

# **IQRF OS**

## **Operating System**

**version 2.09**

## **Reference Guide**



**Simple way to smarter wireless solutions**

## Content

Quick reference.....	4
Parameters.....	4
Functions.....	4
OS functions.....	6
Control.....	6
reset.....	6
iqrfSleep.....	7
debug.....	8
batteryValueOK.....	9
getTemperature.....	10
Active waiting.....	11
waitMS.....	11
waitDelay.....	11
Timing on background.....	12
startCapture.....	12
captureTicks.....	12
startDelay.....	13
isDelay.....	13
LED indication.....	14
setOnPulsingOUT2.....	14
setOffPulsingOUT2.....	14
pulsingOUT2.....	15
pulseOUT2.....	15
stopOUT2.....	16
EEPROM.....	17
eeReadByte.....	17
eeReadData.....	17
eeWriteByte.....	18
eeWriteData.....	19
RAM.....	20
readFromRAM.....	20
writeToRAM.....	21
setPIR1.....	22
Buffers, data blocks.....	23
clearBufferINFO.....	23
copyBufferINFO2COM.....	23
copyBufferINFO2RF.....	24
copyBufferRF2COM.....	24
copyBufferRF2INFO.....	24
copyBufferCOM2RF.....	25
copyBufferCOM2INFO.....	25
compareBufferINFO2RF.....	25
copyMemoryBlock.....	26
moduleInfo.....	27
applInfo.....	27
SPI.....	28
enableSPI.....	28
disableSPI.....	28
startSPI.....	29
stopSPI.....	29
getStatusSPI.....	30
RF.....	31
setTXpower.....	31
RFTXpacket.....	31
RFRXpacket.....	33
Networking.....	34
setNetworkOne.....	34
setNetworkTwo.....	34
setNetworkingOff.....	35

setCoordinatorMode.....	36
setNodeMode.....	36
setNetworkFilteringOn.....	37
setNetworkFilteringOff.....	37
setLoggingOn.....	38
setLoggingOff.....	38
getNetworkParams.....	39
Routing.....	40
setRoutingOn.....	40
setRoutingOff.....	40
Bonding – Node only.....	41
bondRequest.....	41
amIBonded.....	42
removeBond.....	43
wipeBondNR.....	43
Bonding – Coordinator only.....	44
bondNewNode.....	44
isBondedNode.....	45
removeBondedNode.....	45
rebondNode.....	46
clearAllBonds.....	46
Documentation and Information.....	47
Sales and Service.....	49

## 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

<b>Control</b>		<b>6</b>
<code>reset()</code>	initialization of microcontroller, OS and application	6
<code>iqrfSleep()</code>	enter power saving mode (Sleep)	7
<code>debug()</code>	enter the debug mode	8
<code>uns8 batteryValueOK()</code>	battery check	9
<code>getTemperature()</code>	temperature measurement	10
<b>Active waiting</b>		<b>11</b>
<code>waitMS(ms)</code>	active waiting (time in ms)	11
<code>waitDelay(ticks)</code>	active waiting (time in ticks)	11
<b>Timing on background</b>		<b>12</b>
<code>startDelay(ticks)</code>	start waiting (time in ticks)	13
<code>bit isDelay()</code>	still waiting	13
<code>startCapture()</code>	resets counter of ticks	12
<code>captureTicks()</code>	get number of ticks counted	12
<b>LED indication</b>		<b>14</b>
<code>setOnPulsingOUT2(ticks)</code>	LED On time setting (for blinking)	14
<code>setOffPulsingOUT2(ticks)</code>	LED Off time setting (for blinking)	14
<code>pulsingOUT2()</code>	LED activation (blinking on background)	15
<code>pulseOUT2()</code>	single LED pulse (flash on background)	15
<code>stopOUT2()</code>	LED off, blinking stopped	16
<b>EEPROM</b>		<b>17</b>
<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	19
<b>RAM</b>		<b>20</b>
<code>uns8 readFromRAM(addr)</code>	read one byte	20
<code>writeToRAM(addr, data)</code>	write one byte	21
<code>setPIR1(x)</code>	setting of microcontroller peripheral flags	22
<b>Buffers, data blocks</b>		<b>23</b>
<code>clearBufferINFO()</code>	bufferINFO clearing	23
<code>copyBufferINFO2COM()</code>	copy bufferINFO to bufferCOM	23
<code>copyBufferINFO2RF()</code>	copy bufferINFO to bufferRF	24
<code>copyBufferRF2COM()</code>	copy bufferRF to bufferCOM	24
<code>copyBufferRF2INFO()</code>	copy bufferRF to bufferINFO	24
<code>copyBufferCOM2RF()</code>	copy bufferCOM to bufferRF	25
<code>copyBufferCOM2INFO()</code>	copy bufferCOM to bufferINFO	25
<code>bit compareBufferINFO2RF(length)</code>	comparison of bufferINFO and bufferRF	25
<code>copyMemoryBlock(uns16 from, uns16 to, uns8 length)</code>	copy any data block to any position	26
<code>moduleInfo()</code>	get info about transceiver module and OS	27
<code>appInfo()</code>	copy info about application from EEPROM to bufferINFO	27
<b>SPI</b>		<b>28</b>
<code>enableSPI()</code>	SPI communication line activation	28
<code>disableSPI()</code>	SPI communication line deactivation	28
<code>startSPI(length)</code>	SPI packet transmission	29
<code>stopSPI()</code>	SPI stopping	29
<code>bit getStatusSPI()</code>	SPI status, update SPI flags	30

<b>RF</b>		<b>31</b>
<b>setTXpower</b> (level)	RF power setting (7 levels)	31
<b>RFTXpacket</b> ()	send a packet from buffer RF via RF	33
bit <b>RFRXpacket</b> ()	receive a packet via RF to buffer RF	33
<b>Networking</b>		<b>34</b>
<b>setNetworkOne</b> ()	network 1 selected	34
<b>setNetworkTwo</b> ()	network 2 selected	34
<b>setNetworkingOff</b> ()	networking disabled	35
<b>setCoordinatorMode</b> ()	device is the Coordinator	36
<b>setNodeMode</b> ()	device is a Node	31
<b>setNetworkFilteringOn</b> ()	packets accepted from current network only	37
<b>setNetworkFilteringOff</b> ()	packets accepted from both networks	37
<b>setLoggingOn</b> ()	communication with adjacent nodes logged	38
<b>setLoggingOff</b> ()	communication with adjacent nodes not logged	38
uns8 <b>getNetworkParams</b> ()	get information about the network	39
<b>Routing</b>		<b>40</b>
<b>setRoutingOn</b> ()	device works as a router	40
<b>setRoutingOff</b> ()	device does not work as a router	40
<b>Bonding - Node</b>		<b>41</b>
bit <b>bondRequest</b> ()	request for bonding	41
bit <b>amIBonded</b> ()	is the Node bonded?	42
<b>removeBond</b> ()	unbonding	43
<b>wipeBondNR</b> ()	unbonding and reserved node number cancellation	43
<b>Bonding - Coordinator</b>		<b>44</b>
bit <b>bondNewNode</b> ()	bonding a Node	44
bit <b>isBondedNode</b> (n)	is the Node bonded?	45
<b>removeBondedNode</b> (n)	unbonding a Node	45
bit <b>rebondNode</b> (n)	rebonding a Node	46
<b>clearAllBonds</b> ()	clearing of all bonds	46

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

## OS functions

### Control

#### reset

<b>Function</b>	Initialization of microcontroller, OS and application
<b>Purpose</b>	If needed, it is possible to initialize the application and run the program from the very beginning again.
<b>Syntax</b>	<code>void reset ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–TO = 0
<b>Preconditions</b>	–
<b>Remarks</b>	<p>This is forced watchdog overflow reset. To distinguish the power down reset from this one the –TO (STATUS . 4) status flag is intended.</p> <ul style="list-style-type: none"> <li>–TO = 0 after <code>reset ()</code> execution</li> <li>–TO = 1 after power up</li> </ul> <p>Refer to the PIC datasheet [8] (STATUS register) for details.</p>
<b>Side effects</b>	It is strictly specified which RAM registers are initialized and which are unchanged. Refer to the PIC datasheet [8] (Reset) for details.
<b>See also</b>	–
<b>Example</b>	<pre>if (Error == 1)     reset ();</pre>

### iqrfsleep

<b>Function</b>	Entering the power saving mode (Sleep)
<b>Purpose</b>	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.
<b>Syntax</b>	<code>void iqrfsleep()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	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>
<b>Preconditions</b>	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 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.
<b>Remarks</b>	There are no restriction concerning Sleep and wake-up due to OS from OS v2.03. All features are under user's control. See example E01-TR [10].
<b>Side effects</b>	Global interrupt enable (GIE) is controlled by OS again after wake-up.
<b>See also</b>	–
<b>Example1</b>	<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>
<b>Example2</b>	<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 iqrfsleep();       // Put the module into Sleep mode RBIF = 0;         // Clear flag if (buttonPressed) // If button is pressed { ... }           // ... </pre>

## debug

<b>Function</b>	Enter the debug mode
<b>Purpose</b>	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.
<b>Syntax</b>	<code>void debug ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	OS directly returns no value but supports using W (PIC accumulator) to identify which of the debug points is currently active.
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• Debug should be used with corresponding development kit (e.g. CK-USB-02/03) and the IQRF IDE development environment.</li> <li>• 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"> <li>• Pins C5 to C8 are configured for SPI in respective TRIS bits (C7 in, the others out). It is arranged by OS by default.</li> <li>• 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.</li> <li>• SPI need not be enabled by <code>enableSPI</code></li> </ul> </li> </ul>
<b>Remarks</b>	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].
<b>Side effects</b>	<ul style="list-style-type: none"> <li>• <code>param1</code> to <code>param4</code> are not displayed</li> <li>• Watchdog is cleared while in Debug mode</li> </ul>
<b>See also</b>	–
<b>Example</b>	<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>



**batteryValueOK**

<b>Function</b>	Get a reference value for battery check from EEPROM.
<b>Purpose</b>	Power supply measurement to check whether the voltage is sufficient, especially for battery applications.
<b>Syntax</b>	uns8 <b>batteryValueOK()</b>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	Return value is factory calibrated constant with respect to piece-to-piece variability of device parameters.
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• The <code>THRESHOLD</code> voltage must be selected according to desired limit.</li> <li>• Battery check is based on A/D measurement. See example E10-BATTERY [10].</li> </ul>
<b>Remarks</b>	Power supply (on the C3 SIM pin) is measured by internal A/D converter of the microcontroller. Supply voltage depending on the battery condition is used as a reference while the measured input voltage is stable (LED voltage drop). The A/D converter must be controlled via the user program. The A/D result is stored in the <code>ADRESH</code> special function PIC register.
<b>Side effects</b>	–
<b>See also</b>	–
<b>Example</b>	<pre>// #define THRESHOLD    1        // 3.0 V // #define THRESHOLD    5        // 2.9 V // #define THRESHOLD    10       // 2.8 V for BL-ER14250 battery  ...                               // include A/D power measurement here                                    // A/D result is in ADRESH  pomi = batteryValueOK();          // Get calibrated value from EEPROM if (pomi &gt; ADRESH)                 // Comparison     if (pomi-ADRESH &gt; THRESHOLD)  // Exhausted?         _LED = 1;                 // Yes: LED on</pre>

### getTemperature

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

## Active waiting

### waitMS

<b>Function</b>	Wait specified number of miliseconds
<b>Purpose</b>	Time delay generation
<b>Syntax</b>	<code>void waitMS (ms)</code>
<b>Parameters</b>	ms - time to wait in miliseconds (1 - 255)
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	This function can be combined with <code>waitDelay</code> , <code>startCapture</code> and <code>captureTicks</code> .
<b>Remarks</b>	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].
<b>Side effects</b>	–
<b>See also</b>	<code>waitDelay</code> , <code>startDelay</code>
<b>Example</b>	<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

<b>Function</b>	Wait specified number of ticks
<b>Purpose</b>	Time delay generation
<b>Syntax</b>	<code>void waitDelay (ticks)</code>
<b>Parameters</b>	ticks – time to wait in 10 ms intervals (1 - 255)
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	This function can be combined with <code>waitMS</code> .
<b>Remarks</b>	This is the active waiting (on OS foreground). No other operation runs on OS foreground during waiting.
<b>Side effects</b>	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.
<b>See also</b>	<code>waitMS</code> , <code>startDelay</code>
<b>Example</b>	<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

<b>Function</b>	Reset and start the Capture timer
<b>Purpose</b>	Initialization of time measurement or delay generation
<b>Syntax</b>	<code>void startCapture ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	This function can be combined with <code>waitMS</code> .
<b>Remarks</b>	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.
<b>Side effects</b>	Functionality is affected by <code>bondRequest</code> , <code>bondNewNode</code> and <code>RFRXpacket</code> .
<b>See also</b>	<code>captureTicks</code>
<b>Example</b>	See <code>captureTicks</code>

### captureTicks

<b>Function</b>	Get number of ticks counted from the last <code>startCapture</code> and <code>captureTicks</code> calling.
<b>Purpose</b>	Measurement of elapsed time.
<b>Syntax</b>	<code>void captureTicks ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output value</b>	<ul style="list-style-type: none"> <li>• <code>param3</code>: ticks counted from the last <code>startCapture</code> (0 - 65535)</li> <li>• <code>param4</code>: ticks counted from the last <code>captureTicks</code> (0 - 65535)</li> </ul>
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• <code>startCapture</code> should be used at least once before.</li> <li>• 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.</li> </ul>
<b>Remarks</b>	See example E05–DELAYS [10]
<b>Side effects</b>	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].
<b>See also</b>	<code>startCapture</code>
<b>Example</b>	<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

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

### isDelay

<b>Function</b>	Information whether specified delay is still in progress
<b>Purpose</b>	Time measurement or delay generation
<b>Syntax</b>	bit <b>isDelay</b> ()
<b>Parameters</b>	–
<b>Return value</b>	<ul style="list-style-type: none"> <li>• 1: still in progress</li> <li>• 0: elapsed</li> </ul>
<b>Output values</b>	–
<b>Preconditions</b>	startDelay should be used before.
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• The Delay timer measures specified time period. The result is available via the isDelay function.</li> <li>• Tip: the clrwdt instruction should be used to avoid unintentional watchdog reset during the delay.</li> <li>• See example E05-DELAYS [10].</li> </ul>
<b>Side effects</b>	–
<b>See also</b>	startDelay
<b>Example</b>	<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

### setOnPulsingOUT2

<b>Function</b>	LED On time setting
<b>Purpose</b>	Specification of the "On" time for LED (either for a single flash or for blinking)
<b>Syntax</b>	<code>void setOnPulsingOUT2 (ticks)</code>
<b>Parameters</b>	<code>uns8 ticks</code> : number of ticks (10 ms system intervals) (1-255)
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	–
<b>Remarks</b>	Default value is 5 (50 ms).
<b>Side effects</b>	–
<b>See also</b>	<code>setOffPulsingOUT2</code> , <code>pulsingOUT2</code> , <code>pulseOUT2</code>
<b>Example</b>	See <code>setOffPulsingOUT2</code>

### setOffPulsingOUT2

<b>Function</b>	LED Off time setting
<b>Purpose</b>	Specification of the "Off" time for LED (for blinking)
<b>Syntax</b>	<code>void setOffPulsingOUT2 (ticks)</code>
<b>Parameters</b>	<code>uns8 ticks</code> : number of ticks (10 ms system intervals) (1-255)
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	–
<b>Remarks</b>	Default value is 20 (200 ms).
<b>Side effects</b>	–
<b>See also</b>	<code>setOnPulsingOUT2</code> , <code>pulsingOUT2</code>
<b>Example</b>	<pre> // Change blinking to 250 ms On / 750 ms Off setOnPulsingOUT2(25); // 250 ms On setOffPulsingOUT2(75); // 750 ms Off </pre>

### pulsingOUT2

<b>Function</b>	LED blinking
<b>Purpose</b>	Continuous LED blinking on OS background
<b>Syntax</b>	<code>void pulsingOUT2 ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	Blinking times should be defined in advance by <code>setOnPulsingOUT2</code> and <code>setOffPulsingOUT2</code> . The appropriate PIC pin is configured as an output automatically.
<b>Remarks</b>	Blinking continues until it is stopped by the user (e.g. by <code>stopOUT2</code> ).
<b>Side effects</b>	–
<b>See also</b>	<code>setOnPulsingOUT2</code> , <code>setOffPulsingOUT2</code> , <code>stopOUT2</code> , <code>pulseOUT2</code>
<b>Example1</b>	<code>pulsingOUT2 (); // continuous blinking on OS background</code>
<b>Example1</b>	<pre>// Blinking for 2 s pulsingOUT2 (); // blinking for 2 s on OS background waitDelay(200); // 2 s delay generated on foreground stopOUT2 (); // Stop blinking</pre>

### pulseOUT2

<b>Function</b>	Single LED flash
<b>Purpose</b>	LED flash on OS background
<b>Syntax</b>	<code>void pulseOUT2 ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	Flash time should be defined in advance by <code>setOnPulsingOUT2</code> . The appropriate PIC pin is configured as an output automatically.
<b>Remarks</b>	The on-board LED(s) can also be directly controlled on OS foreground using C commands for manipulating the <code>_OUT2</code> output (a pin the LED is connected to) and corresponding control bit (TRISB.7).
<b>Side effects</b>	<ul style="list-style-type: none"> <li>• The appropriate PIC pin is not restored to the state before <code>pulsingOUT2/pulseOUT2</code> (<code>TRISB.7 == 0, _OUT2 == 0</code> after finishing on background).</li> <li>• Background activity overrides setting of <code>OUT2</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.</li> </ul>
<b>See also</b>	<code>setOnPulsingOUT2</code> , <code>pulsingOUT2</code> , <code>stopOUT2</code>
<b>Example</b>	<pre>setOnPulsingOUT2(10); // 100 ms On pulseOUT2 (); // Single 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>

## stopOUT2

<b>Function</b>	LED off, blinking stopped
<b>Purpose</b>	Stops the LED activity on OS background
<b>Syntax</b>	<code>void stopOUT2 ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	–
<b>Remarks</b>	–
<b>Side effects</b>	<ul style="list-style-type: none"> <li>• The appropriate PIC pin is not restored to the state before pulsingOUT2/pulseOUT2 (TRISB.7 == 0, _OUT2 == 0 after finishing on background).</li> <li>• Background activity overrides setting of OUT2 (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.</li> </ul>
<b>See also</b>	pulsingOUT2, pulseOUT2
<b>Example1</b>	<pre>pulsingOUT2 ();           // Start blinking on OS background ...                       // Blinking continues during any operation stopOUT2 ();             // Stop blinking</pre>
<b>Example2</b>	<pre>pulseOUT2 ();           // LED On on OS background ...                     // continuously lighting during any operation                         // until specified time expired stopOUT2 ();           // or LED is switched Off by this command</pre>



## EEPROM

### eeReadByte

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

### eeReadData

<b>Function</b>	Read a block of specified length from specified location in EEPROM to <code>bufferINFO</code>
<b>Purpose</b>	Block access to EEPROM
<b>Syntax</b>	<code>void eeReadData (addr, length)</code>
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• <code>uns8 addr</code>: address in EEPROM (0 to 0xBF - length + 1). See EEPROM map [2].</li> <li>• <code>uns8 length</code>: number of bytes to be read (1 to 35)</li> </ul>
<b>Return value</b>	–
<b>Output values</b>	<code>bufferINFO[0 to length - 1]</code>
<b>Preconditions</b>	–
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• Direct user access to EEPROM (using registers <code>EECONx</code> etc.) is not allowed for security reasons, specialized OS functions are intended for this.</li> <li>• EEPROM area dedicated to OS (locations 0xC0 or higher) is not accessible. See example E04–EEPROM [10].</li> </ul>
<b>Side effects</b>	• Any attempt to read from protected area above 0xBF leads to reading from EEPROM location 0xA0.
<b>See also</b>	<code>eeReadByte</code> , <code>eeWriteByte</code> , <code>eeWriteData</code>
<b>Example1</b>	<code>eeReadData(10, 16); // copy 16B from EEPROM from address 10 to bufferINFO // bufferINFO[0] = EEPROM[10] // ... // bufferINFO[15] = EEPROM[25]</code>
<b>Example2</b>	<code>// 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]</code>

### eeWriteByte

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

<b>Function</b>	Write a block of specified length from <code>bufferINFO</code> to specified location in EEPROM
<b>Purpose</b>	Block access to EEPROM
<b>Syntax</b>	<code>void eeWriteData (addr, length)</code>
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• <code>uns8 addr</code>: address in EEPROM . See EEPROM map [2]. <ul style="list-style-type: none"> <li>• (0 to 0xBF - length + 1) for Node</li> <li>• (0xA0 to 0xBF - length + 1) for Coordinator</li> </ul> </li> <li>• <code>uns8 length</code>: number of bytes to be written from <code>bufferINFO</code> (1 to 35)</li> </ul>
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	–
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• Direct user access to EEPROM (using registers <code>EECONx</code> etc.) is not allowed for security reasons, specialized OS functions are intended for this.</li> <li>• EEPROM area dedicated to OS (locations 0xC0 or higher) is not accessible. See example E04–EEPROM [10].</li> </ul>
<b>Side effects</b>	• Any attempt to write to protected area above 0xBF leads to writing to EEPROM location 0xA0.
<b>See also</b>	<code>eeReadByte</code> , <code>eeReadData</code> , <code>eeWriteByte</code>
<b>Example1</b>	<pre>eeWriteData(10,16); // copy 16B from bufferINFO to EEPROM to address 10                   // EEPROM[10] = bufferINFO[0]                   //      ...                   // EEPROM[25] = bufferINFO[15]</pre>
<b>Example2</b>	<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**

<b>Function</b>	Read one byte from specified location in RAM
<b>Purpose</b>	Indirect access to RAM registers
<b>Syntax</b>	<code>uns8 readFromRAM(addr)</code>
<b>Parameters</b>	<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.
<b>Return value</b>	Value read from specified location
<b>Output values</b>	–
<b>Preconditions</b>	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.
<b>Remarks</b>	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].
<b>Side effects</b>	–
<b>See also</b>	<code>writeToRAM</code>
<b>Example1</b>	<pre>IRP = 0; X = readFromRAM(bufferCOM + 10); IRP = 1; Y = readFromRAM(bufferRF + 10);</pre>
<b>Example2</b>	<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>
<b>Example3</b>	<pre>// Correct: IRP = 1; X = readFromRAM(0x90) - 1; // Perfect X = readFromRAM(0x190) - 1; // Also works thanks to the compiler feature</pre>
<b>Example4</b>	<pre>// Not allowed. The compiler uses FSR in such cases. for (i=0; i&lt;5; i++)     A[i] = bufferRF[i];</pre>
<b>Example5</b>	<pre>// Correct for (i=0; i&lt;5; i++)     A[i] = readFromRAM(bufferRF + i);</pre>

### writeToRAM

<b>Function</b>	Write one byte to specified location in RAM
<b>Purpose</b>	Indirect access to RAM registers
<b>Syntax</b>	<code>void writeToRAM(addr, value)</code>
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• <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.</li> <li>• <code>uns8 value</code>: value to be written</li> </ul>
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• 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.</li> <li>• Avoid writing to RAM areas dedicated to OS and to PIC special function registers otherwise OS can collapse. See RAM map [2].</li> <li>• Some locations are restricted from writing. In doubt, refer to IQRF support by the manufacturer [6].</li> </ul>
<b>Remarks</b>	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.
<b>Side effects</b>	–
<b>See also</b>	<code>readFromRAM</code> , <code>copyMemoryBlock</code>
<b>Example1</b>	<pre>IRP = 0; writeToRAM(bufferCOM + 10, 2 * X); IRP = 1; writeToRAM(bufferRF + 10, 2 * X);</pre>
<b>Example2</b>	<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>
<b>Example3</b>	<pre>// Correct: IRP = 1; writeToRAM(0x90, 201);                  // Perfect writeToRAM(0x190, 201);                // Also works thanks to the compiler feature</pre>
<b>Example4</b>	<pre>// Not allowed. The compiler uses FSR in such cases. for (i=0; i&lt;5; i++)     bufferRF[i] = i;</pre>
<b>Example5</b>	<pre>// Correct for (i=0; i&lt;5; i++)     writeToRAM(bufferRF + i, i);</pre>

**setPIR1**

<b>Function</b>	Setup microcontroller flags in PIR1
<b>Purpose</b>	Access to control flags for PIC internal peripherals (I <sup>2</sup> C, A/D, ...).
<b>Syntax</b>	<code>void setPIR1 (value)</code>
<b>Parameters</b>	<code>uns8 value</code> : value to be written
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	–
<b>Remarks</b>	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].
<b>Side effects</b>	–
<b>See also</b>	–
<b>Example</b>	<code>setPIR1(0) // ADIF = 0; SSPIF = 0;</code>

**Buffers, data blocks**
**clearBufferINFO**

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

**copyBufferINFO2COM**

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

### copyBufferINFO2RF

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

### copyBufferRF2COM

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

### copyBufferRF2INFO

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



### copyBufferCOM2RF

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

### copyBufferCOM2INFO

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

### compareBufferINFO2RF

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

### copyMemoryBlock

<b>Function</b>	Copy specified RAM block to specified location
<b>Purpose</b>	Copy memory block within RAM
<b>Syntax</b>	<code>void copyMemoryBlock (from, to, length)</code>
<b>Parameters</b>	<ul style="list-style-type: none"> <li>• <code>uns16 from</code>: starting address of the block to be copied</li> <li>• <code>uns16 to</code>: destination address</li> <li>• <code>uns8 length</code>: block length in bytes</li> </ul>
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• Copying is allowed also between different memory banks but both blocks are not allowed to cross bank boundaries (0x7F-0x80, 0xFF-0x100, 0x17F-0x180).</li> <li>• Unlike <code>writeToRAM</code>, "long" addresses (up to 0x1FF) are used. IRP is no object.</li> <li>• 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..</li> </ul>
<b>Remarks</b>	This function can access RAM within the whole memory area. See RAM map [2] and example E06 - RAM [10].
<b>Side effects</b>	–
<b>See also</b>	<code>writeToRAM</code>
<b>Example</b>	<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

<b>Function</b>	Store Module data to <code>bufferINFO</code>									
<b>Purpose</b>	Get information about transceiver module and OS									
<b>Syntax</b>	<code>void moduleInfo ()</code>									
<b>Parameters</b>	-									
<b>Return value</b>	-									
<b>Output values</b>	<code>bufferINFO[0 to 7]</code>									
<b>Preconditions</b>	-									
<b>Remarks</b>	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.										
<b>Side effects</b>	-									
<b>See also</b>	<code>appInfo</code>									
<b>Example</b>	<pre> uns24 SN @ bufferInfo; uns8 OSv @ bufferInfo[4]; moduleInfo(); // Now SN == module serial number // and OSv == OS version </pre>									

### appInfo

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

**SPI**
**enableSPI**

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

**disableSPI**

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

### startSPI

<b>Function</b>	Indicate data ready to Master and provide data from <code>bufferCOM</code> to Master according to Master's clock (typically on OS background).
<b>Purpose</b>	Initiate SPI packet transmission from Slave (request to Master)
<b>Syntax</b>	<code>void startSPI (length)</code>
<b>Parameters</b>	<code>uns8 length</code> : number of bytes to be sent (0 to 35)
<b>Return value</b>	–
<b>Output values</b>	SPI Status is switched to <i>SPI data ready</i> .
<b>Preconditions</b>	SPI must be enabled by the <code>enableSPI</code> function before.
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• SPI runs on OS background except of the Slow mode case (during RF receiving).</li> <li>• <code>startSPI(0)</code> is useful for recovering SPI from communication failures (e.g. the CRC mismatch).</li> <li>• See SPI Implementation in the IQRF platform [5] and example E07-SPI [10].</li> </ul>
<b>Side effects</b>	–
<b>See also</b>	<code>enableSPI</code> , <code>disableSPI</code> , <code>stopSPI</code> , <code>getStatusSPI</code>
<b>Example1</b>	<pre> // Slave -&gt; 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>
<b>Example2</b>	See <code>getStatusSPI</code>

### stopSPI

<b>Function</b>	Stop SPI communication
<b>Purpose</b>	Suspend SPI transmissions whenever it suits to Slave (e.g. not to lose previous data in <code>bufferCOM</code> )
<b>Syntax</b>	<code>void stopSPI ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	SPI Status is switched to <i>User stop</i> .
<b>Preconditions</b>	–
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• SPI transmission is stopped but SPI remains active (enabled). Communication can continue after next <code>startSPI</code>.</li> <li>• <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.</li> <li>• See SPI Implementation in the IQRF platform [5] and example E07-SPI [10].</li> </ul>
<b>Side effects</b>	Current packet is lost by both sides if SPI communication is running on background at this moment.
<b>See also</b>	<code>enableSPI</code> , <code>disableSPI</code> , <code>startSPI</code> , <code>getStatusSPI</code>
<b>Example</b>	See <code>getStatusSPI</code>

### getStatusSPI

<b>Function</b>	Update SPI flags and packet length and check whether SPI is busy
<b>Purpose</b>	Provide application program with information about current SPI status
<b>Syntax</b>	bit <code>getStatusSPI ()</code>
<b>Parameters</b>	–
<b>Return value</b>	<ul style="list-style-type: none"> <li>• 1 – SPI busy</li> <li>• 0 – SPI not busy</li> </ul>
<b>Output values</b>	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.
<b>Preconditions</b>	SPI must be enabled by <code>enableSPI</code>
<b>Remarks</b>	See SPI Implementation in the IQRF platform [5] and example E07-SPI [10].
<b>Side effects</b>	–
<b>See also</b>	<code>enableSPI</code> , <code>disableSPI</code> , <code>startSPI</code> , <code>stopSPI</code>
<b>Example</b>	<pre> // Master -&gt; 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:     PacketLength = param1; // Store packet length to user variable     stopSPI(); // Prohibit Master from transmitting     copyBufferCOM2INFO(); // until received packet is stored by the user     startSPI(0); // 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>

**RF**
**setTXpower**

<b>Function</b>	Set RF output power
<b>Purpose</b>	Change RF range
<b>Syntax</b>	<code>void setTXpower (level)</code>
<b>Parameters</b>	<code>uns8 level</code> : 1 (min.) to 7 (max.)
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	–
<b>Remarks</b>	Actual output power is non-linear and depends on TR type. See example E03–TR [10].
<b>Side effects</b>	–
<b>See also</b>	RFTXpacket
<b>Example</b>	<code>setTXpower (7); // Max. RF output power</code>

**RFTXpacket**

<b>Function</b>	Send RF packet of specified length from <code>bufferRF</code> to specified address
<b>Purpose</b>	RF transmission
<b>Syntax</b>	<code>void RFTXpacket ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• Peer-to-peer topology: <ul style="list-style-type: none"> <li>• <code>PIN = 0</code> (Peer-to-peer)</li> <li>• <code>DLEN = packet length in bytes (0 to 64)</code></li> <li>• Prepare data to send in <code>bufferRF[0]</code> to <code>bufferRF[DLEN - 1]</code> (if <code>DLEN ≠ 0</code>)</li> <li>• Set RF output power via <code>setTXpower</code></li> </ul> </li> <li>• IQMESH: <ul style="list-style-type: none"> <li>• <code>PIN = 0x80</code> (IQMESH)</li> <li>• Other network related parameters should also be specified</li> </ul> </li> </ul> See IQRF OS User's guide [1] and IQMESH specification [4].
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• Unlike SPI, RF communication does not run on OS background. This function is active on foreground until the packet is sent.</li> <li>• Duration depends on TR type, routing algorithm and packet length.</li> <li>• 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.</li> <li>• See examples E01–TX, E03–TR and E09–LINK [10].</li> </ul>
<b>Side effects</b>	<ul style="list-style-type: none"> <li>• <code>OPTION.7 = 1</code> (Port B pull-ups disabled)</li> <li>• <code>bufferRF[DLEN]</code> and <code>bufferRF[DLEN+1]</code> are destroyed</li> </ul>
<b>See also</b>	<code>RFRXpacket</code> , <code>setTXpower</code> and (in case of IQMESH) also other RF functions

<b>Example1</b>	<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>
<b>Example2</b>	<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>
<b>Example3</b>	<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

<b>Function</b>	Receive RF packet to <code>bufferRF</code> and provide related information
<b>Purpose</b>	RF receiving
<b>Syntax</b>	bit <code>RFRXpacket()</code>
<b>Parameters</b>	–
<b>Return value</b>	<ul style="list-style-type: none"> <li>• 1 – packet received</li> <li>• 0 – packet not received</li> </ul>
<b>Output values</b>	<p><code>DLEN</code> = packet length</p> <p><code>_NTWPACKET</code>: valid if <code>RFRXpacket</code> return value == 1 only:</p> <ul style="list-style-type: none"> <li>• 1 – networking packet received</li> <li>• 0 – non-networking packet received</li> </ul> <p>Other related networking information in case of IQMESH.</p>
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• Timeout in number of 10 ms ticks should be specified in <code>toutRF</code> (1 to 255)</li> <li>• Peer-to-peer topology: nothing else</li> <li>• IQMESH: network related parameters (filtering, ...) should be predefined</li> </ul> <p>See IQRF OS User's guide [1] and IQMESH specification [4].</p>
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• 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 does not terminate the reception.</li> <li>• If the packet is sent when the address (or a routing device) is not executing this function the packet is lost.</li> <li>• Peer-to-peer topology: All packets in range are received.</li> <li>• IQMESH: Device receives only packets intended for it and non-networking packets depending on filtering mode – see <code>setFilteringOn</code> and <code>setFilteringOff</code>.</li> <li>• See examples E02–RX, E03–TR, E09–LINK and E11–MEASUREMENT [10].</li> </ul>
<b>Side effects</b>	<ul style="list-style-type: none"> <li>• Update PIN before every <code>RFTXpacket</code> folowed after <code>RFRXpacket</code>.</li> <li>• SPI is switched to Slow mode during <code>RFRXpacket</code> activity.</li> <li>• Result of <code>captureTicks</code> is destroyed if <code>startCapture</code> is active on background at the same time.</li> <li>• <code>OPTION.7 = 1</code> (Port B pull-ups disabled).</li> <li>• <code>OPTION.6 = 1</code> (Interrupt on rising edge of INT pin). See PIC datasheet (Option register) [8].</li> <li>• <code>bufferRF[DLEN]</code> and <code>bufferRF[DLEN+1]</code> is destroyed.</li> </ul>
<b>See also</b>	<code>RFTXpacket</code> and (in case of IQMESH) also other RF functions
<b>Example1</b>	<pre> // Peer-to-peer topology toutRF = 10;           // RF timeout 100 ms if RFTXpacket();      // 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>
<b>Example2</b>	IQMESH: See <code>setNodeMode</code> and <code>setNetworkFilteringOn</code> .

## Networking

### setNetworkOne

<b>Function</b>	Select IQMESH networking, Network 1
<b>Purpose</b>	Switch (from IQMESH Network 2 or Peer-to-peer) to IQMESH Network 1
<b>Syntax</b>	<code>void setNetworkOne ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	For IQMESH only. Default OS mode is Peer-to-peer.
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• 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.</li> <li>• See example E11 - DATACENTER [10].</li> <li>• This settings affects both <code>RFRXpacket</code> and <code>RFTXpacket</code> features.</li> </ul>
<b>Side effects</b>	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.
<b>See also</b>	<code>setNetworkTwo</code> , <code>setNetworkingOff</code> , <code>setCoordinatorMode</code> , <code>setNodeMode</code>
<b>Example</b>	See <code>setNodeMode</code>

### setNetworkTwo

<b>Function</b>	Select IQMESH networking, Network 2
<b>Purpose</b>	Switch (from IQMESH Network 1 or Peer-to-peer) to IQMESH Network 2
<b>Syntax</b>	<code>void setNetworkTwo ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	For IQMESH only. Default OS mode is Peer-to-peer.
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• 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.</li> <li>• See example E11 - MEASUREMENT [10].</li> <li>• This settings affects <code>RFRXpacket</code> and <code>RFTXpacket</code> features.</li> </ul>
<b>Side effects</b>	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.
<b>See also</b>	<code>setNetworkOne</code> , <code>setNetworkingOff</code> , <code>setCoordinatorMode</code> , <code>setNodeMode</code>
<b>Example</b>	See <code>setNodeMode</code>

**setNetworkingOff**

<b>Function</b>	Select Peer-to-peer mode
<b>Purpose</b>	Switch from IQMESH to Peer-to-peer
<b>Syntax</b>	void <b>setNetworkingOff</b> ()
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	–
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• Default OS mode is Peer-to-peer.</li> <li>• This settings affects <code>RFRXpacket</code> and <code>RFTXpacket</code> features.</li> <li>• PIN is not affected immediately but it is cleared after subsequent <code>RFRXpacket</code> or <code>RFTXpacket</code>.</li> </ul>
<b>Side effects</b>	–
<b>See also</b>	<code>setNetworkOne</code> , <code>setNetworkTwo</code> , <code>setCoordinatorMode</code> , <code>setNodeMode</code>
<b>Example</b>	<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

<b>Function</b>	Set Coordinator mode
<b>Purpose</b>	Assign the TR module as a network Coordinator
<b>Syntax</b>	<code>void setCoordinatorMode ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	For IQMESH only
<b>Remarks</b>	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> .
<b>Side effects</b>	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.
<b>See also</b>	<code>setNetworkOne</code> , <code>setNetworkTwo</code> , <code>setNodeMode</code>
<b>Example</b>	See <code>setNodeMode</code>

### setNodeMode

<b>Function</b>	Set Node mode
<b>Purpose</b>	Assign the TR module as a Node
<b>Syntax</b>	<code>void setNodeMode ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	For IQMESH only
<b>Remarks</b>	Every TR module can work as a Coordinator or a Node. This settings affects both <code>RFRXpacket</code> and <code>RFTXpacket</code> .
<b>Side effects</b>	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.
<b>See also</b>	<code>setNetworkOne</code> , <code>setNetworkTwo</code> , <code>setNodeMode</code>
<b>Example</b>	<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

<b>Function</b>	Start filtering incoming non-networking packets and packets coming from non-current network(s).
<b>Purpose</b>	To receive packets from current network only.
<b>Syntax</b>	<code>void setNetworkFilteringOn ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	This affects the <code>RFRXpacket</code> return value.
<b>Preconditions</b>	For IQMESH only. Default OS condition is Filtering Off.
<b>Remarks</b>	–
<b>Side effects</b>	–
<b>See also</b>	<code>setNetworkFilteringOff</code> , <code>RFRXpacket</code>
<b>Example</b>	<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

<b>Function</b>	Stop filtering incoming packets from the point of view the packet is coming from.
<b>Purpose</b>	To receive all packets ( non-networking packets as well as packets from all networks).
<b>Syntax</b>	<code>void setNetworkFilteringOff ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	This affects the <code>RFRXpacket</code> return value.
<b>Preconditions</b>	For IQMESH only. Default OS condition is Filtering Off.
<b>Remarks</b>	Network 1 or 2 is automatically selected according to last received packet in this mode (except of non-networking packets).
<b>Side effects</b>	–
<b>See also</b>	<code>setNetworkFilteringOn</code> , <code>RFRXpacket</code>
<b>Example</b>	<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

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

### setLoggingOff

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



## Routing

### setRoutingOn

<b>Function</b>	Routing enabled
<b>Purpose</b>	Outgoing packets will be delivered via routing devices on background.
<b>Syntax</b>	<code>void setRoutingOn ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	For IQMESH only. Default OS condition is Routing off. <b>Not implemented</b> , reserved for future OS versions. <b>Do not use in this OS version.</b>
<b>Remarks</b>	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"> <li><code>_ROUTEF = 1</code>: routing enabled. Routing vector RTV[0 to 3] should be specified.</li> <li><code>_ROUTEF = 0</code>: routing disabled.</li> </ul>
<b>Side effects</b>	–
<b>See also</b>	<code>setRoutingOff</code>
<b>Example</b>	See RFTXpacket, example 3

### setRoutingOff

<b>Function</b>	Routing disabled
<b>Purpose</b>	Outgoing packets will not be delivered via routing devices on background.
<b>Syntax</b>	<code>void setRoutingOff ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	–
<b>Preconditions</b>	For IQMESH only. Default OS condition is Routing off. <b>Not implemented</b> , reserved for future OS versions. <b>Do not use in this OS version.</b>
<b>Remarks</b>	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"> <li><code>_ROUTEF = 1</code>: routing enabled. Routing vector RTV[0 to 3] should be specified.</li> <li><code>_ROUTEF = 0</code>: routing disabled.</li> </ul>
<b>Side effects</b>	–
<b>See also</b>	<code>setRoutingOn</code>
<b>Example</b>	See RFTXpacket, example 2



## Bonding – Node only

### bondRequest

<b>Function</b>	Ask Coordinator via RF for bonding to its network. Bond the Node in cooperation with Coordinator and record it to EEPROM.
<b>Purpose</b>	Request by the Node to be included to the network on both Coordinator's and Node's sides.
<b>Syntax</b>	bit <code>bondRequest ()</code>
<b>Parameters</b>	–
<b>Return value</b>	<ul style="list-style-type: none"> <li>• 1 – Node has been bonded</li> <li>• 0 – Node has not been bonded</li> </ul>
<b>Output values</b>	<ul style="list-style-type: none"> <li>• The <code>amIBonded</code> function starts to return value == 1 whenever is called while the Node is bonded by <code>bondRequest</code> not beeing unbonded by <code>removeBond</code> or <code>wipeBondNR</code>.</li> <li>• Coordinator is not affected at all.</li> <li>• <code>param2</code>: Node address (if successfully bonded only). Not guarranted for future OS versions.</li> </ul>
<b>Preconditions</b>	For IQMESH only. To avoid possible bonding of an unintentional Node, bonding operates with the lowest RF power (it is arranged by OS). Therefore the Node should be placed on minimal distance from Coordinator during bonding.
<b>Remarks</b>	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 an 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.
<b>Side effects</b>	<p>The following values are modified and not restored:</p> <ul style="list-style-type: none"> <li>• WDTCON</li> <li>• DLEN</li> <li>• PIN</li> <li>• toutRF</li> <li>• RF power = 1</li> <li>• OPTION.7 = 1 (Port B pull-ups disabled)</li> <li>• OPTION.6 = 1 (Interrupt on rising edge of PIC INT pin)</li> <li>• <code>bufferRF[0 to 63] = 0</code></li> <li>• <code>bufferINFO[0 to 35] = 0</code></li> <li>• Result of <code>captureTicks</code> is destroyed if <code>startCapture</code> is active on background at the same time.</li> </ul>
<b>See also</b>	<code>bondNewNode</code> , <code>amIBonded</code> , <code>removeBond</code> , <code>rebondNode</code> , <code>wipeBondNR</code> , <code>getNetworkParams</code>
<b>Example1</b>	<pre> pulsingLED();           // LED blinking indicates attempt to bond (max. 10 s) if (bondRequest()) {     stopLED();           // if successfully bonded     _RLED=1;             // LED On     waitDelay(100);     // for 1 s } stopLED(); </pre>
<b>Example2</b>	See <code>amIBonded</code>

### amIBonded

<b>Function</b>	Is the Node bonded?
<b>Purpose</b>	Test whether the Node is bonded on Node's side
<b>Syntax</b>	bit <code>amIBonded()</code>
<b>Parameters</b>	–
<b>Return value</b>	<ul style="list-style-type: none"> <li>• 1 – Node is bonded (after <code>bondRequest</code> not being unbonded by <code>removeBond</code> or <code>wipeBondNR</code>)</li> <li>• 0 – Node is not bonded: <ul style="list-style-type: none"> <li>• no <code>bondRequest</code> has ever been successfully executed</li> <li>• after <code>removeBond</code> or <code>wipeBondNR</code></li> </ul> </li> </ul>
<b>Output values</b>	–
<b>Preconditions</b>	For IQMESH only. Result is not depended on the Coordinator at all.
<b>Remarks</b>	See example E11 - MEASUREMENT [10].
<b>Side effects</b>	–
<b>See also</b>	<code>bondRequest</code> , <code>removeBond</code> , <code>wipeBondNR</code>
<b>Example</b>	<pre>while (!amIBonded()) // Request for being bonded (if not bonded yet) {     bondRequest(); // Repeatedly try to bond     clrwdt(); } // until successful setTXpower(7); // Max. RF output power for future communication</pre>

### removeBond

<b>Function</b>	Remove the Node from the network and record it to EEPROM.
<b>Purpose</b>	Exclude the Node from the network on Node's side and keep its Node number reserved for possible future rebonding.
<b>Syntax</b>	<code>void removeBond ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	<ul style="list-style-type: none"> <li>• 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>.</li> <li>• Just this value is affected but the Node keeps the Node number still stored (for possible future rebonding with the same Node number).</li> <li>• Coordinator is not affected at all.</li> </ul>
<b>Preconditions</b>	For IQMESH only. Max. 239 Nodes can be bonded to single network.
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• See example E11 - MEASUREMENT [10].</li> <li>• 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.</li> <li>• <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.</li> </ul>
<b>Side effects</b>	• 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.
<b>See also</b>	<code>bondRequest</code> , <code>bondNewNode</code> , <code>amIBonded</code> , <code>rebondNode</code> , <code>wipeBondNR</code>
<b>Example</b>	<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

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

## Bonding – Coordinator only

### bondNewNode

<b>Function</b>	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.
<b>Purpose</b>	Include a new Node to the network
<b>Syntax</b>	bit <code>bondNewNode ()</code>
<b>Parameters</b>	–
<b>Return value</b>	<ul style="list-style-type: none"> <li>• 1 – bonding successful, Node included to the list of bonded Nodes</li> <li>• 0 – bonding unsuccessful, Node not included to the list of bonded Nodes</li> </ul>
<b>Output values</b>	<ul style="list-style-type: none"> <li>• <code>param2</code>: Node number</li> <li>• The <code>isBondedNode</code> function starts to return value == 1 whenever is called while the Node is in the list of bonded Nodes.</li> </ul>
<b>Preconditions</b>	<ul style="list-style-type: none"> <li>• For IQMESH only.</li> <li>• 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.</li> <li>• To avoid possible bonding of an unintentional Node, bonding operates with the lowest RF power (it is arranged by OS). Therefore the Node should be placed on minimal distance from Coordinator during bonding.</li> </ul>
<b>Remarks</b>	<ul style="list-style-type: none"> <li>• See example E11 - DATACENTER [10].</li> <li>• If no requesting Node is detected during 10 s period this function terminates.</li> <li>• Network number is derived from Coordinator ID which ensures unique identification of various networks.</li> <li>• 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.</li> </ul>
<b>Side effects</b>	<p>The following values are modified and not restored:</p> <ul style="list-style-type: none"> <li>• WDTCON modified</li> <li>• toutRF modified</li> <li>• RF power = 1</li> <li>• OPTION.7 = 1 (Port B pull-ups disabled)</li> <li>• OPTION.6 = 1 (Interrupt on rising edge of INT pin)</li> <li>• <code>bufferRF[0 to 63] = 0</code></li> <li>• <code>bufferINFO[0 to 35] = 0</code></li> <li>• Result of <code>captureTicks</code> is destroyed if <code>startCapture</code> is active on background at the same time.</li> </ul>
<b>See also</b>	<code>bondRequest</code> , <code>removeBondedNode</code> , <code>rebondNode</code> , <code>isBondedNode</code>
<b>Example</b>	<pre> if (bondNewNode())           // Bonding successful ? {                             // Yes:     NodeNumber = param2;     ... } else {                             // No:     ...                       // Arrange necessary steps } </pre>

### isBondedNode

<b>Function</b>	Is specified Node in the list of bonded Nodes?
<b>Purpose</b>	Test whether the Node is bonded on Coordinator's side
<b>Syntax</b>	bit <code>isBondedNode (n)</code>
<b>Parameters</b>	uns8 n: Node number
<b>Return value</b>	<ul style="list-style-type: none"> <li>• 1 – Node is in the list of bonded Nodes</li> <li>• 0 – Node is not in the list of bonded Nodes</li> </ul>
<b>Output values</b>	–
<b>Preconditions</b>	For IQMESH only. The result is not affected by the Node at all.
<b>Remarks</b>	–
<b>Side effects</b>	–
<b>See also</b>	<code>bondNewNode</code> , <code>removeBondedNode</code> , <code>rebondNode</code> , <code>clearAllBonds</code>
<b>Example</b>	<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

<b>Function</b>	Remove a Node from the list of bonded Nodes by Coordinator in EEPROM
<b>Purpose</b>	Exclude the Node from the network on Coordinator's side
<b>Syntax</b>	void <code>removeBondedNode (n)</code>
<b>Parameters</b>	uns8 n: Node number
<b>Return value</b>	–
<b>Output values</b>	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.
<b>Preconditions</b>	For IQMESH only
<b>Remarks</b>	<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.
<b>Side effects</b>	
<b>See also</b>	<code>bondNewNode</code> , <code>isBondedNode</code> , <code>clearAllBonds</code> , <code>removeBond</code>
<b>Example</b>	<pre> removeBondedNode(28); // Coordinator assumes Node #28 to be                        // out of the network from now on </pre>

### rebondNode

<b>Function</b>	Put a Node back to the list of bonded Nodes by Coordinator in EEPROM
<b>Purpose</b>	Include the Node to the network again on Coordinator's side
<b>Syntax</b>	bit <code>rebondNode (n)</code>
<b>Parameters</b>	uns8 n: Node number
<b>Return value</b>	reserved for future OS versions
<b>Output values</b>	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.
<b>Preconditions</b>	For IQMESH only. Avoid rebonding a Node not being bonded ever before.
<b>Remarks</b>	<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.
<b>Side effects</b>	
<b>See also</b>	<code>bondNewNode</code> , <code>removeBondedNode</code> , <code>isBondedNode</code>
<b>Example</b>	<pre>rebondNode(28);           // Coordinator assumes Node #28 to be                            // back in the network from now on</pre>

### clearAllBonds

<b>Function</b>	Remove all Nodes from the list of bonded Nodes by Coordinator in EEPROM
<b>Purpose</b>	Excluding all Nodes from the network on Coordinator's side
<b>Syntax</b>	void <code>clearAllBonds ()</code>
<b>Parameters</b>	–
<b>Return value</b>	–
<b>Output values</b>	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.
<b>Preconditions</b>	For IQMESH only
<b>Remarks</b>	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.
<b>Side effects</b>	• <code>bufferINFO[0 to 35] = 0</code>
<b>See also</b>	<code>removeBondedNode</code>
<b>Example</b>	<pre>clearAllBonds()          // Exclude all currently bonded nodes from the network</pre>

## Documentation and Information

- 1 **IQRF OS** User's guide  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=130](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=130)
- 2 **RAM map** and **EEPROM map**, IQRF OS User's guide, Appendix  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=130](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=130)
- 3 **IQRF** home page [www.iqrf.org](http://www.iqrf.org)
- 4 **IQMESH** specification [www.iqmesh.org/iqmesh](http://www.iqmesh.org/iqmesh)
- 5 **SPI** specification  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=90](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=90)
- 6 **IQRF support** site [www.iq-esupport.com](http://www.iq-esupport.com)
- 7 **TR-xxx-21A** datasheet:  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=88](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=88)  
**TR-xxx-11A** datasheet  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=70](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=70)  
**TR-xxx-31B** datasheet:  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=109](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=109)  
**TR-xxx-32B** datasheet:  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=128](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=128)  
**TR-xxx-52B** datasheet:  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=129](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=129)
- 8 **PIC16LF88** datasheet: <http://ww1.microchip.com/downloads/en/DeviceDoc/30487c.pdf>
- 9 **PIC16LF886** datasheet: <http://ww1.microchip.com/downloads/en/DeviceDoc/30487c.pdf>
- 9 **IQRF IDE**  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=89](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=89)
- 10 **Examples** (included in the StartUp Package)  
[www.iq-esupport.com/index.php?dload=Download&\\_m=downloads&\\_a=downloadfile&downloaditemid=96](http://www.iq-esupport.com/index.php?dload=Download&_m=downloads&_a=downloadfile&downloaditemid=96)

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

## Index

amlBonded.....	33
applInfo.....	21
batteryValueOK.....	7
bondNewNode.....	35
bondRequest.....	33
captureTicks.....	10
clearAllBonds.....	36
clearBufferINFO.....	17
compareBufferINFO2RF.....	20
copyBufferCOM2INFO.....	19
copyBufferCOM2RF.....	19
copyBufferINFO2COM.....	18
copyBufferINFO2RF.....	18
copyBufferRF2COM.....	18
copyBufferRF2INFO.....	19
copyMemoryBlock.....	20
debug.....	7
disableSPI.....	22
eeReadByte.....	14
eeReadData.....	14
eeWriteByte.....	14
eeWriteData.....	15
enableSPI.....	21
getNetworkParams.....	31
getStatusSPI.....	23
getTemperature.....	8
iqrfSleep.....	6
isBondedNode.....	35
isDelay.....	11
moduleInfo.....	20
pulseOUT2.....	13
pulsingOUT2.....	12
readFromRAM.....	15
rebondNode.....	36
removeBond.....	34
removeBondedNode.....	36
reset.....	6
RFRXpacket.....	25
RFTXpacket.....	25
setCoordinatorMode.....	28
setLoggingOff.....	30
setLoggingOn.....	30
setNetworkFilteringOff.....	29
setNetworkFilteringOn.....	29
setNetworkingOff.....	27
setNetworkOne.....	27
setNetworkTwo.....	27
setNodeMode.....	28
setOffPulsingOUT2.....	12
setOnPulsingOUT2.....	12
setPIR1.....	17
setRoutingOff.....	32
setRoutingOn.....	32
setTXpower.....	24
startCapture.....	10
startDelay.....	11
startSPI.....	22
stopOUT2.....	13
stopSPI.....	23
waitDelay.....	9
waitMS.....	9
wipeBondNR.....	34
writeToRAM.....	16



---

## 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](http://www.microrisc.com)

**Partners and distribution:**

please visit [www.iqrf.org/partners](http://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)*

---

<b>Website</b>	<b><a href="http://www.iqrf.org">www.iqrf.org</a></b>
<b>E-mail</b>	<b><a href="mailto:sales@iqrf.org">sales@iqrf.org</a></b>
<b>On-line support</b>	<b><a href="http://iq-esupport.com">http://iq-esupport.com</a></b>



Simple way to smarter wireless solutions