ECE371 – DESIGN PROJECT #2, FALL 2017

Electrical and Computer Engineering Department
Portland State University
Portland, OR 97207

REFERENCES

Hall Text Chapter 4 and Chapter 5; The BeagleBone Black System C.1 Reference Manual on http://elinux.org/Beagleboard:BeagleBoneBlack; and the TI Sitara AM 335X manual at www.ti.com/lit/ug/spruh73p/spruh73p.pdf

IMPORTANT FIRST STEP

As with any design project, it is very important to read through the entire project description to get an overview before starting to work through the project. Make notes on the requirements, deliverables, and timelines, if given, etc. These notes are the start of your design logbook.

INTRODUCTION

You will be developing this project for one of the BeagleBone Black boards connected to stations on the west end of the Tek Lab and in the Intel Lab. There are three major parts to this design project.

1. In the first part you will learn is to learn how to control GPIO pins and how to turn the 4 BeagleBone Black USR LEDs on and off in a specified pattern with delay loop timing.

2. In the second part of this project, you will develop an interrupt procedure that services an interrupt request from a push-button switch connected to GPIO1_31. The first time the button is pushed, the interrupt procedure will start the LED display pulsing. The next time the button is pushed, the interrupt procedure will stop the pulsing LED display.

3. In the third part you will use a Sitara AM3358 Timer on an interrupt basis to time the LED lighting, instead of using a delay loop

GOALS AND OBJECTIVES

The major goals of this exercise are:

1. To give you some practice following the "Fast is Slow" rule, after you use up the "5-minute rule". In fact this exercise is specifically designed to be relatively easy, if you follow the "Fast is Slow" rule but, due to the amount of detail, very difficult and prone to many errors, if you try to just speed through it without carefully studying all the reference material and developing detailed task lists as described in the text.
2. To give you some practice in working through and using text examples and data sheets to quickly and efficiently develop programs that work with GPIO pins, an interrupt controller and a timer.
3. To show you how to successfully develop a multi-part program by systematically building and testing one piece at a time.

The specific objectives of this exercise are that at the end of this exercise you should be able to:

1. Output highs or lows on GPIO pins.
2. Initialize a GPIO pin as an input or as an output
3. Set up a "flag" that can be used to switch back and forth between two different actions.
4. Utilize AM3358 GPIO pins as interrupt inputs.
5. Initialize and use an AM 335X Interrupt Controller.
6. Write an interrupt service procedure that services a IRQ interrupt request.

DELIVERABLES

A. An **as-you-go** design log that shows your thinking, all the steps you took, and results at each step. **Make sure to create quality documentation that you would like to receive, if you were assigned to take over the project halfway through it.**

B. A clearly written algorithm for each of the development sections of your program, for the final LED pulsing program, and for the final button driven LED program, and for the final version that uses a timer instead of a delay loop.

C. A signed off copy of the .s file for the delay-loop flasher and a signed off copy of the .s file for the timer LED program. Note that your .s files should have full headers and comments. (Be prepared to answer questions at signoff.)

D. A written and signed statement that, "I developed and wrote this program by myself with NO help from anyone except the instructor and/or the T.A. and I did not give any assistance to anyone else." (Any evidence of joint work will result in project grades of zeros for all parties involved.)

Grading rubric is as follows:

GRADING KEY FOR ECE 371 DESIGN PROJECTS D.V. HALL FALL 2017

LOG (DETAILED AND "AS YOU GO"	**20 MAX**	_____
TASK LISTS/ALGORITHMS (DETAILED AND COMPLETE)	**20 MAX**	_____
WORKING PROGRAMS (Demos required)	**40 MAX**	_____
COMMENTS (CLEAR AND USEFUL)	**10 MAX**	_____
OVERALL ORGANIZATION	**10 MAX**	_____
TOTAL	**100 MAX**	_____

PROCEDURE OVERVIEW

Part 1 Lighting LEDs

1. Study the BeagleBone Black System Manual to determine which GPIO pins are connected to the 4 USR LEDs and the logic level required to turn on one of the LEDs.
2. Write the **high level algorithm** for a program that lights LED0 for 1 second, then adds LED1 so that two LEDS are on for one second, then adds LED2 so that 3 LEDs are on for one second, then adds LED3 so that all 4 LEDs are on for one second, turns off all the LEDs for one second and then repeats the pattern over and over and over. (Theaters used to have lights cycle through a rotating display like this.)
3. Carefully work through the section of Hall Chapter 4 that describes the AM3358 GPIO pins and the memory mapped registers that control them. As part of this, determine the registers that control the GPIO pins connected to the LEDs. Also study the section that shows how to set up bit templates for working with the GPIO registers.
4. Set up the templates needed for the GPIO pins you are using.
5. Determine values and addresses you need to output a high or output a low on a GPIO pin.
6. Determine the RMW sequence of instructions and addresses required to program the GPIO pins for the LEDs as outputs.
7. Develop the **low level algorithm**, as shown in chapter 4, for your program, including the delay loop
8. Write and carefully check the assembly language program.
9. Build, Load, Run, and Debug the program. Note that to single step through the program easily you can initially use a very small delay constant, so you don't have to single step forever to get from one LED to the next.
10. When your program works, make an electronic copy that you will use for developing the next part of the project. Key point is that once you get something working, you save it and use a copy to develop the next stage. That way you always have a working program section to go back to.

Part 2 Creating an Interrupt Procedure for Servicing a Button Push

1. Carefully read through Hall Chapter 5 and work through the development of the button service program. Your log should parallel the development steps described in the text section for the example program. Develop an initialization list, such as those in the text, with the values needed for this project where the pushbutton is on GPIO1_30. (Show all thinking, labeled templates, etc.)
2. Use this detailed initialization list to modify the required steps of the assembly language program in Figure 5-14 as needed for this project. (Note that, as discussed in the text, you have to modify the startup file to intercept the system IRQ response, instead of using the usual method of hooking the interrupt vector, due to the way the BeagleBone Black is set up.)
3. Write an algorithm for the Button Service procedure that will start the LED rotating action the first time it is called and turn the LED rotating action off the next time it is called. In general, one way to do this is to set aside a memory location that you toggle back and forth to keep track of whether the LEDs are pulsing or not. You then use this to determine whether to start or stop the pulsing when a button press produces an IRQ interrupt request.
4. Write the assembly language program.
5. To test the program, set a breakpoint at the start of the INT_DIRECTOR procedure and run the program. If all is well, execution should go to the breakpoint when the button is pushed. If execution doesn't make it to the breakpoint, mentally work through your code very carefully again to make sure all the bits in the control words, etc. are initialized as they should be. Your log should be helpful in doing this.
6. When execution gets to the INT_DIRECTOR procedure correctly, set a breakpoint at the start of the BUTTON_SVC procedure and step execution to that point.
7. If execution gets to the start of BUTTON_SVC correctly, then you can step through the rest of the program and back to the wait loop to wait for the next button push.
8. When this is all working, save it and make a copy that you will use for developing the next addition to the program.
9. Take a break and give a couple of cheers!

Part 3 Using a Programmable Timer to control LED switching times

For this part of your program, you will use interrupts from Timer3 to determine when to switch from one LED to the next, instead of using a delay loop. The Timer 2 program discussed in text chapter 5 shows you how to set up a timer to produce interrupts at desired time intervals. In ECE 371 Design Project #2 part 1, you learned how to use a dedicated memory location to keep track of whether an LED is on or off. For this program you need to figure out some simple way to keep track of which of the 4 LEDs is on when a timer interrupt occurs and take appropriate action.

The timer section of Chapter 5 tells you almost everything you need to add the timer capability to your button program, if you study it VERY carefully. Of course, you are using Timer 3 instead of Timer 2, so you have to modify those parts as needed. You also have to integrate your rotating LED actions into the Program.

Signoff

1. Demonstrate the basic LED rotate with delay loop program and the final button-push/timer to TA or Instructor for signoffs on .s files. Be prepared to answer questions about the details of your program.
2. Congratulate yourself. You have successfully created a very real program that uses multiple interrupts and a timer as is done in all embedded systems.
3. Turn in a hard copy of your documentation for the project. You do not need to put it in a fancy binder. Just fasten all the pages together in some way so they do not become separated during

grading, etc. Projects will be available in first week of ECE 372 classes or they can be picked up from my office during winter quarter.