AN015
RF Modem Reference Design

By K.H. Torvmark

Keywords

- *Reference design*
- *RS-232*
- *CC1000*
- *Application Example*
- *Modem*
- *Interfacing CC1000 with SPI*
- *Cable replacement*
- *Protocol example*
- *Microchip PIC*

Introduction

This reference design for an RF modem demonstrates both low-level and protocol-level design using the CC1000 RF transceiver. Software and hardware are described in detail

A pair of RF modems function as a wireless replacement for an ordinary RS-232 serial cable. The modem will function with equipment and software supporting hardware handshaking and half-duplex communication. The function of the modem is completely transparent, but data rate, RF frequency and other parameters are user-definable.

The hardware is designed as a motherboard in which a CC1000PP plug-and-play module can be inserted. The use of modules allows the basic RF modem design to be used with different RF modules.

The software handles transceiver configuration and data flow.

Table of Contents

Description

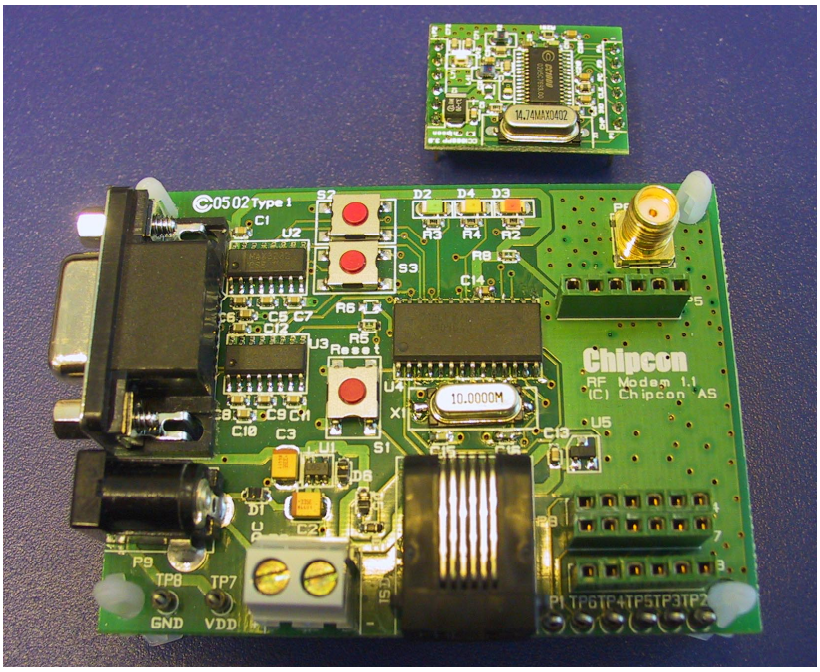


Figure 1: Photo of the RF modem and the CC1000PP module

The RF modem communicates with a PC or other RS-232 devices over an RS-232 link. In transmit mode, the modem reads data from the RS-232 serial interface, creates packets containing the data, and transmits these packets over the RF link. In receive mode, it looks for valid packets. When one is received, the data contained within the packet is extracted and sent to the attached RS-232 device or PC. A CC1000PP plug-and-play module (described in detail in [1]) is used for RF communication.

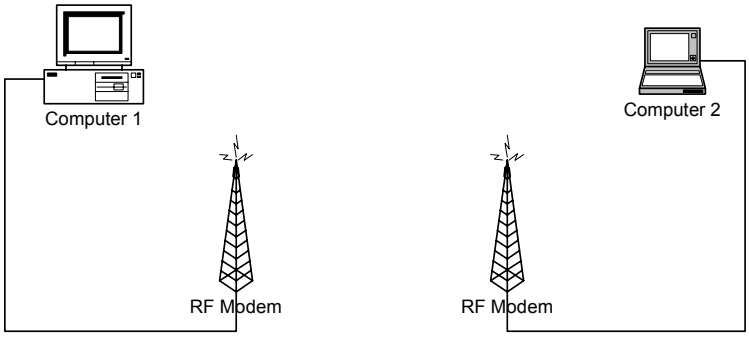


Figure 2: Basic RF modem System

In most cases, an RS-232 serial port is full duplex. As the RF link is a half-duplex link (data cannot travel in both directions at the same time), some form of data flow control is necessary. The RF modem is designed to use hardware handshaking to tell the PC when it is able to receive data and when it is not.

Configuration

The RF modem can be configured by holding down button 2 (S3) when power to the modem is switched on or when the modem is restarted by pressing the reset button (S1). If the modem is connected to a PC running a suitable terminal-emulation program such as the HyperTerminal program furnished with most versions of Microsoft Windows, the user sees the Configuration menu similar to the one shown below. The menu gives the user acces to a set of configuration and test features and enables the user to change the data rate and various RF parameters. The data can then be saved into non-volatile EEPROM data memory, so that the settings are retained when power is switched off. Communication between the PC and the RF modem occurs at 57600 baud, 8 data-bits, 1 stop-bit, and no parity.

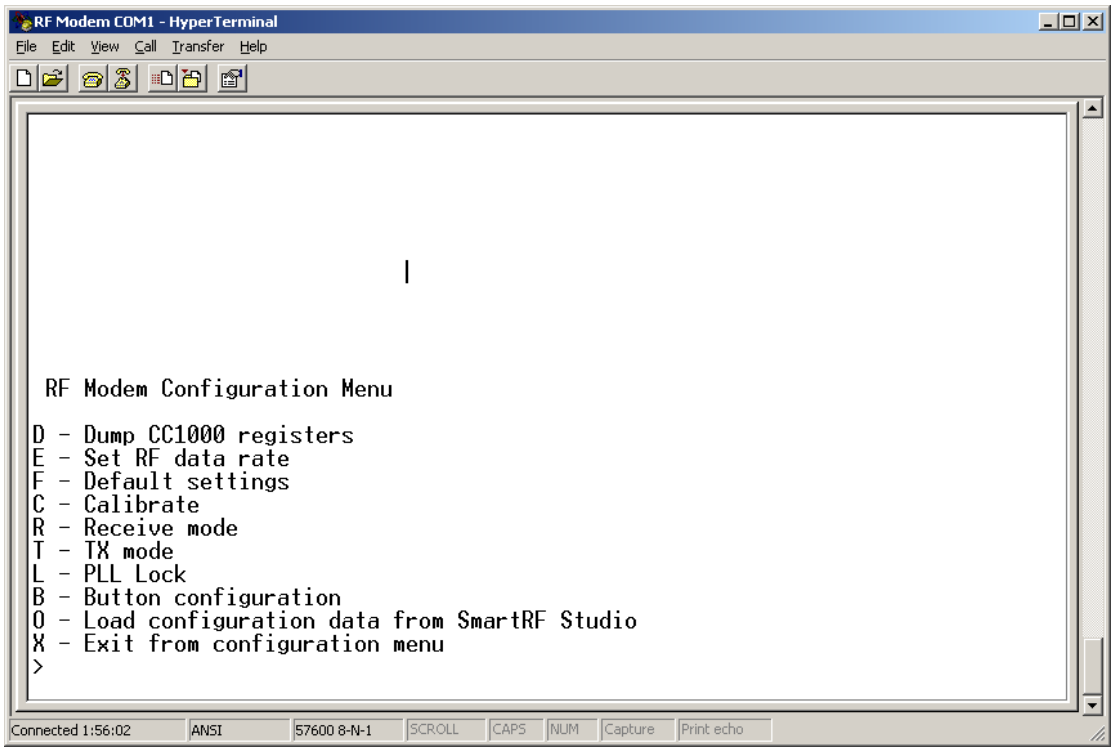


Figure 3: Configuration menu

Table 1 provides a summary of each function in the Configuration menu.

Function	Description
D - Dump CC1000 registers	This function dumps the contents of the CC1000's registers to the terminal program. It can be used to examine the configuration of the CC1000 to see if it is being configured correctly.
E – Set RF data rate	This function sets the data rate of the RF link and specifies the type of coding. For a link to function, both modems must be set to the same setting. Note that the microcontroller is not fast enough to function with data rates above 19.2kbit/s. This setting is stored in EEPROM.
F – Default settings	This function enables the user to reset the configuration to one of the three defaults. The setting selected depends on the Chipcon plug-and-play module that is inserted: 433 MHz (Europe) requires the CC1000PP-433;, 869.85 MHz (Europe) or 905 MHz (US) requires the CC1000PP-868.
C – Calibrate	This function calibrates the CC1000. A message is displayed indicating whether on not the calibration was successful (determined by whether the PLL is in lock).
R – RX mode	Puts the CC1000 into RX mode.
T – TX mode	Puts the CC1000 into TX mode.
L – PLL lock	Reads out the PLL lock status. This is useful for troubleshooting.
B – Button configuration	This function lets the user specify the character to be transmitted when the buttons on the RF modem are pressed.
O – Load configuration from SmartRF® Studio	This function enables the user to import data from SmartRF® Studio 3.20. When selected, the user is prompted to send register data from SmartRF® Studio. From the File Menu in Register View, a register data file can be generated by selecting Print registers to file. Using Microsoft's HyperTerminal, this file can be transmitted to the RF modem by choosing Send Text File from the Transfer menu. Other terminal-emulation programs should have a similar function.
X – Exit from configuration menu	This option closes the Configuration menu and starts normal operation.

Table 1: Configuration menu options

Parameter	Allowable settings
RF data rate	0.3 kbit/s Manchester, 0.6 kbit/s NRZ, 0.6 kbit/s Manchester, 1.2 kbit/s NRZ, 1.2 kbit/s Manchester, 2.4 kbit/s NRZ, 2.4 kbit/s Manchester, 4.8 kbit/s NRZ, 4.8 kbit/s Manchester, 9.6 kbit/s NRZ, 9.6 kbit/s Manchester, 19.2 kbit/s NRZ, 19.2 kbit/s Manchester, 38.4 kbit/s NRZ, 38.4 kbit/s Manchester, 76.8 kbit/s NRZ Note: Data rates above 19.2 kbit/s do not work due to workload on the microcontroller Default data rate is 4.8 kbit/s Manchester.
Button 1 character	Any ASCII character (default: 'Y')
Button 2 character	Any ASCII character (default: 'Z')
Other RF parameters	Three factory configurations (433 MHz, 869 MHz, 905 MHz). Settings can be downloaded from SmartRF® Studio into the modem. Default : 433 MHz.

Table 2: Configurable parameters

Basic operation

After it has been properly set up, the RF modem can be used to replace an RS-232 cable with a wireless link. Hardware handshaking must be used. For a simple test of the system, use a terminal emulation program. Any characters typed on one computer will appear on the monitor of the other computer.

The range of the system depends on the RF frequency, data rate, output power and type of antenna used. Environmental conditions will also affect the range; interference from nearby radio sources may reduce range drastically. If the system is used indoors, a phenomenon known as multi-path fading may be a problem, but can usually be resolved by moving the RF modem around a bit or changing the RF frequency.

In normal operation, data packets are sent when the buttons on the modem are pressed. The character sent, however, depends on the settings made in the Configuration menu. This functionality exists to be able to use the RF modem as a remote controller for a PC.

Three LEDs on the RF modem are used as status indicators. The green LED (D2) functions as a power-on indicator, the yellow LED (D4) is lit when the modem is transmitting RF data or when button 2 (S3) is pressed, and the red LED (D3) is lit when the modem is receiving RF data or when button 1 (S2) is pressed.

Hardware

Figure 4 shows a block diagram of the RF modem. The RS-232 signals are level-converted to interface to the standard 3V CMOS signal levels used by the RF modem. The RS-232 data lines are connected to the built-in UART of the Microchip microcontroller. The RF modem functions as data communication equipment (DCE) according to the RS-232 definitions, so the RF Modem can be connected to a PC using a standard serial cable.

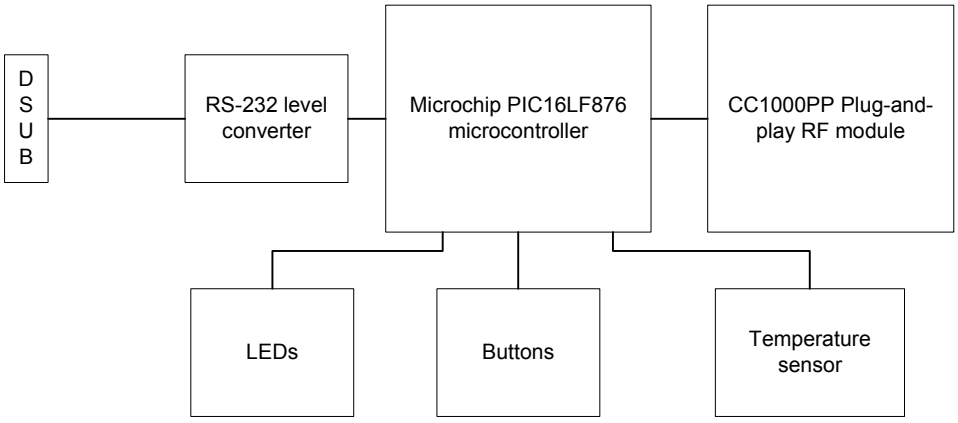


Figure 4: RF Modem hardware block diagram

The RF Modem has two primary buttons, three LEDs and a reset button for resetting the microcontroller. A temperature sensor is included for crystal temperature drift compensation; a feature not implemented in the software.

This hardware platform is very versatile; and by simply reprogramming the microcontroller, can be used for many purposes other than a modem.

Microcontroller

The PIC16LF876 microcontroller from Microchip performs all the control functions in the RF Modem. The microcontroller runs on a 10 MHz crystal clock. It interfaces to the RS-232 port using its internal UART, which can be configured to use all standard RS-232 communication baud rates. Communication with the CC1000 is handled via the PIC's built-in SPI interface, or by using general I/O-pins.

The PIC16LF876 has 8 kilobytes of program memory, sufficient for these types of applications. It has a data memory of 368 bytes and 256 bytes of non-volatile EEPROM data memory.

PORTC		
PC7	I	DATA_IN (RS-232: TD)
RC6	O	DATA_OUT (RS-232: RD)
RC5	O	PDATA (Output)
RC4	I	PDATA (Input)
RC3	O	CLOCK / PCLK
RC2	O	STROBE / PALE
RC1	I/O	DIO
RC0	O	READY (RS-232: CTS)

PORTB		
PB7	I	ICD pin
RB6	I	ICD pin
RB5	I	RX_TX (RS-232: RTS)
RB4	I	PD (RS-232: DTR)
RB3	I	LVP programming pin
RB2	I	CHP_OUT (CC1000)
RB1	O	AWAKE (RS-232: DSR)
RB0	I	DCLK

PORTA		
RA5	I/O	TD LED / Button 2
RA4	O	Carrier Detect LED / SYNC (RS-232: CD)
RA3	I	Unused analog input
RA2	I/O	RD LED / Button 1
RA1	I	RSSI input
RA0	I	Temperature sensor analog input

Table 3: I/O pin usage for the PIC16LF876 microcontroller

The microcontroller can be reprogrammed by using the In-Circuit Debugger (ICD) module available from Microchip. This module plugs into the modular jack on the RF modem. For further details, consult the documentation from Microchip [2].

The following settings should be used in Microchip's MPLAB software when reprogramming the PIC:

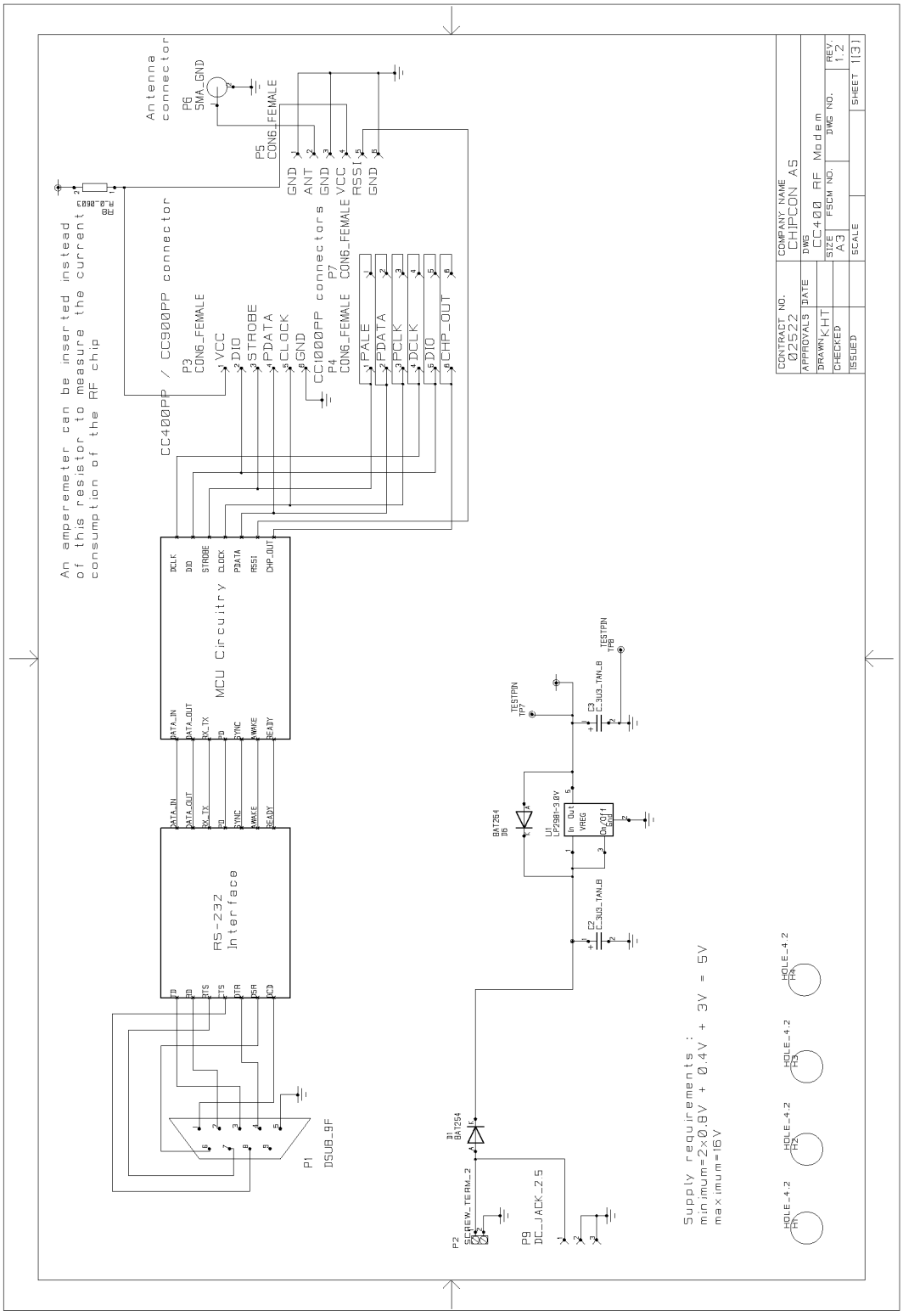
Oscillator: HS
Watchdog: OFF
Power-up timer: ON
Brown-out detect: OFF
Low-voltage program: OFF
Code Protect Data EE: OFF
Flash Memory Write: No Memory written to by EECON
Code protect Code protection OFF
Erase all before program must be turned OFF

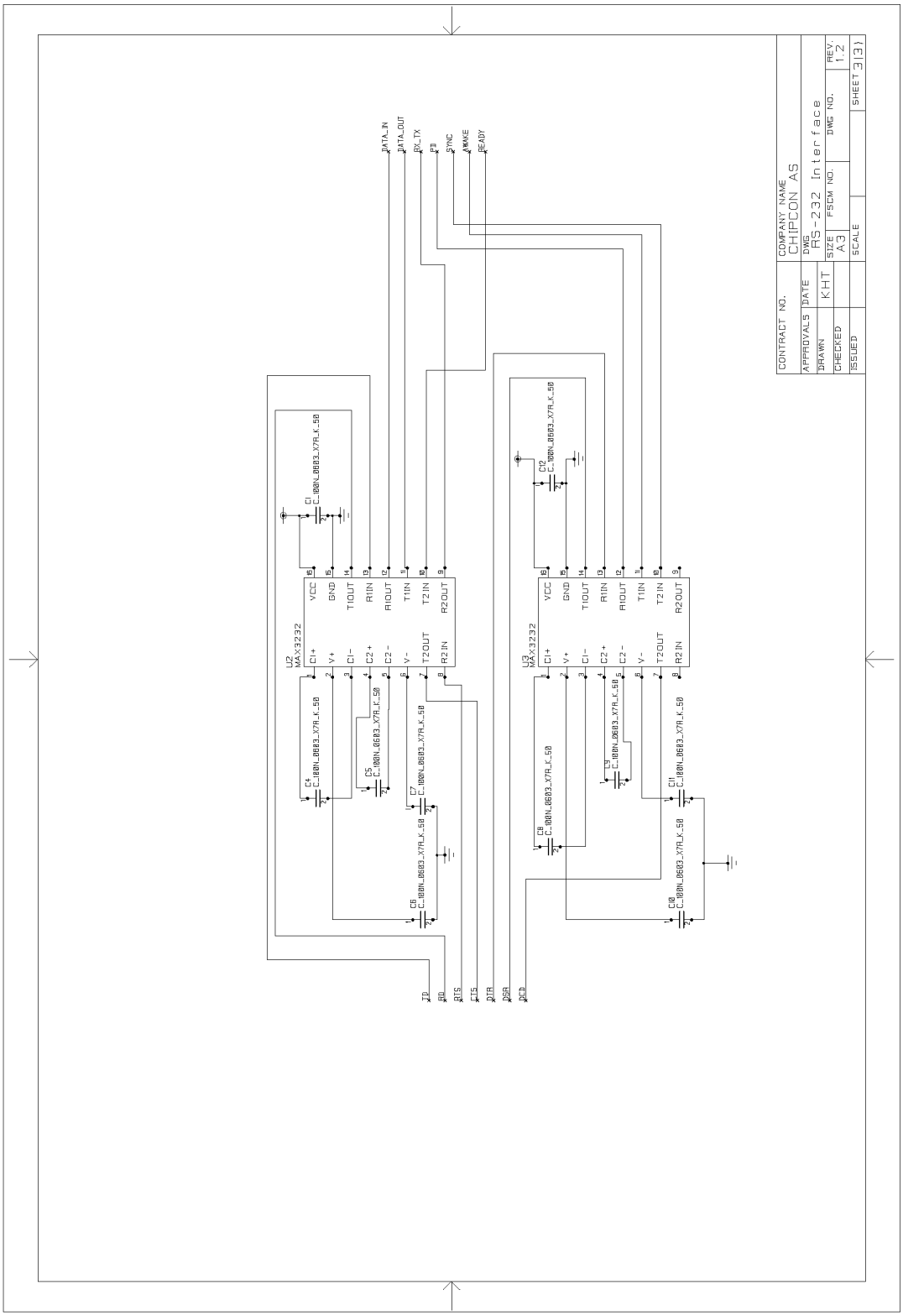
Temperature sensor

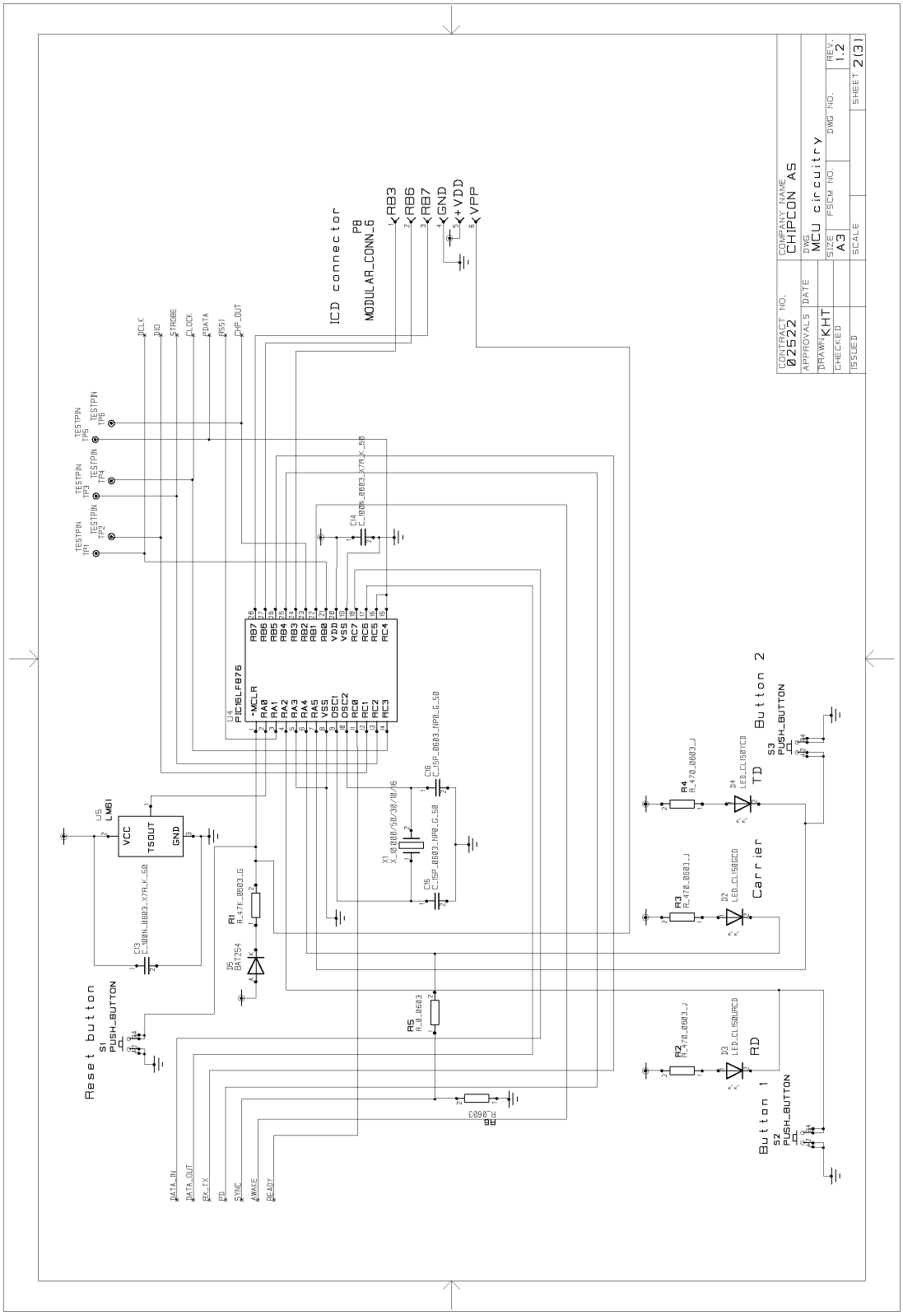
An analogue temperature sensor is located on the RF modem PCB right under the plug-and-play module. This temperature sensor is connected to an ADC input on the microcontroller, and can be used for compensating transceiver crystal drift due to temperature. Although the ideal placement for such a sensor would be right next to the transceiver crystal, this is not possible in this reference design, because the crystal is located on another PCB. Chipcon recommends that the sensor and the crystal be positioned as close to each other as possible in temperature compensation applications. See [4] for more information about the temperature sensor.

The current version of the RF Modem software does not implement temperature drift compensation.

Schematics


An amperemeter can be inserted instead of this resistor to measure the current consumption of the RF chip
CC400PP / CC900PP connector
P3 CON6_FEMALE
VCC
DIO
STROBE
PDATA
CLOCK
GND
CC1000PP connectors
P4 CON6_FEMALE
P7 CON6_FEMALE
PALE
PDATA
PCLK
PDIO
CHP_OUT
P5 CON6_FEMALE
GND
ANT
GND
VCC
RSSI
GND
Antenna connector
P6 SMA_GND
MCU Circuitry
RS-232 Interface
P1 DSUB_9F
U1 LP2981-3.0V
BAT254
C2 C_3U3_TANLB
C3 C_3U3_TANLB
TESTPIN TP7
TESTPIN TP8
P2 SCREW_TERM_2
P9 DC_JACK_2.5
Supply requirements :
minimum=2x0.8V + 0.4V + 3V = 5V
maximum=16V
HOLE_4.2
CONTRACT NO. 02522
APPROVALS
DRAWN KHT
CHECKED
ISSUED
COMPANY NAME CHIPCON AS
DWG CC400 RF Modem
SIZE A3
FSCM NO.
DWG NO.
REV. 1.2
SCALE
SHEET 1(1)


DATA_IN
DATA_OUT
RX_TX
SYNC
MAKE
READY
U2
MAX3232
U3
MAX3232
VCC
GND
T1OUT
R1IN
R1OUT
T1IN
T2IN
R2OUT
C1+
V+
C1-
C2+
C2-
V-
T2OUT
R2IN
C1
C2
C4
C5
C7
C8
C9
C10
C11
C12
C13
C_100N_0603_X7R_K_50
DTR
DSR
DCD
RTS
CTS
RD
TD
COMPANY NAME
CHIPCON AS
DWG:
RS-232 interface
CONTRACT NO.
APPROVALS
DATE
DRAWN
KHT
CHECKED
ISSUED
SIZE
A3
FSCM NO.
DWG NO.
REV.
1.2
SCALE
SHEET 3 OF 3


ICD connector
P8
MODULAR_CONN_6
RB3
RB6
RB7
GND
+VDD
VPP
MCU circuitry
CHIPCON AS
Reset button
S1
PUSH_BUTTON
Button 1
S2
PUSH_BUTTON
Button 2
S3
PUSH_BUTTON
Carrier TD
RD
PIC16LF876
LM61
TSOUT
VCC
GND
DATA_IN
DATA_OUT
RX_TX
CD
SYNC
AWAKE
READY

Bill of materials

Ref.	Description	Value	Part
C1	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C2	Capacitor, tantalium	3.3uF	C_3U3_TAN_B
C3	Capacitor, tantalium	3.3uF	C_3U3_TAN_B
C4	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C5	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C6	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C7	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C8	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C9	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C10	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C11	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C12	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C13	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C14	Capacitor 0603	100nF	C_100N_0603_X7R_K_50
C15	Capacitor 0603	15pF	C_15P_0603_NP0_G_50
C16	Capacitor 0603	15pF	C_15P_0603_NP0_G_50
D1	Diode, Si		BAT254
D2	LED, green, SMD		LED_CL150GCD
D3	LED, red, SMD		LED_CL150URCD
D4	LED, yellow, SMD		LED_CL150YCD
D5	Diode, Si		BAT254
D6	Diode, Si		BAT254
H1	Circuit Board Support		Distance 12.5mm
H2	Circuit Board Support		Distance 12.5mm
H3	Circuit Board Support		Distance 12.5mm
H4	Circuit Board Support		Distance 12.5mm
P1	D-Sub, 9 pin, female		DSUB_9F
P2	2 pin terminal, screw		SCREW_TERM_2
P3	Connector, 0.9 mm pin, female		CON6_FEMALE
P4	Connector, 0.9 mm pin, female		CON6_FEMALE
P5	Connector, 0.9 mm pin, female		CON6_FEMALE
P6	SMA connector, straight		SMA
P7	Connector, 0.9 mm pin, female		CON6_FEMALE
P8	6-pin modular jack		MODULAR_CONN_6
P9	DC jack, 2.5mm center pin		DC_JACK_2.5
R1	Resistor 0603	47kΩ	R_47K_0603_G
R2	Resistor 0603	470Ω	R_470_0603_J
R3	Resistor 0603	470Ω	R_470_0603_J
R4	Resistor 0603	470Ω	R_470_0603_J
R5	Resistor 0603	0Ω	R_0_0603
R8	Resistor 0603	0Ω	R_0_0603
S1	Push button, SMD		PUSH_BUTTON
S2	Push button, SMD		PUSH_BUTTON
S3	Push button, SMD		PUSH_BUTTON
TP1	Testpoint		TESTPIN
TP2	Testpoint		TESTPIN
TP3	Testpoint		TESTPIN
TP4	Testpoint		TESTPIN
TP5	Testpoint		TESTPIN
TP6	Testpoint		TESTPIN
TP7	Testpoint		TESTPIN

TP8	Testpoint		TESTPIN
U1	National Semiconductor LP2981 3.0V low drop-out regulator		LP2981-3.0V
U2	Maxim RS-232 Transceiver, SO-16		MAX3232
U3	Maxim RS-232 Transceiver, SO-16		MAX3232
U4	Microchip PIC16LF876 microcontroller, SO-28		PIC16LF876
U5	National Semiconductor LM61 temperature sensor, SOT23		LM61
X1	10.000 MHz Crystal, 16pF load, HC-49-SMD		X_10.000/50/30/10/16

Software

The software for the PIC microcontroller is written for the IAR PIC16 C-compiler.

The highest priority task of the software is performed by the external interrupt handler, which is triggered by transitions in the DCLK clock coming from the CC1000.

The main program handles state transitions, writes data in the RX buffer to the UART, reads any incoming data from the UART and stores it in the TX buffer, and handles the time-out and button debouncing timers.

Configuration of the CC1000 is performed either using general I/O pins or the PIC's SPI interface. If the symbol 'SPI' is defined, the SPI code is used; otherwise the slower general I/O pin-based code is used. For more details on CC1000 configuration issues, see Chipcon application note AN009.

Resource usage

Most of the microcontroller's SRAM data memory is used for buffering the incoming and outgoing data streams. When data arrives from the UART, the UART main program stores the data in the TX buffer, and an RF packet is sent only after a timeout or when the buffer is full.

When data is received via RF, the data is buffered in the RX ring buffer, and sent to the PC via the UART.

The EEPROM data memory is used for non-volatile storage of configuration parameters. These parameters may be changed by entering the configuration mode at start up. The required data is read from the EEPROM memory as needed.

Packet protocol

The protocol chosen for the RF modem is a simple variable-length packet protocol.

Field	Length	Format, usage
Preamble	32 bits	Alternating 0s and 1s
Start-of-Frame / Unique Identifier	2 bytes	0x33CC for modem application
Unit address	1 byte	0 for broadcast, 1-255 for unit address, not currently used
Data length	1 byte	Length of data payload
Data	Variable	Data payload, maximum length is 64 bytes

Table 4: Data protocol

The maximum packet length for the RF modem software is 64 bytes. If the modem receives a packet a greater data length than this, the packet is discarded.

Source files

The software consists of several source code files. A quick overview of the files is provided in the table below.

File name	Short description
rfmodem_cc1000.c	Main program source file
modemhw.h	Header file defining the I/O pin usage
cc1000.c	Function library for configuring and using CC1000. See application note AN009 for more information.
cc1000.h	Header file for cc1000.c, also includes definitions for all the registers of the CC1000.
configure.c	Configuration menu source file
configure.h	Header file for configure.c
getchar.c	Implements the C library function getchar() using the UART
putchar.c	Implements the C library function putchar() using the UART
simpleio.c	Implements simple replacements for standard I/O functions like printf() and scanf(). Uses the getchar() and putchar() functions implemented in getchar.c and putchar.c
simpleio.h	Header file for simpleio.c
interrupt.c	Contains the RF modem interrupt handler
main.h	Header file used by interrupt.c to gain access to variables shared with the main program

Table 5: Summary of software source files

All of these files can be downloaded from Chipcon's web site at www.chipcon.com.

All other files referenced are standard ANSI C library files and should be provided by the compiler vendor.

Software implementation

The software is implemented as a state machine with three different states.

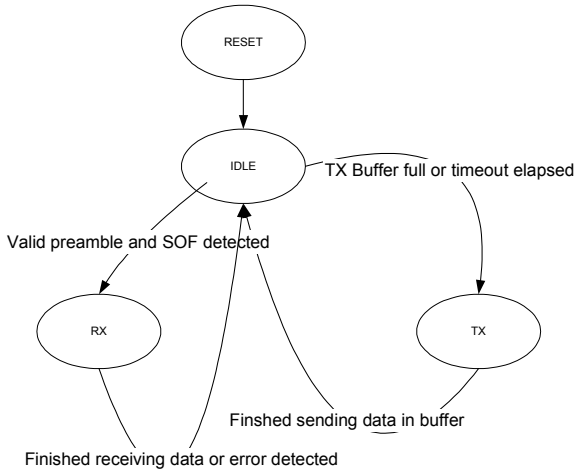


Figure 5: State diagram

In the IDLE state, the modem looks for a valid incoming preamble and for data from the RS-232 interface. If the RF modem detects a valid incoming preamble and it is followed by a valid start-of-frame (SOF) word, the modem enters the RX state. If the transmit buffer is full or if the

timeout period has expired since the last character was received from the RS-232 interface, the modem enters the TX state.

In the RX state, header data is handled in the interrupt handler, data is buffered in a circular buffer and transmitted via the RS-232 interface in the main program.

In the TX state, the data in the transmit buffer is sent to the CC1000 for transmission.

State switching is performed by the main program; the interrupt routine updates a NextState variable when a state change is to occur.

Main program
The figure below is a flowchart of the main program, implemented in the main() function in rfmodem_cc1000.c.

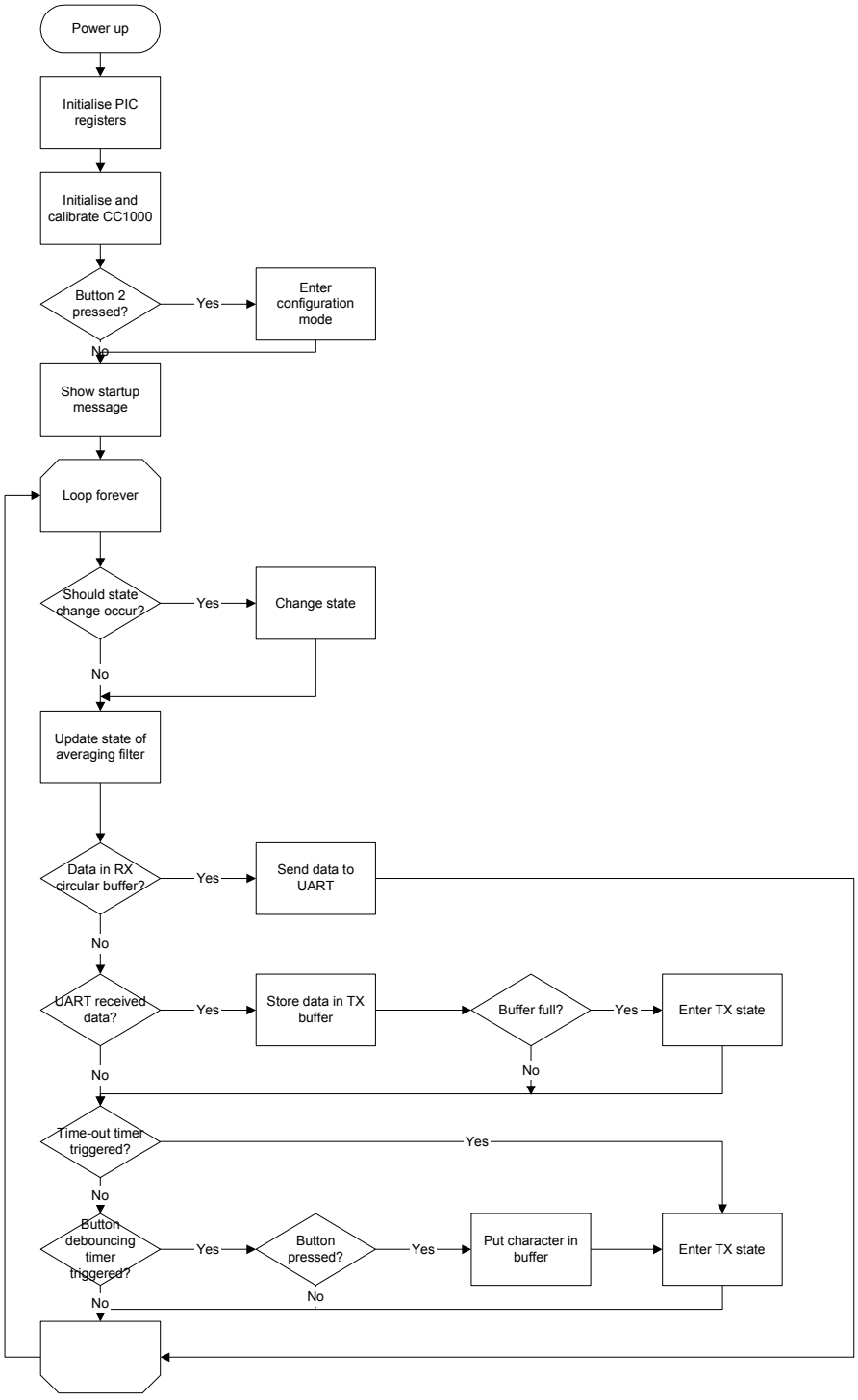


Figure 6: Main program flowchart

Interrupt handler

The figure below is a flow chart of the interrupt handler, implemented in interrupt.c for interrupts generated by transitions on the DCLK pin of the CC1000. The processing that is done depends on the current state of the software.

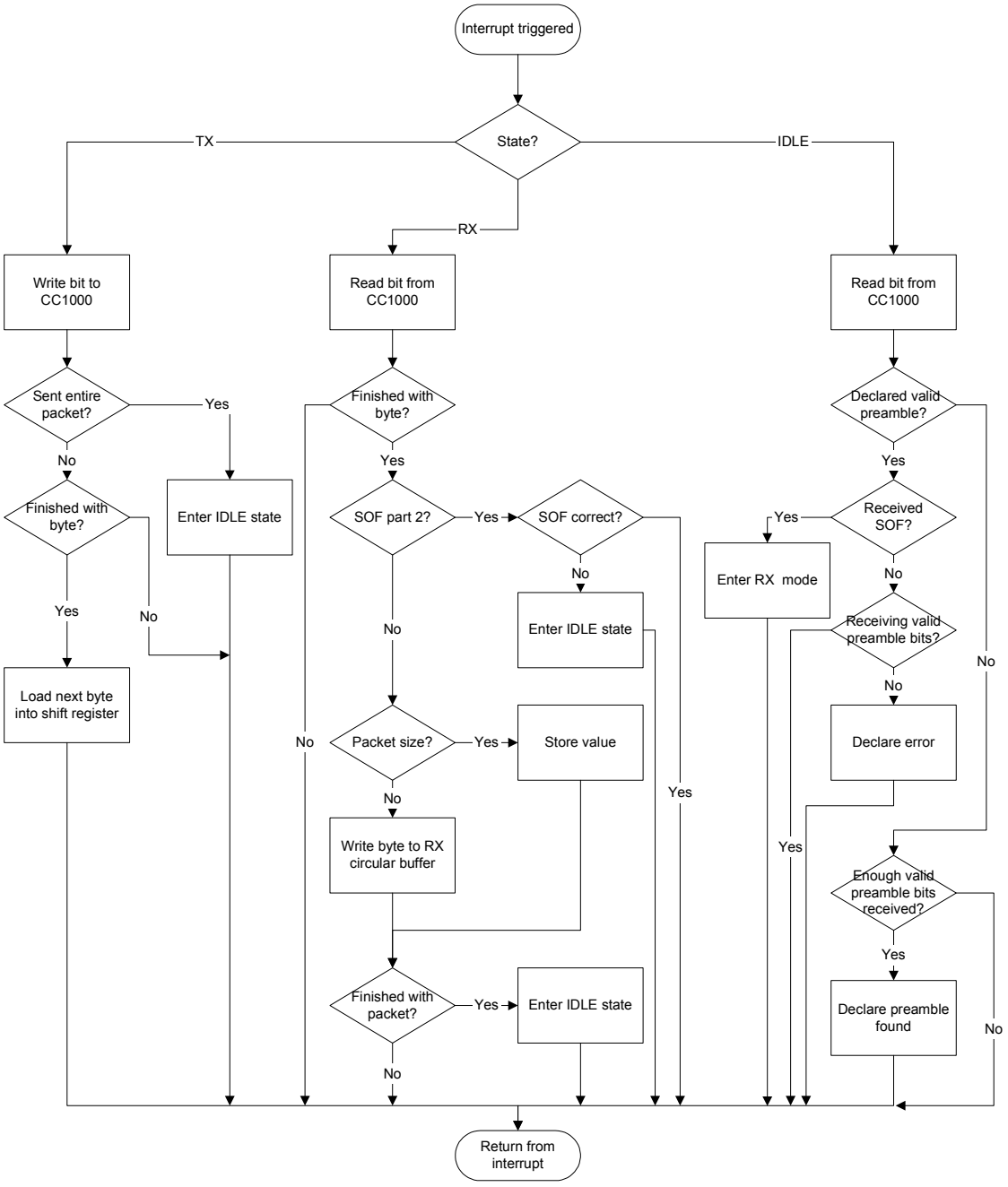


Figure 7: Interrupt handler flow chart

Operation

The diagrams below, generated from a logic analyzer, illustrate the operation of the RF Modem and show how the I/O lines behave during communication.

The figure blow shows a packet consisting of a single ASCII ‘a’ being received. Once a valid preamble is detected, the averaging filter is locked. The lock averaging filter command to the CC1000 can be seen to the left in this figure, appearing as a short transition in the PALE, PCLK and PDATA signals. The receiver then receives the first part of the SOF and, once this is accepted, it enters the RX state. The READY signal is set high (indicating to the PC that it will not accept serial data), and the RX LED is lit (active low). The header data is processed, indicating that this packet contains only one byte of payload data. This data byte is stored in the receive ring-buffer. As soon as the last bit of the message is received, the averaging filter is unlocked (the second transition of the PALE, PCLK and PDATA signals). Packet reception is then complete. The ring buffer is processed when sufficient time is available and the data is sent to the UART (signal on DATA_OUT line).

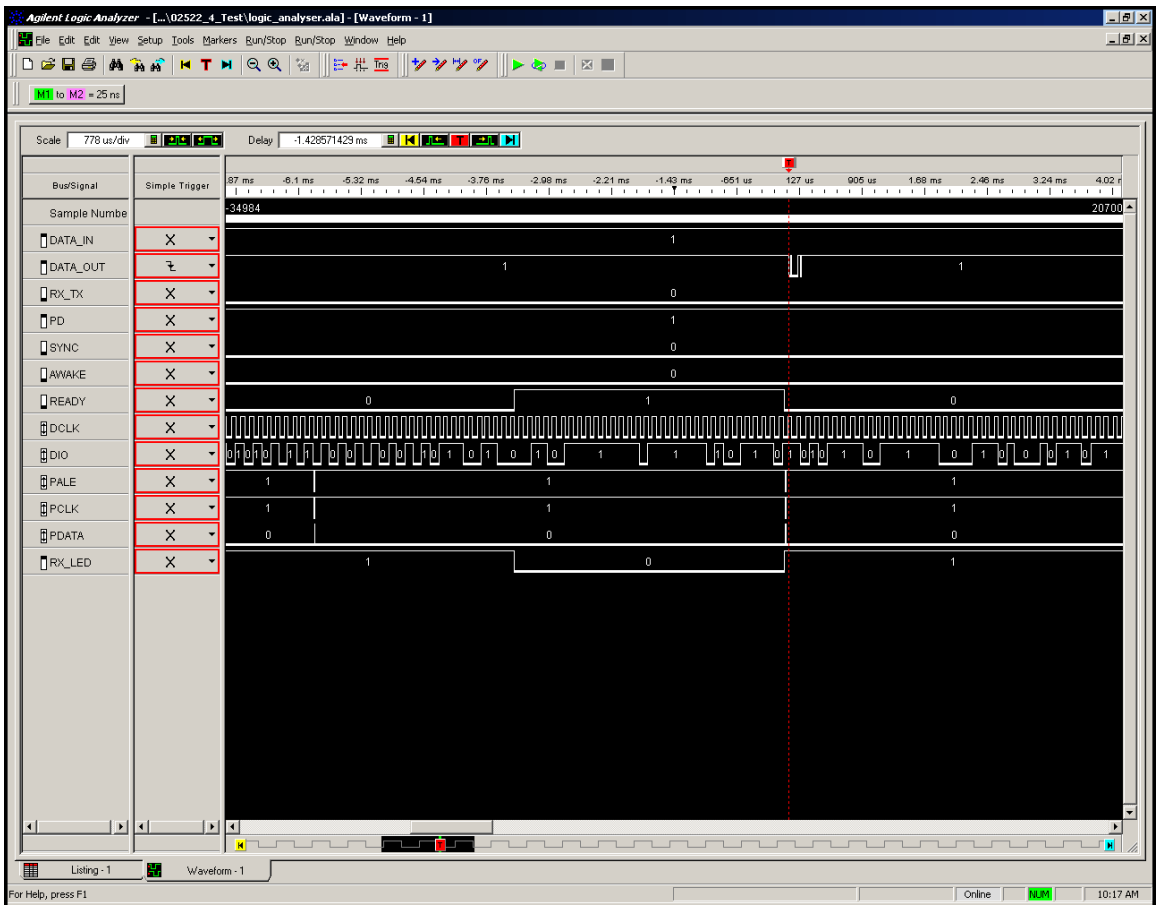


Figure 8: I/O lines during reception of a short packet

The figure below shows the CC1000 being configured by using the SPI interface of the microcontroller. The configuration is very fast; programming a register takes 9 us. The SPI interface is configured for reading data on the falling edge of the clock and the clock idle state is set to HIGH to communicate with the CC1000. As the transmit and receive lines of the SPI interface are connected together to form a two-wire interface, the SPI output is set to be an input when reading data from the CC1000.

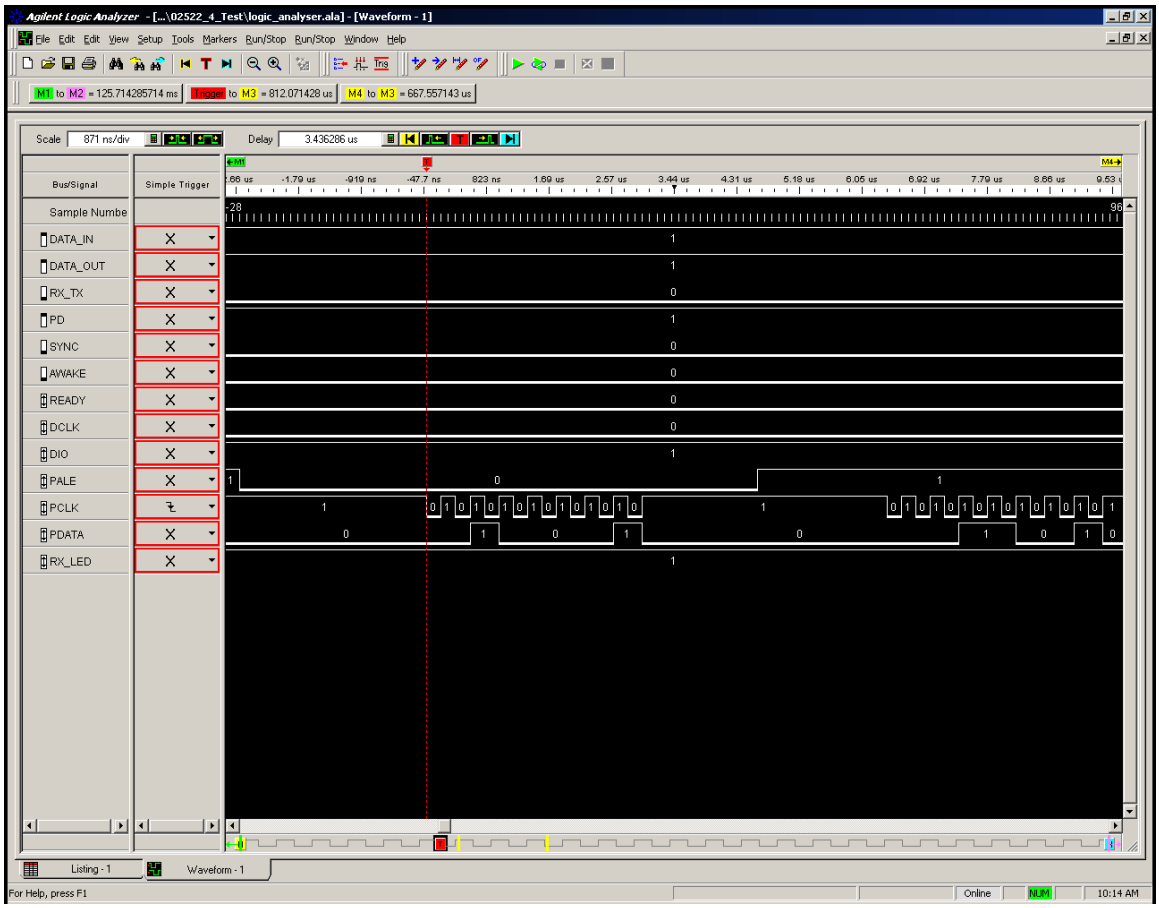


Figure 9: I/O lines during CC1000 configuration

Figure 10 shows a long string of ASCII characters being received ('1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'). This string is 62 characters long, and therefore split into two packets. In between the two packets is also a rejected packet, the RF Modem goes into the RX state, but the second half of the SOF is not correct, and the modem returns to the IDLE state.

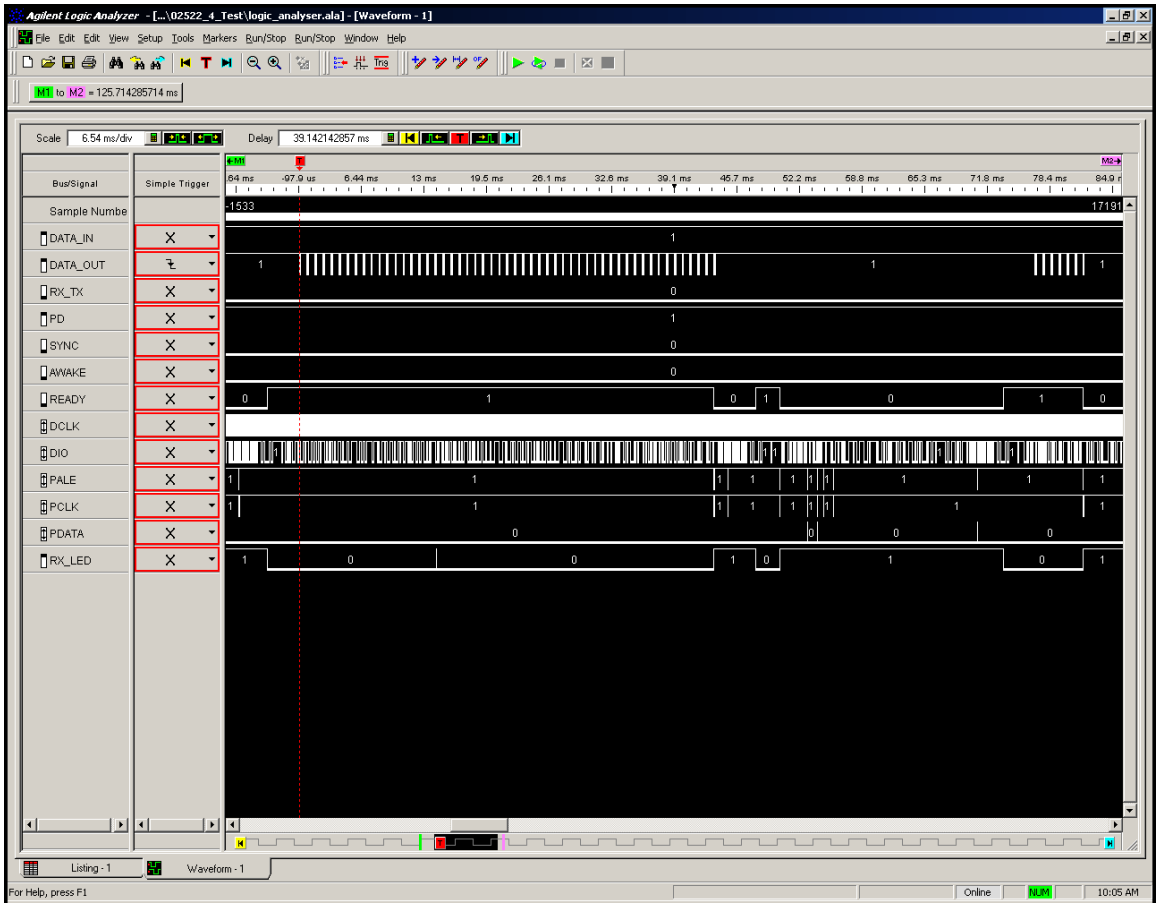


Figure 10: I/O lines during reception of long packets

The figure below is an enlargement of the middle of the transmission shown in Figure 10. The time between UART transmissions is 670us.

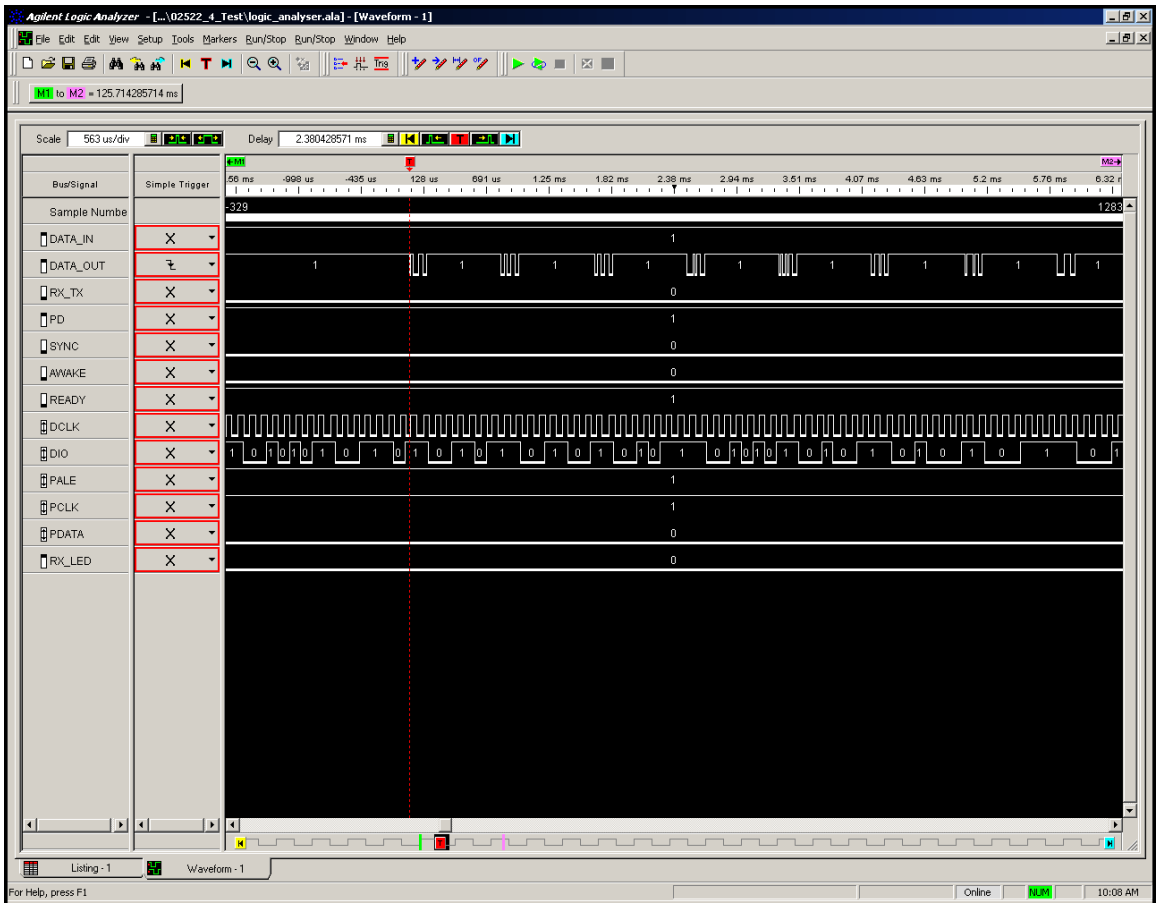


Figure 11: Close-up of middle of long packet

Performance

The C-code presented in this application note works for data rates up to 19.2kbit/s. At rates faster than this, the interrupt routine does not execute fast enough to keep up with the data flow. For a data rate of 38.4kbit/s, the PIC16 microcontroller has only time to execute 65 instructions for each bit. The machine code created by the compiler is not fast enough to do this, but a handwritten interrupt routine in assembly can be able to execute fast enough. Due to time by the microcontroller during mode switching, packet loss is higher at 19.2kbit/s than at the lower data rates. Implementing an optimized interrupt handler in assembly is left as an exercise for the reader.

Operation at 76.8kbit/s is not possible with the current design, as this leaves only 32 instructions for each bit – too few to implement the functionality needed. While the CC1000 has no problems operating at a data rate of 76.8 kbit/s, the microcontroller does. There are two ways to reach this data rate; either use a faster microcontroller or a byte-oriented serial interface to handle the data interface.

The PIC16F876 has a maximum clock rate of 10 MHz when run on a 3.0V supply voltage. It processes instructions at one-fourth this rate, executing 2.5 million instructions per second (MIPS). Preliminary information from Microchip indicates that the PIC18F242 may be a viable alternative. The PIC18 is pin-compatible, and can run at 20 MHz (5 MIPS) at 3.0V. The instruction set is also more efficient, so the PIC18 should be able to handle a 76.8kbit/s data

rate. Because the instruction set is different, the source code would have to be ported to IAR's PIC18 compiler.

Several other available microcontrollers run at speeds of up to 1MIPS/MHz clock. For instance the new megaAVR microcontrollers from Atmel can run at up to 8 MHz with a 3.0V supply voltage, giving a performance of 8 MIPS. This makes operation with very high data rates possible.

Another way of increasing the maximum data rate is to use a byte-oriented serial port to interface with the CC1000. A USART or an SPI interface can be used. This has not been done in this reference design because the USART is used for RS-232 interface and the SPI interface is used for CC1000 configuration. In a design that does not need an RS-232 interface, the USART can be connected to the data interface of the CC1000 instead. This will decrease the microcontroller's processing-load by a factor of almost 8. Byte synchronization will have to be handled by looking for the end of the SOF, and then starting USART reception at the correct time.

Modifications

The RF modem software can be modified for different applications:

- For low-power use, the RF modem software can be modified so that the receiver polls regularly instead of being on all the time. If this is implemented, the preamble should be lengthened so that it lasts longer than the polling period.
- Support for repeaters can be quite easily added by implementing a hop counter.
- For adding error correction and/or detection, an ACK/NAK handshake protocol can be added, together with CRC checking. Some form of forward error correction (FEC) can be beneficial.
- Temperature compensation of the reference crystal may be added for applications where very good frequency accuracy is important.

Source code and PCB documentation

The PCB docuemtnation files (including Gerber-format files for PCB production) and full source code for the RF modem software can be downloaded from Chipcon's web site at www.chipcon.com

References

[1] CC1000PP Plug-and-play reference design, Chipcon 2002
[2] MPLAB ICD User's Guide, Microchip Technology Inc. 2000
[3] Microchip PIC16F87X Data Sheet, Microchip Technology Inc. 1999
[4] LM61 2.7V, SOT-23 or TO-92 Temperature Sensor Data Sheet, National Semiconductor Corporation 2001

Contact Information

Chipcon is a world-wide supplier of RFICs. For further information on the products from Chipcon please contact us or visit our web site. An updated list of distributors is also available at our web site.

Chipcon AS
Gaustadalléen 21
N-0349 Oslo,
NORWAY

Telephone : (+47) 22 95 85 44
Telefax : (+47) 22 95 85 46
E-mail : wireless@chipcon.com
Web site : http://www.chipcon.com

Disclaimer