Final Project Report

Application Programming Interface

For Radiofrequency Transceiver

Senior Independent Project(ECE4910)

Student: Jaehyun Kim

Supervising Faculty: Bruce Land

Purpose

The purpose of this project is to develop an Application Programming Interface(API) for 2.4 GHz radiofrequency(RF) transceiver, nRF24L01+, by Nordic Semiconductor. The API is to be used by any engineers who wish to use the transceiver to easily call functions in API to implement systems using wireless communications using the RF transceiver.

Introduction

Wireless communications play very important role in many of electronics. The application of wireless communication system includes not only the areas described by home electronics but also manufacturing, industrial Local Area Networks, military purposes, or in any other areas where wired transmission of information is either impossible or impractical. Wireless communication can be implemented by methods including but not limited to radiofrequency, infrared light, sound wave and visible lights. Basically, any form of media that is capable of transferring energy from the source to destination can be used for wireless communication. In our project, we resort to implement wireless communication by means of radiofrequency electromagnetic wave.

Motivation

Radiofrequency signals(RF signal) are the most widely used means of wireless communication in home electronics, next to infrared light. Unlike infrared light, however, RF-signals have advantage in the fact that the signal can travel around obstacles to reach the destination. Also, RF-signals allow reliable communications across wide range of distances, typically, ranges that are much(more than 10 ~ 1000 folds) greater than the range offered by any infrared transceiver pair.

On the market, there are numerous vendors that offer Application Specific Integrated Circuit(ASIC) specifically designed to generate and control wireless signals that requires only few external passive elements, at very low prices. Depending on the base carrier frequency, RF-signal transceiver ASICs cost from as low as less than $2 for devices having sub GHz carrier frequency, to $20 dollars for device operating at 2.4GHz.For our purposes, we chose RF-signal drivers with 2.4GHz carrier frequency because of its versatility. Higher base frequency allows more versatility in terms of size. RF-signals at 2.4GHz allow us to make antenna size very small for the same antenna gain, thus allowing very small device size. An engineer with experience in using microcontroller(MCU) for implementation of digital systems, can easily implement simple wireless communication systems using a general purpose microcontroller and an RF-signal generator/driver ASIC.

Theory of operation

a. *RF-signal driver operating modes*

When the device is powered down, the quiescent current falls to very low level, allowing the system to save power, especially in case of mobile application where battery life is important. When the device is powered up The RF-signal driver, or RF-signal transceiver primarily operates in one of 2 modes, *Primary Receiver(PRX)*, or *Primary Transmit(PTX)*. When the device is in PRX mode, the device constantly monitors the channel at specified frequency, which is also user configurable from 2.4GHz to 2.527 GHz, and specified data rate(250KHz, 1MHz, or 2MHz), for the right address. If the device determines that the address is right, then it stores that in Receive First-In-First-Out register(RX-FIFO), whose size is 32 bytes, and sends Acknowledgement(ACK) signal to the transmitter, so that the transmitter device (TX-device) knows that the data packet has been safely received. When the device is in PTX mode, it waits for the user(meaning microcontroller who acts as master in serial communication SPI) to write to TX-FIFO, whose size is also 32 bytes. As soon as there are fresh data in the TX-FIFO and the CE pin is driven low(by MCU's general purpose input-output pin(GPIO)). After the data is transmitted, the transmitting device switches momentarily to PRX mode and listens for ACK signal sent by the device receiving data. After waiting for given amount of time(user specifiable), the transmitting device tries to send the packet again for given number of times(also user specifiable). After the given number of times of retry, the transmitter gives up on sending and the packet is lost. Note that the user can opt not to wait for the ACK signal, and not to retry at all.

b. *Microcontrollers to RF-signal driver communication.*

The communication between the two device is carried out by the Serial Peripheral Interface Bus(SPI). SPI protocol has 8 total modes of operation and this particular device uses rising leading edge with sampling at the leading edge, with sending MSBits first. The SPI communication between the two devices uses 4 signals including MISO, MOSI, SCK, and CSN. MISO, MOSI, and SCK are connected to the MCU pins that are dedicated to the SPI communication. CSN pin can be assigned to any GPIO of MCU. The start of the communication is initiated by the MCU lowering the CSN pin. After that, MCU transmits data through MOSI pin and simultaneously receives data through MISO pin. MCU terminates the data transaction by pulling CSN pin up. The first byte received by MCU is always the contents of STATUS

MCU can configure the RF-signal driver through two mechanisms. MCU can either write to the registers in the RF-signal driver, or directly issue command. The two methods are mutually exclusive in functionality. That is , some configurations must be done by directly issuing commands cannot be done by writing to some registers and vice versa. Note that when the user wishes to write to the specific registers in RF-signal driver, it issues command called W_REGISTER and clock in the data to be written to the register.

Drive CSN low	Command Byte	Drive CSN high

Fig1.1 Configuring by directly issuing command.

Drive CSN Low	Command Byte(Write Register with lower bits specifying the register in interest)	Byte(s) to be written to that register.	Driver CSN High

Fig1.2 Configuration by writing to the registers in RF-signal driver

If the user wishes to transmit data in the air, the user writes 0 to PRIM_RX bit in CONFIG register, to turn it into PTX device and starts clocking in the data to be transmitted after issuing W_TX_PAYLOAD. Note that the data in TX-FIFO will only be transmitted when the CE pin is driven high.

Drive CSN Low	Command Byte(Write Register with lower bits specifying CONFIG register)	Write 0 to PRIM_RX bit. Make sure to keep the other bits intact.	Driver CSN High

Fig1.3 Turning the device in PTX device for data transmission.

Drive CSN Low	Command Byte(W_TX_PAYLOAD)	Byte(s) to be written to the TX_FIFO, LSBytes first.	Driver CSN High

Fig1.4 Writing to TX-FIFO

The device transitions to PRX mode when PRIM_RX bit in CONFIG register is written 1. If the user wishes to read data received while the device is in PRX mode, the user issues R_RX_PAYLOAD and reads data clocked in through MISO pin.

Note that the number of bytes to be transmitted on air is agreed between the two stations(two autonomous systems communicating through the air by use of nRF24L01+) unless they agree to use Dynamic Payload Length(DPL) feature offered by the RF-signal driver.

The RF-signal driver also offers IRQ pin which serves as the source of interrupt for the MCU. The pin is driven low from the idle state(HIGH), when one of the three source of the interrupt occurs. The IRQ pin is driven low when 1) PTX device has successfully finished transmitting the data(TX_DS), 2) PRX device has successfully received transmitted data(RX_DR), and 3) PTX has failed to transmit the data after the maximum number of retry specified by the user. Using the IRQ pins, MCU can sense the important events happening while carrying out other processes. Note that each of three source of interrupts can be suppressed by writing 1 to interrupt mask bits in CONFIG register.

Instead of using IRQ option, the user can instead choose to pole the interrupts flag inside the RF-signal driver. The interrupt flags are located in STATUS register, and the user has to manually clear them.

Development Process

For detailed description of the how each demo works, please read the readme files attached in the appendix. Note that throughout the development process, the functions in API were debugged an modified. The final versions of each demo programs uses the finalized version of API.

c. Getting the first packet to work

Most of the time spent in developing API was spent in this part. In this stage, we debugged, MCU to RF-signal driver communication, configuration of RF-signal driver, and finally the communication between two RF-signal drivers. In the experiment, we kept one driver in PRX mode, and the other driver in PTX mode. Thus, the data transmission was one-sided.

d. Simple Exchange Counter

In this system, we tried to switch the drivers between PTX and PRX device. The two systems that each consists of a single MCU and a RF-driver, was configured to receive and transmit. Each device receives a byte from the other device, adds 1 to it, and sends it back to the transmitter. The main problem in this stage of development was switching the driver between PRX mode and PTX mode. We still haven't figured out the source of the problem, but the solution we found was to power down the device before each transition.

e. Rock Paper Scissors game

In this system, two players can play simple rock-paper-scissors game. The prompts asking for choice and displaying of the result was done through RS232 communication between each MCU and PC. In this stage, we tried to create an interesting application of the API, so that the possible user of API can enjoy learning how to use API.

f. Multiciever

In this system, we tried to implement one base station which receives data from multiple satellite stations. The main problem in this stage of development was multiplexing between the multiple stations. Several possible methods of implementation were considered. The RF-driver allows storing up to 6 addresses. Upon the reception of the data, the base station in PRX mode can check RX_PW_PX registers to determine which pipe the data was received. However, we determined that the method was cumbersome. Plus, the method had possible problems when one of the satellite stations misbehaved, which will gunk up the system. Instead, we decided to implement the software time-division multiplexing. During each slot, the base station would listen only for satellite station with specific address. For next slot, the base station will change its pipe address and listens for next satellite stations. This method has big inefficiencies in power consumption and the processing bandwidth of the satellite station since the method requires the satellite stations to virtually continuously send data until it is received. To ameliorate such inefficiencies, we considered token passing method where the base station would first deliver token(via transmitting pre-agreed byte) to a satellite station with specific

address and listens for that station upon the reception of ACK packet. That way, the satellite station can just wait for the RX_DR interrupt while processing other important functions. This method however was not implemented due to the time constraints.

Conclusion

The driver did not provide the robust wireless communication environment as we hoped for. During the development process, we realized that there are certain ways that we have to use the driver to get it to work. We could not figure out the specific reasoning behind the specific ways we have to control the driver. Due to time constraints, we could not further debug the API for it to be completely deterministic and fool-proof. However, the basic structure and the functionality was established and any user with moderate experience in digital systems using microcontroller can benefit from developed API.

Appendix A. Demo-code description

i) Exchange Counter

A) Setup Guide.

- The program in folder ‘a’ is downloaded to the MCU of side A.
- The program in folder ‘b’ is downloaded to the MCU of side B.
- Before the game starts, both sides should have the program downloaded.
- They should have identical pipe address(modify ‘#define PIPE_ADDRESS XXXXXXX’ part).
- The pipe address should not be used by any other systems using the RF module with same carrier frequency(this example uses 4.001GHz).
- UART connection should be available for the both system.

B) Operation Description.

- Side A starts by sending 0 to side B.
- Side A waits until side B receives and sends back ACK packet.
- Side A displays that ‘0’ has been transmitted.
- Side B displays that ‘0’ has been received.
- Side B increments the received number(0 + 1 = 1) and sends it back to side A.
- Side B waits until side A receives and sends back ACK packet.
- Side B displays that ‘1’ has been transmitted.
- Side A displays that ‘1’ has been received.
- Side A increments ‘1’(1+1=2) and sends it back to side B.
- The system repeats the cycle.

ii) Rock Paper Scissors

A) Setup Guide.

- The program in folder ‘a’ is downloaded to the MCU of side A.
- The program in folder ‘b’ is downloaded to the MCU of side B.
- Before the game starts, both sides should have the program downloaded.
- They should have identical pipe address(modify ‘#define PIPE_ADDRESS XXXXXXX’ part).
- The pipe address should not be used by any other systems using the RF module with same carrier frequency(this example uses 4.001GHz).
- UART connection should be available for the both system.

B) Operation Description.

- Player A always starts first.
- Player A is prompted to choose from Rock, Paper, and scissors by entering 1~3 which corresponds to the choices in RPS game.
- If Player A chooses valid choice, the information is transmitted to Player B and Player B is prompted to choose his choice.
- If Player B chooses valid choice, it displays the result of the game, for both players and goes back to the first stage prompting Player A for the choice again.

iii) Multiciever

A) Setup Guide.

- The program in folder ‘a’ is downloaded to the MCU of satellite station 0 and 1(with some modification).
- The program in folder ‘base’ is downloaded to the MCU of based station.
- They should have identical pipe address(modify ‘#define PIPE_ADDRESS_X XXXXXXX’ part).
- The pipe address should not be used by any other systems using the RF module with same carrier frequency(this example uses 4.001GHz).
- UART connection should be available for the base station system.

B) Operation Description.

- Base station only receives and satellite stations only send.
- Base station allocates window for each stations(50ms). If no data is received within that window from the corresponding satellite station, base station switches to the next station.
- Base station switches to the next station simply by changing its receive address.
- Satellite station-0 has address PIPE_ADDRESS_0 and sends alphabet a~z.
- Satellite station-1 has address PIPE_ADDRESS_1 and sends alphabet A~Z.
- All satellite stations attempt to send continuously.

Appendix B. Schematics