RFduino Reference v1.0

For a general Arduino reference please see **http://arduino.cc/en/Reference/HomePage**

BLE Stack

- RFduinoBLE.begin()

This function starts the BLE Stack and begins advertising.
Example:
RFduinoBLE.begin();

- RFduinoBLE.end()

This function stops the BLE Stack and stops advertising.
Example:
RFduinoBLE.end();

- RFduinoBLE.deviceName

This variable allows you to set the BLE device name as it will appear when advertising
Example:
RFduinoBLE.deviceName = "RFduino"; //Sets the device name to RFduino

- RFduinoBLE.advertismentData

This variable allows you to set the BLE advertisment data.
Example:
RFduinoBLE.advertismentData = "Unit A"; //Will include Unit A in the advertisement packet.
Note: Advertisment length and deviceName length must be <= 18 bytes

- RFduinoBLE.advertisementInterval

This variable allows you to set the BLE advertisment interval in milliseconds.
Example:
RFduinoBLE.advertisementInterval = 100; //Sets the interval to 100ms

- RFduinoBLE.txPowerLevel

This variable allows you to set the BLE transmit power in dBm. You can select any value between -20 to +4 dBm in 4dBm increments. (ex. -20, -16, -12, -8, -4, 0, +4)
Example:
RFduinoBLE.txPowerLevel = +4; //Sets the transmit power to max +4dBm

- RFduinoBLE.send()

This function allows you to send data via BLE. RFduinoBLE.send(char data) or RFduinoBLE.send(const char *data, int len);
Example:
RFduinoBLE.send = (1); //Sends a 1
or
RFduinoBLE.send = (myarray, 5); //Sends a character array called myarray with a length of 5

- RFduinoBLE.sendByte()

This function allows you to send a Byte via BLE.
Example:
uint8_t myByte = 50;
RFduinoBLE.sendByte = (myByte); //Sends myByte

- RFduinoBLE.sendInt()

This function allows you to send a INT via BLE.
Example:
int myByte = 5000;
RFduinoBLE.sendInt = (myByte); //Sends myByte

- RFduinoBLE.sendFloat()

This function allows you to send a float via BLE.
Example:
float myNumber = 16.49;
RFduinoBLE.sendFloat = (myNumber); //Sends myNumber

- RFduinoBLE.radioActive

This function allows you to check whether the radio is active or not. Since the radio take priority over resources when it is active, this is very useful in timing critical applications, where you can wait until the radio is off to run your critical code.
Example:

```
// Wait while the Radio is active
while (RFduinoBLE.radioActive)
;
// Timing Critical Code goes here
```

BLE Callbacks

- RFduinoBLE_onAdvertisement()

This function allows you to run a piece of code everytime the radio advertises.
Example:

```
void RFduinoBLE_onAdvertisement(bool start){
// Insert code here
}
```

- RFduinoBLE_onConnect()

This function allows you to run a piece of code everytime you connect to the radio.
Example:

```
void RFduinoBLE_onConnect(){
// Insert code
}
```

- RFduinoBLE_onDisconnect()

This function allows you to run a piece of code everytime you disconnect to the radio.
Example:

```
void RFduinoBLE_onDisconnect(){
// Insert code here
}
```

- RFduinoBLE_onReceive()

This function returns data from the radio.
Example:

```
void RFduinoBLE_onReceive(char *data, int len){
uint8_t myByte = data[0]; // store first char in array to myByte
Serial.println(myByte); // print myByte via serial
}
```

- RFduinoBLE_onRSSI()

This function returns the dBm signal strength after connecting
Example:

```
void RFduinoBLE_onRSSI(int rssi){
Serial.println(rssi); // print rssi value via serial
}
```

Sleep and Wake

- RFduino_ULPDelay()

This function sets the module in an ultra low power delay for the amount of time specified.
RFduino_ULPDelay(uint64_t ms);
Example:

```
RFduino_ULPDelay(350); // 350 milliseconds
RFduino_ULPDelay(SECONDS(350)); //350 seconds
RFduino_ULPDelay(MINUTES(350)); //350 minutes
RFduino_ULPDelay(HOURS(10)); // 10 hours
RFduino_ULPDelay(DAY(3)); // 3 days
RFduino_ULPDelay(INFINITE); // Stay in ultra low power mode until interrupt from the BLE or pinWake()
```

- RFduino_pinWake()

This function configures a pin to wake up the device
Example:

```
pinMode(5, INPUT); // set pin 5 to input
RFduino_pinWake(5, HIGH); // configures pin 5 to wake up device on a high signal
```

- RFduino_pinWoke()

This function allows you to test whether a pin caused a wakeup
Example:

```
RFduino_ULPDelay(INFINITE); // stay in ULP forever
if (RFduino_pinWoke(5))
//do something here if pin 5 caused us to wake up
RFduino_resetPinWake(5); // reset state of pin that caused wakeup
```

Note: You must resetPinWake otherwise you will be stuck in the pinWoke loop.

- RFduino_resetPinWake()

This function resets the state of apin that caused a wakeup. You must reset this after using a pinWoke function otherwise you will be stuck in your pinWoke loop.
Example:

```
if (RFduino_pinWoke(5))
//do something here if pin 5 caused us to wake up
RFduino_resetPinWake(5); // reset state of pin that caused wakeup
```

- RFduino_pinWakeCallback()

This function configures a pin to wake the device and exectue a callback. RFduino_pinWakeCallback(uint32_t ulPin, uint32_t dwWake, pin_callback_t callback) ;
Example:

```
pinMode(6, INPUT); // set pin 6 to input
RFduino_pinWakeCallback(6, HIGH, myPinCallback); // configure pin 6 to wakeup the device and run function "myPinCallback"
```

- RFduino_systemReset()

This function resets the system.
Example:
RFduino_systemReset();

- RFduino_systemOff()

This function turns the system off into an ultra low power state where it can be waken up via a pinWake.
Example:
RFduino_systemOff();

Misc

- RFduino_temperature()

This function returns a sample from the on-chip temperature sensor. RFduino_temperature(int scale)
Example:
float temp = RFduino_temperature(CELSIUS); // returns temperature in Celsius and stores in float temp
or
float temp = RFduino_temperature(FAHRENHEIT); // returns temperature in Celsius and stores in float temp