ECSE426 Microprocessor Systems - Fall 2009

Experiment 3: Wireless Transceiver Interfacing
Demos: Oct. 22, 23
Lab Notes: Oct. 25

Objectives

This experiment will familiarize you with the SPI interface and will teach you how to configure and use a modern wireless transceiver operating in the 2.4 GHz ISM band. You will be re-using the USART in asynchronous mode to build a menu-driven interface to the wireless transceiver, allowing you to manipulate many of its parameters interactively. The USART (in synchronous mode) will be used to communicate with the wireless transceiver and send data packets.

Background

CC2500 Wireless Transceiver

The Texas Instruments CC2500 was selected for its versatility, low complexity and very integrated functionality. You are strongly encouraged to read and understand the following sections in its datasheet:
1, 2, 3, 4.9, 5, 6, 8, 9, 10,11, 15 (PktLen < 256), 18 (FEC is optional, but FYI), 19, 20, 21, 29, 32 (Find the registers that you will need to control).

MSP430 SPI

You will need to read and understand the details of the MSP430 USART in SPI mode and the procedure to configure it for a compatible operation with the CC2500 transceiver.

You are required to use the SPI port by passing the data through the CPLD (MSP430 Port 5). You will need to program the CPLD such that it propagates the SPI traffic to pins on header H1.

MSP430 USART in asynchronous mode

You will re-use the USART driver code (asynchronous mode) from experiment 2 to send commands and data to your MSP430 that is controlling the CC2500.

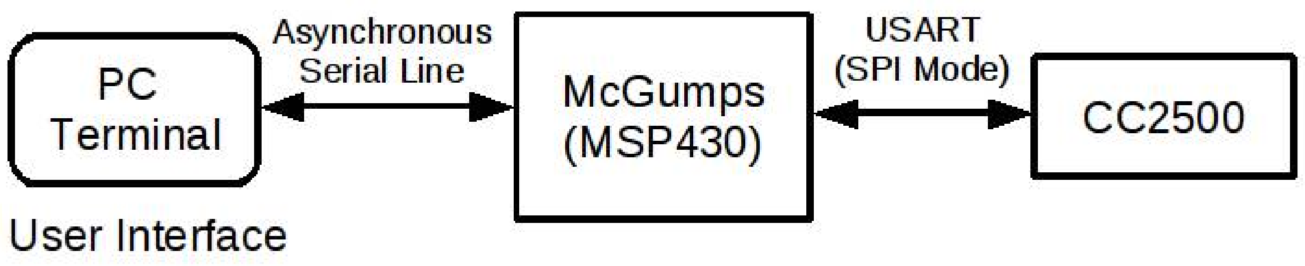

Illustration 1: Overview of the system

Tasks

Hardware SPI interface timing capture

Your first task is to be able to send SPI frames that are compatible with the CC2500. For this you will need to understand the SPI timing diagram of the CC2500 and configure the MSP430 USART module. You will then use the logic analyzer (Ant16) available in the lab to capture:
–A strobe command to reset the chip (SRES)
–A write to PKTLEN with a value of 0x55
–A read from the SYNC0 register which we expect to be 0x91 (reset value) if the register was not modified.

Your capture should explicitly name the signals and be recorded as part of your lab notes. Those captures will be graded.

Note : You may first send the SPI frames with no CC2500 chip connected and ensure that the chip select and bits that you send match what the CC2500 is expecting. You can also do a loopback test of the SPI (MISO connected to MOSI) to ensure that you can send and receive SPI data correctly. Once you confirm proper operation with the logic probe, you can hook up the CC2500 and then capture the required logic analyzer shots.

User Interface to modify and display the CC2500 registers

Your second task is to build a user interface to dump the state (registers) of the transceiver. Through this interface, you should display a list of options (via a small menu) to the computer to which the serial port is connected. A menu item should be provided to dump all the registers. The dump should list all the registers in the following format:

```
Address : NAME : Value@Address
0x00    : IOCFG0 = 0xVV
0x01    : IOCFG1 = 0xVV
[...] and so on [...]
```

(where VV is the value read at the Address)

Another menu element will allow you to modify the content of the register at a given address. When modifying a register in the transceiver, the user interface must first read the current value in the register, display it with the new value being prompted in an intuitive way, for example:

```
Register Modification for PKTLEN:
Value = 0x25 changed to => 0x<User enters a new value here>
```

If the user presses the escape key, the register modification should be aborted, the enter key will accept the changes.

Send and receive data packets

Your third task is to be able to send and receive data packets of less than 64 bytes. Your test interface should ask you for a destination address, let you type in a string (limited to the max packet length that you support) and then send the information to that address.

You will need to demonstrate proper operation of your radio interface by sending and receiving a few packet from the wireless echo test module in the lab. The proper channels, frequency and other RF parameters will be provided to you.

Your program is not required to be constantly scanning for packets. However, there must be a way to force the CC2500 to go in "receive mode" and listen to traffic. When a packet is captured, it must be displayed in hexadecimal format in a way that is easy to comprehend.

Important Considerations

- Don't forget to get your RF2500 board re-programmed by the TA such that the MSP430F22xx on that board does not interfere with your SPI communication with the transceiver. Note that you can do this yourself too by exploring the CD that is part of the kit and installing the TI coding environment or IAR Workbench on your own computer. You only need to program the MSP430F22xx such that all the pins are inputs and then execute a while(1) or enter a low power standby mode that blocks the processor from continuing execution.
- Ensure that you correctly wire up your RF2500 module such that it receives 3.3V. Test all the SPI bus pins by using the GPIO control on your MSP430 and ensure that the CPLD is passing the information correctly to the transceiver.
- Some application notes are available on WebCT to help you write an interface to the CC2500. TI also provides driver code for the CC2500. You can inspire yourself from those to complete your work. You may even re-use the TI driver code, provided that you cite your sources and that you retain the copyright header information. However, it is your job to make the driver code work with Rowley on the McGumps board is you choose this route.

Bonus Features

- The on-chip FLASH memory in the MSP430 is in-system programmable, meaning that the MSP430 can modify its own FLASH without any external programmer. It is thus possible to use a given block of this non-volatile memory to store parameters that will be preserved across reset and power cycles. Saving and restoring configuration data from FLASH will be a required feature in the final project, so you may attempt this feature now if you have time. In the context of assignment #3, this can be put to good use by saving and restoring the radio transceiver parameters.