EECE 690/890
Digital Radio Hardware Design

Team 3
Assignment 2

Informal Design Review: Thurs 10/15/98
Deliverables Due: Tues 10/20/98

Introduction

This is the second in a series of assignments designed to guide you through the tasks needed to complete the software design. Specific tasks are detailed for each team member so that each person has a well-defined job and deliverable (material to be turned in by the due date). However, the tasks are also interdependent, so you should work together when appropriate.

We will have an informal design review at the end of next week. You should have the bulk of your code written and printed out so that we can go over it as a group and be sure the interfaces will all work.

Attached to this assignment is an additional page giving a preview of remaining tasks that will need to be completed before the PDR. As always, you are encouraged to work ahead if you can.

Firmware Engineer 1 Tasks

Your main task for this assignment is to write code specified in the Top-Level Executive flowchart discussed in class (or a suitable derivative of this flowchart approved by your teammates and instructor). A somewhat detailed list of suggested subtasks is given below to help you organize your work.

- If you have not already done so, get a notebook and start putting items into it. You should collect the following items that will be needed and/or helpful in later steps:
  - Include file p17c44.inc and the one created by Test/Diagnostic Engineer in Assignment 1 which gives names of ports/pins.
  - Delay routine/code written by Firmware Engineer 2 in first assignment.
  - The example code given out with the first assignment.
  - The firmware flowcharts handed out in class.

- Review the Top-Level Executive flowchart and determine what supporting routines you will need. Some of the routines needed and possible sources for the code are:
  - Initialization function (being written by Firmware Engineer 2 in assignment below). Just agree on an interface (function name).
  - Function to pause for 1 second (adapt delay routines from old program, or get from Firmware Engineer 2's first assignment).
  - Function to power up receiver (to be written by you in this assignment - see include file for ports/pins needed to be set/reset).
  - Function for programming synthesizer (being written by Firmware Engineer 2 in assignment below). Just agree on an interface (function name and argument(s)).
  - Function to pause for synth/data-slicer settling time determined in Homework #5 (adapt delay routines from old program, or get from Firmware Engineer 2's first assignment).
  - Function to check for a signal present (to be written by you in a future assignment).
  - Function to check for a valid ID (to be written by you in this assignment).
  - Function to power-down/sleep (adapt from power up code above).
- Write a pseudo-code description of the Top-Level Executive. You should be more detailed than the flowchart and you should call out the names of the functions identified above that perform the different actions needed. Keep your Top-Level Executive simple! Delegate most actions to called functions. Pseudo code for lower level functions is not required at this stage.
- Write the top-level executive code and the called routines (except for the Valid ID check). You may develop the executive first and then the called routines, or the other way around. You will know the best approach.
- Test your code.
  - Use stubs for the initialization, synth programming, and packet checking code.
  - Use conditional compilation to "comment out" delay routines so that the simulation can run fast enough.
- ***DOCUMENT YOUR CODE*** with appropriate comments before each function, and for each major section!
- Check with your teammates to be sure that you all agree on interfaces and are ready to integrate the code together!!

Your deliverables are listed below:

- Your pseudo-code description of the top-level executive.
- Your (documented) code - including any header files you used.
- A description of how you tested your code and if it seemed to work. Your "coverage" in the test should be complete. That is, make sure all branches have been tested.

Firmware Engineer 2 Tasks

Your main tasks in this assignment are to develop code that initializes the microcontroller and the transceiver circuitry at power-up, and code that programs the synthesizer. This will be needed by your teammates, and by Team 1 in testing the synthesizer circuits (which may actually be done before final layout and assembly of the phone). A list of suggested subtasks is given below to help you organize your work.

- If you have not already done so, get a notebook and start putting items into it. You should collect the following items that will be needed and/or helpful in later steps:
  - Standard include file p17c44.inc, plus the one created by your company's Test/Diagnostic Software Engineer in Assignment 1 which gives names of ports/pins.
  - Delay routine/code written in first assignment by you.
  - Port initialization code developed in first assignment by Firmware Engineer 1.
  - The example code given out with the first assignment.
  - The firmware flowcharts handed out in class.
  - Data sheet for PIC 17C44 microcontroller.
- Review the flowcharts to understand where your code fits into the overall design.
- Review the 17c44 data sheet to understand what initializations are needed in the uC.
  - Read about the startup and watchdog timer functions.
  - Read about the states of special function registers at startup.
  - Determine what, if any, interrupts need to be masked off and how to do this.
- Review the project block diagram to understand what initializations are needed in the phone. You may want to talk with other teams in your company to get some of this information. **DOCUMENT WHAT YOU LEARN!** It will help you as well as others.

  Some of the things that you should consider in initialization include:
  - Initializing I/O port directions as needed (e.g. RSSI = input, TX = output, etc.)
  - Programming the synthesizer registers. Although some programming will be done when the frequencies are scanned, there are registers that will remain static. The best thing would be to just initialize the synth for normal operation on channel 0. Talk to Team 1 and review the example code handed out in class, but remember that this code was for a slightly different synthesizer IC. Review the synth data sheet to understand the differences.
  - Setting up the PLD to a known state (and resetting it?). Talk to Team 2.
  - Initializing the LCD. (See the example code given out in class, and/or work with the Test/Diagnostic Engineer who is developing code for talking to the LCD.)
- Write pseudo-code descriptions of the main initialization routine (function). You should call out the names of the functions identified above that perform the different actions needed. Keep your code simple in this routine. Delegate most actions to called functions.
- Write pseudo-code descriptions of the called routines. Some of these will be simple, but others, such as the synth initialization, will require more work, some careful thought, and information from other members of your company (Talk to Team 1!). When possible, port code from the program given out with the first assignment.

- Write and document the init routines.
- Modify your synth init routine to provide one that can be used to set the synth to any specified channel from 1 to 10. You may not need to program all the registers, so this code may be a subset of (or a function called by) the synth init routine.
- Test the routines.

Your deliverables are listed below:

- Your pseudo-code description of your routines.
- Your (documented) code - including any header files you used.
- A description of how you tested your code and if it seemed to work. Your "coverage" in the test should be complete. That is, make sure all branches have been tested.

Test / Diagonostic Engineer Tasks

Your main tasks for this assignment are to develop a "Hardware Test Bench" for testing code written for the uC and to port the LCD display code that was written for the old synthesizer/display board (see code handed out with first assignment).

The first activity will allow you to test the LCD code more easily than on the software simulator in MPLAB, and will also provide a method for other members of your company/team to test their software and circuits. For example, Firmware Engineer 2 may want to try testing the "ringer" code written in assignment 1 and Team 1 may want to try testing the synthesizer circuits in conjunction with Firmware Engineer 2's code.

A list of suggested subtasks is given below to help you organize your work.

- If you have not already done so, get a notebook and start putting items into it. You should collect the following items that will be needed and/or helpful in later steps:
  - Include file p17c44.inc, plus the one you created in Assignment 1 which gives names of ports/pins.
  - Delay routine/code written by Firmware Engineer 2 in first assignment.
  - The example code given out with the first assignment.
  - The firmware flowcharts handed out in class.
  - The data sheet for the PIC17C44 uC.
  - The data sheet for the LCD.
- Decide if you will use byte-mode or nibble-mode programming for the LCD based on how many pins are available on the uC (see your first assignment and talk to other teams to resolve any uncertainties).
- Read the section of the data sheet for the uC that describes how to design an RC or crystal oscillator for the uC clock. This only takes a few external components. You may want to

design an RC circuit for the test bench since we do not yet have the necessary crystals. (What frequency does your company plan to run the uC at?)

- Review the data sheet for the LCD and sketch a schematic for your complete "Hardware Test Bench", which includes both the uC and the LCD. Keep it simple. You will probably want to build it on a "proto board" to save time and effort. Don't forget to use a potentiometer for contrast adjust and to include any necessary bypass caps.
- Construct your circuit on a protoboard using wires only as long as necessary. Plan on using a ribbon cable to connect the LCD to the proto-board. You can do this by plugging one of the in-line pin strips available in the electronics shop downstairs into the proto board. There are also single-turn and multi-turn pots in the shop that are suitable for use on a protoboard.
- Study/document the LCD display programming needed. See the LCD data sheet as well as the code given out with the first assignment.
- Port the code for displaying messages on the LCD. (Work with other team members to define the best interface for this.)
- Write code that displays Hello World on the LCD and test it. (We have several PIC17C44's on order, and one LCD display in the Comm lab at present).

Your deliverables are listed below:

- The schematic of your test board.
- Your (documented) code - including any header files you used.
- A demonstration of your working board/code.

Team 3 Future Assignments

The following gives an overview of the tasks remaining after task 2.

Firmware Engineer 1

- Study/document the interface to the PLD and write code for sending/receiving packet type and security ID and for checking to see if a valid ID has been received.
- Write call init code and develop a test plan for it.
- Test the code.
- Integrate with Firmware Engineer 2's code.

Firmware Engineer 2

- Test your ringer code written in first assignment.
- Write Call reception code and develop test plan for it.
- Test the code.
- Integrate with Firmware Engineer 1's code.

Test/Diagnostic Engineer 1

- Work with other members of your team to test their code on the hardware test bench.
- Work with RF team to test synthesizer interface.
- Write diagnostic code for placing phone in various modes that will be useful when phone is first activated.