MBWBLUE Function DLL

Manual (English)

11/2017

Michael Rac GmbH
Am Hirtenfeld 51
91522 Ansbach
GERMANY

Email: mrg@michaelrac.com

Michael Rac GmbH
Am Hirtenfeld 51
91522 Ansbach
ALLEMAGNE

Courriel: mrg@michaelrac.com

Michael Rac GmbH
Am Hirtenfeld 51
91522 Ansbach
DEUTSCHLAND

Email: mrg@michaelrac.com

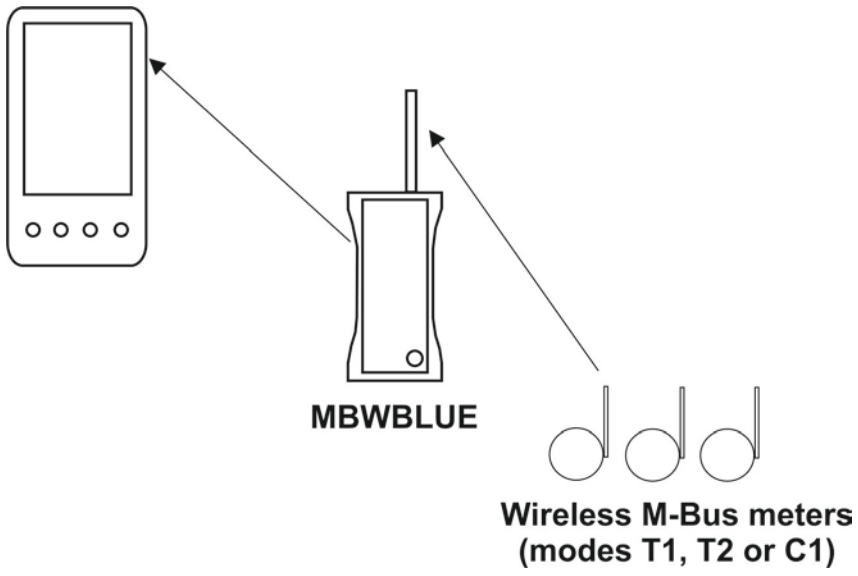
Table of Contents

MBWBLUE Function DLL

Introduction

The MBWBLUE Function DLL is a software module for reading and interpreting the data created by the MBWBLUE device. Its intention is to guide software developers who want to integrate the MBWBLUE data readings in their own software. It is assumed that the reader is familiar with the MBWBLUE device and its principle of operation.

The MBWBLUE is a radio receiver for unidirectional M-Bus Mode T1 consumption meters (electricity, gas, water, heat and others). The received radio telegrams of the consumption meters are stored into an internal memory and are immediately transmitted to a handheld computer using its Bluetooth interface, if a Bluetooth connection is established. If there is no Bluetooth connection the MBWBLUE collects all radio telegrams (up to 2000) and transmits them at once as soon as the Bluetooth connection is reestablished.



This DLL is used to read and interpret the radio telegrams received by the MBWBLUE.

System Requirements

Before installing the MBWBLUE Function DLL, please check if your PC complies with the minimum requirements:

- Windows 7 or 8 operating system (updated to the latest version)
- 1 GHz processor
- 1 GB memory
- 20 MB free hard disk space
- Bluetooth interface

Installation

The installation file **MBWBLUE_FunctionDLL_Setup.exe** has to be started on your PC. The standard installation path:

C:\Program Files\MBWBLUE

should be used, if possible.

If there is already a former version of the software installed, you have to remove this version prior to installing the current version.

Distribution

The file containing the MBWBLUE functions is called **MBT1ReceiverLib.dll** and can be found under **C:\Program Files\MBWBLUE** (if you have chosen this installation directory).
For distributing this file with your software it is only necessary to copy this file and the file “ftd2xx.dll” to the target system and register **MBT1ReceiverLib.dll** with the Windows operating system:

regsvr32.exe "c:\program files\MBWBLUE\MBT1ReceiverLib.dll"

You have to replace the directory name with the directory of your software installation.

COM Module Properties

The interface object to access in the COM (component object model) module of MBT1ReceiverLib.dll is called:

PC Version:
MBT1ReceiverLib.MBT1Receiver.1

IID: **9DF0B13A-8587-4C44-B584-BE209D51D647**

CLSID: **D4B4EC90-052B-422C-B8D3-F36197C7FC23**

Mobile Version:
MBT1ReceiverLibM.MBT1Receiver.1

IID: **FC4A0C4F-5715-4D3A-A272-E9E65ED0D5A7**

CLSID: **5D8DA86A-A188-4726-8C32-882E357188C4**

COM Interface

An application software accesses the methods and properties of the MBWBLUE Function DLL by its COM (component object model) interface. There are read-only and read / write properties of two types:

long 32 bit signed integer value
BSTR string (used for COM)

Usually the software development system is automatically converting the two different types to their respective version for your system (e.g. int and CString, integer and string …).

Floating point values and date / time values are returned as strings in the following form:

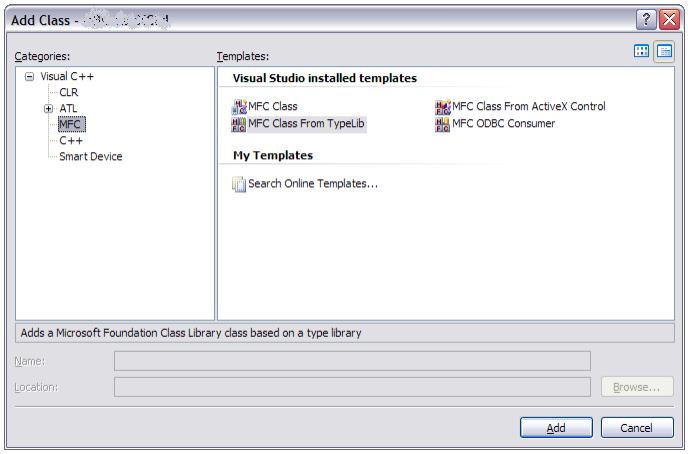
floating point
"3.456" or "2.34E-2" (BSTR)
date / time
"2006-01-01" or "2006-01-01 00:00" or "2006-01-01 00:00:00"
YYYY-MM-DD or YYYY-MM-DD HH:MM or YYYY-MM-DD HH:MM:SS

To integrate a COM module into your application software you have to check the manual of your software development system (check for integration of COM or OLE).

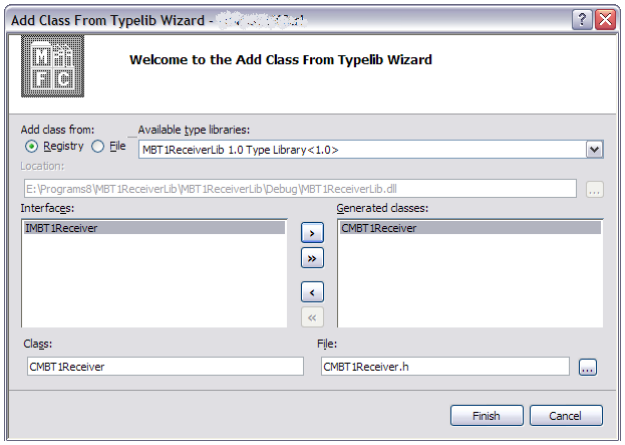
For **Visual Basic and VBA (Visual Basic for Application)** it can be done very easily (remember to have the **MBT1ReceiverLib.dll** registered on your system):

```
Sub MBWBLUEDLLTest()
    Set MBWBLUE = CreateObject("MBT1ReceiverLib.MBT1Receiver.1")
                                                    'invoke MBWBLUE Function DLL
'*************************************************************************************************
    MBWBLUE.AddressInterpretationSetting = 0        'how to decode the 6 bytes address
                                                    'field of the telegram (not
                                                    'standardized)
                                                    '0 = auto
                                                    '1 = address-generation-medium
                                                    '2 = generation-medium-address
                                                    '3 = 6 bytes (LSB first)
                                                    '4 = SPDE [CNNCCNNNNNN]
                                                    '5 = 6 bytes hexadecimal (LSB
                                                    'first)
    MBWBLUE.LanguageSetting = 1                     'language setting
                                                    '0 = English
                                                    '1 = German
                                                    '2 = French
    MBWBLUE.DecimalSetting = 1                      'decimal setting
                                                    '0 = decimal point, e.g. 0.123
                                                    '1 = decimal comma, e.g. 0,123
'*************************************************************************************************
End Sub
```

For **Visual C++ (Visual Studio 2005 / 2008)** you first have to add a class to your project, chose **Project -> Add Class**:



and select **MFC Class From TypeLib**.



Select **MBT1ReceiverLib 1.0 Type Library** from the list of registered modules and generate the class **CMBT1Receiver** from **IMBT1Receiver**.

- This will create the wrapper class for the MBT1ReceiverLib. You have to include the function header file **CMBT1Receiver.h** in your project.
- Check if in your stdafx.h the following lines are present:

```
#include <afxdisp.h>        // MFC OLE automation classes
#include <objbase.h>        // DCOM support
```

- Use the code frame below to invoke (and remove afterwards the MBWBLUE Function DLL:

```
#include "CMBT1Receiver.h"

CMBT1Receiver  MBWBLUE;

const IID IID_MBT1Mod     =
{0x9DF0B13A,0x8587,0x4C44,{0xB5,0x84,0xBE,0x20,0x9D,0x51,0xD6,0x47}};

const CLSID CLSID_MBT1Mod =
{0xD4B4EC90,0x052B,0x422C,{0xB8,0xD3,0xF3,0x61,0x97,0xC7,0xFC,0x23}};

HRESULT       hr;
IID           ActIID;
CLSID         ActCLSID;
MULTI_QI      mqi;

ActIID        =IID_MBT1Mod;
ActCLSID      =CLSID_MBT1Mod;

MBWBLUE=NULL;
mqi.pIID=&ActIID;
mqi.pItf=NULL;
mqi.hr=NULL;
hr = CoCreateInstanceEx(ActCLSID, NULL, CLSCTX_SERVER, NULL, 1, &mqi);

if (FAILED(hr))
{
       AfxMessageBox("Unable to load MBWBLUE Function DLL!");
       MBWBLUE =NULL;
}
else MBWBLUE.AttachDispatch((IDispatch*)mqi.pItf);

//************** You may now use the methods and properties of the software module,
MBWBLUE.put_AddressInterpretationSetting(0)              'how to decode the 6 bytes address
                                                         'field of the telegram (not
                                                         'standardized)
                                                         '0 = auto
                                                         '1 = address-generation-medium
                                                         '2 = generation-medium-address
                                                         '3 = 6 bytes (LSB first)
                                                         '4 = SPDE [CNNCCNNNNNN]
                                                         '5 = 6 bytes hexadecimal (LSB
                                                         'first)
MBWBLUE.put_LanguageSetting(1)                           'language setting
                                                         '0 = English
                                                         '1 = German
                                                         '2 = French
MBWBLUE.put_DecimalSetting(1)                            'decimal setting
                                                         '0 = decimal point, e.g. 0.123
                                                         '1 = decimal comma, e.g. 0,123

//************** If you are finished using the software module (usually on exiting your
//************** application) release the module and clean up
if (MBWBLUE!=NULL) MBWBLUE.ReleaseDispatch();
```

Methods and Properties

This chapter contains the methods and properties of the MBWBLUE Function DLL in alphabetical order.

AddressInterpretationSetting

[property, long, read / write]

The radio telegram address interpretation is defined differently for different manufacturer. Usually the Function DLL tries to correctly interpret the address field of a radio telegram, however, it is possible that this is not working correctly for some manufacturer.

Parameter:	none
Values:	0 = auto
	1 = address-generation-medium
	2 = generation-medium-address
	3 = 6 bytes (LSB first)
	4 = SPDE [CNNCCNNNNNN]
	5 = 6 bytes hexadecimal (LSB first)
Default Value:	0 = auto

BlueBatteryLowWarn

[property, long, read only]

Returns the battery low indicator of the device. This value is refreshed every time a radio telegram is received and read (during radio reading).

Parameter:	none	
Values:	0	battery is still OK
	else	battery low warning
Default Value:	0	battery is still OK

Note: This property is only available for DLL versions 1.38 and higher and device firmware versions 1.007 and higher.

BlueExtSerSendData

[method]

Switches the external serial interface of the MBWBLUE on using the given parameters and sends data over the serial interface. The communication has finished if the CommunicationThreadRuns property is reset to zero (polling). After the data transmission has finished the interface is automatically switched off.

Parameter: Baud rate:

300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800

Parity:

0: 8 bit data no parity
1: 8 bit data + parity odd
2: 8 bit data + parity even

Timeout:
in 0.35 seconds, default = 4 (1.4 seconds)
This is the timeout for waiting for an answer from an external device connected to the serial interface of the MBWBLUE.

Data:
Data string to send in HEXASCII format (max. 245 bytes)
e.g. 0x10, 0x40, 0xFE, 0x3E, 0x16
must be send as "1040FE3E16"

BlueExtSerSendDataWU

[method]
(MBWBLUE firmware version 1.010 and higher only)

Switches the external serial interface of the MBWBLUE on using the given parameters and sends a wake-up command and afterwards data over the serial interface. The communication has finished if the CommunicationThreadRuns property is reset to zero (polling). After the data transmission has finished the interface is automatically switched off.
This version of the command includes the possibility to send a wake-up sequence over the serial interface (e.g. optical interface 1010101010 sequence). To do so the parameter WUByte must be filled with the appropriate byte (e.g. 0x55) and the parameter WURepetitions with the appropriate number of repetitions of this byte to form a bit sequence of a defined length at the selected baud rate. The WUByte is always sent without parity bit using the selected baud rate. Between the wake-up sequence and the data a pause of 50 ms is automatically introduced.

Parameter:

Baud rate:
300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800

Parity:
0: 8 bit data no parity
1: 8 bit data + parity odd
2: 8 bit data + parity even

Timeout:
in 0.35 seconds, default = 6 (2.1 seconds)
This is the timeout for waiting for an answer from an external device connected to the serial interface of the MBWBLUE.

WUByte:
Byte to be used as wake-up command, usually 0x55 (010101010101 sequence) or 0x00 (1000000000 sequence)

WURepetitions:
Number of repetitions of the wake-up byte

Data:
Data string to send in HEXASCII format (max. 245 bytes)
e.g. 0x10, 0x40, 0xFE, 0x3E, 0x16
must be send as "1040FE3E16"

Examples for wake-up sequences (0x55 = 85):

2.2 seconds 0101010101... sequence according to EN1434-3:

300 baud: WUByte = 85, WURepetitions = 66
2400 baud: WUByte = 85, WURepetitions = 528
9600 baud: WUByte = 85, WURepetitions = 2112

BlueExtSerRecData

[property, BSTR, read only]

Returns the data received from the external serial interface, if there are any. Before invoking this property the user has to wait until CommunicationThreadRuns is reset to zero (polling).

Parameter:	none	
Values:	""	no data received
	else	Data received over the serial interface
Default Value:	""	no data received

CommunicationThreadBreak

[property, long, read / write]

Interrupts the currently running data reading or serial interface communication.

Parameter:	none
Values:	0 = no break
	1 = break data reading / serial interface communication
Default Value:	0 = no break

CommunicationThreadRuns

[property, long, read only]

Indicates whether the data reading is still running or if it is stopped.

Parameter:	none
Values:	0 = no data reading / serial interface communication running
	1 = data reading / serial interface communication running
Default Value:	0 = no data reading / serial interface communication running

DecimalSetting

[property, long, read / write]

This property gives the decimal divider, whether it is a point or a comma.

Parameter:	none	
Values:	0 = decimal point, e.g. 0.123	
	1 = decimal comma, e.g. 0,123	
Default Value:	0 = decimal point, e.g. 0.123	

FirmwareVersion

[property, BSTR, read only]

After having invoked the ReadParameter method, the FirmwareVersion can be read if the CommunicationThreadRuns property is zero again (communication has finished).

Parameter:	none	
Values:	""	invalid
	"1.005"	current firmware version
Default Value:	""	invalid

LanguageSetting

[property, long, read / write]

This property sets the language used by the Function DLL

Parameter:	none
Values:	0 = English
	1 = German
	2 = French
Default Value:	0 = English

NextRadioTelegram

[property, BSTR, read only]

If the MBWBLUE radio telegram reading is running (StartRadioReading), this property returns the next telegram received, if there is any.

Parameter:	none	
Values:	"FF"	no radio telegram available
	else	received radio telegram, use RADExtractDecipherValid to decipher and decode the radio telegram
Default Value:	"FF"	no radio telegram available

RADCIField

[property, long, read only]

Returns the CI field of the current radio telegram
Parameter: none
Values: Radio telegram CI field (e.g. “7A”)
Default Value: "" empty

RADDatarecordDIF

[property, BSTR, read only]
(long DRNumber)

Returns the DIF of data record [DRNumber] of the current radio telegram in hexadecimal form.
Parameter: DRNumber [1… RADNumberOfDatarecords]
Values: DIF field of the datarecord (e.g. “07”)
Default Value: "" empty

RADDatarecordUnit

[property, BSTR, read only]
(long DRNumber)

Returns the physical unit of data record [DRNumber] of the current radio telegram
Parameter: DRNumber [1… RADNumberOfDatarecords]
Values: Physical unit of the datarecord (e.g. “m3”)
Default Value: "" empty

RADDatarecordValue

[property, BSTR, read only]
(long DRNumber)

Returns the value of data record [DRNumber] of the current radio telegram
Parameter: DRNumber [1… RADNumberOfDatarecords]
Values: Value of the datarecord (e.g. “123,232” or “2009-01-01”)
Default Value: "" empty

Note: The value may be a value or a date / time expression. Depending on the language setting the date / time expressions are presented as shown beneath:

English: "2009-01-01" / "2009-01-01 01:23"
German: "01.01.2009" / "01.01.2009 01:23"
French: "01/01/2009" / "01/01/2009 01h23"

RADDatarecordVIF

[property, BSTR, read only]
(long DRNumber)

Returns the VIF of data record [DRNumber] of the current radio telegram in hexadecimal form.
Parameter: DRNumber [1… RADNumberOfDatarecords]
Values: VIF field of the datarecord (e.g. “16”)
Default Value: "" empty

RADDeviceAddress

[property, BSTR, read only]

Returns the address of the current radio telegram.
Parameter: none
Values: Address of the current radio telegram
Default Value: "" empty

RADExtractDecipherValid

[property, long, read only]
(BSTR Radio telegram string from NextRadioTelegram)

Deciphers, decodes and interprets a radio telegram received by the MBWBLUE. This is usually the telegram string returned by NextRadioTelegram. If the radio telegram is valid you may use any of the other RAD methods and properties to get meter data (meter address, count, etc.).
Parameter: [Radio telegram string from NextRadioTelegram]
Values: 0 radio telegram invalid
1 radio telegram valid

RADExtractRecTime

[property, BSTR, read only]
(BSTR Radio telegram string from NextRadioTelegram)

Returns the receiving time of the radio telegram string (usually returned by NextRadioTelegram).
Parameter: [Radio telegram string from NextRadioTelegram]
Values: [RecTime] receiving time
English: “2009-01-01 01:23:00"
German: "01.01.2009 01:23:00"
French: "01/01/2009 01:23:00"

RADExtractSignalStrength

[property, long, read only]
(BSTR Radio telegram string from NextRadioTelegram)

Returns the RSSI (received signal strength indicator) from the radio telegram string (usually returned by NextRadioTelegram).

Parameter:	[Radio telegram string from NextRadioTelegram]	
Values:	[RSSI]	received signal strength in %; for signal strength in dBm subtract -98 dBm

RADGeneration

[property, BSTR, read only]
Returns the generation of the current radio telegram.

Parameter:	none
Values:	Generation of the current radio telegram (e.g. “87”)
Default Value:	"" empty

RadioPasskey

[property, BSTR, read / write]
(long RadioPassNumber)
This property contains the radio passkeys for deciphering DES64 enciphered radio telegrams. Up to 3 passkeys can be programmed which are tried out one after another.

Parameter:	RadioPassNumber [1..3]
Values:	Radio passkey 64 bit, (e.g. "FFFFFFFFFFFFFFFF")
Default Value:	"" empty

RadioPasskey128

[property, BSTR, read / write]
(long RadioPassNumber)
This property contains the radio passkeys for deciphering AES128 enciphered radio telegrams. Up to 3 passkeys can be programmed which are tried out one after another.

Parameter:	RadioPassNumber [1..3]
Values:	Radio passkey 128 bit, (e.g. "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF")
Default Value:	"" empty

RADManufacturer

[property, BSTR, read only]

Returns the manufacturer code of the current radio telegram.
Parameter: none
Values: Manufacturer code of the current radio telegram
Default Value: "" empty

RADMedium

[property, BSTR, read only]

Returns the medium code of the current radio telegram.
Parameter: none
Values: Medium code of the current radio telegram (e.g. "07")
Default Value: "" empty

RADNumberOfDatarecords

[property, long, read only]

Returns the number of datarecords within the current radio telegram.
Parameter: none
Values: Number of datarecords
Default Value: 0

RADSendRadioTelegram

(long RMode, long RSet, long RSpeed, BSTR RAddr, long RTelLen, BSTR RTelg) [method]

(only for bidirectional devices with firmware version 2.xx)

Prepares a radio telegram to be sent to a specific radio device. This command stores the given radio telegram to an internal buffer. As soon as a radio telegram from the device with the given address is received, this radio telegram is sent.
Parameter:

RMode:	Radio mode:	0x00	= wireless M-Bus Mode T2
		else	= invalid
RSet:	Options:	0x01	= transmit radio telegram once if a telegram from the given address has been received
		0x02	= transmit radio telegram always if a telegram from the given address has been received

0x04 = transmit once if any telegram is received
0x08 = transmit always if any telegram is received
0x10 = transmit once immediately
else = invalid

If you have chosen one of the options which transmits if any telegram is received then the address field of the telegram to send (byte 2 to byte 7) is automatically replaced by the address field of the received telegram.

RSpeed — Transmission speed (chiprate) e.g. 32768 chip/s

RAddr — The address of the radio device to send the radio telegram to
e.g.: " 4823785634120102" contains
the manufacturer code "4823" = "RAC"
the serial number "78563412" = "12345678"
the version number "01"
the device type "02" = electricity

RTelLen — The number of bytes to send (radio telegram length)

RTelg — The radio telegram to send in hexadecimal format (e.g. "19444823..."). It must contain all CRC bytes but neither preamble nor synchronization word. The radio telegram is automatically transferred to Manchester code.

RADSendRadioTelegram2

(long RIndex, long RNumber, long RMode, long RDelay, BSTR RAddr, long RTelLen, BSTR RTelg)
[method]

(only for bidirectional devices with firmware version 3.17 and up)

Prepares a radio telegram to be sent to a specific radio device. This command stores the given radio telegram to an internal buffer. As soon as a radio telegram from the device with the given address is received, this radio telegram is sent. A maximum of 5 different radio telegrams for 5 different radio meters may be prepared (RIndex=1, 2, 3, 4, 5).

Parameter:

RIndex: Index of buffer:
1, 2, 3, 4, 5
RIndex < 1 → RIndex=1
RIndex > 5 → RIndex=5

RNumber: Number of emissions
How often the radio telegram is to be sent
0x00 = send always if meter is received
0x01 … 0xFF = send 1 to 255 times

RMode: Radio mode:
0x00 = auto, according to received telegram (T2, C2 or S2)
bidirectional bit in config is ignored
0x10 = auto, according to received telegram (T2, C2 or S2)
bidirectional bit in config is checked

RDelay :	Delay	2..254 ms delay between receiving and sending 255 = automatic setting: S2 = 10 ms T2 = 2 ms C2 = 100 ms (delay bit = 0) C2 = 1000 ms (delay bit =1)
RAddr	The address of the radio device to send the radio telegram to e.g.: " 4823785634120102" contains the manufacturer code "4823" = "RAC" the serial number "78563412" = "12345678" the version number "01" the device type "02" = electricity	
RTelLen	The number of bytes to send (radio telegram length)	
RTelg	The radio telegram to send in hexadecimal format (e.g. "19444823..."). It must contain all CRC bytes but neither preamble nor synchronization word. For S2 and T2 the radio telegram is automatically transferred to Manchester code.	

RADReadSendRadioTelegram2Buffer

(long RIndex)
[method]

(only for bidirectional devices with firmware version 3.17 and up)

Reads out the content of the send radio telegram buffers of the device. The communication has finished if the CommunicationThreadRuns property is reset to zero (polling). After CommunicationThreadRuns is zero, it is possible to read the configured parameters (see also RADSendRadioTelegram2):

RADSendRadioTelegram2Index	[property, long, read only]	RIndex
RADSendRadioTelegram2Number	[property, long, read only]	RNumber
RADSendRadioTelegram2Counter	[property, long, read only]	Counter on how often the radio telegram has already been sent
RADSendRadioTelegram2Mode	[property, long, read only]	RMode
RADSendRadioTelegram2Delay	[property, long, read only]	RDelay
RADSendRadioTelegram2TelgAddr	[property, BSTR, read only]	RAddr
RADSendRadioTelegram2TelgLen	[property, long, read only]	RTelLen
RADSendRadioTelegram2TelgBuf	[property, BSTR, read only]	RTelg

Shall not be called if StartRadioReading has been invoked.

RADSignature

[property, BSTR, read only]

Returns the signature of the current radio telegram

Parameter:	none
Values:	Signature field of the current radio telegram (e.g. "0000")
Default Value:	"" empty

RADStatus

[property, BSTR, read only]

Returns the status of the current radio telegram

Parameter:	none
Values:	Status field of the current radio telegram (e.g. "00")
Default Value:	"" empty

RADTransCount

[property, BSTR, read only]

Returns the transmission counter of the current radio telegram

Parameter:	none
Values:	Transmission counter field of the current radio telegram (e.g. "1F")
Default Value:	"" empty

ReadParameter

[method]

Starts reading the firmware version and the serial number of the device. The reading has finished if the CommunicationThreadRuns property is reset to zero (polling). After the reading has finished the properties FirmwareVersion and SerialNumber can be used.

Parameter:	none
Values:	none
Default Value:	none

SerialNumber

[property, BSTR, read only]

After having invoked the ReadParameter method, the SerialNumber can be read if the CommunicationThreadRuns property is zero again (communication has finished).

Parameter:	none	
Values:	""	invalid
	"MBWBLUE ABB0102"	serial number of the device
Default Value:	""	invalid

StartRadioReading

[method]

Starts reading the incoming radio telegrams from the MBWBLUE. This process continues until it is stopped by setting CommunicationThreadBreak to 1. During the time the radio telegram reading is enabled, the user may call NextRadioTelegram for retrieving the radio telegrams received by the MBWBLUE.

Parameter:	none
Values:	none
Default Value:	none

Example Code for MBWBLUE reading (VBA)

```
Sub MBWBLUEReading()
'*************************************************************************************************************
   MBT1COM = 12                        'MBWBLUE device is given the virtual COM port number COM12
                                       'change this value according to your COM port
'*************************************************************************************************************
   For RowCounter = 1 To 50                        'clear all cells
    For LineCounter = 1 To 200
      Cells(LineCounter, RowCounter) = ""
    Next LineCounter
   Next RowCounter
   Cells(1, 4) = "Running"

'*************************************************************************************************************
   Set MBWBLUE = CreateObject("MBT1ReceiverLib.MBT1Receiver.1") 'invoke MBT1ReceiverLib software module
'*************************************************************************************************************
                                                        'general settings
   MBWBLUE.RadioPasskey(1) = "FFFFFFFFFFFFFFFF"         'set 64 bit radio deciphering pass key 1 (if available)
   MBWBLUE.RadioPasskey(2) = "FFFFFFFFFFFFFFFF"         'set 64 bit radio deciphering pass key 2 (if available)
   MBWBLUE.RadioPasskey(3) = "FFFFFFFFFFFFFFFF"         'set 64 bit radio deciphering pass key 3 (if available)
   MBWBLUE.RadioPasskey128(1) = "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF" 'Set 128 bit radio deciphering pass key 1 (if available)
   MBWBLUE.RadioPasskey128(2) = "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF" 'Set 128 bit radio deciphering pass key 2 (if available)
   MBWBLUE.RadioPasskey128(3) = "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF" 'Set 128 bit radio deciphering pass key 3 (if available)

'*************************************************************************************************************
   MBWBLUE.CurrentCOMPort = MBT1COM                    'set the virtual COM port for the MBWBLUE
                                                       'check the Bluetooth manager for the correct setting
   MBWBLUE.ReadParameter                               'start with reading the parameter
   Do
      DoEvents
   Loop While MBWBLUE.CommunicationThreadRuns <> 0
   Cells(1, 1) = "MBWBLUE"
   Cells(1, 2) = MBWBLUE.SerialNumber                  'print out serial number of MBWBLUE device
   Cells(2, 1) = "Firmware"
   Cells(2, 2) = MBWBLUE.FirmwareVersion               'print out firmware version of MBWBLUE device
   If MBWBLUE.SerialNumber = "" Then                   'if the serial number is empty there is no device connected
      Cells(3, 1) = "No MBT1Device connected"
      GoTo MBWBLUEReadingEnd
   End If
'*************************************************************************************************************

   Cells(4, 1) = "Reception Time"
   Cells(4, 2) = "Manufacturer"
   Cells(4, 3) = "Address"
   Cells(4, 4) = "Signal Strengh [%]"
   Cells(4, 5) = "Generation"
   Cells(4, 6) = "Medium"
   Cells(4, 7) = "CI Field"
   Cells(4, 8) = "Transmission Count"
   Cells(4, 9) = "Status"
   Cells(4, 10) = "Signature"
   For Counter = 1 To 15
      CellStr = "Value "
      CellStr = CellStr + Str(Counter)
      Cells(4, ((Counter * 2) + 9)) = CellStr
      CellStr = "Unit "
      CellStr = CellStr + Str(Counter)
      Cells(4, ((Counter * 2) + 10)) = CellStr
   Next Counter

   TelegramRow = 5
   StartTime = Now
   MBWBLUE.StartRadioReading                   'start radio telegram reading

'*************************************************************************************************************
   Do                                                  'read out the MBWBLUE device for 15 seconds and display all values
    DoEvents
    TelegramStr = MBWBLUE.NextRadioTelegram
    If Left(TelegramStr, 2) <> "FF" Then               'if the telegram string contains FF there is no telegram available
      TelValuesValid = MBWBLUE.RADExtractDecipherValid(TelegramStr)
                                                       'get the deciphering successful flag
```

```
      MBWBLUE.TelegramInterpret Mid(TelegramStr, 17, 500), TelValuesValid
                                                    'interpret the telegram and its meter counts
      Cells(TelegramRow, 1) = MBWBLUE.RADExtractRecTime(TelegramStr)
      Cells(TelegramRow, 2) = MBWBLUE.RADManufacturer
      Cells(TelegramRow, 3) = MBWBLUE.RADDeviceAddress
      Cells(TelegramRow, 4) = MBWBLUE.RADExtractSignalStrength(TelegramStr)
      Cells(TelegramRow, 5) = MBWBLUE.RADGeneration
      Cells(TelegramRow, 6) = MBWBLUE.RADMedium
      Cells(TelegramRow, 7) = MBWBLUE.RADCIField
      Cells(TelegramRow, 8) = MBWBLUE.RADTransCount
      Cells(TelegramRow, 9) = MBWBLUE.RADStatus
      Cells(TelegramRow, 10) = MBWBLUE.RADSignature 'read the different values of the telegram header

      NumberOfValues = MBWBLUE.RADNumberOfDatarecords  'read the different meter counts
      For Counter = 1 To NumberOfValues
         Cells(TelegramRow, ((Counter * 2) + 9)) = MBWBLUE.RADDatarecordValue(Counter)
         Cells(TelegramRow, ((Counter * 2) + 10)) = MBWBLUE.RADDatarecordUnit(Counter)
      Next Counter
      TelegramRow = TelegramRow + 1
    End If
  Loop While Now < StartTime + TimeValue("0:00:15")        'read the MBWBLUE device for 15 seconds
'*************************************************************************************************************************

  MBWBLUE.CommunicationThreadBreak = 1                  'Stop radio telegram reading

MBWBLUEReadingEnd:
  Cells(1, 4) = "Stopped"
End Sub
```

Example Code for using the serial interface (VBA)

```
Sub MBWBLUExtSerial()
'*************************************************************************************************************
   MBT1COM = 12                    'MBWBLUE device is given the virtual COM port number COM12
                                   'change this value according to your COM port

'*************************************************************************************************************
   Set MBWBLUE = CreateObject("MBT1ReceiverLib.MBT1Receiver.1") 'invoke MBT1ReceiverLib software module
'*************************************************************************************************************
   MBWBLUE.CurrentCOMPort = MBT1COM                'set the virtual COM port for the MBWBLUE
                                                     'check the Bluetooth manager for the correct setting
   MBWBLUE.ReadParameter                           'start with reading the parameter
   Do
      DoEvents
   Loop While MBWBLUE.CommunicationThreadRuns <> 0
   Cells(1, 1) = "MBWBLUE"
   Cells(1, 2) = MBWBLUE.SerialNumber               'print out serial number of MBWBLUE device
   Cells(2, 1) = "Firmware"
   Cells(2, 2) = MBWBLUE.FirmwareVersion            'print out firmware version of MBWBLUE device
   If MBWBLUE.SerialNumber = "" Then                'if the serial number is empty there is no device connected
      Cells(3, 1) = "No MBT1Device connected"
      GoTo MBWBLUEReadingEnd
   End If
'*************************************************************************************************************
   SRequest = "1040FE3E16"
   MBWBLUE.BlueExtSerSendData 3, 2, 4, SRequest     'Initialize external serial interface and send a request command
   Do
      DoEvents
   Loop While MBWBLUE.CommunicationThreadRuns <> 0  'Wait until the external serial interface request has finished

   Cells(4, 1) = MBWBLUE.BlueExtSerRecData          'Print out received data

MBWBLUEReadingEnd:
   Cells(1, 4) = "Stopped"
End Sub
```