Wireless Communications Library

MICRO-RM2.4 Radio Module

MicroRidge

Measurement Collection Specialists
Connect Any Gage into Any Software

MICRO-RM2.4 Communications Library

Created: Monday, August 28, 2017 at 12:23 PM in Sunriver, Oregon.

This Document has been formatted for double-sided printing.

Table of Contents

1 Introduction

This document provides details on the use of the MicroRidge Wireless Communications Library (RM2.4 Library) for the MICRO-RM2.4 Radio Module. The library (MRdg_MC_MICRO-RM24-02_Rel.r90) is used to develop Mobile Module type products that will communicate with MobileCollect Bases manufactured by or licensed from MicroRidge. By using this RM2.4 Library from MicroRidge, you will save yourself many man-hours and months of development time and cost. MobileCollect Bases available to receive information from applications built with the RM2.4 Library include:

- USB Base connected to a Windows PC
- USB MicroBase connected to a Windows PC
- RS-232/USB Base connected to any computer that has a DB9 serial port or USB port
- ASI 600 Series Handheld computer

The Bases with a USB port can be connected to any computer that is supported by the FTDI USB drivers. (www.ftdichip.com).

This RM2.4 Library is only for use with the IAR compiler for the Atmel AVR. No support is provided for any other compiler.

Current RM2.4 Library version:	LRM24-02-01MHZ-R.014
Release date:	August 23, 2017
IAR Compiler version used to build the library:	6.80.8

For details on the MICRO-RM2.4 Radio Module refer to the MICRO-RM2.4 Radio Module web page at www.microridge.com/wl_micro-rm24.htm.

Communications Library Software License

The license fee for the use of the RM2.4 Library is included as part of the cost of the MICRO-RM2.4 Radio Module. All MICRO-RM2.4 Radio Modules are shipped with the RM2.4 Library support enabled. A version of the MICRO-RM2.4 Radio Module is also available without support for the RM2.4 Library.

Order Information

The MICRO-RM2.4 Radio Module can be ordered directly from MicroRidge Systems. Contact MicroRidge 19 for current pricing and delivery schedules.

The Radio Modules can be ordered with or without support for the RM2.4 Library. This RM2.4 Library provides wireless ZigBee communications support for the MICRO-RM2.4 Radio Module. The RM2.4 Library has been compiled with the IAR Embedded Workbench for the AVR microcontroller and you must use an appropriate version of the IAR compiler in order to use the RM2.4 Library.

The part numbering code for the RM2.4 Library is as follows:

L =	Library support enabled via the embedded MICRO-RM2.4 Radio Module serial number
N =	No library support enabled

The MICRO-RM2.4 Radio Modules are in tubes and bulk packaging.

TU =	Tubes of 23 modules
SG =	Single pieces up to 22 modules

When purchasing tubes, only full tubes of 23 pieces can be ordered. Partial tubes of less than 23 Radio Modules cannot be ordered.

Part Number	Description	Typical Lead Times
MICRO-RM2.4-L-TU	Radio Module with RM2.4 Library support in tubes of 23 modules. This is the standard product packaging.	Less than 2 weeks
MICRO-RM2.4-N-TU	Radio Module with no wireless library support in tubes of 23 modules.	Greater than 1 month. Minimum quantities may apply. Call MicroRidge for details.
MICRO-RM2.4-L-SG	Radio Module with RM2.4 Library support. This item can only be ordered in quantities of 22 or less.	Less than 2 weeks
MICRO-RM2.4-N-SG	Radio Module with no wireless library support. This item can only be ordered in quantities of 22 or less.	Greater than 1 month. Minimum quantities may apply. Call MicroRidge for details.

RoHS 2
Compliant

The MICRO-RM2.4 Radio Module is in compliance with the European Union Directive on the restricted use of certain hazardous substances (RoHS/RoHS2 Directive). For more information review the ***RoHS Declaration of Compliance*** document on the MicroRidge web.

2 Library Functions

The MicroRidge RM2.4 Library consists of 2 files:

- MRdg_MC_MICRO-RM24-02_Rel.r90 (Communications Library)
- MRdg_Include_MICRO-RM24.h (Include file)

These files are only supported for the AVR compiler from IAR. The previous version of the Wireless Communications Library was for the Atmel ATZB-24-A2 Radio Module used a communications library and a Silicon Serial Number Chip that had to be purchased from MicroRidge. The MICRO_RM2.4 Radio Module does use the Silicon Serial Number Chip.

The following sections describe the functions that can be called from the library. The function prototypes, along with macros and structures, are included in the MRdg_Include_MICRO-RM24.h file.

2.1 Initialization

These functions must be called before you can use radio communications. These functions were only tested at a clock speed of 1.00 MHz.

uint8_t MRdg__Startup_Initialization (uint8_t ucRF_Chan)

The channel number must be in the range of 11 to 26. Channel 14 is reserved for firmware updates.

This function will call the following:

- MRdg__Set_Clock_Speed_to_Default ()
- MRdg__Radio_Initialization (uint8_t ucRF_Chan)

BOOL MRdg__Radio_Initialization (uint8_t ucRF_Chan)

Initialize the radio. You should not have to call this function since it is called by MRdg__Startup_Initialization (...). If you do call and the return value is FALSE, you can call MRdg__Get_Radio_Init_Error () to get the error number.

uint8_t MRdg__Get_Radio_Init_Error (void)

The error generated by the call to MRdg__Radio_Initialization (...). Refer to the RADIO_ERR__ macros in MRdg_Include_MICRO-RM24.h.

2.2 Radio Communications

The radio communication functions are listed below. These functions were only tested at a clock speed of 1.00 MHz.

Initialization

void MRdg__Initialization_Radio_RxD_Frames (void)

Initialize the radio RxD frames used to receive data packets from a Base. There are NUM_FRAME_BUFFER frames available to store the received data.

void MRdg__Enable_Transceiver_in_PRR1_Register (void)

Enable the PRTRX24 bit (Power Reduction Transceiver) in the PRR1 register. You may not need to call this function. This function is called by the following functions:

- BOOL MRdg__Radio_Initialization (uint8_t ucRF_Chan)
- uint8_t MRdg__Set_Radio_State (uint8_t ucNew_State)

void MRdg__Disable_Transceiver_in_PRR1_Register (void)

Disable the PRTRX24 bit (Power Reduction Transceiver) in the PRR1 register.

Radio State

void MRdg__Keep_Radio_On (BOOL bKeep_On)

If bKeep_On is TRUE, the radio will not enter the low power sleep mode. Normally you will set bKeep_On to FALSE.

void MRdg__Enter_Radio_Sleep_Mode (void)

Put the radio into the sleep mode.

uint8_t MRdg__Set_Radio_State (uint8_t ucNew_State)

Set the state of the radio. Use RADIO_RX_ON to wake up and turn on the radio.

uint8_t MRdg__Get_Radio_State (void)

Get the current state of the radio.

Operating Channel

void MRdg__Set_Operating_Channel (uint8_t ucChan)

Set the radio channel. The channel number must be in the range of 11 to 26. Channel 14 is reserved for firmware updates.

uint8_t MRdg__Get_Operating_Channel (void)

Get the current operating channel. The returned value will be in the range from 11 to 26.

Transmit & Receive Data

uint8_t MRdg__Get_Number_of_RxD_Frames (void)

Get the number of RxD frames in the buffer that have been received from a Base. The return value will be from 0 to NUM_FRAME_BUFFER.

BOOL MRdg__Read_Next_RxD_Frame (RADIO_RX_FRAME* pFrame, uint8_t* pRSSI)

Read the next RxD frame from the buffer. The RADIO_RX_FRAME structure is defined in MRdg_Include_MICRO-RM24.h. The value returned in pRSSI is the received strength signal indicator. The following is a guide for the meaning of this RSSI value:

- Strong signal <= 75
- Moderate singnal = 76 to 90
- Weak (but usable) signal > 90

uint16_t MRdg__Update_CRC_Value (uint16_t unCRC, uint8_t ucData)

Calculate the CRC checksum. This function must be called for each byte to be included in the checksum calculation.

- unCRC = Current value. Initialize to 0 before you state the sequence.
- ucData = Next byte to be included in the CRC calculation
- Return value = New calculated CRC value

void MRdg__Build_and_Send_Attention_Packet (uint8_t* pPan_ID)

Call this function before you send the real data packet to the radio. I believe I had to do this in order to get the Atmel radio to talk to the older XBee radios. This may not be needed if you are doing MICRO-RM2.4 (transmitter) to MICRO-RM2.4 (receiver) communications. If you are not sure, it is ok to send this packet.

uint8_t MRdg__Send_Packet_to_Radio (uint8_t* pTxD_Packet, uint8_t ucLength)

Send the data packet to the radio.

- pTxD_Packet = Pointer to the data packet to be sent
- ucLength = Length of the packet. The length value must include the final 2 bytes for the automatic checksum calculation. The maximum packet length = MAX_FRAME_LENGTH.

2.3 Serial Number

The Serial Number is preset in all Radio Modules that support the RM2.4 Library. If the Serial Number is not valid or cannot be read, the library will not transmit any measurement packets to a Base.

BOOL MRdg__RM24_SN (SILICON_SN* pSN);

Read the Serial Number from the MICRO-RM2.4 Radio Module.

Return values:

- TRUE = The CRC read from the Radio Module and the calculated CRC are equal
- FALSE = Could not read the S/N, or the CRC values (calculated & read) are not equal

pSN = Pointer to the SILICON_SN structure as described in MRdg_Include_MICRO-RM24.h.

2.4 Encryption

If you are sending measurement packets to a Base that contains a MicroRidge MICRO-RM2.4 or Atmel ATZB-24-A2 Radio Module, you must use encryption. If you are sending to a Base that contains an XBee module, you cannot use encryption.

BOOL MRdg__Can_Measurements_Be_Encripted (uint8_t* pBase_NW_SN)

Can the measurement packet to be sent to the Base be encrypted? If this function returns TRUE, you must encrypt the measurements with the function list below. This function should be called after you have associated the module with a Base.

pBase_NW_SN = Pointer to the 8 byte serial number that the module is associated with

uint8_t MRdg__Encript_Measurement (uint8_t ucNum_Meas_Bytes, uint8_t* pucMeas_Data)

Encrypt the measurement packet before sending it to the Base. The size of the encrypted packet will always be an even number of bytes. If your packet is 12 bytes, the encrypted packet will be 12 bytes. If your packet is 13 bytes long, the encrypted packet will be 14 bytes. The encryption algorithm is proprietary to MicroRidge and it will not be released to other companies.

Input parameters:

- ucNum_Meas_Bytes = Number of bytes in the unencrypted measurement
- pucMeas_Data = Pointer to the unencrypted measurement

Return parameters:

- The encrypted measurement is placed back in pucMeas_Data
- Return value = Number of bytes in the encrypted measurement

Warning: The value of ucNum_bytes must be less than 100. There are no checks to be sure this is TRUE.

2.5 Timers & Clock Speed

Timer 2 is used by the library functions and must not be used by your application. You must set up the timer when your application starts. To reduce power consumption, you will probably want to disable the timer if your application goes to sleep. Be sure to enable the timer when your application wakes up.

void MRdg__Setup_Timer_2 (void)

Set up timer 2. Timer 2 is an 8-bit timer.

void MRdg__Enable_Timer_2 (void)

Enable timer 2. Be sure the timer is enabled if you are calling any of the library functions or the radio is receiving data.

void MRdg__Disable_Timer_2 (void)

Disable timer 2. This is done for power saving.

uint8_t MRdg__Get_Clock_Speed (void)

Get the current clock speed. The return values will be CLK_SPEED__ as defined in MRdg_Include_MICRO-RM24.h

void MRdg__Set_Clock_Speed_to_Default (void)

Set the clock speed to the default value. This is the same as calling MRdg__Set_Clock_Speed_to__1_0000_MHz ().

void MRdg__Set_Clock_Speed_to__1_0000_MHz (void)

Set the clock speed to 1.00 MHz. This is the frequency that should be used when doing radio communications.

void MRdg__Set_Clock_Speed_to__4_0000_MHz (void)

Set the clock speed to 4.00 MHz.

void MRdg__Set_Clock_Speed_to__8_0000_MHz (void)

Set the clock speed to 8.00 MHz.

uint8_t MRdg__Get_Clock_Source (void)

The clock source set by the AVR low fuse must be the Internal RC Oscillator or the Transceiver Oscillator Crystal. Normally you would use one of the Transceiver Oscillator Crystal options. The Transceiver Oscillator Crystal is part of the MICRO-RM2.4 Radio Module and operates at 16 MHz. The Internal RC Oscillator might be required if you are having data from an RS-232 device wake up the Radio Module. This function reads the fuse low fuse setting to determine the clock source.

When this function is called it will return one of the following values:

CLK_SRC_XTAL_OSC	Clock source is the 16 MHz Transceiver Crystal Oscillator or some other unsupported clock source.
CLK_SRC_RC_OSC	Clock source is the 16 MHz Calibrated RC Oscillator.

BOOL MRdg__Calibrate_RM24_RC_Osc (void)

If you are using the 16 MHz Calibrated RC Oscillator you should calibrate this oscillator since the actual oscillator speed will vary with supply voltage and temperature. This calibration function uses the 16 MHz Transceiver Crystal Oscillator as a reference source. Typically, a calibration will take less than 5 msec. If a new calibration is successful, the calibration will update the OSCCAL oscillator calibration value.

2.6 Miscellaneous

There are several miscellaneous functions that can be called.

void MRdg__Get_Library_Version_Number (uint8_t* pText, uint8_t ucMax_Len)

Get the version number of the RM2.4 Library. A typical response is LRM24-02-01MHZ-R.013.

- pText = Pointer for the destination version number string
- ucMax_Len = Maximum length of the version number string, including the final '\0'

char __farflash* MRdg__Get_Library_Link_Date (void)

Get the link date of the library. The return value will be a pointer to a string in flash. The library (and therefore your application) must store the predefined strings in flash memory (--string_literals_in_flash), not in SRAM. A typical response is 10-12-16 15:16:24.

BOOL MRdg__Currently_Running_Debug_Firmware (void)

The return value from this function should always be FALSE. If you get a TRUE response, contact MicroRidge.

uint16_t MRdg__Get_Battery_Voltage (BOOL bRestore_Sleep_Mode)

Get the battery voltage. The voltage is an analog value returned from an A/D converter. Depending on the battery voltage, the voltage will be returned in steps of .050 or .075 volts.

bRestore_Sleep_Mode = Should the sleep mode be restored when leaving this function?

Output value = Battery voltage * 1000

3 Packet Formats

Data packet formats used to send copyright and measurement information to a Base are described on the following pages. These are packets that will be processed by Bases. As you develop your application, you may find that you need additional packet structures.

The copyright packet is simply information that is sent to all Bases that are operating on the same channel as the Mobile Module. A Base will only process this copyright packet if it is in the Setup Mode.

The measurement packet will only be processed by the Base that matches the network serial number and Pan ID contained in the measurement packet.

You can view the packet information being sent by a Base or Mobile Module by sending the <!:P1 command to the Base. If you want to view the packets, you should have a Base that is only used for viewing the packets. You should not use the same Base to receive measurement data and view the packets that are being sent.

3.1 Copyright Packet

Copyright data sent from Mobile Module to Base. Base must be in Setup Mode. Put the Mobile Module into the Setup Mode and press the Setup button on the Mobile Module to send data. This is one of sever packets for the copyright data

Columns:
- Byte Number
- ASCII value
- Hex value
- Character if between ASCII 21 and 127
- Description

```
  0   65   41   A  +---       Frame control field.  Should be 41h (Byte 0)
                              & 88h (Byte 1) for XBee compaibility
  1  136   88      +
  2   26   1a      ----       Packet counter #1, increment by 1 for each
                              new packet
  3  255   ff      +---       Pan ID:  Set both bytes to ffh
  4  255   ff      +
  5  255   ff      +---       16 bit Network ID of the target:  Set to ffh
                              ffh 0h 0h
  6  255   ff      +
  7    0    0      +
  8    0    0      +
  9  115   73   s  ----       Packet counter #2, increment by 1 for each
                              new packet
 10    0    0      ----       Unknown byte, set to 0
 11   67   43   C  ----       Frame ID
 12   13    d      ----       Copyright information
 13   77   4d   M  +
 14  105   69   i  +
 15   99  63  c  +
 16  114  72  r  +
 17  111  6f  o  +
 18   82  52  R  +
 19  105  69  i  +
 20  100  64  d  +
 21  103  67  g  +
 22  101  65  e  +
 23   32  20     +
 24   67  43  C  +
 25  111  6f  o  +
 26  109  6d  m  +
 27  109  6d  m  +
 28  117  75  u  +
 29  110  6e  n  +
 30  105  69  i  +
 31   99  63  c  +
 32   97  61  a  +
 33  116  74  t  +
 34  105  69  i  +
 35  111  6f  o  +
 36  110  6e  n  +
 37  115  73  s  +
 38   32  20     +
 39   76  4c  L  +
 40  105  69  i  +
 41   98  62  b  +
 42  114  72  r  +
 43   97  61  a  +
 44  114  72  r  +
 45  121  79  y  +
 46   32  20     +
 47   68  44  D  +
 48  101  65  e  +
 49  109  6d  m  +
 50  111  6f  o  +
 51   32  20     +
 52   64  40  @  +
 53   32  20     +
 54   50  32  2  +
 55   46  2e  .  +
 56   52  34  4  +
 57   32  20     +
 58   71  47  G  +
 59   72  48  H  +
 60  122  7a  z  +
 61   13   d     +
 62   67  43  C  +
 63  111  6f  o  +
 64  112  70  p  +
 65  121  79  y  +
 66   114   72   r  +
 67   105   69   i  +
 68   103   67   g  +
 69   104   68   h  +
 70   116   74   t  +
 71    32   20      +
 72    40   28   (  +
 73    67   43   C  +
 74    41   29   )  +
 75    32   20      +
 76    50   32   2  +
 77    48   30   0  +
 78    48   30   0  +
 79    57   39   9  +
 80    45   2d   -  +
 81    50   32   2  +
 82    48   30   0  +
 83    49   31   1  +
 84   161   a1      +---        CRC = afa1
 85   175   af      +
```

3.2 Measurement Attention Packet

Measurement attention packet sent from Mobile Module to Base. This packet is built and send by the Communications Library

Columns:
- Byte Number
- ASCII value
- Hex value
- Character if between ASCII 21 and 127
- Description

```
  0    65   41   A  +---        Frame control field.  Should be 41h (Byte 0)
                                & 88h (Byte 1) for XBee compaibility
  1   136   88      +
  2   176   b0      ----        Always 0xb0 (actually the counter)
  3    37   25   %  +---        PAN ID (2 bytes in reverse order)
  4   101   65   e  +
  5     0    0      +---        Just use these bytes, don't ask why
  6     0    0      +
  7     0    0      +
  8     0    0      +
  9     1    1      +
 10     0    0      +
 11    36   24   $  +
 12    46   2e      +---        CRC = dc2e
 13   220   dc      +
```

3.3 Measurement Packet

Measurement packet from Mobile Module to Base. The measurement in this packet is encrypted

Columns:
- Byte Number
- ASCII value
- Hex value
- Character if between ASCII 21 and 127
- Description

```
 0   65  41  A  +---       Frame control field.  Should be 41h (Byte 0)
                           & 88h (Byte 1) for XBee compaibility
 1  140  8c     +
 2   18  12     ----       Packet counter #1, increment by 1 for each
                           new packet
 3   37  25  %  +---       PAN ID (2 bytes in reverse order)
 4  101  65  e  +
 5    1   1     +---       Base network serial number (bytes in reverse
                           order)
 6  169  a9     +
 7   73  49  I  +
 8   89  59  Y  +
 9   16  10     +
10    0   0     +
11    0   0     +
12  236  ec     +
13    0   0     +---       XBee options byte (Bytes 5 & 6 in 0x81 cmd),
                           always 0
14    0   0     +
15    0   0     ----       RSSI, always 0
16    0   0     +---       XBee options byte (Byte 8 in 0x81 cmd),
                           always 0
17   81  51  Q  +---       Frame ID for Base unit processing
18   63  3f  ?  +---       Network ID of Mobile Module
19    0   0     +
20    0   0     +
21   13   d     +
22  162  a2     +
23   66  42  B  +
24   91  5b  [  +
25    1   1     +
26  107  6b  k  +---       Event number (unique value for each
                           measurement)
27   63  3f  ?  +
28   17  11     +---       Control bits variable #1
29   17  11     +---       Control bits variable #2
30    0   0     +---       Control bits variable #3
31   51  33  3  +---       Mobile Module ID
32   70  46  F  +
33    52   34   4  +
34    50   32   2  +
35    53   35   5  +
36    66   42   B  +
37    69   45   E  +---       Base station ID
38    67   43   C  +
39    52   34   4  +
40    57   39   9  +
41    65   41   A  +
42    57   39   9  +
43     6    6      +---       Number of bytes inthe measurement prefix
44    51   33   3  +---       Measurement prefix/suffix
45    70   46   F  +
46    52   34   4  +
47    50   32   2  +
48    53   35   5  +
49    66   42   B  +
50     0    0      +
51     0    0      +
52   193   c1      +---       Battery voltage with radio off (only for
                           XBee, set to battery on volts for Atmel)
53   193   c1      +---       Battery voltage with radion on
54    22   16      +---       Number of bytes in the measurement data
55    43   2b   +  +---       Encrypted measurement data
56   199   c7      +
57    38   26   &  +
58    47   2f   /  +
59   165   a5      +
60   208   d0      +
61   228   e4      +
62    58   3a   :  +
63    45   2d   -  +
64   203   cb      +
65   133   85      +
66    60   3c   <  +
67   165   a5      +
68   159   9f      +
69    76   4c   L  +
70   106   6a   j  +
71   100   64   d  +
72   144   90      +
73    46   2e   .  +
74   111   6f   o  +
75     7    7      +
76   162   a2      +
77   209   d1      +---       CRC = 95d1
78   149   95      +
```

3.4 Base Acknowledgement Packet

Measurement acknowledgement packet sent from Base to Mobile Module.

Columns:
- Byte Number
- ASCII value
- Hex value
- Character if between ASCII 21 and 127
- Description

```
 0    65   41   A  +---       Frame control field
 1   140   8c      +
 2    15    f      ----       Packet counter #1, increment by 1 for each
                              new packet
 3    37   25   %  +---       PAN ID (2 bytes in reverse order)
 4   101   65   e  +
 5     1    1      +---       Mobile Module serial number (bytes in
                              reverse order)
 6    91   5b   [  +
 7    66   42   B  +
 8   162   a2      +
 9    13    d      +
10     0    0      +
11     0    0      +
12    63   3f   ?  +
13     0    0      +---       Unknown bytes, always 0
14     0    0      +
15   104   68   h  ----       Packet counter #2, increment by 1 for each
                              new packet
16     0    0      +---       Unknown byte, always 0
17   114   72   r  +---       Frame ID
18    52   34   4  +---       RSSI of measurement packet received by the
                              Base
19   107   6b   k  +---       Event number (must match event number used
                              with measurement)
20    63   3f   ?  +
21    45   2d   -  +---       CRC = bc2d
22   188   bc      +
```

4 Support

MicroRidge Systems will provide reasonable levels of support for the MicroRidge Wireless Communications Library. If you need additional functionality in the library, please contact us.

Email:

Support:	support@microridge.com
Sales:	sales@microridge.com
Information:	info@microridge.com

Phone:

Support:	541.593.1656
Sales:	541.593.3500
Main office:	541.593.1656
Fax:	541.593.5652

Mailing Address:

MicroRidge Systems, Inc.

PO Box 3249

Sunriver, OR 97707-0249

Shipping Address:

MicroRidge Systems, Inc.

56888 Enterprise Drive

Sunriver, OR 97707

Note: There is no mail delivery to this address. This address should only be used for UPS and FedEx package delivery services. DHL does not deliver to this area in Oregon.

Web: www.microridge.com