
EEPROM		17
<code>uns8 eeReadByte(addr)</code>	read one byte	17
<code>eeReadData(addr, length)</code>	read a block	17
<code>eeWriteByte(addr, data)</code>	write one byte	18
<code>eeWriteData(addr, length)</code>	write a block	18
RAM		19
<code>uns8 readFromRAM(addr)</code>	read one byte	19
<code>writeToRAM(addr, data)</code>	write one byte	20
<code>setPIR1(x)</code>	setting of microcontroller peripheral flags	21
Buffers, data blocks		22
<code>clearBufferINFO()</code>	bufferINFO clearing	22
<code>copyBufferINFO2COM()</code>	copy bufferINFO to bufferCOM	22
<code>copyBufferINFO2RF()</code>	copy bufferINFO to bufferRF	23
<code>copyBufferRF2COM()</code>	copy bufferRF to bufferCOM	23
<code>copyBufferRF2INFO()</code>	copy bufferRF to bufferINFO	23
<code>copyBufferCOM2RF()</code>	copy bufferCOM to bufferRF	24
<code>copyBufferCOM2INFO()</code>	copy bufferCOM to bufferINFO	24
<code>bit compareBufferINFO2RF(length)</code>	comparison of bufferINFO and bufferRF	24
<code>copyMemoryBlock(uns16 from, uns16 to, uns8 length)</code>	copy any data block to any position	25
<code>moduleInfo()</code>	get info about transceiver module and OS	26
<code>appInfo()</code>	copy info about application from EEPROM to bufferINFO	26

SPI		27
<code>enableSPI ()</code>	SPI communication line activation	27
<code>disableSPI ()</code>	SPI communication line deactivation	27
<code>startSPI (length)</code>	SPI packet transmission	28
<code>stopSPI ()</code>	SPI stopping	29
<code>bit getStatusSPI ()</code>	SPI status, update SPI flags	30
RF		31
<code>setTXpower (level)</code>	RF power setting (7 levels)	31
<code>setRFspeed (speed)</code>	select RF bit rate	31
<code>setRFband (band)</code>	select RF band (868 MHz or 916 MHz)	32
<code>setRFchannel (channel)</code>	select RF channel	32
<code>setRFmode (mode)</code>	select RF power management mode	33
<code>checkRF (level)</code>	detect incoming RF signal	34
<code>RFpacket ()</code>	send a packet from <code>bufferRF</code> via RF	37
<code>bit RFRpacket ()</code>	receive a packet via RF to <code>bufferRF</code>	37
Networking		38
<code>setNetworkOne ()</code>	network 1 selected	38
<code>setNetworkTwo ()</code>	network 2 selected	38
<code>setNetworkingOff ()</code>	networking disabled	39
<code>setCoordinatorMode ()</code>	device is the Coordinator	40
<code>setNodeMode ()</code>	device is a Node	35
<code>setNetworkFilteringOn ()</code>	packets accepted from current network only	41
<code>setNetworkFilteringOff ()</code>	packets accepted from both networks	41
<code>setLoggingOn ()</code>	communication with adjacent nodes logged	42
<code>setLoggingOff ()</code>	communication with adjacent nodes not logged	42
<code>uns8 getNetworkParams ()</code>	get information about the network	43
Routing		44
<code>setRoutingOn ()</code>	outgoing packets routed via other devices on background	44
<code>setRoutingOff ()</code>	no routing for outgoing packets	44
Bonding - Node		45
<code>bit bondRequest ()</code>	request for bonding	45
<code>bit amIBonded ()</code>	is the Node bonded?	46
<code>removeBond ()</code>	unbonding	47
<code>wipeBondNR ()</code>	unbonding and reserved node number cancellation	47
Bonding - Coordinator		48
<code>bit bondNewNode ()</code>	bonding a Node	48
<code>bit isBondedNode (n)</code>	is the Node bonded?	49
<code>removeBondedNode (n)</code>	unbonding a Node	49
<code>bit rebondNode (n)</code>	rebonding a Node	50
<code>clearAllBonds ()</code>	clearing of all bonds	50

Unless otherwise stated, all functions are the `void` type and all their parameters are the `uns8` type.

OS functions**Control****reset**

Function	Initialization of microcontroller, OS and application
Purpose	If needed, it is possible to initialize the application and run the program from the very beginning again.
Syntax	<code>void reset ()</code>
Parameters	–
Return value	–
Output values	–TO = 0
Preconditions	–
Remarks	This is forced watchdog overflow reset. To distinguish the power down reset from this one the –TO (STATUS . 4) status flag is intended. –TO = 0 after reset () execution –TO = 1 after power up Refer to the PIC datasheet [8] (STATUS register) for details.
Side effects	It is strictly specified which RAM registers are initialized and which are unchanged. Refer to the PIC datasheet [8] (Reset) for details.
See also	–
Example	<pre>if (Error == 1) reset ();</pre>

iqrfSleep

Function	Setting the TR module in power saving mode (Sleep)
Purpose	Easy and efficient power management. This function, once called, puts the module into the Sleep mode. Wake-up can be caused by power off/on, watchdog timeout or on the C5 pin change.
Syntax	<code>void iqrfSleep ()</code>
Parameters	–
Return value	–
Output values	The STATUS flags <code>-PD (STATUS . 3)</code> and <code>-TO (STATUS . 4)</code> are setup by this function: <code>-TO = 1, -PD = 0</code>
Preconditions	This functions operates like the PIC machine instruction Sleep. Additionally, OS suspends all HW resources that are under its control (RF circuitry, timers, internal PIC pins, LEDs etc.). The user should do the same for resources used by the application before entering the Sleep mode to achieve minimal power consumption. For wake-up on pin change the required sequence should be executed. Wake-up on pin change is default disabled.
Remarks	There are no restriction concerning Sleep and wake-up due to OS from OS v2.03. All features are under user's control. RBIF flag is not cleared to allow to distinguish wake-up type. See example E01-TR [10].
Side effects	Global interrupt enable (GIE) is controlled by OS again after wake-up.
See also	<code>setRFsleep</code>
Example1	<pre> // minimize consumption (depends on resources used by the user) Motor = 0; // Stop the motor ADON = 0; // Disable A/D converter SWDTEN = 0; // Disable watchdog iqrfSleep(); // Put the module into Sleep mode </pre>
Example2	<pre> // wake-up on pin change. See example E01-TX. GIE = 0; // Disable all interrupts RBIE = 1; // Enable wake-up on pin change // It was default disabled by OS up to v2.03 // automatically enabled before iqrfSleep // and automatically disabled after iqrfSleep // Now wake-up on pin change is fully under user control SWDTEN = 0; // Disable watchdog to save ~2µA iqrfSleep(); // Put the module into Sleep mode RBIF = 0; // Clear flag if (buttonPressed) // If button is pressed { ... } // ... </pre>

setRFsleep

Function	Setting RF circuitry in power saving mode (Sleep)
Purpose	Easy and efficient power management. This function, once called, puts all RF circuitry in Sleep mode.
Syntax	<code>void setRFsleep ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	0.6 mA typ. is saved. RF response is prolonged for 2 ms typ., 7 ms max. due to wake-up. Wake-up can be caused by <code>RFTXpacket</code> , <code>RFRXpacket</code> , <code>checkRF</code> or <code>getSupplyVoltage</code> .
Side effects	–
See also	<code>iqrfSleep</code> , <code>getSupplyVoltage</code> , <code>checkRF</code> , <code>RFTXpacket</code> , <code>RFRXpacket</code>
Example	<code>setRFsleep(); // Put the RF circuitry in Sleep mode</code>

debug

Function	Enter the debug mode
Purpose	IQRF OS directly supports debugging and testing. It is possible to stop the application wherever you need and display internal values (variables, RAM registers, EEPROM etc.) and then continue later on.
Syntax	<code>void debug ()</code>
Parameters	–
Return value	–
Output values	OS directly returns no value but supports using W (PIC accumulator) to identify which of the debug points is currently active.
Preconditions	<ul style="list-style-type: none"> • Debug should be used with corresponding development kit (e.g. CK-USB-02/03) and the IQRF IDE development environment. • To avoid possible HW collision with respect to user application, <code>debug</code> operates only under the following conditions: <ul style="list-style-type: none"> • Pins C5 to C8 are configured for SPI in respective TRIS bits (C7 in, the others out). It is arranged by OS by default. • The <i>Check Mode</i> function is enabled in IQRF IDE. Otherwise no communication on these pins is initiated by debug tools even though TR is in debug mode until the <i>Check Mode</i> is enabled. • SPI need not be enabled by <code>enableSPI</code>
Remarks	Number of <code>debug()</code> instances is unlimited. The application is running until a <code>debug</code> function is encountered. Then the program is stopped and the module is switched to the debug mode allowing IQRF IDE to display values. The module stays in the debug mode until the user selects the <i>Skip Debug</i> button. Then the application program continues running until another <code>debug</code> function is encountered and so on. See IQRF IDE Help and example E04-EEPROM [10].
Side effects	<ul style="list-style-type: none"> • <code>param1</code> to <code>param4</code> are not displayed • Watchdog is cleared while in Debug mode
See also	–
Example	<pre>if (compareBufferINFO2RF(4)) W = 1; // match else W = 2; // mismatch debug(); // Skip Debug 1 or 2 will be displayd here according the result</pre>

getSupplyVoltage

Function	Power supply measurement (up to 3.8 V)
Purpose	Battery check for discharge-sensitive batteries
Syntax	uns8 getSupplyVoltage ()
Parameters	–
Return value	level = 1, 2, ...15 Voltage > 2.25 V + level × 0.1 V
Output values	–
Preconditions	–
Remarks	<ul style="list-style-type: none"> • Internal power supply voltage is checked. • In case of TR-52B it is the LDO output but not actual battery voltage. This value is 3.0 V typ. if battery is O.K. and drops down if battery is low. • To evaluate the battery, take into consideration your battery type and power supply circuitry with respect to diodes and other possible voltage drops. • The detector circuit has a built-in 50 mV hysteresis.
Side effects	The RF circuitry wakes up (in case of sleeping).
See also	setRFsleep
Example	<pre>if (getSupplyVoltage () > 7) ... // Voltage > 2.95V else ... // Low battery</pre>

getTemperature

Function	For TR-52B only: Converts the analog temperature sensor output to digital value.
Purpose	Temperature measurement
Syntax	void getTemperature ()
Parameters	–
Return value	–
Output values	Unsigned 10b result is stored in ADRES (ADRESH and ADRESL PIC special function registers), right justified.
Preconditions	–
Remarks	<p>Temperature can be measured with on-board temperature sensor and internal A/D converter of the microcontroller. Temperature (in °C) evaluation: $T_a = ADRES \times 75/256 - 50$.</p> <p>See example E08–TEMPERATURE [10].</p> <p>Temperature accuracy can be improved from ±2 °C (typical) to ±0.1 °C by individual calibration. Refer to IQRF support [6].</p>
Side effects	–
See also	–
Example1	<pre>// Temperature measurement uns16 temperature; getTemperature (); // Temperature measurement temperature.high8 = ADRESH&0x03; // 10 b result is stored in ADRESH temperature.low8 = ADRESL; // and ADRESL</pre>
Example2	<pre>// Conversion to °C temperature = 75/256 * temperature - 50 [°C] temperature *= 75; temperature >>= 8; temperature -= 50; // temperature.high8 = tens of °C // temperature.high8 = units of °C</pre>

Active waiting

waitMS

Function	Wait specified number of miliseconds
Purpose	Time delay generation
Syntax	<code>void waitMS (ms)</code>
Parameters	ms - time to wait in miliseconds (1 - 255)
Return value	–
Output values	–
Preconditions	This function can be combined with <code>waitDelay</code> , <code>startCapture</code> and <code>captureTicks</code> .
Remarks	This is an active waiting (on OS foreground). No other operation runs on OS foreground during waiting. Time precision depends on internal RC oscillator. Thus, the delay can vary with temperature etc. See respective PIC datasheet [8].
Side effects	–
See also	<code>waitDelay</code> , <code>startDelay</code>
Example	<pre>waitMS(10); // Delay 10 ms. Program stays here for the whole 10 ms period ... // and continues here just after the period elapsed.</pre>

waitDelay

Function	Wait specified number of ticks
Purpose	Time delay generation
Syntax	<code>void waitDelay (ticks)</code>
Parameters	ticks – time to wait in 10 ms intervals (1 - 255)
Return value	–
Output values	–
Preconditions	This function can be combined with <code>waitMS</code> .
Remarks	This is the active waiting (on OS foreground). No other operation runs on OS foreground during waiting.
Side effects	This function should not be combined with <code>startDelay</code> . Internal ticks are based on internal RC oscillator. Thus, the delay can vary with temperature etc. See respective PIC datasheet [8]. For short time delays <code>waitMS</code> is more precise due to latency.
See also	<code>waitMS</code> , <code>startDelay</code>
Example	<pre> // LED on for 0.5 s _LED = 1; waitDelay(50); // Delay 500 ms. Program stays here for 500 ms _LED = 0; // and continues here just after the period elapsed.</pre>

Timing on background

startCapture

Function	Reset and start the Capture timer
Purpose	Initialization of time measurement or delay generation
Syntax	<code>void startCapture ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	This function can be combined with <code>waitMS</code> .
Remarks	Capture timer is a resettable counter of OS ticks (10 ms system intervals) running on OS background. This function clears the counter and starts counting.
Side effects	Functionality is affected by <code>bondRequest</code> , <code>bondNewNode</code> and <code>RFRXpacket</code> .
See also	<code>captureTicks</code>
Example	See <code>captureTicks</code>

captureTicks

Function	Get number of ticks counted from the last <code>startCapture</code> and <code>captureTicks</code> calling.
Purpose	Measurement of elapsed time.
Syntax	<code>void captureTicks ()</code>
Parameters	–
Return value	–
Output value	<ul style="list-style-type: none"> • <code>param3</code>: ticks counted from the last <code>startCapture</code> (0 - 65535) • <code>param4</code>: ticks counted from the last <code>captureTicks</code> (0 - 65535)
Preconditions	<ul style="list-style-type: none"> • <code>startCapture</code> should be used at least once before. • To ensure correct operation the counter must not overflow. That is why <code>captureTicks</code> should be called max. ~655 s after last <code>startCapture</code> or <code>captureTicks</code> calling.
Remarks	See example E05–DELAYS [10]
Side effects	Functionality is affected by <code>bondRequest</code> , <code>bondNewNode</code> and <code>RFRXpacket</code> . Internal ticks are based on internal RC oscillator. Thus, the delay can vary with temperature etc. See respective PIC datasheet [8].
See also	<code>startCapture</code>
Example	<pre>startCapture(); // Reset counter of ticks waitMS(200); // Delay 200 ms captureTicks(); // param3 == 20 waitMS(150); // Delay 150 ms captureTicks(); // param3 == 35, param4 == 15 startCapture(); // Reset counter of ticks waitMS(100); // Delay 100 ms captureTicks(); // param3 == 10</pre>

startDelay

Function	Preset and start the Delay timer
Purpose	Initialization of time measurement or delay generation
Syntax	void startDelay (ticks)
Parameters	uns8 ticks: number of ticks (10 ms system intervals) to be measured (1-255)
Return value	–
Output values	–
Preconditions	This function can be combined with waitMS.
Remarks	The Delay timer measures specified time period on OS background. The result is available via the isDelay function.
Side effects	This function does not work properly if the waitDelay, RFRXpacket or RFTXpacket functions are active.
See also	isDelay, waitDelay
Example	See isDelay

isDelay

Function	Information whether specified delay is still in progress
Purpose	Time measurement or delay generation
Syntax	bit isDelay ()
Parameters	–
Return value	<ul style="list-style-type: none"> • 1: still in progress • 0: elapsed
Output values	–
Preconditions	startDelay should be used before.
Remarks	<ul style="list-style-type: none"> • The Delay timer measures specified time period. The result is available via the isDelay function. • Tip: the clrwdt instruction should be used to avoid unintentional watchdog reset during the delay. • See example E05-DELAYS [10].
Side effects	–
See also	startDelay
Example	<pre> // LED on for 1 s _LED = 1; startDelay(100); // Start 1 sec delay counting on OS background while (isDelay()) // Wait until the delay is over { clrwdt(); // Any useful operation on OS foreground can be ... // performed during waiting } _LED = 0; // Continue here after 1 sec </pre>

LED indication

setOnPulsingLED

Function	LEDs On time setting (red as well as green)
Purpose	Specification of the "On" time for LEDs (either for a single flash or for blinking)
Syntax	<code>void setOnPulsingLED (ticks)</code>
Parameters	<code>uns8 ticks</code> : number of ticks (10 ms system intervals) (1-255)
Return value	–
Output values	–
Preconditions	–
Remarks	Default value is 5 (50 ms).
Side effects	–
See also	<code>setOffPulsingLED</code> , <code>pulsingLEDR</code> , <code>pulseLEDR</code> , <code>pulsingLEDG</code> , <code>pulseLEDG</code>
Example	See <code>setOffPulsingLED</code>

setOffPulsingLED

Function	LEDs Off time setting (red as well as green)
Purpose	Specification of the "Off" time for LEDs (for blinking)
Syntax	<code>void setOffPulsingLEDR (ticks)</code>
Parameters	<code>uns8 ticks</code> : number of ticks (10 ms system intervals) (1-255)
Return value	–
Output values	–
Preconditions	–
Remarks	Default value is 20 (200 ms).
Side effects	–
See also	<code>setOnPulsingLED</code> , <code>pulsingLEDR</code> , <code>pulsingLEDG</code>
Example	<pre> // Change blinking to 250 ms On / 750 ms Off setOnPulsingLEDR(25); // 250 ms On setOffPulsingLEDR(75); // 750 ms Off </pre>

pulsingLEDR

Function	Red LED blinking
Purpose	Continuous red LED blinking on OS background
Syntax	<code>void pulsingLEDR ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	Blinking times should be defined in advance by <code>setOnPulsingLED</code> and <code>setOffPulsingLED</code> . The appropriate PIC pin is configured as an output automatically.
Remarks	Blinking continues until it is stopped by the user (e.g. by <code>stopLEDR</code>).
Side effects	–
See also	<code>setOnPulsingLED</code> , <code>setOffPulsingLED</code> , <code>stopLEDR</code> , <code>pulseLEDR</code>
Example1	<code>pulsingLEDR (); // continuous blinking on OS background</code>
Example1	<pre>// Blinking for 2 s pulsingLEDR (); // blinking for 2 s on OS background waitDelay(200); // 2 s delay generated on foreground stopLEDR (); // Stop blinking</pre>

pulseLEDR

Function	Single red LED flash
Purpose	Red LED flash on OS background
Syntax	<code>void pulseLEDR ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	Flash time should be defined in advance by <code>setOnPulsingLED</code> . The appropriate PIC pin is configured as an output automatically.
Remarks	The on-board LEDs can also be directly controlled on OS foreground using C commands for manipulating the <code>_LEDR</code> output (the pin the red LED is connected to) and corresponding control bit (TRISA.2).
Side effects	<ul style="list-style-type: none"> • The appropriate PIC pin is not restored to the state before <code>pulsingLEDR/pulseLEDR</code> (TRISA.2 == 0, <code>_LEDR</code> == 0 after finishing on background). • Background activity overrides setting of <code>_LEDR</code> (the on-board red LED pin) made by the application. Thus, attention should be taken when using direct setup of LED output pin combined with LED functions. Possible background LED routines can overwrite the status set by direct manipulation with the pin.
See also	<code>setOnPulsingLEDR</code> , <code>pulsingLEDR</code> , <code>stopLEDR</code>
Example	<pre>setOnPulsingLEDR(10); // 100 ms On pulseLEDR (); // Single red LED flash for 100 ms on OS background ... // Program continues immediately, // not waiting until the delay expires. // LED will be switched off after 100 ms automatically</pre>

stopLEDR

Function	Red LED off, blinking stopped
Purpose	Stops the red LED activity on OS background
Syntax	<code>void stopLEDR ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	–
Side effects	<ul style="list-style-type: none"> • The appropriate PIC pin is not restored to the state before pulsingLEDR/pulseLEDR (TRISA.2 == 0, _LEDR == 0 after finishing on background). • Background activity overrides setting of _LEDR (the on-board LED pin) made by the application. Thus, attention should be taken when using direct setup of LED output pin combined with LED functions. Possible background LED routines can overwrite the status set by direct manipulation with the pin.
See also	pulsingLEDR, pulseLEDR
Example1	<pre>pulsingLEDR(); // Start blinking on OS background ... // Blinking continues during any operation stopLEDR(); // Stop blinking</pre>
Example2	<pre>pulseLEDR(); // Red LED On on OS background ... // continuously lighting during any operation ... // until specified time expired stopLEDR(); // or LED is switched Off by this command</pre>

pulsingLEDG

Function	Green LED blinking
Purpose	Continuous green LED blinking on OS background
Syntax	<code>void pulsingLEDG ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	Blinking times should be defined in advance by setOnPulsingLED and setOffPulsingLED. The appropriate PIC pin is configured as an output automatically.
Remarks	Blinking continues until it is stopped by the user (e.g. by stopLEDG).
Side effects	–
See also	setOnPulsingLED, setOffPulsingLED, stopLEDG, pulseLEDG
Example1	<pre>pulsingLEDG(); // continuous blinking on OS background</pre>
Example1	<pre>// Blinking for 2 s pulsingLEDG(); // blinking for 2 s on OS background waitDelay(200); // 2 s delay generated on foreground stopLEDG(); // Stop blinking</pre>

pulseLEDG

Function	Single green LED flash
Purpose	Green LED flash on OS background
Syntax	<code>void pulseLEDG ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	Flash time should be defined in advance by <code>setOnPulsingLED</code> . The appropriate PIC pin is configured as an output automatically.
Remarks	The on-board LEDs can also be directly controlled on OS foreground using C commands for manipulating the <code>_LEDG</code> output (the pin the green LED is connected to) and corresponding control bit (TRISB.7).
Side effects	<ul style="list-style-type: none"> • The appropriate PIC pin is not restored to the state before <code>pulsingLEDG/pulseLEDG</code> (<code>TRISB.7 == 0, _LEDG == 0</code> after finishing on background). • Background <code>activity</code> overrides setting of <code>_LEDG</code> (the on-board green LED pin) made by the application. Thus, attention should be taken when using direct setup of LED output pin combined with LED functions. Possible background LED routines can overwrite the status set by direct manipulation with the pin.
See also	<code>setOnPulsingLEDG</code> , <code>pulsingLEDG</code> , <code>stopLEDG</code>
Example	<pre>setOnPulsingLEDG(10); // 100 ms On pulseLEDG(); // Single green LED flash for 100 ms on OS background ... // Program continues immediately, // not waiting until the delay expires. // LED will be switched off after 100 ms automatically</pre>

stopLEDG

Function	Green LED off, blinking stopped
Purpose	Stops the green LED activity on OS background
Syntax	<code>void stopLEDG ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	–
Side effects	<ul style="list-style-type: none"> • The appropriate PIC pin is not restored to the state before <code>pulsingLEDG/pulseLEDG</code> (<code>TRISB.7 == 0, _LEDG == 0</code> after finishing on background). • Background <code>activity</code> overrides setting of <code>_LEDG</code> (the on-board LED pin) made by the application. Thus, attention should be taken when using direct setup of LED output pin combined with LED functions. Possible background LED routines can overwrite the status set by direct manipulation with the pin.
See also	<code>pulsingLEDG</code> , <code>pulseLEDG</code>
Example1	<pre>pulsingLEDG(); // Start blinking on OS background ... // Blinking continues during any operation stopLEDG(); // Stop blinking</pre>
Example2	<pre>pulseLEDG(); // Green LED On on OS background ... // continuously lighting during any operation // until specified time expired stopLEDG(); // or LED is switched Off by this command</pre>

EEPROM

eeReadByte

Function	Read one byte from specified location in EEPROM
Purpose	Access to EEPROM
Syntax	<code>uns8 eeReadByte (addr)</code>
Parameters	<code>uns8 addr</code> : address in EEPROM (0 to 0xBF). See EEPROM map [2].
Return value	Value read from specified EEPROM location (0 to 255)
Output values	–
Preconditions	–
Remarks	<ul style="list-style-type: none"> • Direct user access to EEPROM (using registers <code>EECONx</code> etc.) is not allowed for security reasons, specialized OS functions are intended for this. • EEPROM area dedicated to OS (locations 0xC0 or higher) is not accessible. See example E04–EEPROM [10].
Side effects	Any attempt to read from protected area above 0xBF leads to reading from EEPROM location 0xA0.
See also	<code>eeReadData</code> , <code>eeWriteByte</code> , <code>eeWriteData</code>
Example1	<pre>i = eeReadByte(0); // store 1 byte from EEPROM from address 0 to i</pre>
Example2	<pre>// Illegal access: Avoid access to EEPROM locations 192 (0xC0) or higher i = eeReadByte(200); // Reading from protected area is redirected to 160 (0xA0)</pre>

eeReadData

Function	Read a block of specified length from specified location in EEPROM to <code>bufferINFO</code>
Purpose	Block access to EEPROM
Syntax	<code>void eeReadData (addr, length)</code>
Parameters	<ul style="list-style-type: none"> • <code>uns8 addr</code>: address in EEPROM (0 to 0xBF - length + 1). See EEPROM map [2]. • <code>uns8 length</code>: number of bytes to be read (1 to 32)
Return value	–
Output values	<code>bufferINFO[0 to length - 1]</code>
Preconditions	–
Remarks	<ul style="list-style-type: none"> • Direct user access to EEPROM (using registers <code>EECONx</code> etc.) is not allowed for security reasons, specialized OS functions are intended for this. • EEPROM area dedicated to OS (locations 0xC0 or higher) is not accessible. See example E04–EEPROM [10].
Side effects	• Any attempt to read from protected area above 0xBF leads to reading from EEPROM location 0xA0.
See also	<code>eeReadByte</code> , <code>eeWriteByte</code> , <code>eeWriteData</code>
Example1	<pre>eeReadData(10, 16); // copy 16B from EEPROM from address 10 to bufferINFO // bufferINFO[0] = EEPROM[10] // ... // bufferINFO[15] = EEPROM[25]</pre>
Example2	<pre>// Illegal access: Avoid access to EEPROM locations 192 (0xC0) or higher eeReadData(200, 16); // EEPROM address 160 is used instead of protected area // bufferINFO[0] = EEPROM[160] // ... // bufferINFO[15] = EEPROM[160]</pre>

eeWriteByte

Function	Write one byte to specified location in EEPROM
Purpose	Access to EEPROM
Syntax	<code>void eeWriteByte (addr, data)</code>
Parameters	<ul style="list-style-type: none"> • <code>uns8 addr</code>: address in EEPROM (0 to 0xBF for Node, 0xA0 to 0xBF for Coordinator). See EEPROM map [2]. • <code>uns8 data</code>: value to be written (0 to 255)
Return value	–
Output values	–
Preconditions	–
Remarks	<ul style="list-style-type: none"> • Direct user access to EEPROM (using registers <code>EECONx</code> etc.) is not allowed for security reasons, specialized OS functions are intended for this. • EEPROM area dedicated to OS (locations 0xC0 or higher) is not accessible. See example E04–EEPROM [10].
Side effects	• Any attempt to write to protected area above 0xBF leads to writing to EEPROM location 0xA0.
See also	<code>eeReadByte</code> , <code>eeReadData</code> , <code>eeWriteData</code>
Example1	<pre>eeWriteByte(191, 0x75) // store 0x75 to EEPROM to address 191 eeWriteByte(0x80, X) // copy X to EEPROM to address 0x80</pre>
Example2	<pre>// Illegal access: Avoid access to EEPROM locations 192 (0xC0) or higher eeWriteByte(198, 0x75); // Writing to protected area is redirected to 160 (0xA0) X = eeReadByte(160); // X == 0x75 will be read after illegal writing</pre>

eeWriteData

Function	Write a block of specified length from <code>bufferINFO</code> to specified location in EEPROM
Purpose	Block access to EEPROM
Syntax	<code>void eeWriteData (addr, length)</code>
Parameters	<ul style="list-style-type: none"> • <code>uns8 addr</code>: address in EEPROM . See EEPROM map [2]. <ul style="list-style-type: none"> • (0 to 0xBF - length + 1) for Node • (0xA0 to 0xBF - length + 1) for Coordinator • <code>uns8 length</code>: number of bytes to be written from <code>bufferINFO</code> (1 to 32)
Return value	–
Output values	–
Preconditions	–
Remarks	<ul style="list-style-type: none"> • Direct user access to EEPROM (using registers <code>EECONx</code> etc.) is not allowed for security reasons, specialized OS functions are intended for this. • EEPROM area dedicated to OS (locations 0xC0 or higher) is not accessible. See example E04–EEPROM [10].
Side effects	• Any attempt to write to protected area above 0xBF leads to writing to EEPROM location 0xA0.
See also	<code>eeReadByte</code> , <code>eeReadData</code> , <code>eeWriteByte</code>
Example1	<pre>eeWriteData(10,16); // copy 16B from bufferINFO to EEPROM to address 10 // EEPROM[10] = bufferINFO[0] // ... // EEPROM[25] = bufferINFO[15]</pre>
Example2	<pre>// Illegal access: Avoid access to EEPROM locations 192 (0xC0) or higher eeWriteData(200,16); // EEPROM address 160 is used instead of protected area // EEPROM[160] = bufferINFO[0] // ... // EEPROM[160] = bufferINFO[15]</pre>

RAM
readFromRAM

Function	Read one byte from specified location in RAM
Purpose	Indirect access to RAM registers
Syntax	<code>uns8 readFromRAM(addr)</code>
Parameters	<code>uns8 addr</code> : memory location address. "Short" address within respective bank pair (up to 0xFF) should be used (e.g. 0xC5 for both 0xC5 and 0x1C5). <i>But using long addresses (up to 0x1FF, e.g. 0x1C5) also works because the compiler trims the overlapping bit to correct 8b value.</i> See Preconditions as well.
Return value	Value read from specified location
Output values	–
Preconditions	Before using this function, the IRP bank select bit must correspond to the location of the register (IRP=0 for RAM bank 0 and 1, IRP=1 for RAM bank 2 and 3). See the PIC datasheet [8] (Memory organization) and RAM map [2] for details.
Remarks	RAM can be accessed either directly (using common C commands like <code>X = Y;</code>) or indirectly. But indirect access using the FSR register is not allowed. Due to security reasons all instructions using FSR are removed during Upload. To avoid unintended behavior all constructions using addressing via FSR (either by the user or by the compiler) should be omitted. Instead of this IQRF OS provides complete support for indirect (byte oriented) addressing using extra system functions <code>readFromRAM</code> and <code>writeToRAM</code> . See example E06–RAM [10].
Side effects	–
See also	<code>writeToRAM</code>
Example1	<pre>IRP = 0; X = readFromRAM(bufferCOM + 10); IRP = 1; Y = readFromRAM(bufferRF + 10);</pre>
Example2	<pre>// Not correct: IRP = 0; // To access 0x190 IRP must be set to 1 X = readFromRAM(0x190) - 1; // 0x90 is addressed instead of expected 0x190</pre>
Example3	<pre>// Correct: IRP = 1; X = readFromRAM(0x90) - 1; // Perfect X = readFromRAM(0x190) - 1; // Also works thanks to the compiler feature</pre>
Example4	<pre>// Not allowed. The compiler uses FSR in such cases. for (i=0; i<5; i++) { A = bufferRF[i]; ... }</pre>
Example5	<pre>// Correct for (i=0; i<5; i++) { A = readFromRAM(bufferRF + i); ... }</pre>

writeToRAM

Function	Write one byte to specified location in RAM
Purpose	Indirect access to RAM registers
Syntax	<code>void writeToRAM(addr, value)</code>
Parameters	<ul style="list-style-type: none"> • <code>uns8 addr</code>: memory location address. Unlike <code>copyMemoryBlock</code>, "short" address within respective bank pair (up to 0xFF) should be used (e.g. 0xC5 for both 0xC5 and 0x1C5). But using long addresses (up to 0x1FF, e.g. 0x1C5) also works because the compiler trims the overlapping bit to correct 8b value. See Preconditions as well. • <code>uns8 value</code>: value to be written
Return value	–
Output values	–
Preconditions	<ul style="list-style-type: none"> • Before using this function, the IRP bank select bit must correspond to the location of the register (IRP=0 for RAM bank 0 and 1, IRP=1 for RAM bank 2 and 3). See the PIC datasheet [8] (Memory organization) and RAM map [2] for details. • Avoid writing to RAM areas dedicated to OS and to PIC special function registers otherwise OS can collapse. See RAM map [2]. • Some locations are restricted from writing. In doubt, refer to IQRF support by the manufacturer [6].
Remarks	RAM can be accessed either directly (using common C commands like <code>X = Y;</code>) or indirectly. But indirect access using the FSR register is not allowed. Due to security reasons all instructions using FSR are removed during Upload. To avoid unintended behavior all constructions using addressing via FSR (either by the user or by the compiler) should be omitted. Instead of this IQRF OS provides complete support for indirect (byte oriented) addressing using extra system functions <code>readFromRAM</code> and <code>writeToRAM</code> . See example E06–RAM [10]. Certain RAM locations are not accessible at all for security reasons.
Side effects	–
See also	<code>readFromRAM</code> , <code>copyMemoryBlock</code>
Example1	<pre>IRP = 0; writeToRAM(bufferCOM + 10, 2 * X); IRP = 1; writeToRAM(bufferRF + 10, 2 * X);</pre>
Example2	<pre>// Not correct: IRP = 0; // To access 0x190 IRP must be set to 1 writeToRAM(0x190, 201); // 0x90 is addressed instead of expected 0x190</pre>
Example3	<pre>// Correct: IRP = 1; writeToRAM(0x90, 201); // Perfect writeToRAM(0x190, 201); // Also works thanks to the compiler feature</pre>
Example4	<pre>// Not allowed. The compiler uses FSR in such cases. for (i=0; i<5; i++) bufferRF[i] = i;</pre>
Example5	<pre>// Correct for (i=0; i<5; i++) writeToRAM(bufferRF + i, i);</pre>

setPIR1

Function	Setup microcontroller flags in PIR1
Purpose	Access to control flags for PIC internal peripherals (I ² C, A/D, ...).
Syntax	<code>void setPIR1 (value)</code>
Parameters	<code>uns8 value</code> : value to be written
Return value	–
Output values	–
Preconditions	–
Remarks	PIR1 belongs to protected registers not user accessible in standard ways for security reasons. Thus, to allow access to some PIC internal peripherals, the specialized function is available to setup the appropriate flags in the PIR1 control register. See PIC datasheets [8].
Side effects	–
See also	–
Example	<code>setPIR1(0) // ADIF = 0; SSPIF = 0;</code>

Obsolete

Buffers, data blocks
clearBufferINFO

Function	Clear <code>bufferINFO</code>
Purpose	<code>bufferINFO</code> clearing
Syntax	<code>void clearBufferINFO ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	Complete <code>bufferINFO</code> (35 B) is cleared (filled with zeros). See example E06 - RAM [10].
Side effects	–
See also	<code>copyBufferINFO2COM</code> , <code>copyBufferINFO2RF</code> , <code>copyBufferRF2INFO</code> , <code>copyBufferCOM2INFO</code> , <code>compareBufferINFO2RF</code> , <code>copyMemoryBlock</code>
Example	<code>clearBufferINFO ();</code>

copyBufferINFO2COM

Function	Copy <code>bufferINFO</code> to <code>bufferCOM</code>
Purpose	Data transfer between buffers
Syntax	<code>void copyBufferINFO2COM ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	35 B is copied. See example E06 - RAM [10].
Side effects	–
See also	<code>clearBufferINFO</code> , <code>copyBufferINFO2RF</code> , <code>copyBufferRF2COM</code> , <code>copyBufferRF2INFO</code> , <code>copyBufferCOM2RF</code> , <code>copyBufferCOM2INFO</code> , <code>compareBufferINFO2RF</code> , <code>copyMemoryBlock</code>
Example	<code>copyBufferINFO2COM ();</code>

copyBufferINFO2RF

Function	Copy <code>bufferINFO</code> to <code>bufferRF</code>
Purpose	Data transfer between buffers
Syntax	<code>void copyBufferINFO2RF ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	35 B is copied. See example E06 - RAM [10].
Side effects	–
See also	<code>clearBufferINFO</code> , <code>copyBufferINFO2COM</code> , <code>copyBufferRF2COM</code> , <code>copyBufferRF2INFO</code> , <code>copyBufferCOM2RF</code> , <code>copyBufferCOM2INFO</code> , <code>compareBufferINFO2RF</code> , <code>copyMemoryBlock</code>
Example	<code>copyBufferINFO2RF () ;</code>

copyBufferRF2COM

Function	Copy <code>bufferRF</code> to <code>bufferCOM</code>
Purpose	Data transfer between buffers
Syntax	<code>void copyBufferRF2COM ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	35 B is copied. See example E06 - RAM [10].
Side effects	–
See also	<code>clearBufferINFO</code> , <code>copyBufferINFO2RF</code> , <code>copyBufferINFO2COM</code> , <code>copyBufferRF2INFO</code> , <code>copyBufferCOM2RF</code> , <code>copyBufferCOM2INFO</code> , <code>compareBufferINFO2RF</code> , <code>copyMemoryBlock</code>
Example	<code>copyBufferRF2COM () ;</code>

copyBufferRF2INFO

Function	Copy <code>bufferRF</code> to <code>bufferINFO</code>
Purpose	Data transfer between buffers
Syntax	<code>void copyBufferRF2INFO ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	35 B is copied. See example E06 - RAM [10].
Side effects	–
See also	<code>clearBufferINFO</code> , <code>copyBufferINFO2COM</code> , <code>copyBufferINFO2RF</code> , <code>copyBufferRF2COM</code> , <code>copyBufferCOM2RF</code> , <code>copyBufferCOM2INFO</code> , <code>compareBufferINFO2RF</code> , <code>copyMemoryBlock</code>
Example	<code>copyBufferRF2INFO () ;</code>

copyBufferCOM2RF

Function	Copy <code>bufferCOM</code> to <code>bufferRF</code>
Purpose	Data transfer between buffers
Syntax	<code>void copyBufferCOM2RF ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	35 B is copied. See example E06 - RAM [10].
Side effects	–
See also	<code>clearBufferINFO</code> , <code>copyBufferINFO2COM</code> , <code>copyBufferINFO2RF</code> , <code>copyBufferRF2COM</code> , <code>copyBufferRF2INFO</code> , <code>copyBufferCOM2INFO</code> , <code>compareBufferINFO2RF</code> , <code>copyMemoryBlock</code>
Example	<code>copyBufferCOM2RF ();</code>

copyBufferCOM2INFO

Function	Copy <code>bufferCOM</code> to <code>bufferINFO</code>
Purpose	Data transfer between buffers
Syntax	<code>void copyBufferCOM2INFO ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	35 B is copied. See example E06 - RAM [10].
Side effects	–
See also	<code>clearBufferINFO</code> , <code>copyBufferINFO2COM</code> , <code>copyBufferINFO2RF</code> , <code>copyBufferRF2COM</code> , <code>copyBufferRF2INFO</code> , <code>copyBufferCOM2RF</code> , <code>copyMemoryBlock</code>
Example	<code>copyBufferCOM2INFO ();</code>

compareBufferINFO2RF

Function	Compare <code>bufferINFO</code> and <code>bufferRF</code> with respect to specified length
Purpose	Buffer comparison
Syntax	<code>bit compareBufferINFO2RF (length)</code>
Parameters	<code>uns8 length</code> : number of bytes to be compared (1 to 35)
Return value	<ul style="list-style-type: none"> • 1 – match • 0 – mismatch
Output values	–
Preconditions	–
Remarks	See example E06 - RAM [10].
Side effects	–
See also	<code>clearBufferINFO</code> , <code>copyBufferINFO2RF</code> , <code>copyBufferRF2INFO</code>
Example	<pre>if (!compareBufferINFO2RF(32)) // Compare 32 B then Error = 1; // Error if mismatch</pre>

copyMemoryBlock

Function	Copy specified RAM block to specified location
Purpose	Copy memory block within RAM
Syntax	<code>void copyMemoryBlock (from, to, length)</code>
Parameters	<ul style="list-style-type: none"> • <code>uns16 from</code>: starting address of the block to be copied • <code>uns16 to</code>: destination address • <code>uns8 length</code>: block length in bytes
Return value	–
Output values	–
Preconditions	<ul style="list-style-type: none"> • Copying is allowed also between different memory banks but both blocks are not allowed to cross bank boundaries (0x7F-0x80, 0xFF-0x100, 0x17F-0x180). • Unlike <code>writeToRAM</code>, "long" addresses (up to 0x1FF) are used. IRP is no object. • Avoid writing to RAM areas dedicated to OS and to PIC special function registers otherwise OS can collapse. See RAM map [2]. Some locations are restricted from writing due to security reasons..
Remarks	This function can access RAM within the whole memory area. See RAM map [2] and example E06 - RAM [10].
Side effects	–
See also	<code>writeToRAM</code>
Example	<pre>copyMemoryBlock(0x15D, 0x1DD, 4); // copy 4B block from 0x15D to 0x1DD copyMemoryBlock(bufferRF+10, bufferCOM+1, 8); // 8 bytes copied: // bufferCOM[1] = bufferRF[10] ... bufferCOM[8] = bufferRF[17]</pre>

moduleInfo

Function	Store Module data to <code>bufferINFO</code>								
Purpose	Get information about transceiver module and OS								
Syntax	<code>void moduleInfo ()</code>								
Parameters	-								
Return value	-								
Output values	<code>bufferINFO[0 to 7]</code>								
Preconditions	-								
Remarks	address in <code>bufferInfo</code>	7	6	5	4	3	2	1	0
	meaning	OS build		PIC type	OS version	Coordinator / Node	serial number		
	Module ID								
	See IQRF OS User's guide [1] (Identification) for Module data format details.								
Side effects	-								
See also	<code>appInfo</code>								
Example	<pre> uns24 SN @ bufferInfo; uns8 OSv @ bufferInfo[4]; moduleInfo (); // Now SN == module serial number // and OSv == OS version </pre>								

appInfo

Function	Store Application data from EEPROM to <code>bufferINFO</code>							
Purpose	Get information about user application							
Syntax	<code>void appInfo ()</code>							
Parameters	-							
Return value	-							
Output values	<code>bufferINFO[0 to 31]</code>							
Preconditions	-							
Remarks	See IQRF OS User's guide [1] (Identification and Appendix, Memory maps).							
Side effects	-							
See also	<code>moduleInfo</code>							
Example1	<pre> appInfo (); // Copy Application data from EEPROM to bufferINFO copyBufferINFO2RF (); // and then to bufferRF </pre>							
Example2	<pre> #pragma packedCdataStrings 0 // Application data to EEPROM after compilation #pragma cdata[__EEAPPINFO] = "Application data, I'm user #01 " bufferINFO[0] = '2'; // Dynamic change of application data eeWriteData(__EEAPPINFO+29,1); // #01 changed to #02 appInfo (); // "Application data, I'm user #02 " is read </pre>							

SPI
enableSPI

Function	Activate SPI communication module and related pins
Purpose	Enable SPI communication
Syntax	<code>void enableSPI ()</code>
Parameters	–
Return value	–
Output values	SPI Status is switched to <i>SPI ready, communication mode</i> .
Preconditions	–
Remarks	<ul style="list-style-type: none"> • The PIC internal SPI hardware module and appropriate pins (C5 – C8) are configured and activated as SPI Slave. • See SPI Implementation in IQRF TR modules [5] and example E07-SPI [10].
Side effects	Related pins can not be used as general I/Os until SPI is disabled via <code>disableSPI</code> .
See also	<code>disableSPI</code> , <code>startSPI</code> , <code>stopSPI</code> , <code>getStatusSPI</code>
Example	See <code>getStatusSPI</code>

disableSPI

Function	Switch SPI HW module off and configure SPI pins as I/Os
Purpose	Disable SPI communication
Syntax	<code>void disableSPI ()</code>
Parameters	–
Return value	–
Output values	SPI Status is switched to <i>SPI not active</i> .
Preconditions	–
Remarks	The PIC internal SPI hardware module is disabled and related pins (C5 – C8) are reconfigured as general I/Os. See SPI Implementation in IQRF TR modules [5] and example E07-SPI [10].
Side effects	<ul style="list-style-type: none"> • The appropriate PIC pins are not restored to the state before <code>enableSPI</code> calling. • Current packet is lost by both sides if SPI communication is running on background at this moment.
See also	<code>enableSPI</code> , <code>startSPI</code> , <code>stopSPI</code> , <code>getStatusSPI</code>
Example	See <code>getStatusSPI</code>

startSPI

Function	Indicate ready to Master.
Purpose	<ul style="list-style-type: none"> Initiate SPI packet transmission from Slave (request to Master). Provide data from <code>bufferCOM</code> to Master according to Master's clock (on OS background). <code>startSPI(0)</code> indicates to Master that the Slave is ready to receive data (<code>bufferCOM</code> not full).
Syntax	<code>void startSPI (length)</code>
Parameters	<code>uns8 length</code> : number of bytes to be sent (0 to 41)
Return value	–
Output values	SPI Status is switched to: <ul style="list-style-type: none"> <i>SPI data ready</i> – after <code>startSPI(1 to 41)</code> <i>SPI ready, Communication mode</i> – after <code>startSPI(0)</code>.
Preconditions	SPI must be enabled by the <code>enableSPI</code> function before.
Remarks	<ul style="list-style-type: none"> SPI runs on OS background. <code>startSPI(0)</code> is also useful for recovering SPI from communication failures (e.g. the CRC mismatch). See SPI Implementation in IQRF TR modules [5] and example E07-SPI [10].
Side effects	–
See also	<code>enableSPI</code> , <code>disableSPI</code> , <code>stopSPI</code> , <code>getStatusSPI</code>
Example1	<pre> // Slave -> Master bufferCOM[0] = "I"; bufferCOM[1] = "Q"; enableSPI(); startSPI(2); // Request to Master is active on background from now ... // and the program just continues here </pre>
Example2	See <code>getStatusSPI</code>

stopSPI

Function	Stop SPI communication
Purpose	Suspend SPI transmissions whenever it suits to Slave
Syntax	<code>void stopSPI ()</code>
Parameters	–
Return value	–
Output values	SPI Status is switched to <i>User stop</i> .
Preconditions	–
Remarks	<ul style="list-style-type: none"> • <code>stopSPI</code> is useful e.g. to avoid violation during preparation data to <code>bufferCOM</code>. • SPI transmission is stopped but SPI remains active (enabled). Communication can continue after next <code>startSPI</code>. • <code>stopSPI</code> is not needed after successful SPI reception to protect data received in <code>bufferCOM</code>. Data is protected by OS (and SPI status stays in mode 3F) until the slave allows further communication e.g. by the <code>startSPI (0)</code>. • <code>startSPI</code> and <code>stopSPI</code> are not fully complementary. Receiving is allowed just after <code>enableSPI</code> without previous <code>startSPI</code>, <code>startSPI</code> is meaningful after previous <code>startSPI</code> not followed by <code>stopSPI</code> etc. • See SPI Implementation in the IQRF TR modules [5] and example E07-SPI [10].
Side effects	Current packet is lost by both sides if SPI communication is running on background at this moment.
See also	<code>enableSPI</code> , <code>disableSPI</code> , <code>startSPI</code> , <code>getStatusSPI</code>
Example	<pre> if (!getStatusSPI()) // If SPI is not in progress { stopSPI(); // Prohibit Master from transmitting bufferCOM[0] = ... // Prepare data to send bufferCOM[1] = ... startSPI(2); // Request to send } </pre>

getStatusSPI

Function	Update SPI flags and packet length and check whether SPI is busy
Purpose	Provide application program with information about current SPI status
Syntax	bit <code>getStatusSPI ()</code>
Parameters	–
Return value	<ul style="list-style-type: none"> • 1 – SPI busy • 0 – SPI not busy
Output values	param1: received packet length param2.3 (<code>_SPIRX</code>): 1 – Something received on SPI. param2.4 (<code>_SPICRCok</code>): 1 – The last received SPI CRCM was O.K.
Preconditions	SPI must be enabled by <code>enableSPI</code>
Remarks	<ul style="list-style-type: none"> • Output values (<code>param2</code>) has different format than SPI status sent to the Master. • See SPI Implementation in IQRF TR modules [5] and example E07-SPI [10].
Side effects	–
See also	<code>enableSPI</code> , <code>disableSPI</code> , <code>startSPI</code> , <code>stopSPI</code>
Example1	<pre> // Master -> Slave enableSPI(); // Master is allowed to transmit from now Receive: clrwdt(); if (getStatusSPI()) // Wait until SPI is not busy goto Receive; if (_SPIRX) // Anything received? { // Yes: if (!_SPICRCok) // CRCM matched? { // No: startSPI(0); // Restart SPI goto Receive; // and try to receive again. } // Yes: // BufferCOM is automatically protected now // not to be overwritten by next SPI packet. // Thus, stopSPI is not necessary here. PacketLength = param1; // Store packet length to user variable. copyBufferCOM2INFO(); // Store received packet startSPI(0); // and then allow Master to transmit again. } else goto Receive; // Nothing received yet // ... Continue here after successful receiving waitMS(1); // Time for finishing startSPI(0) on background disableSPI(); // otherwise Master's CRCS check fails. // The delay depends on Master application. </pre>
Example2	<pre> enableSPI(); startSPI(2); // 2 B to send to master while (getStatusSPI()) // Wait until SPI is not busy waitMS(1); ... // Now the transfer is finished </pre>

RF
setTXpower

Function	Set RF output power
Purpose	Change RF range
Syntax	<code>void setTXpower (level)</code>
Parameters	uns8 level: 0 (min.) to 7 (max. – default) See datasheet of TR module, Table 2.
Return value	–
Output values	–
Preconditions	–
Remarks	–
Side effects	–
See also	RFTXpacket
Example	<code>setTXpower (7); // Max. RF output power</code>

setRFspeed

Function	Select RF bit rate
Purpose	Select RF bit rate
Syntax	<code>void setRFspeed (speed)</code>
Parameters	uns8 speed: <ul style="list-style-type: none"> • 1 1.2 kb/s (preliminary) • 2 19.2 kb/s (default) • 3 57.6 kb/s (preliminary) • 4 86.2 kb/s (preliminary)
Return value	–
Output values	–
Preconditions	–
Remarks	<ul style="list-style-type: none"> • Non-default bit rates are provisionally intended for experimental purposes only. • Routing is supported for 19.2 kb/s only
Side effects	RF channel must be specified after every bit rate change.
See also	setRFchannel
Example1	<pre>setRFspeed(1); // 1.2 kb/s selected setRFchannel(...); // channel must be selected then</pre>
Example2	<pre>setRFspeed(2); // 19.2 kb/s selected setRFchannel(...); // channel must be selected then</pre>

setRFband

Function	Select RF frequency band
Purpose	Select 868 MHz or 916 MHz band
Syntax	<code>void setRFband (band)</code>
Parameters	uns8 band: • 0 868 band MHz (default) • 1 916 band MHz
Return value	–
Output values	–
Preconditions	–
Remarks	Default channel is set (52 for 868 MHz band or 104 for 916 MHz band).
Side effects	–
See also	setRFchannel
Example1	<code>setRFband(1); // 916 MHz band selected</code>
Example2	<code>setRFband(0); // 868 MHz band selected</code>

setRFchannel

Function	Set RF channel
Purpose	Select free RF channel for not interfered communication
Syntax	<code>void setRFchannel (channel)</code>
Parameters	uns8 channel: see IQRF OS User's guide, Appendix 2, Channel map
Return value	–
Output values	–
Preconditions	RF channel must be specified after every bit rate change.
Remarks	–
Side effects	–
See also	setRFspeed
Example	<code>setRFband(0); // 868 MHz band selected</code> <code>setRFspeed(3); // 57.6 kb/s bit rate selected</code> <code>setRFchannel(25); // 868.15 MHz channel selected</code>

setRFmode

Function	Set RF mode
Purpose	Specify power management and signal filtering modes for RF transmission and receipt
Syntax	<code>void setRFmode (mode)</code>
Parameters	<p>uns8 mode: 00STFFRR in binary</p> <ul style="list-style-type: none"> • RR: 00 STD RX mode 01 LP RX mode 10 XLP RX mode 11 SSF RX (signal strength filtering) mode • FF: Filter incoming signal in LP, XLP and SSF RX modes (RR ≠ 0). Signal with lower level is ignored. Relative RF range is shortened due to this filtration: <ul style="list-style-type: none"> 00 relative range 45 % 01 relative range 30 % 10 relative range 15 % 11 relative range 10 % • T: 0 send packets with standard preambles 1 send packets with preambles prolonged for ~3 ms (for LP receiving) • S: 0 send packets once 1 send packets twice (for XLP receiving)
Return value	–
Output values	–
Preconditions	Non-STD RX modes are intended for bit rate 19.2 kb/s only.
Remarks	Default value is mode = 0.
Side effects	RF circuitry and MCU is temporarily set to sleep during low power RX modes. Thus, all tasks running on OS background can be untimely canceled. To avoid this, use <code>setRFmode</code> after finishing background tasks. See Example 2.
See also	<code>checkRF</code>
Example1	<pre>setRFmode(0b00010001); // RX: LP, lowest filtering (0) // TX: prolonged preambles, packets sent once setRFmode(0b00111110); // RX: XLP, highest filtering (3) // TX: prolonged preambles, packets sent twice setRFmode(0b00000101); // RX: LP, low filtering (1) // TX: standard preambles, packets sent once setRFmode(0b00000000); // RX: STD, no filtering // TX: standard preambles, packets sent once setRFmode(0b00001011); // RX: SSF, high filtering (2) // TX: standard preambles, packets sent once</pre>
Example2	<pre>while (getStatusSPI()) // wait for finishing SPI on background clrwdt(); disableSPI(); SWDTEN = 0; // possibly disable watchdog for lower consumption setRFmode(0b00010001); // and go to LP mode then</pre>

checkRF

Function	Check incoming RF signal strength for specified level.
Purpose	Incoming RF signal detection to start RF receiving.
Syntax	<code>bit checkRF(level)</code>
Parameters	<p>uns8 level: 0 to 64</p> <p>Higher level requires stronger signal. Relative RF range is shortened due to this filtration according the datasheet of the TR module, Table 3.</p>
Return value	<ul style="list-style-type: none"> • 0: Signal with specified level or higher not detected • 1: Signal with specified level or higher detected
Output values	Signal strength is also available as a relative value in the ADRESH (one of PIC SFR registers). Higher value means stronger signal.
Preconditions	This function is not intended for LP RX mode.
Remarks	<ul style="list-style-type: none"> • Higher level means lower sensitivity which requires stronger signal resulting in higher immunity against interferences. • Checking takes ~1 ms and consumes ~9.5 mA. • This function is intended for fast response and power consumption reduction in STD RX mode. • Received packets should have prolonged preambles.
Side effects	–
See also	setRFmode
Example1	<pre> // Fast response receiving in STD mode setRFmode(0b00010000); // RX: STD, no filtering // TX: prolonged preambles, packets sent once if (checkRF(5)) // Detect signal with 84 % { if (RFRXpacket()) // Duration according to toutRF only if packet is sent. { // toutRF can be optimized for expected packet length. ... // Transmitter should send packets with prolonged // preambles. } } // Otherwise only ~1 ms is spent. ... time-critical section can be placed here </pre>
Example2	<pre> if (checkRF(10)) // Detect signal with 64 % ... </pre>
Example3	<pre> // RF signal strength analyzer SWDTEN = 0; // disable watchdog while (1) if (checkRF(3)) pulseLEDR(); // LED flash if signal level >= 3 detected </pre>

RFTXpacket

Function	Send RF packet of specified length from <code>bufferRF</code> .
Purpose	RF transmission
Syntax	<code>void RFTXpacket ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	<ul style="list-style-type: none"> • Peer-to-peer topology: <ul style="list-style-type: none"> • PIN = 0 (Peer-to-peer) • DLEN = packet length in bytes (0 to 64) • Prepare data to send in <code>bufferRF[0]</code> to <code>bufferRF[DLEN - 1]</code> (if <code>DLEN ≠ 0</code>) • Set RF output power via <code>setTXpower</code> • IQMESH: <ul style="list-style-type: none"> • PIN = 0x80 (IQMESH) • Other network related parameters should also be specified See IQRF OS User's guide [1] and IQMESH specification [4].
Remarks	<ul style="list-style-type: none"> • Unlike SPI, RF communication does not run on OS background. This function is active on foreground until the packet is sent. • Duration depends on TR type, routing algorithm and packet length. • To enable comfortable acknowledge, routing vector used by Coordinator is delivered to Node in reversed order ready for immediate usage. To take advantage of this the communication should be initiated by Coordinator. • See examples E01–TX, E03–TR, E09–LINK [10] and E11–DATACENTER [10].
Side effects	<ul style="list-style-type: none"> • <code>bufferRF[DLEN]</code> and <code>bufferRF[DLEN+1]</code> are destroyed • The RF circuitry wakes up (in case of sleeping).
See also	<code>RFRXpacket</code> , <code>setTXpower</code> , <code>setRFmode</code> and (in case of IQMESH) also other RF functions

Example1	<pre> // Peer-to-peer topology PIN=0; // Peer-to-peer (update also after every RFRXpacket // before every RFTXpacket) setNetworkingOff(); bufferRF[0] = "I"; // Data to send bufferRF[1] = "Q"; DLEN = 2; // 2 B packet RFTXpacket(); // Send the packet to all Peer-to-peer Nodes in range // and to all IQMESH Nodes having set filtering off // Program stays here until the packet is sent ... // and then continues </pre>
Example2	<pre> // IQMESH without routing, packet from Coordinator to Node #10 PIN = 0; // PIN preclearing (update also after every RFRXpacket // before every RFTXpacket) setNetworkOne(); // Select Network 1. The NTWF flag (PIN.7) is set here. // setCoordinatorMode(); Select Coordinator mode - not necessary // in this OS versions (already done by setNetworkOne) bufferRF[0] = "I"; // Data to send bufferRF[1] = "Q"; DLEN = 2; // 2 B packet RX = 10; // Packet for Node #10 // _ROUTEF = 0; // Routing disabled - not necessary (default by OS) RFTXpacket(); // Send the packet to IQMESH Node #10 in this network // Reception depends on the Node (its current network // or filtering) </pre>
Example3	<pre> // IQMESH with routing // Packet from Coordinator to Node #10 via Nodes 1,2,3 and 4 PIN = 0; // PIN preclearing (update also after every RFRXpacket // before every RFTXpacket) setNetworkOne(); // Select Network 1. The NTWF flag (PIN.7) is set here. // setCoordinatorMode(); Select Coordinator mode - not necessary // in this OS versions (already done by setNetworkOne) bufferRF[0] = "I"; // Data to send bufferRF[1] = "Q"; DLEN = 2; // 2 B packet RX = 10; // Packet for Node #10 _ROUTEF = 1; // Routing enabled for outgoing packets // RTDEF = 0; // Standard routing, 4+1 hops. Reserved // for future OS versions (various routing types) RTV[0] = 1; // Routing vector. This will not be necessary to RTV[1] = 2; // specify in future OS versions. RTV[2] = 3; RTV[3] = 4; RFTXpacket(); // Send the packet to IQMESH Node #10 in this network // Reception depends on the Node (its current network // or filtering) // Routing vector is reversed (4,3,2,1) by OS for Node </pre>

RFRXpacket

Function	Receive RF packet to <code>bufferRF</code> and provide related information
Purpose	RF receiving
Syntax	bit <code>RFRXpacket()</code>
Parameters	–
Return value	<ul style="list-style-type: none"> • 1 – packet received • 0 – packet not received
Output values	<p><code>DLEN</code> = packet length. This variable is destroyed if the receipt is not successful.</p> <p><code>PIN</code> is updated according to packet received. This variable is destroyed if the receipt is not successful.</p> <p><code>_NTWPACKET</code>: valid if <code>RFRXpacket</code> return value == 1 only:</p> <ul style="list-style-type: none"> • 1 – networking packet received • 0 – non-networking packet received <p>Other related networking information in case of IQMESH.</p>
Preconditions	<ul style="list-style-type: none"> • Timeout in number of 10 ms ticks should be specified in <code>toutRF</code> (1 to 255) • Peer-to-peer topology: nothing else • IQMESH: network related parameters (filtering, ...) should be predefined <p>See IQRF OS User's guide [1] and IQMESH specification [4].</p>
Remarks	<ul style="list-style-type: none"> • Unlike SPI, RF communication does not run on OS background. This function is active on foreground until the packet is received or timeout expired. Timeout during packet receiving terminates the reception. • If the packet is sent when the address (or a routing device) is not executing this function the packet is lost. • Peer-to-peer topology: All packets in range are received. • IQMESH: Device receives only packets intended for it and non-networking packets depending on filtering mode – see <code>setFilteringOn</code> and <code>setFilteringOff</code>. • See examples E02–RX, E03–TR, E09–LINK and E11–MEASUREMENT [10].
Side effects	<ul style="list-style-type: none"> • Update <code>PIN</code> before every <code>RFTXpacket</code> folowed after <code>RFRXpacket</code>. • Result of <code>captureTicks</code> is destroyed if <code>startCapture</code> is active on background at the same time. • <code>bufferRF[DLEN]</code> and <code>bufferRF[DLEN+1]</code> is destroyed. • The RF circuitry wakes up (in case of sleeping).
See also	<code>RFTXpacket</code> , <code>setRFmode</code> , <code>checkRF</code> and (in case of IQMESH) also other RF functions
Example1	<pre> // Peer-to-peer topology toutRF = 10; // RF timeout 100 ms if RFRXpacket(); // Try to receive RF packet. Packet received? // Program stays here until the packet is received // or the timeout is expired. { // Yes: copyBufferRF2INFO(); // Store received data PacketLength = DLEN; // and possibly other info (packet length, ...) } else { // No: ... // Timeout expired. Arrange respective operations. } </pre>
Example2	IQMESH: See <code>setNodeMode</code> and <code>setNetworkFilteringOn</code> .

Networking

setNetworkOne

Function	Select IQMESH networking, Network 1
Purpose	Switch (from IQMESH Network 2 or Peer-to-peer) to IQMESH Network 1
Syntax	<code>void setNetworkOne ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	For IQMESH only. Default OS mode is Peer-to-peer.
Remarks	<ul style="list-style-type: none"> • IQMESH allows every TR module to work in more independent networks. This OS version supports 2 networks and working as a Coordinator in Network 1 and as a Node in Network 2. • See example E11 - DATACENTER [10]. • This settings affects both <code>RFRXpacket</code> and <code>RFTXpacket</code> features. • PIN is not affected immediately but not until subsequent <code>RFRXpacket</code> or <code>RFTXpacket</code>.
Side effects	TR module is switched to Coordinator mode by this function in this OS version. Thus, <code>setNetworkOne</code> is equivalent to <code>setCoordinatorMode</code> in this OS version. This is not guaranteed for future OS versions.
See also	<code>setNetworkTwo</code> , <code>setNetworkingOff</code> , <code>setCoordinatorMode</code> , <code>setNodeMode</code>
Example	See <code>setNodeMode</code>

setNetworkTwo

Function	Select IQMESH networking, Network 2
Purpose	Switch (from IQMESH Network 1 or Peer-to-peer) to IQMESH Network 2
Syntax	<code>void setNetworkTwo ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	For IQMESH only. Default OS mode is Peer-to-peer.
Remarks	<ul style="list-style-type: none"> • IQMESH allows every TR module to work in more independent networks. This OS version supports 2 networks and working as a Coordinator in Network 1 and a Node in Network 2. • See example E11 - MEASUREMENT [10]. • This settings affects <code>RFRXpacket</code> and <code>RFTXpacket</code> features. • PIN is not affected immediately but not until subsequent <code>RFRXpacket</code> or <code>RFTXpacket</code>.
Side effects	TR module is switched to Node mode by this function in this OS version. Thus, <code>setNetworkTwo</code> is equivalent to <code>setNodeMode</code> in this OS version. This is not guaranteed for future OS versions.
See also	<code>setNetworkOne</code> , <code>setNetworkingOff</code> , <code>setCoordinatorMode</code> , <code>setNodeMode</code>
Example	See <code>setNodeMode</code>

setNetworkingOff

Function	Select Peer-to-peer mode
Purpose	Switch from IQMESH to Peer-to-peer
Syntax	void setNetworkingOff ()
Parameters	–
Return value	–
Output values	–
Preconditions	–
Remarks	<ul style="list-style-type: none"> • Default OS mode is Peer-to-peer. • This settings affects <code>RFRXpacket</code> and <code>RFTXpacket</code> features. • PIN is not affected immediately but it is cleared after subsequent <code>RFRXpacket</code> or <code>RFTXpacket</code>.
Side effects	–
See also	<code>setNetworkOne</code> , <code>setNetworkTwo</code> , <code>setCoordinatorMode</code> , <code>setNodeMode</code>
Example	<pre> setNetworkOne (); // IQMESH selected ... // TR is assigned as a Coordinator and communicates // in IQMESH networking mode here // setNetworkingOff (); // Switch to Peer-to-peer mode ... // Now TR communicates without networking support </pre>

setCoordinatorMode

Function	Set Coordinator mode
Purpose	Assign the TR module as a network Coordinator
Syntax	<code>void setCoordinatorMode ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	For IQMESH only
Remarks	Every TR module can work as a Coordinator or a Node. Just one Coordinator in single network is allowed. Avoid dynamic switching the Coordinator from device to device in a network. This settings affects both <code>RFRXpacket</code> and <code>RFTXpacket</code> .
Side effects	TR module is switched to IQMESH, Network 1 in this OS version. Thus, <code>setCoordinatorMode</code> is equivalent to <code>setNetworkOne</code> in this OS version. This is not guaranteed for future OS versions.
See also	<code>setNetworkOne</code> , <code>setNetworkTwo</code> , <code>setNodeMode</code>
Example	See <code>setNodeMode</code>

setNodeMode

Function	Set Node mode
Purpose	Assign the TR module as a Node
Syntax	<code>void setNodeMode ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	For IQMESH only
Remarks	Every TR module can work as a Coordinator or a Node. This settings affects both <code>RFRXpacket</code> and <code>RFTXpacket</code> .
Side effects	TR module is switched to IQMESH, Network 2 in this OS version. Thus, <code>setNodeMode</code> is equivalent to <code>setNetworkTwo</code> in this OS version. This is not guaranteed for future OS versions.
See also	<code>setNetworkOne</code> , <code>setNetworkTwo</code> , <code>setNodeMode</code>
Example	<pre> // Forwarding a packet from Network 2 to Network 1 setNetworkTwo(); // These 2 functions are equivalent in this OS version // setNodeMode(); // In either case the result is the same. if (RFRXpacket()) // Receive a packet in Network 2 { setNetworkOne(); // Then switch to Network 1 // setCoordinatorMode(); These 2 functions are equivalent in this OS version RX=5; RFTXpacket(); // and forward the packet to addressee 5 in Network 1 } </pre>

setNetworkFilteringOn

Function	Start filtering incoming non-networking packets and packets coming from non-current network(s).
Purpose	To receive packets from current network only.
Syntax	<code>void setNetworkFilteringOn ()</code>
Parameters	–
Return value	–
Output values	This affects the RFRXpacket return value.
Preconditions	For IQMESH only. Default OS condition is Filtering Off.
Remarks	–
Side effects	–
See also	setNetworkFilteringOff, RFRXpacket
Example	<pre> setNetworkFilteringOn(); // Start filtering incoming packets RFRXpacket(); // Return value == 1 if the packet came // from current network only. // Return value == 0 if // the packet came from non-current network(s) // or it is a non-networking packet // or no packet came in time at all. </pre>

setNetworkFilteringOff

Function	Stop filtering incoming packets from the point of view the packet is coming from.
Purpose	To receive all packets (non-networking packets as well as packets from all networks).
Syntax	<code>void setNetworkFilteringOff ()</code>
Parameters	–
Return value	–
Output values	This affects the RFRXpacket return value.
Preconditions	For IQMESH only. Default OS condition is Filtering Off.
Remarks	Network 1 or 2 is automatically selected according to last received packet in this mode (except of non-networking packets).
Side effects	–
See also	setNetworkFilteringOn, RFRXpacket
Example	<pre> setNetworkFilteringOff(); // Stop filtering incoming packets RFRXpacket(); // Return value == 1 if // the packet came from current network // or from non-current network(s) // or it is a non-networking packet // Return value == 0 if // no packet came in time at all </pre>

setLoggingOn

Function	Start logging communication with adjacent devices
Purpose	To keep track of devices in range
Syntax	<code>void setLoggingOn ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	For IQMESH only. Default OS condition is Logging off. Not implemented. Reserved for future OS versions (Discovery service etc.). Do not use in this OS version.
Remarks	–
Side effects	–
See also	setLoggingOff
Example	<pre>setLoggingOn(); // Start logging ... // All communication with adjacent devices is logged now</pre>

setLoggingOff

Function	Stop logging communication with adjacent devices
Purpose	Not to track of devices in range
Syntax	<code>void setLoggingOff ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	For IQMESH only. Default OS condition is Logging off. Not implemented. Reserved for future OS versions (Discovery service etc.). Do not use in this OS version.
Remarks	–
Side effects	–
See also	setLoggingOn
Example	<pre>setLoggingOff(); // Stop logging ... // No communication with adjacent devices is logged now</pre>

Routing

setRoutingOn

Function	Routing enabled
Purpose	Outgoing packets will be delivered via routing devices on background.
Syntax	<code>void setRoutingOn ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	For IQMESH only. Default OS condition is Routing off. Not implemented , reserved for future OS versions. Do not use in this OS version.
Remarks	To select wheter the transmitting packet should be delivered via routing devices use the <code>_ROUTEF</code> flag (PIN.5) instead of this function: <ul style="list-style-type: none"> <code>_ROUTEF = 1</code>: routing enabled. Routing vector RTV[0 to 3] should be specified. <code>_ROUTEF = 0</code>: routing disabled.
Side effects	–
See also	<code>setRoutingOff</code>
Example	See RFTXpacket, example 3

setRoutingOff

Function	Routing disabled
Purpose	Outgoing packets will not be delivered via routing devices on background.
Syntax	<code>void setRoutingOff ()</code>
Parameters	–
Return value	–
Output values	–
Preconditions	For IQMESH only. Default OS condition is Routing off. Not implemented , reserved for future OS versions. Do not use in this OS version.
Remarks	To select wheter the transmitting packet should be delivered via routing devices use the <code>_ROUTEF</code> flag (PIN.5) instead of this function: <ul style="list-style-type: none"> <code>_ROUTEF = 1</code>: routing enabled. Routing vector RTV[0 to 3] should be specified. <code>_ROUTEF = 0</code>: routing disabled.
Side effects	–
See also	<code>setRoutingOn</code>
Example	See RFTXpacket, example 2

Bonding – Node only

bondRequest

Function	Ask Coordinator via RF for bonding to its network. Bond the Node in cooperation with Coordinator and record it to EEPROM.
Purpose	Request by the Node to be included to the network on both Coordinator's and Node's sides.
Syntax	bit <code>bondRequest()</code>
Parameters	–
Return value	<ul style="list-style-type: none"> • 1 – Node has been bonded • 0 – Node has not been bonded
Output values	<ul style="list-style-type: none"> • The <code>amIBonded</code> function starts to return value == 1 whenever is called while the Node is bonded by <code>bondRequest</code> not being unbonded by <code>removeBond</code> or <code>wipeBondNR</code>. • Coordinator is not affected at all. • <code>param2</code>: Node address (if successfully bonded only). Not guaranteed for future OS versions.
Preconditions	For IQMESH only.
Remarks	Bonding is a mutual relationship between Coordinator and Node. Coordinator assigns a Node number (1 to 239) to the Node which serves as Node address within the network. (Coordinator itself has the address 0.) Bonding accomplishes via exchanging system RF packets and results are stored in system part of internal EEPROMs. The user can access results and change them via other functions related to bonding. See example E11 - MEASUREMENT, E11 - DATACENTER [10] and IQMESH specification [4]. This function is active until successfully finished or fixed 10 s timeout expired. RF power is not affected (from OS v2.10).
Side effects	<p>The following values are modified and not restored:</p> <ul style="list-style-type: none"> • DLEN • PIN • toutRF • bufferRF[0 to 63] destroyed • bufferINFO[0 to 34] destroyed • Result of <code>captureTicks</code> is destroyed if <code>startCapture</code> is active on background at the same time. • Watchdog is disabled during this operation and enabled after finishing
See also	<code>bondNewNode</code> , <code>amIBonded</code> , <code>removeBond</code> , <code>rebondNode</code> , <code>wipeBondNR</code> , <code>getNetworkParams</code>
Example1	<pre> pulsingLED(); // LED blinking indicates attempt to bond (max. 10 s) if (bondRequest()) { // if successfully bonded stopLED(); _RLED=1; // LED On waitDelay(100); // for 1 s } stopLED(); </pre>
Example2	See <code>amIBonded</code>

amIBonded

Function	Is the Node bonded?
Purpose	Test whether the Node is bonded on Node's side
Syntax	bit <code>amIBonded()</code>
Parameters	–
Return value	<ul style="list-style-type: none">• 1 – Node is bonded (after <code>bondRequest</code> not being unbonded by <code>removeBond</code> or <code>wipeBondNR</code>)• 0 – Node is not bonded:<ul style="list-style-type: none">• no <code>bondRequest</code> has ever been successfully executed• after <code>removeBond</code> or <code>wipeBondNR</code>
Output values	–
Preconditions	For IQMESH only. Result is not depended on the Coordinator at all.
Remarks	See example E11 - MEASUREMENT [10].
Side effects	–
See also	<code>bondRequest</code> , <code>removeBond</code> , <code>wipeBondNR</code>
Example	<pre>while (!amIBonded()) // Request for being bonded (if not bonded yet) { bondRequest(); // Repeatedly try to bond clrwdt(); } // until successful</pre>

removeBond

Function	Remove the Node from the network and record it to EEPROM.
Purpose	Exclude the Node from the network on Node's side and keep its Node number reserved for possible future rebonding.
Syntax	<code>void removeBond ()</code>
Parameters	–
Return value	–
Output values	<ul style="list-style-type: none"> • The <code>amIBonded</code> function starts to return value <code>== 0</code> whenever is called until the Node is bonded again via <code>bondRequest</code>. • Just this value is affected but the Node keeps the Node number still stored (for possible future rebonding with the same Node number). • Coordinator is not affected at all.
Preconditions	For IQMESH only. Max. 239 Nodes can be bonded to single network.
Remarks	<ul style="list-style-type: none"> • See example E11 - MEASUREMENT [10]. • For rebonding use <code>bondRequest</code> again (reserved Node number will be offered to Coordinator then). The Coordinator accepts it if this number is still free. • <code>removeBond</code> relates to Node only and <code>removeBondedNode</code> and <code>rebondNode</code> relate to Coordinator only. The other side is not informed by OS about changes made by these functions. If synchronization is needed it should be done by the application.
Side effects	<ul style="list-style-type: none"> • The effect of <code>removeBond</code> will not surface in Coordinator until <code>bondRequest/bondNewNode</code> procedure is invoked afterward. That is why Coordinator can use the number reserved for this Node for another Node meanwhile (after possible <code>removeBondedNode</code>). In this case the Coordinator will assign a different number.
See also	<code>bondRequest</code> , <code>bondNewNode</code> , <code>amIBonded</code> , <code>rebondNode</code> , <code>wipeBondNR</code>
Example	<pre>removeBond(); // Remove the bond. Node number remains reserved for this // Node to be bonded with the same Node number in the future.</pre>

wipeBondNR

Function	Unbond the Node from the network and cancel the Node number reservation for given Node. Record it to EEPROM.
Purpose	Exclude the Node from the network on Node's side and wipe the Node number (not to keep it for the future)
Syntax	<code>void wipeBondNR ()</code>
Parameters	–
Return value	–
Output values	<ul style="list-style-type: none"> • The <code>amIBonded</code> function starts to return value <code>== 0</code> whenever is called until the Node is bonded again. • Additionally, the Node number is wiped not supporting future rebonding with the same Node number. • Coordinator is not affected at all.
Preconditions	For IQMESH only
Remarks	The number is released for future bonding to any Node. See example E11 - MEASUREMENT [10]. The effect will not surface in Coordinator until <code>bondRequest/bondNewNode</code> procedure is invoked.
Side effects	–
See also	<code>removeBond</code> , <code>rebondNode</code> , <code>bondNewNode</code>
Example	<pre>wipeBondNR(); // Remove the bond and release the Node number to be used // by any Node in the future.</pre>

Bonding – Coordinator only
bondNewNode

Function	Look for bond requesting devices and bond a new Node by Coordinator on a Node's request via RF. Allocate the Node number and assign the Network number and send both to Node via RF. If successful, the Node is bonded to the network on both Coordinator's and Node's sides and is included to the list of bonded Nodes provided by Coordinator in EEPROM.
Purpose	Include a new Node to the network
Syntax	bit <code>bondNewNode ()</code>
Parameters	–
Return value	<ul style="list-style-type: none"> • 1 – bonding successful, Node included to the list of bonded Nodes • 0 – bonding unsuccessful, Node not included to the list of bonded Nodes
Output values	<ul style="list-style-type: none"> • <code>param2</code>: Node number • The <code>isBondedNode</code> function starts to return value == 1 whenever is called while the Node is in the list of bonded Nodes.
Preconditions	<ul style="list-style-type: none"> • For IQMESH only. • Coordinator accomplishes bonding on request from Node via RF. When this function is executing the <code>bondRequest</code> function must just be active in the Node.
Remarks	<ul style="list-style-type: none"> • See example E11 - DATACENTER [10]. • If no requesting Node is detected during 10 s period this function terminates. • Network number is derived from Coordinator ID which ensures unique identification of various networks. • In case of rebonding the original (reserved) Node number is preferred to be assigned. If it is just occupied by another Node a different Node number is assigned. • RF power is not affected (from OS v2.10).
Side effects	<p>The following values are modified and not restored:</p> <ul style="list-style-type: none"> • <code>toutRF</code> modified • <code>bufferRF[0 to 63]</code> = destroyed • <code>bufferINFO[0 to 34]</code> = destroyed • Result of <code>captureTicks</code> is destroyed if <code>startCapture</code> is active on background at the same time. • Watchdog is disabled during this operation and enabled after finishing
See also	<code>bondRequest</code> , <code>removeBondedNode</code> , <code>rebondNode</code> , <code>isBondedNode</code>
Example	<pre> if (bondNewNode()) // Bonding successful ? { // Yes: NodeNumber = param2; ... } else { // No: ... // Arrange necessary steps } </pre>

isBondedNode

Function	Is specified Node in the list of bonded Nodes?
Purpose	Test whether the Node is bonded on Coordinator's side
Syntax	bit isBondedNode (n)
Parameters	uns8 n: Node number
Return value	<ul style="list-style-type: none"> • 1 – Node is in the list of bonded Nodes • 0 – Node is not in the list of bonded Nodes
Output values	–
Preconditions	For IQMESH only. The result is not affected by the Node at all.
Remarks	–
Side effects	–
See also	bondNewNode, removeBondedNode, rebondNode, clearAllBonds
Example	<pre> if isBondedNode(28) // Is Node #28 bonded ? { // Yes: ... // Coordinator assumes Node #28 to be bonded } else { // No: ... // Coordinator assumes Node #28 not to be bonded } </pre>

removeBondedNode

Function	Remove a Node from the list of bonded Nodes by Coordinator in EEPROM
Purpose	Exclude the Node from the network on Coordinator's side
Syntax	void removeBondedNode (n)
Parameters	uns8 n: Node number
Return value	–
Output values	The <code>isBondedNode</code> function starts to return value == 0 whenever is called while the Node is not in the list of bonded Nodes. The Node is not affected at all.
Preconditions	For IQMESH only
Remarks	<code>removeBondedNode</code> and <code>rebondNode</code> relate to Coordinator only and <code>removeBond</code> relates to Node only. The other side is not informed by OS about changes made by these functions. If synchronization is needed it should be done by the application.
Side effects	–
See also	bondNewNode, isBondedNode, clearAllBonds, removeBond
Example	<pre> removeBondedNode(28); // Coordinator assumes Node #28 to be // out of the network from now on </pre>

rebondNode

Function	Put a Node back to the list of bonded Nodes by Coordinator in EEPROM
Purpose	Include the Node to the network again on Coordinator's side
Syntax	bit <code>rebondNode (n)</code>
Parameters	uns8 n: Node number
Return value	reserved for future OS versions
Output values	The <code>isBondedNode</code> function starts to return value == 1 whenever is called while the Node is in the list of bonded Nodes. The Node is not affected at all.
Preconditions	For IQMESH only. Avoid rebonding a Node not being bonded ever before.
Remarks	<code>removeBondedNode</code> and <code>rebondNode</code> relate to Coordinator only and <code>removeBond</code> relates to Node only. The other side is not informed by OS about changes made by these functions. If synchronization is needed it should be done by the application.
Side effects	–
See also	<code>bondNewNode</code> , <code>removeBondedNode</code> , <code>isBondedNode</code>
Example	<pre>rebondNode(28); // Coordinator assumes Node #28 to be // back in the network from now on</pre>

clearAllBonds

Function	Remove all Nodes from the list of bonded Nodes by Coordinator in EEPROM
Purpose	Excluding all Nodes from the network on Coordinator's side
Syntax	void <code>clearAllBonds ()</code>
Parameters	–
Return value	–
Output values	The <code>isBondedNode</code> function starts to return value == 0 whenever is called while the Node is not in the list of bonded Nodes. Nodes are not affected at all.
Preconditions	For IQMESH only
Remarks	See example E11 - DATACENTER [10]. Coordinator will start to assign Node numbers from 0 for Nodes with wiped Node numbers. For not wiped ones will try to restore the original Node numbers.
Side effects	<code>bufferINFO[0 to 34]</code> destroyed
See also	<code>removeBondedNode</code>
Example	<pre>clearAllBonds(); // Exclude all currently bonded nodes from the network</pre>

Documentation and Information

- 1 IQRF OS User's guide www.iqrf.org/weben/downloads.php?id=155
- 2 **RAM map** and **EEPROM map**, IQRF OS User's guide, Appendix 1 [1]
- 3 **IQRF** home page www.iqrf.org
- 4 **IQMESH** specification www.iqmesh.org/iqmesh
- 5 **SPI** specification www.iqrf.org/weben/downloads.php?id=85
- 6 **IQRF support** site www.iq-esupport.com
- 7 **TR-52B** datasheet: www.iqrf.org/weben/downloads.php?id=91
- 8 **PIC16F886** datasheet: www.iqrf.org/weben/downloads.php?id=126
- 9 **IQRF IDE**: www.iqrf.org/weben/downloads.php?id=86
- 10 **Basic examples** (included in the StartUp Package): www.iqrf.org/weben/downloads.php?id=112

If you need a help or more information please visit IQRF support pages [6] and Submit a Ticket with your request. A lot of information is also available in the IQRF OS User's guide [1] and on the IQRF home page [3].

Document revision

- 100310 Description of SPI functions improved
- 100301 OS v2.11
- 100209 OS v2.10, first release for TR-52B, TR-53B and compatibles

Index

amlBonded.....	46
applInfo.....	26
bondNewNode.....	48
bondRequest.....	45
captureTicks.....	11
checkRF.....	34
clearAllBonds.....	50
clearBufferINFO.....	22
compareBufferINFO2RF.....	24
copyBufferCOM2INFO.....	24
copyBufferCOM2RF.....	24
copyBufferINFO2COM.....	22
copyBufferINFO2RF.....	23
copyBufferRF2COM.....	23
copyBufferRF2INFO.....	23
copyMemoryBlock.....	25
debug.....	8
disableSPI.....	27
eeReadByte.....	17
eeReadData.....	17
eeWriteByte.....	18
eeWriteData.....	18
enableSPI.....	27
getNetworkParams.....	43
getStatusSPI.....	30
getSupplyVoltage.....	9
getTemperature.....	9
iqrfSleep.....	7
isBondedNode.....	49
isDelay.....	12
moduleInfo.....	26
pulseLEDG.....	16
pulseLEDR.....	14
pulsingLEDG.....	15
pulsingLEDR.....	14
readFromRAM.....	19
rebondNode.....	50
removeBond.....	47
removeBondedNode.....	49
reset.....	6
RFRXpacket.....	37
RFTXpacket.....	35
setCoordinatorMode.....	40
setLoggingOff.....	42
setLoggingOn.....	42
setNetworkFilteringOff.....	41
setNetworkFilteringOn.....	41
setNetworkingOff.....	39
setNetworkOne.....	38
setNetworkTwo.....	38
setNodeMode.....	40
setOffPulsingLED.....	13
setOnPulsingLED.....	13
setPIR1.....	21
setRFband.....	32
setRFchannel.....	32
setRFmode.....	33
setRFsleep.....	7
setRFspeed.....	31
setRoutingOff.....	44
setRoutingOn.....	44
setTXpower.....	31
startCapture.....	11
startDelay.....	12
startSPI.....	28
stopLEDG.....	16
stopLEDR.....	15
stopSPI.....	29
waitDelay.....	10
waitMS.....	10
wipeBondNR.....	47
writeToRAM.....	20

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 permission 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)

Website	www.iqrf.org
E-mail	sales@iqrf.org
On-line support	http://iq-esupport.com



Simple way to smarter wireless solutions