Keywords: 2-way, remote control, RC, home automation, transceiver, ISM, RF, radio, PCB, layout, schematic, hardware, firmware, microcontroller, antenna

REFERENCE DESIGN 5406 INCLUDES: √Tested Circuit √Schematic √BOM √Description √Test Data √Software √Layout

LFRD004: 2-Way Remote Control Reference Design

By: Martin Stoehr, Principal Member of the Technical Staff, Applications
May 28, 2012

Abstract: This reference design provides a complete demonstration platform for using industrial/scientific/medical radio frequency (ISM-RF) products in a 2-way remote control (RC) application. This document includes the hardware, firmware, and system structure requirements for implementing a simple 2-way RC design.

More Information
- Wireless Home
- Application Notes and Tutorials
- EV Kit Software
- Technical Support

General Description

Click here for an overview of the wireless components used in a typical radio transceiver.

The MAX7032 transceiver reference design (RD) is a self-contained evaluation platform for exercising the product as a simple 2-way remote control (RC) demonstrator system. With the use of the Maxim USB-to-JTAG board (MAXQJTAG-USB), the MAXQ610 on both boards can be programmed by the end user.

The 2-way RC boards provide for basic human interaction through a single momentary switch input and three LEDs for visual feedback. These boards are designed to be compact, providing a self-contained transceiver with the radio, microcontroller, and multiple "ports" for connecting various inputs to the system if desired. The design provides a small footprint MMCX connector for a variety of antenna-mounting options. Input to the microcontrollers (MCUs) can be configured to use up to six of the spare ports. This board can be operated from any 3V power source (1.7V to 3.6V for the MAXQ610, 2.1V to 3.6V for the MAX7032).

Both boards come preprogrammed with identical operational firmware and mirrored target addresses to demonstrate a simple wireless 2-way RC system. Gerber files and PADS[1] schematic and layout files are available for simple cut-and-paste designs of just the radio section or the full radio and MCU implementation.

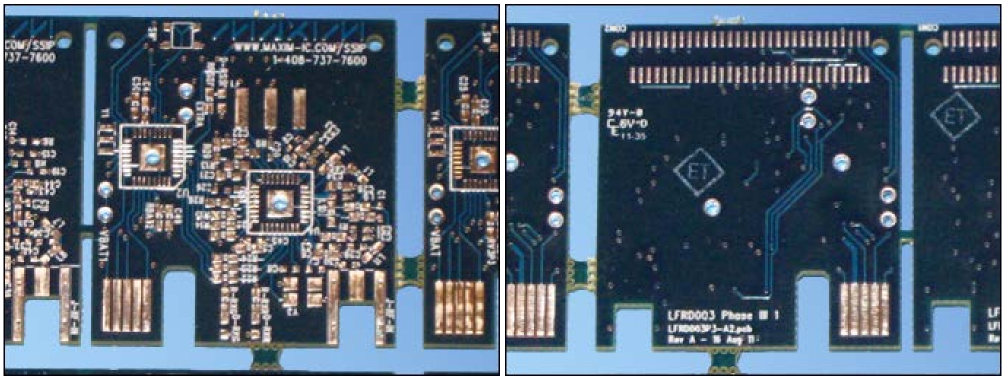
Features

- Proven printed circuit board (PCB) layout
- Proven component parts list
- Preprogrammed transceiver (TRX) pair for quick demonstration capabilities
- Free MAXQ® microcontroller programming tools available for flexible operation

Quick Start

1. Pull the two RC boards out of box and connect the batteries.
2. Connect an antenna/cable to each of the MMCX connectors.
3. Position the radios within about 1m to 2m of each other and press the switch on one of the boards. The 2-way remote control will broadcast a frame and receive an echo back, indicated by the green LED.

Two-Way RC Board Description

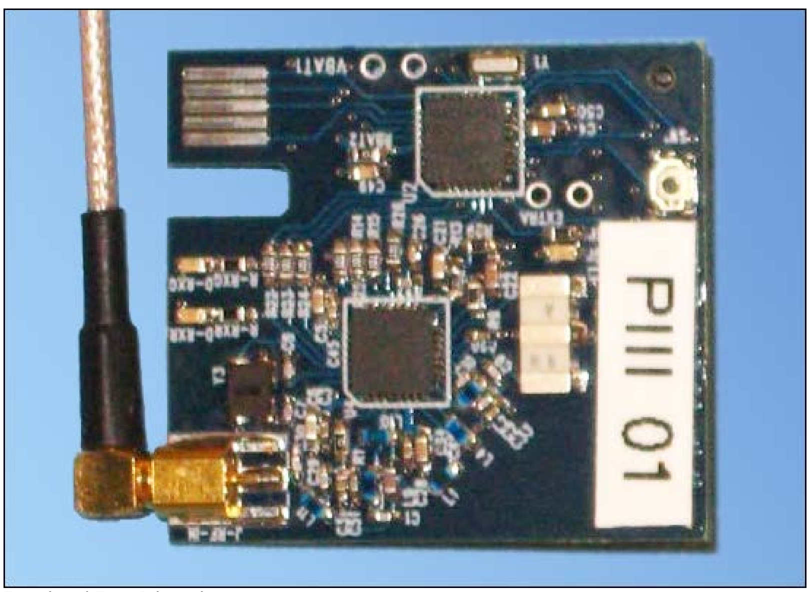


Unpopulated Rev A board

Form Factor

The LFRD004 was designed as a demonstration platform for the MAX7032 transceiver as well as the MAXQ610 microcontroller. This reference design targets a low-cost, low-BOM-count RF link. Both boards are built with identical hardware, the only difference being separate hard-coded (firmware) addresses. They both incorporate a MAX7032 radio IC mated with a MAXQ610 microcontroller that comes preprogrammed to operate as the 2-way remote control demonstration system.

Both boards were designed to allow the end user to program the MAXQ610 through a JTAG interface. In its smallest form, the boards are 3cm x 3cm, and they include a footprint for a high-density connector, which is not used in this implementation (see reference design 5404, "LFRD003: Water Meter AMR Reference Design" for more information).



Populated Rev A board

I/Os and Switches

Power is typically supplied to the boards by a 3.6V lithium battery or a pair of AA batteries, but can alternately be delivered through the JTAG interface. The power connection is available through a 100mil header (VBAT1), with both supply and ground pins.

Switch Function Table

Switch	Position	Function	µC Connection
SW'	Momentary	User TX	Pin 11, P1.2
EXTRA	GPIO	NA	Pin 4, P0.3

I/O Edge Connectors		
Signal	**Description**	**µC Connection**
JTAG-1	TCK — clock	Pin 24, P2.4
JTAG-2	GND — ground	
JTAG-3	TDO — data out	Pin 27, P2.7
JTAG-4	VBAT — external supply	
JTAG-5	TMS — master select	Pin 26, P2.6
JTAG-6	nRST — reset	Pin 28, Reset
JTAG-7	N/A	
JTAG-8	N/A	
JTAG-9	TDI — data in	Pin 25, P2.5
JTAG-10	GND — ground	

LED Indicator Table		
LED	**Function**	**µC Connection**
D-RSSI'	TX/receive signal indicator LED	Pin 7, P0.6
D-RXR	Red LED (DIO1')	Pin 12, P1.3
D-RXG	Green LED (DIO2')	Pin 8, P0.7

Data Frame Structure

The basic structure of the data frame is ASK modulated, Manchester encoded, and 4.8kbps (0.2083ms/bit). It also has 144 bits per frame (18 bytes or 9, 2-byte words), 30ms per frame, a pause of 70ms between frames, and 3 frame transmissions per burst. For information on Manchester encoding, refer to application note 3435, "Manchester Data Encoding for Radio Communications."

This structure is directly compatible with other reference design communication formats (reference design 5391, "LFRD002: Wireless Automatic Meter Reading Reference Design" and reference design 5404, "LFRD003: Water Meter AMR Reference Design"). Appendix I further describes each section of the remote keyless entry (RKE) frame structure.

Frame Structure																	
Preamble				**ID**				**Function**		**Data**		**Sync**		**Bat**	**Sig**	**Chk Sum**	
FF	FF	FF	FD	04	00	00	00	00	01	00	00	43	21	11	22	01	68

The structure of this frame is arbitrary, but provides an example of the information that can be contained in any frame related to the many industrial, scientific, and medical (ISM) RF applications.

ID Structure			
ID			
04	00	00	00

This design has been preprogrammed to use a 0x04 00 00 XX identification code in the 2-way RC modules, with the last byte being adjusted between the two (typically 0x01 and 0x02). This allows for simultaneous operation of multiple MTR systems within a W-AMR demonstrator.

Function Structure	
Function	
00	00

This reference design uses the Function field to communicate between the two units. Again, this structure can be modified to suit the purposes of the user. In this application, the two bytes of the Function field are used to signal an ECHO command. **Table 1** describes the various functions and their hexadecimal values.

Table 1. Functions and Hexadecimal Values Within the LFRD004 Firmware			
Function	**Frame Value$_h$**		**Source**
	Func[0]	**Func[1]**	
ECHO	00	00	Node

These values are an arbitrary definition for the structure of the Func field and can always be modified by the user.

Firmware Structure

Functional operation of the 2-way RC system is very similar to other LF reference designs. The MAXQ610 is configured to provide a number of inputs to the MAX7032 such as SCLK, DIO, and CS for the SPI interface and DATA, ENABLE, and TR for various other RF controls. The purpose of the MAXQ610 is two-fold: first is for the microcontroller to act as a manager and collect data from any GPIOs configured as such, second is to control the radio and communicate any data upon request. One user input is available to activate the transmission of a basic ECHO data frame. The MAXQ610 and the MAX7032 are configured to be in a "stop mode" unless one of three interrupts occur: a wakeup command from the timer, a switch press from the user, or an incoming edge from the meter port. The last two items will cause an external interrupt to be triggered, whereas the wakeup timer generates an internal interrupt.

The interrupt is serviced by decoding the source of the interrupt, then the appropriate action is taken. A switch press event will cause the microcontroller to go directly into transmit mode and send an ECHO frame. A wakeup command is the most complex of the three processes and involves timers, branching decisions, and possibly both modes (RX and TX) for the MAX7032. See Appendix II for the firmware code.

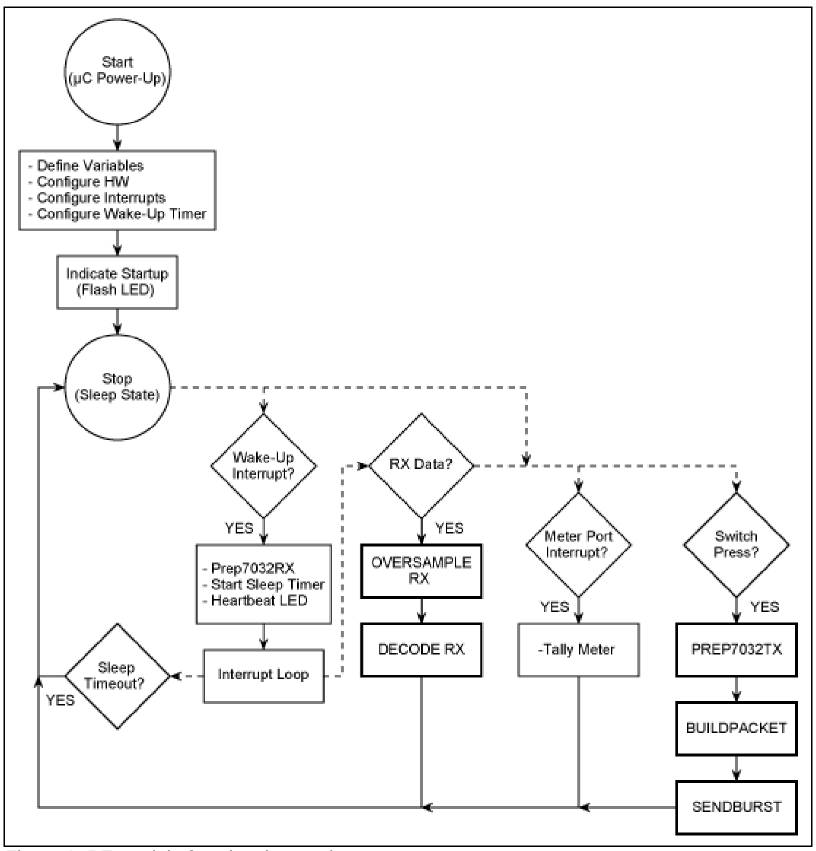


Figure 1. RF module functional operation.

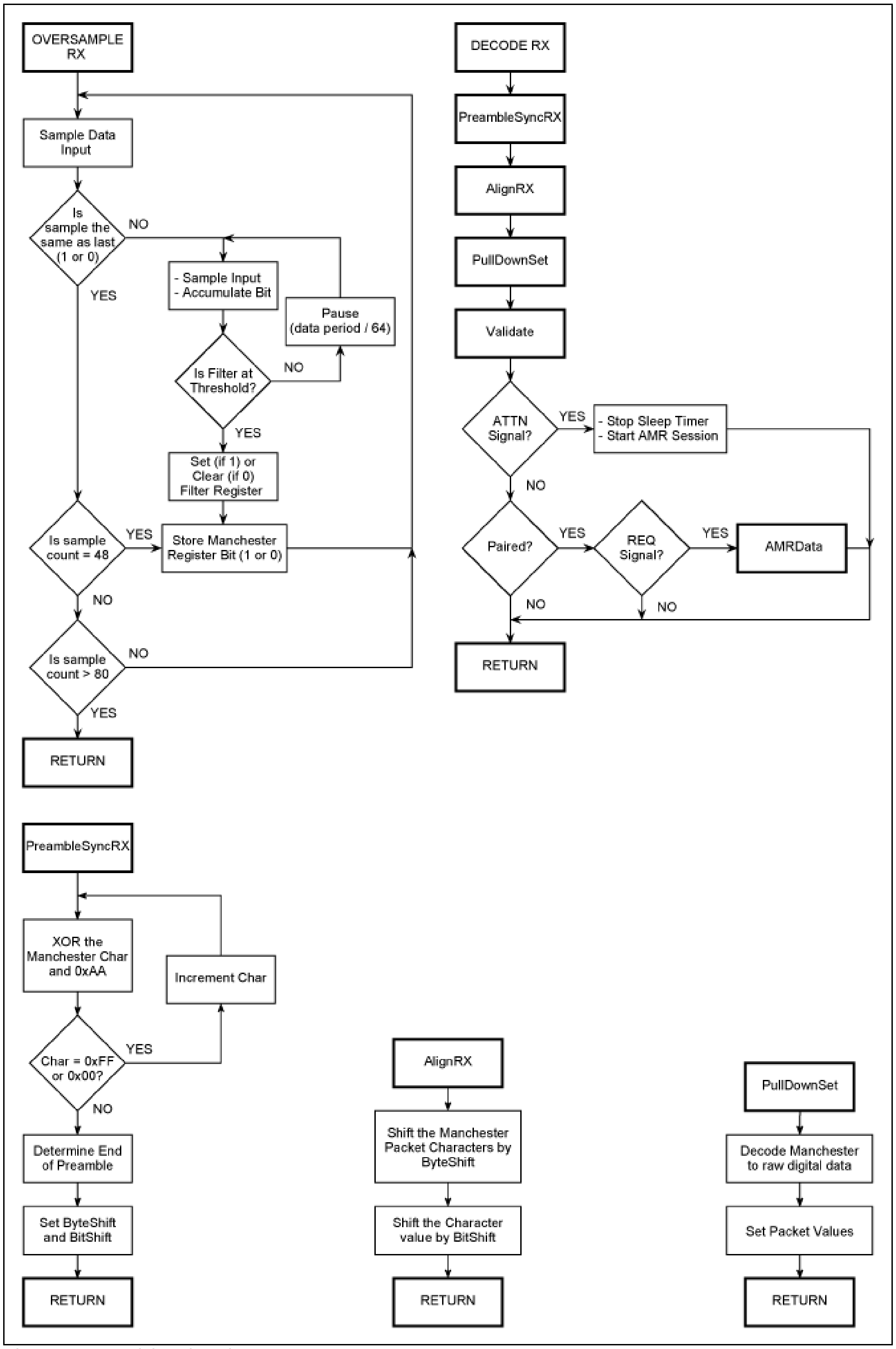


Figure 2. RF module subroutines.

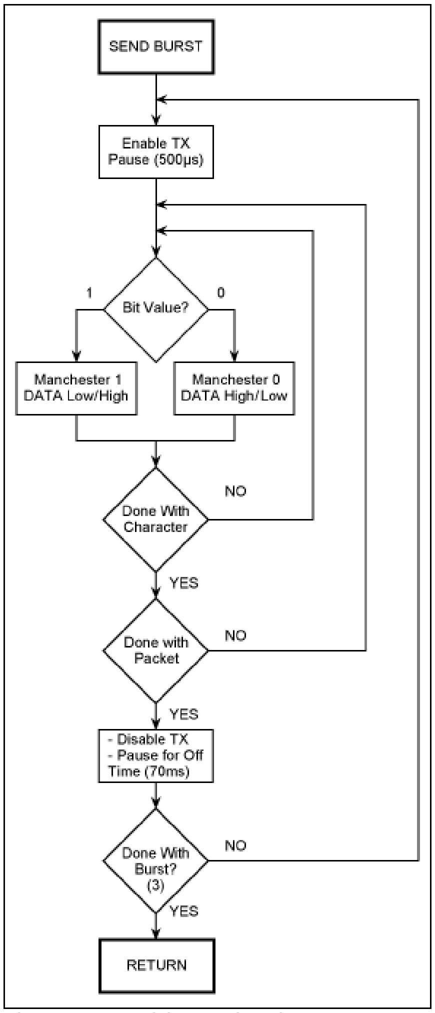


Figure 3. RF module Tx subroutine.

Microcontroller Programming Software

The firmware in this reference design was developed within the IAR Embedded Workbench® (EW) software². A full version of this software (4k KickStart Edition) can be obtained with a limited license that restricts the object code to 4k in size. Commercial licenses can also be purchased from IAR. The IAR EW works in concert with the MAXQUSB-JTAG interface and the programming adapter, to flash the MAXQ610 on both the TX and RX boards.

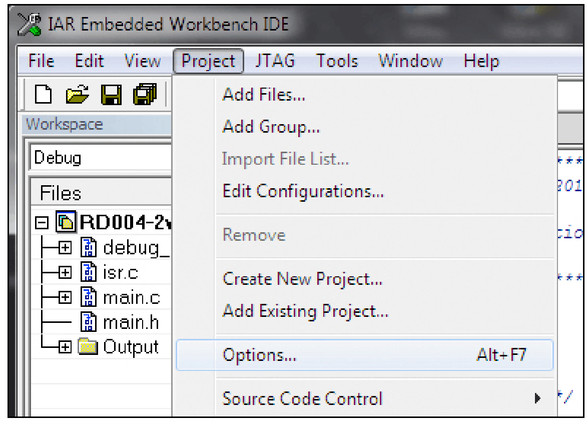
Installation

Please refer to the IAR EW documentation for installation and guidance. The firmware in this project was developed with MAXQ plug-in: IAR EWMAXQ2.20I.

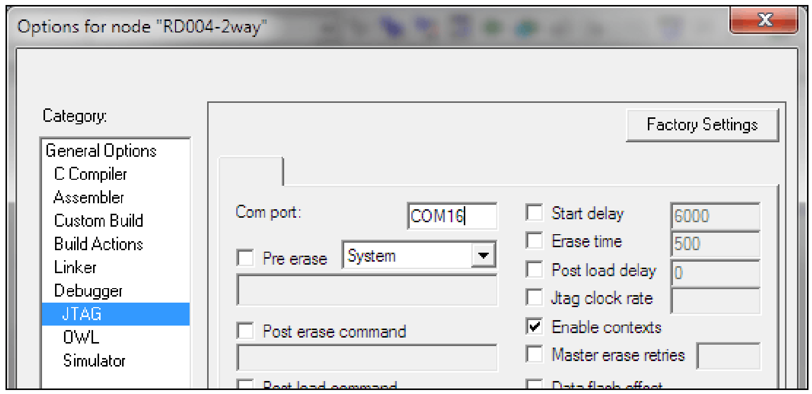
Operation

Be sure to have the USB port properly configured within the IAR EW: Project ⟶ Options... General Options — Debugger — JTAG. The COM Port should be set to match the "USB Serial Port (COM XX)" as indicated in the Windows® Device Manager.

For best performance during programming and debugging of the MAXQ610, we suggest adjusting the advanced settings for the COM port (Device Manager ⟶ USB Serial Port (COM XX) Properties ⟶ Port Settings ⟶ Advanced...). Recommended values for the receive and transmit buffer sizes are 512 bytes. A recommended latency timer of 4ms provides optimal operation.



The USBJTAG translator (MAXQJTAG-USB) board uses a FTDI UART and a MAXQ2000 microcontroller to convert the PC's serial port into a JTAG port. Check the board's firmware to ensure that it is the latest version. To determine the firmware revision, connect the USBJTAG board to a USB port; open the Microcontroller Tool Kit (MTK2) software; select Dumb Terminal in the Select Device window during startup (click OK); choose Options → Configure Serial Port and select the appropriate COM port. Choose the 115200 Speed (click OK); choose Target → Open COMXX...; hit Enter two times, type q and hit Enter again. The terminal will read back the latest firmware revision.



A JTAG interface adapter (JIA) must be used to program these systems because of the different supply levels. The USB port provides a 5V power supply that must be regulated down to the 3.3V level needed on the LFRD004 boards.

The edge connector should be oriented with JTAG pin 1 on the top of the MTR or RDR boards. The system can be programmed with batteries in place or without any batteries installed (3.3V regulated USB power is used). Within the C code, there are some lines that can be uncommented to help debug firmware changes.

Operational Setup and Use

Two-Way Remote Control Functionality

In this reference design, each of the remote control units has been preprogrammed with a semi-unique ID (0x04 00 00 01 and 0x04 00 00 02). This 4-byte value has been flashed into the MAXQ610 as part of the firmware-encoding process and can be easily changed by the user by reprogramming the microcontroller. This system does not require any form of "pairing" in the traditional sense, because the use of this communication protocol only requires one unit to know the ID of the other unit of interest.

With each of these modules, the MCU is programmed to send an ECHO frame when the on-board switch is closed (**Figure 4**). This process will elicit a response from the matching radio module at the other end—the "match" is defined by the destination ID stored in each module. Thus, when the switch is depressed on one of the 2-way RC units, the system will initiate an ECHO frame transmission and the other unit will in turn respond with an Acknowledgement (ACK) frame.

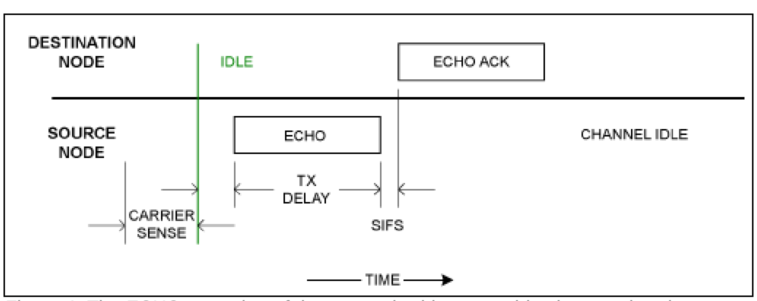


Figure 4. The ECHO operation of the protocol—this protocol implementation does not perform carrier sensing.

When the source unit receives the ACK frame, it will indicate a valid frame by lighting the green LED for about 2s. If the source unit does not receive a valid ACK frame within about 0.5s of transmitting the ECHO frame, then the module will light the red LED instead. Each of the radio modules has the other modules address stored as their destination ID, so each of those modules will act as an ECHO source, using the other as the destination.

Because of this ECHO structure, this routine can be used as a simplified range test. Starting with the two matched radios near each other, a user can slowly separate them, pressing the button as the radios are moved apart. As the "meter" module is moved, every press of the button confirms the two-way radio link, thus the basic operating range can be determined by separating the radios until the green LED no longer comes on.

Range

The predicted range in a flat unobstructed outdoor area is based on the following assumptions.

f_0 = 433.92MHz
P_{PA} = +10dBm
G_T = -18dBi (small loop antenna typical -18dBi)
h_{TX} = 1m
h_{RX} = 1m
G_R = 4.14dBi (ideal ¼λ antenna = 5.14dBi)
L_{ConR1} = -0.57dB
Path loss varies as R^{-4} because of ground bounce interference
RX sensitivity set at -114dBm

The calculated estimate of "open field" range is approximately 370m (see application note 5142, "Radio Link-Budget Calculations for ISM-RF Products" for more information).

Indoor range testing resulted in a consistent range of 30m using a Linx reduced-height antenna on both modules; 35m was achieved with a ¼λ antenna on one unit with it placed ~2m above the floor (cube wall); 30m was also reached in a lab environment with a radio 1m above the floor.

Battery Usage Analysis

Microcontroller

[1.8V nominal core voltage, 1.0V RAM (min) data retention /power-on-reset (POR) voltage]
The MAXQ610 microcontroller uses a maximum of 12µA (with power-fail off) during "stop" mode.
The MAXQ610 microcontroller (with 12MHz SysClk) burns a maximum of 5.1mA during normal operation.

TRX

[2.1V to 3.6V operation]
The MAX7032 transceiver burns a maximum of 8.8µA (3V, 85°C) during sleep mode.
The MAX7032 transceiver requires a typical 12.4mA at 434MHz (20.4mA, max) with "always on" during TX operation; when running at 50% duty cycle at 434MHz, it uses a typical 8.4mA (ASK) and a maximum of 13.6mA.
The MAX7032 transceiver typically needs 6.4mA at 434MHz during RX operation, ASK (3V, 85°C) with a maximum of 8.3mA.

This system has not yet been optimized for low-current "stop" mode.

Hardware Details

Transceiver Specifications

Supply current (I_{DD}) at f_{RF} = 433MHz, TX 50% duty cycle	8.4mA (typ)
Deep-sleep supply current (I_{DD}) at 3V	0.8µA (typ)
Output power (P_{OUT}) into 50Ω	+10.0dBm (typ)
Sensitivity (average power level)	-113dBm (typ)

Component List

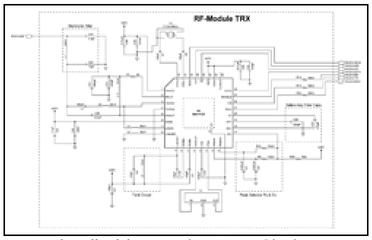
The following table provides a list of components used to populate both of the 2-way remote control boards. It should be noted that Maxim recommends high-quality, wire-wound inductors for components used on both boards.

RF Module		
Designation	**Qty**	**Description**
C48-49	2	CAP, 0.01µF, 10%
C21-22	2	CAP, 0.047µF, 10%
C5-6, C50	3	CAP, 0.1µF, 10%
C4	1	CAP, 1.0µF, 10%
C2	1	CAP, 1.8pF, 5%
C1, C7-10	5	CAP, 100pF, 5%
C29	1	CAP, 10pF, 5%
C20	1	CAP, 1500pF, 5%
C3, C26, C31, C33-34, C45	6	CAP, 220pF, 10%
C27	1	CAP, 470pF, 10%
C28, C30	1	CAP, 6.8pF, 5%
C51	1	CAP, 680pF, 10%
Y1	1	CERAMIC-SMD, 12.000MHz
F1	1	FLT\MURATA\SFTLA10M7FA00-B0, 10.7MHz
L4	1	IND-MOLDED, 10nH, 5%
L2	1	IND-MOLDED, 20nH, 5%
L9-11	3	IND-MOLDED, 22nH, 5%
L1	1	IND-MOLDED, 68nH, 5%
D-RSSI'	1	LED-1, Amber
D-RXG	1	LED-1, Green
D-RXR	1	LED-1, Red
U4	1	MAX7032
U2	1	MAXQ610A-0000+
R1, RBAT2	2	RES, 0Ω
R14-15, R22-26	7	RES, 100Ω
R9, R13	2	RES, 100kΩ
R15	1	RES, 10kΩ
R7	1	RES, 1MΩ
R-RXG, R-RXR, RRSSI'	3	RES, 75Ω
J-RF-IN'	1	Emerson MMCX Jack
SW'	1	SW-SPST-NO-B, SPST NO
Y3	1	XTAL-SMD, 17.63416Mhz

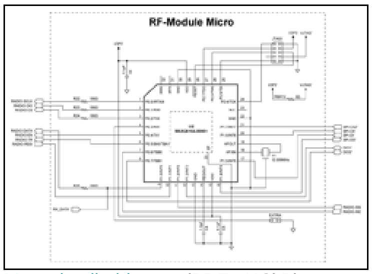
Schematics

(Revision A1: detailed 11"x 17" copy available here.)

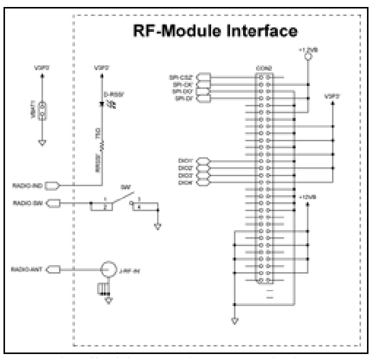
2-Way Radio Control Blocks



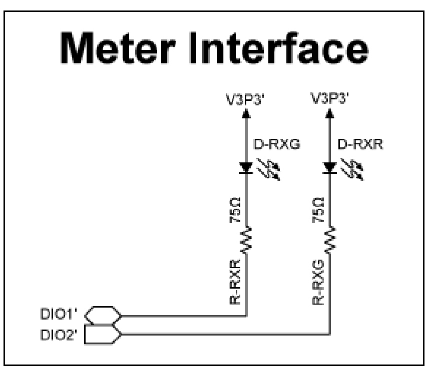
More detailed image (PDF, 443kB)



More detailed image (PDF, 410kB)



More detailed image (PDF, 443kB)

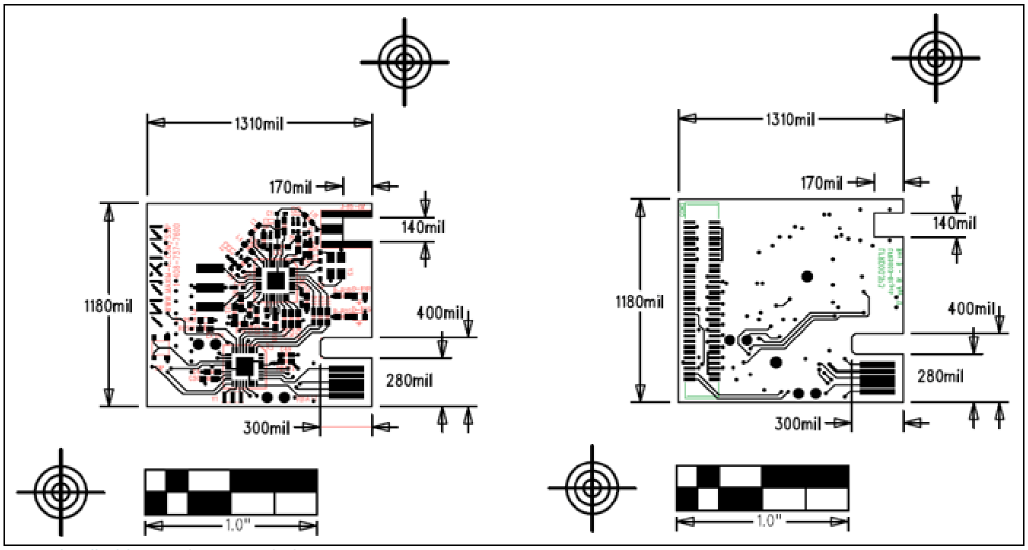


Design Files

Download Design Files
Download Gerber Files

Layout

(Revision A1: detailed scale plots available here.)



More detailed image (PDF, 362kB)

Related Application Notes

Application note 2815, "Calculating the Sensitivity of an ASK Receiver"
Application note 3401, "Matching Maxim's 300MHz to 450MHz Transmitters to Small Loop Antennas"
Application note 3435, "Manchester Data Encoding for Radio Communications"
Application note 3671, "Data Slicing Techniques for UHF ASK Receivers"
Application note 4302, "Small Antennas for 300MHz to 450MHz Transmitters"
Application note 4314, "Getting Started with the MAXQ610 Evaluation Kit (EV Kit) and the IAR Embedded Workbench"
Application note 4465, "Using the Serial Port on the MAXQ610 Microcontroller"
Application note 4636, "Avoid PC-Layout "Gotchas" in ISM-RF Products"
Application note 5142, "Radio Link-Budget Calculations for ISM-RF Products"
Application note 5300, "RF Basics"
Application note 5391, "LFRD002: Wireless Automatic Meter Reading Reference Design"
Application note 5404, "LFRD003: Water Meter AMR Reference Design"

Appendix I: RKE Frame Structure

The basic structure of the data frame is ASK modulated, Manchester encoded, and 4.8kbps (0.2083ms/bit). It also has 144 bits per frame (18 bytes or nine 2-byte words), 30ms per frame, a pause of 70ms between frames, and 3 frame transmissions per burst. For information on Manchester encoding, refer to application note 3435, "Manchester Data Encoding for Radio Communications."

Frame Structure

Preamble				ID				Function		Data		Sync		Bat	Sig	Chk Sum	
FF	FF	FF	FD	03	00	00	01	00	01	00	00	43	21	11	22	01	68

The structure of this frame is arbitrary but has been established to provide an example of the information that can be contained in any frame related to the many ISM-RF applications.

Preamble

Preamble			
FF	FF	FF	FD

The default preamble for this reference design is 4 bytes of high data with a pair of stop bits at the end. Since this is transmitted with Manchester encoding, the waveform appears as a 4.8kHz square wave lasting for 6.67ms. Having a preamble longer than roughly 1ms should provide ample time for the Rx system to wake up, given a strong received signal strength. The MAX1473 has a typical wake-up time of 250µs to start receiving valid data. Extra time can be padded onto this initial wake-up time to allow the Rx system to properly settle the baseband slicing circuit, which in turn provides optimum sensitivity. Providing a preamble of over 6ms gives ample opportunity for a relatively weak signal to wake up the receiver.

Assuming the MAXQ610 microcontroller used in the Rx is sitting in a stop mode, the system will consume a certain number of received bits to start up the microcontroller before it can begin decoding the data stream. The nanopower ring oscillator in the MAXQ610 typically runs at 8kHz (the wake-up timer interval can be $1/f_{NANO}$ to $65535/f_{NANO}$). If the Rx system is configured to use the RXSIG or the RXDATA line as an interrupt generator, the µC would have a warm-up time of $8192 \times t_{HFXIN}$. With t_{HFXIN} = 83.3ns (f_{XCLK} = 12MHz clock), the warm-up time works out to approximately 0.6827µs. This is well within the time available in one preamble transmission, and at a baud rate of 4.8kbps, the µC should be in a warmed-up state within 3.28 bits. The FD pattern at the end of the preamble is used to indicate the following start of the data frame and is the key to synchronizing the received bit stream.

ID Structure

ID			
01	23	45	67

The ID section is configured with 4 bytes of identification. This permits 2^{32} unique identifiers or over 4.29 billion codes. If one byte is used for class identification (256 different car models for example), 2^{24} unique identifiers or over 16 million codes remain. The structure can be modified to suit the purpose of the user. This RD has been preprogrammed to use an identification code as noted above, with the last byte being adjusted between RD systems. This allows for simultaneous operation of multiple, independent RKE systems.

Function Structure

Function	
00	01

This simple reference design has only four input switches on the transmitter, so 2 bytes of "function" is overkill. Again, this structure can be modified to suit the purposes of the user. In this application, when button A is pressed, the function value would be 00 01. When button B is pressed, the function value would be 00 02, and so forth. The function value is used to convey information (with individual buttons having their own bit), so it is possible to press multiple buttons simultaneously. In that instance, if buttons B and D were pressed together, the function value would be represented as 00 0A (this multibutton operation has not been implemented in the design). This is an arbitrary definition for the structure of the function value, and it can always be modified by the user.

Data Structure

Data	
00	00

The data section of this frame is provided for transmitting information, such as a temperature or pressure measurement, a speed indicator, etc. In this design, the data section could work in concert with the function section to convey information whenever a button on the transmitter has been pressed. Again, the use of this data value is arbitrary and can be modified by the user. This operation is not currently implemented in the design.

Sync Structure

Sync	
43	21

The synchronization block is set up to enable encryption coding. Users can work with this section to provide rolling code sync, a public key, etc. This operation is not currently implemented in the design.

Battery Gauge Indicator

Bat
11

This single byte allows the transmitter to send an indication of battery strength to the receiver. This section of the frame could have value when indicating a need to change the Tx battery. This operation is not currently implemented in the design.

Signal Strength Indicator

Sig
22

A possible use for a transceiver configuration, the received signal strength of the return channel could be shared between the nodes. This operation is not currently implemented in the design.

Checksum	
Chk Sum	
01	67

In this reference design, the checksum is used as a go/no-go gate for valid data. The frame values (except the preamble) are summed up during transmission, one byte at a time, and the full sum is tacked on to the end of the frame as the checksum. This value is compared to a received data stream, and a decision to use or discard the frame is made. The format of this checksum operation is arbitrary, but as long as the Tx/Rx and the encode/decode methods are equivalent, the checksum process will operate as intended.

Appendix II: RF Module Firmware

main.h (14 Sep 11, Rev 0.8):

```
/***************************************************************************
 * Copyright (C) 1999-2012 Maxim Integrated, All Rights Reserved.
 *
 * See main.c for additional information.
 *
 *****************************************************************************/

/* Main Subroutines */
  void MasterInt();

/* MAXBee Subroutines */
  void Echo();
  char EchoCheck();

/* AMR Subroutines */
  void AMRData(unsigned char MTRPort);

/* TX Subroutines */
  void SendBurst();
  void BuildPacket();

/* RX Subroutines */
  void DecodeRX();
  void OversampleRX();
  void PreambleSyncRX();
  void AlignRX();
  void PullDownSet();
  char PullDownPair(short int MPIndex);
  char Validate();
  char Paired();

/* 7032 Subroutines */
  void Init7032();
  void Prep7032TX();
  void Prep7032RX();
  void Prep7032Sleep();

/* SPI Subroutines */
  unsigned int Write7032Reg(unsigned char Adr, unsigned char Data);
  unsigned int Read7032Reg(unsigned char Adr);
  void ClockOutSPI(unsigned char InBit, unsigned int Delay);
  unsigned char ClockInSPI(unsigned int Delay);

/* Common Subroutines */
  void Lights7();
  void Pause(long int Count);
  void WriteFlash(unsigned int Address, unsigned int Data);
  unsigned int ReadFlash(unsigned int Address);
  void EraseFlash(unsigned int Address);
  void GoToSleep();
  void ExtISR();
```

```
  void Sleep();
  void WakeUp();

/* ------------------------------------------------------------------------ */
/* Global Constants */
  static int IntFlag = 0;
//  static int SleepTime = 0x0FA00;     // ~ 5.3sec sleep time (almost max), ~2F2B per sec
  static int SleepTime = 0x08D81;       // ~ 3sec sleep time
  static int SetHold = 29;              // default SPI setup and hold time, ~0.05ms
  static char WakeForSession = 0;       // Flag for RX
  static char FWRev = 0x05;             // Firmware revision

/* ------------------------------------------------------------------------ */
/* MAXBee Constants / Variables */
  static char TargetID[4] = {0x03, 0x00, 0x00, 0x3B};  // ID of the "other" radio

/* ------------------------------------------------------------------------ */
/* AMR Constants / Variables */
  static char AMRSession = 0;   // flag for active AMR session
  static char Meter[8][2] =    {{0x00, 0x00},   // initial values for each meter
                                {0x22, 0x22},
                                {0x00, 0x00},
                                {0x00, 0x00},
                                {0x00, 0x00},
                                {0x00, 0x00}};
/* ------------------------------------------------------------------------ */
/* TX Constants / Variables */
  static int PacketChars = 18;
  static char Preamble[4] = {0xFF, 0xFF, 0xFF, 0xFD},
              ID[4] = {0x02, 0x00, 0x01, 0x01},
              Func[2] = {0x00, 0x00},
              Data[2] = {0x00, 0x00},
              Sync[2]= {0x43, 0x21},
              Bat = 0x11,
              Sig = 0x22,
              CheckSum[2] = {0x00, 0x00};
  static char Packet[19];               // store the packet data as a string: 18 char + null
  static int BitMask[8] = {128, 64, 32, 16, 8, 4, 2, 1};  // store a bit mask
  static int TXWUCnt = 273;             // TX Warm-Up Count (~500us)
  static int ManCnt = 61;               // pause time for 1/2 Manchester bit
  static long int TXOffCnt = 40000;     // TX Off time Count

/* ------------------------------------------------------------------------ */
/* RX Constants / Variables */

  static char UnitID[4] = {0x03, 0x00, 0x00, 0x3A};  // ID of the MTR unit
  static char ManPacket[39];  // store the Manchester data of with 2x18+1 char
  // packet (including full preamble) + 2 'noise' characters
  static int ByteShift = 0;
  static int BitShift = 0;

/* ------------------------------------------------------------------------ */
```

See the latest LFRD004-2way*.zip file for all the firmware code.

```
main.c
isr.c

debug_maxq61x.s66
iomacro.h
iomaxq.h
iomaxq61x.h
iomaxq610.h

RD004-2way.dep
RD004-2way.ewd
RD004-2way.ewp
RD004-2way.eww
```

References

1. PADS Logic, version 2005.2 (Build number 2006.53.1); PADS Layout, version 2005.2 (Build number 2006.45.1).
2. EW_UserGuide.pdf, IAR Embedded Workbench IDE User Guide, IAR Systems.

Related Parts		
MAX7032	Low-Cost, Crystal-Based, Programmable, ASK/FSK Transceiver with Fractional-N PLL	Free Samples
MAXQ610	16-Bit Microcontroller with Infrared Module	Free Samples
MAXQUSBJTAG-KIT	Evaluation Kit for the MAXQ USB-to-JTAG Board	

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 5406: http://www.maximintegrated.com/an5406
REFERENCE DESIGN 5406, AN5406, AN 5406, APP5406, Appnote5406, Appnote 5406