Application Notes

for

RDM-A8FZ

RF Transceiver Module in 868 MHz

Confidential and Proprietary Information

Any product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by Reindeer Technologies Private Limited in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. Reindeer Technologies Private Limited shall not be liable for any loss or damage arising from the use of any information in this guide, any error or omission in such information, or any incorrect use of the product.

Reindeer specifically disclaims any and all liability and warranties, implied or expressed, for uses requiring fail-safe performance in which failure of the product could lead to death, serious personal injury, or severe physical or environmental damage such as, but not limited to, life support or medical devices or nuclear applications or on-line control of aircraft, aircraft navigation or communications systems or in air traffic control applications. This product is not designed for and should not be used in any of these applications.

Document Revision History

Version No.	Release Date	Description of Changes
1.0	February 26, 2011	Initial Release

Table of Contents

1. Overview

This application note describes the firmware development in MSP430 microcontroller using RDM-A8FZ, a RF transceiver module based on ADF7023 from Analog Devices. The firmware can be ported to other processors easily as it has been developed in high level language. The firmware running in the 16 bit processor take the UART input data and transmit it as wireless data through RDM-A8FZ. Similarly the wireless packet received by the module is given as MSP430 UART output. The RDM-A8FZ module is connected with the processor through SPI interface.

2. Hardware Block Diagram

It is appropriate to have a glance at the hardware diagrams before we go into analysing the firmware. The block diagram of the hardware is given below (Figure 1). The block diagram shows that the UART of MSP430 controller is connected to RS232 port of PC/Laptop through driver ICs. To test the firmware, after connecting both the units to PC, serial terminal applications like HyperTerminal or docklight or other serial console utilities can be used to transmit and receive the data.

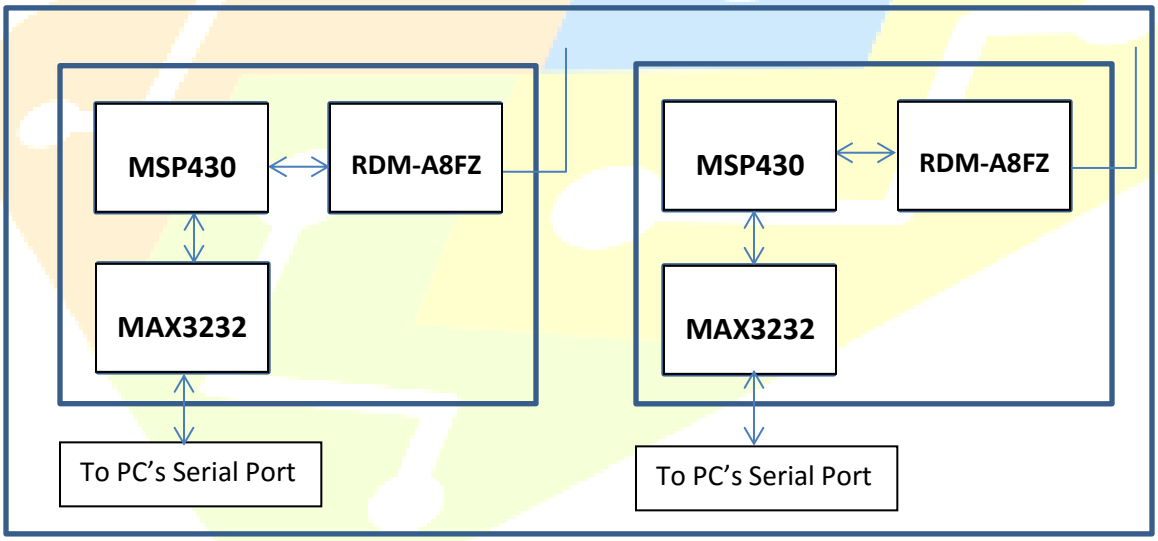

Figure 1: Block Diagram of Hardware Setup

3. Specifications

The sample code is developed for TI MSP430F2XX series microcontrollers. The specifications are mentioned below.

Platform: CCS (Code Composer Studio for MSP430)

Language: Embedded C

Controller: MSP430F2XX seriesDaughter board : RDM-A8FZ

Mother board: TI MSP430F237x EVAL Board

Frequency: 868 MHz

Interface with RDM-A4FZ: SPI

Application: Data communication between RF transceivers.

Serial configuration: 19200bps, Parity: none, Data Bits: 8, Stop Bits: 1

Below mentioned is the list of files used in this application.

- Main.c – This contains the main program
- Msp_adf7023.h – This file contains all the function prototypes and declarations.
- Msp_adf7023.c – This file contains all ADF7023 function definitions.
- hardware.h – This file contains symbolic constant assignment of all the hardware pins.
- includes.h – This file includes all the necessary header files.

4. Explanation of Application Firmware

4.1. Description of ADF7023 Functions

4.1.1. Void Main (void)

The Program enters into the main () function after power-up. All the functions used for configuring the microcontroller and the RF module have been explained below.

4.1.2. void ADF_IssueCommandNW(unsigned char cmd)

Parameters: unsigned char cmd - A single byte command for ADF7023

Return Value: Returns TRUE if the Command has been successfully sent through SPI Interface and FALSE if ADF7023 is not ready to accept the command.

Description: This is the function used to issue command to ADF7023. ADF7023 has an 8 bit RISC processor, which takes the command through SPI from the host processor. This function asserts the CS (Active Low) pin and checks whether the MISO pin of ADF7023 is high for a while. If MISO is not high, then it returns flase. If MISO is high, then it indicates the ADF7023 is ready to take the command. Then the command is sent to ADF7023 through SPI interface.

4.1.3. unsigned char ADF_IssueCommand(unsigned char Cmd)

Parameters: unsigned char cmd - A single byte command for ADF7023

Return Value: Returns TRUE if the Command has been successfully sent through SPI Interface and FALSE if radio controller of ADF7023 is not ready to accept the command.

Description: This is another function used to issue command to ADF7023. This function in turn calls ADF_IssueCommandNW() to give the command to ADF7023. The only exception is that it checks whether the CMD_READY status bit is 1 to indicate that the radio controller of ADF7023 is ready to accept the command. If the radio controler is not ready to accept the command than this function returns FALSE. Otherwise ADF_IssueCommandNW() is called to send the command to ADf7023 and it returns TRUE.

4.1.4. unsigned char ADF_WaitCmdLdr(void)

Parameters: None

Return Value: Returns TRUE when the Command queue of communication processor of ADF7023 is empty (if CMD_READY bit is set) FALSE if MISO output pin of ADF7023 remains low (SPI Interface of ADF7023 is not ready).

Description: This is the function used to wait until the communication processor of ADF7023 is ready to receive the command. If MISO output pin of ADF7023 is low, then the function returns FALSE without waiting for CMD_READY. This function in turn calls ADF_ReadStatus() to read the status word of ADF7023 until the CMD_READY becomes 1 or the MISO output pin of ADF7023 is low.

4.1.5. unsigned char ADF_ReadStatus(ADFSTA_Reg *pStatus)

Parameters: pointer for the variable of structure 'ADFSTA_Reg'

Return Value: Returns TRUE if the ADF7023 status word has been read successfully and FALSE if MISO output pin of ADF7023 remains low (SPI Interface of ADF7023 is not ready).

Description: This is the function used to read the status word of ADF7023. This function asserts the CS pin and checks whether the MISO output pin of ADF7023 is high. If the MISO pin is low then this function returns FALSE as SPI is not ready for communication. Otherwise two NOP commands are given to ADF7023 to retrieve the STATUS word.

4.1.6. void ADF_XMit(unsigned char ucByte,unsigned char *pData)

Parameters: unsigned char ucByte - A byte sent through the SPI interface to ADF7023 communication processor. unsigned char *pData - A byte received as response for the transmitted byte through SPI interface to ADF7023.

Return Value: None

Description: This is the function used to transmit a byte through the SPI interface to ADF7023 communication processor. The first parameter of this function is given through SPI interface to ADF7023. And the response byte of ADF7023 is pointed by the second parameter if it is not NULL.

4.1.7. unsigned char ADF_SyncComms(void)

Parameters: None

Return Value: TRUE if CMD_SYNC is given to ADF7023 successfully and FALSE if the command can not be given to ADF7023.

Description: This is the function used to synchronize the communication processor of ADF7023 with the host processor. This sequence is recommended in the power up. After giving the CMD_SYNC to ADF7023, host processor has to wait for CMD_READY to become 1, to do further communication with the communication processor of ADF7023. This function call is required during power up, WUC wake up and Initialization after CMD_HW_RESET.

4.1.8. unsigned char ADF_MMapRead (unsigned long ulAdr, unsigned long ulLen, unsigned char *pData)

Parameters: unsigned long ulADr - Address to be read in ADF7023 Memory Map
unsigned long ulLen - Length of data to be read in bytes
unsigned char *pData - Starting address of the buffer holding read data

Return Value: TRUE if the specified address has been successfully read. FALSE if the Memory Map read is not successful.

Description: This is the function used to read the specified address in the ADF7023 memory map. The five bit command and 11 bit address is given through the SPI Interface to the communication processor of ADF7023 after checking the MISO pin level. Then a command SPI_NOP is given through SPI which is a 'dummy write' for which the ADF7023 response will be the status word. After this command, the host processor gets the DATA in the successive address locations for each successive SPI_NOP command. This function returns FALSE if the SPI of ADF7023 is not ready.

4.1.9. unsigned char ADF_MMapWrite (unsigned long ulAdr, unsigned long ulLen, unsigned char *pData)

Parameters: unsigned long ulADr - Starting Address of memory location to be write in ADF7023 Memory Map
unsigned long ulLen - Length of data to be read in bytes
unsigned char *pData - Starting address of the buffer holding read data

Return Value: TRUE if the specified address has been successfully read. FALSE if the Memory Map read is not successful.

Description: This is the function used to read the specified address in the ADF7023 memory map. The five bit command and 11 bit address is given through the SPI Interface to the communication processor of ADF7023 after checking the MISO pin level. Then a command SPI_NOP is given through SPI which is a 'dummy write' for which the ADF7023 response will be the status word. After this command, the host processor gets the DATA in the successive address locations for each successive SPI_NOP command. This function returns FALSE if the SPI of ADF7023 is not ready.

4.1.10. void ADF_SetChannelFreq(TyBBRAM *pBBRAM,unsigned long ulChannelFreq)

Parameters: TyBBRAM *pBBRAM - pointer for the structure variable of type TyBBRAM
unsigned long ulChannelFreq – The RF frequency to be set (in Hz)

Return Value: None

Description: This is the function used to set the RF frequency. It eases the programmer's life by allowing him to enter the physical RF frequency in Hz. The expression for calculating the RF frequency of ADF7023 is Frequency(Hz) = FPFD x channel_Freq[23:0] /2^16 Where FPFD is crystal frequency which is 26MHz. The three registers for setting the RF frequency has been updated in this function.

4.1.11. void ADF_SetDataRate(TyBBRAM *pBBRAM,unsigned long ulDataRate)

Parameters: TyBBRAM *pBBRAM - pointer for the structure variable of type TyBBRAM
unsigned long ulDataRate – The RF data rate to be set (in bps)
Return Value: None

Description: This is the function used to set the RF data rate. It eases the programmer's life by allowing him to enter the RF data rate in bits per second. The appropriate registers are updated with the derived value for the particular data rate.

4.1.12. void ADF_SetFreqDev (TyBBRAM *pBBRAM,unsigned long ulFreqDev)

Parameters: TyBBRAM *pBBRAM - pointer for the structure variable of type TyBBRAM
unsigned long ulFreqDev – The RF data rate to be set (in bps)

Return Value: None

Description: This is the function used to set the frequency deviation. The Frequency deviation also can be given in Hertz The appropriate registers are updated with the derived value for the particular data rate.

4.1.13. void ADF_BBRAMDefault(TyBBRAM *pBBRAM)

Parameters: TyBBRAM *pBBRAM - pointer for the structure variable of type TyBBRAM

Return Value: None

Description: This is the function used to initialize most of the registers of ADF7023. It initializes interrupt mask registers, radio configuration registers and set the RF frequency, Frequency Deviation and RF data rate by calling appropriate functions.

4.1.14. void AD_7023_SPISetup(void)

Parameters: None
Return Value: None
Description: This is the function used to setup the host SPI interface to do communication with the communication processor of ADF7023. This function is vital for establishing the communication between host processor and ADF7023. The SPI configurations required to do communication with ADF7023 is given below.

Host SPI Configurations:

1. Master Mode
2. MSB First
3. Synchronous Mode (if USART is used)
4. Inactive level is Low
5. Data is captured on rising edge of the Clock and changed on falling edge.
6. 8 bit character Length
7. 3 Pin SPI
8. SCLK < 5MHz (2MHz is used in this program)

4.1.15. unsigned char ADF_TransmitCarrier(void)

Parameters: None

Return Value: TRUE if the carrier has been transmitted successfully and FALSE if the carrier transmission is not successful.
Description: This is the function used to generate the carrier signal at the particular RF frequency continuously. This is required for testing the transmission performance of the module.

4.1.16. unsigned char ADF_TransmitPreamble(void)

Parameters: None
Return Value: TRUE if the preamble continuous transmission is successful. FALSE if the continuous preamble transmission is not successful.

Description: This is the function used to generate the modulated signal at the particular RF frequency continuously. The preamble gets transmitted out of ADF7023 continuously. This is required for testing the transmission performance of the module.

4.1.17. unsigned char ADF_ReceivePacket(void)

Parameters: None

Return Value: TRUE if ADF7023 goes to PHY_RX State and FALSE if ADF7023 can not be put into PHY_RX State.

Description: This is the function used to put the ADF7023 in PHY_RX State. This function has to be called to enable RF reception to happen.

4.1.18. unsigned char ADF_TransmitPacket(U8 * pucTXData,U8 ucLen)

Parameters: U8 * pucTXData – pointer pointing to the starting address of Transmit data buffer.
U8 ucLen – Length of the Transmit Buffer

Return Value: TRUE if transmit buffer is successfully written into Packet RAM of ADF7023 and FALSE if transmit buffer cannot be written into Packet RAM successfully.

Description: This is the function used to write the transmit buffer into the packet RAM of ADF7023. Once the packet RAM has been updated in PHY_TX state, it gets transmitted through wireless means.

4.1.19. void ADF_CheckInt (void)

Parameters: None

Return Value: None

Description: This is the function used to check the interrupt source by checking the individual bits of interrupt source register. It sets the corresponding flag when the interrupt source is identified. This function will be called inside an Interrupt Service Routine (ISR) to ascertain the source of interrupt and to do further process. And it clear the bits of the Interrupt source register to identify the next interrupt.

4.1.20. void ADF_CheckInt1 (void)

Parameters: None

Return Value: None

Description: It clears both the Interrupt source registers. This is called at the exit of the ISR.

4.1.21. unsigned char ADF_FirstConnect(void)

Parameters: None

Return Value: returns TRUE if host processor successfully connected to ADF7023 and Returns FALSE if the connection to ADF7023 failed.

Description: This is a function called for the first time when the ADF7023 is connected to host processor. It synchronizes the communication with host processor and configures the ADF7023.

4.1.22. unsigned char ADF_ConfigureRadio(TyBBRAM *pBBRAM)

Parameters: Pointer to the TyBBRAM structure

Return Value: returns TRUE if host processor successfully configures the ADF7023 and returns FALSE if the configuration of ADF7023 failed.

Description: This is a function called to configure ADF7023 with by writing the BBRAM register values. This function takes the pointer to the TyBBRAM structure.

4.2. Description of Interrupt Service Routines

ADF7023 IRQ Handler:

In this sample code, PORT2 interrupt of MS430 microcontroller is used to capture ADF7023 IRQ interrupt. In this ISR, ADF_CheckInt() is called initially to check the source of interrupt. That is to check whether the CRC check after RF reception or the completion of RF transmission or other sources of interrupt for ADF7023 has caused the interrupt. In this sample code, the IRQ handler is wriiten to handle the RF reception. As the variable length packet has been used, the first byte of the received packet will be the packet length. So the packet length has been read first and then the packet. Once the packet has been read from the ADF7023 packet RAM, the transceiver is put into receive mode to receive the data.

UART Reception Interrupt Handler

The UART reception handler receives the UART data and stores it in the buffer. A 200ms timeout is used to detect the end of the frame for the serial data. This has been done using TimerA and its interrupt. After every byte reception the millisecond tick timer count is reset and the timerA is started again.

TimerA Interrupt Handler

This 1ms timer interrupt has been started once a byte has been received in the UART. If the time elapsed between two UART data reception is 200ms, then it has been identified as the end of UART data and the UART data will be transmitted wirelessly through ADF7023.

5. Header Files

The sample code contains the following header files:

1. Hardware.h

2. Msp_adf7023.h
3. Prototypes.h
4. Includes.h

Hardware.h

This file contains the assignments of symbolic names for the port pins. The assignment of the symbolic names for port pins makes the program readable.

Msp_adf7023.h

This file contains the address definition of registers in ADF7023 and the associated data structures.

Prototypes.h

This file contains the prototypes of the functions used to interface the ADF7023.

Includes.h

This file includes all the header files required for running the sample code.

6. Using the sample code with other Microcontrollers

As the sample code has been written in high level language, it can be ported to any microcontroller. The hardware.h has to be edited to assign the port pins of the selected microcontroller.

7. Contact Us

7.1. Technical Support

Reindeer Technologies Pvt. Ltd. has built a solid technical support infrastructure so that you can get answers to your questions when you need them. Our technical support engineers are available Mon-Fri between 9:00 am and 6:30 pm Indian standard time.

The best way to reach a technical support engineer is to send an email to *support@reindeersystems.com*. E-mail support requests are given priority because we can handle them more efficiently than phone support requests.

7.2. Sales Support

Our sales department can be reached via e-mail at *sales@reindeersystems.com*or by phone at +91-44-45022335. Our sales department is available Mon-Fri between 9:00 am and 6:30 pm.

Reindeer Technologies Pvt Ltd

Excellence through Innovation™

No. 77, Baskar Colony

Virugambakkam

Chennai – 600092

India.

Phone: 91-44-45022335

Email: sales@reindeersystems.com

Website: www.reindeer-tech.com