ECE 371

DENNIS SOROKIN

PORTLAND STATE UNIVERSITY

12/7/18

Task for part one:

For part one we needed to understand how the gpio pins worked and using them to turn on the four user LED'S on the beaglebone black board. Once you got a LED to light on you then have to light the rest up in a theater marquee pattern with a delay loop.

GPIO templates for setdataout and cleardataout

gpio	31	30	29	28	27	26	25	24	23	22	21	20
bit	0	0	0	0	0	0	0	1	1	1	1	0

Hex	0	1	E

gpio	19	18	17	16	15	14	13	12	11	10	9	8
bit	0	0	0	0	0	0	0	0	0	0	0	0

Hex	0	0	0

gpio	7	6	5	4	3	2	1	0
bit	0	0	0	0	0	0	0	0

Hex	0	0

Hex value for set and clear data out is 0x01E00000

GPIO 21-24 Initialization template

gpio	31	30	29	28	27	26	25	24	23	22	21	20
bit	1	1	1	1	1	1	1	0	0	0	0	1

Hex	F	E	1

gpio	19	18	17	16	15	14	13	12	11	10	9	8
bit	1	1	1	1	1	1	1	1	1	1	1	1

Hex	F	F	F

gpio	7	6	5	4	3	2	1	0
bit	1	1	1	1	1	1	1	1

Hex	F	F

Hex value to for gpio pins 21-24 set as outputs is 0xFE1FFFFF

High level algorithm
Setup the stack for procedures
Turn on the clock
Value to enable clock for a gpio module
Load it to the address of the clock module
Load value to turn off leds 21-24
Get base address for GPIO1
Add offset for cleardata out to base register
Store value to cleardataoutregister

Program GPIO pins 21-24 as outputs
Add output enable register to base address for GPIO1
Read,Modify,Write process

Lights procedure
Loop
Check bit position if in range then continue
If over then reset and go to Clear
Add to counter for checking if over or not
Shift over to next led
Go to a delay function that has a value we just subtract from
Until all lights have been turned on

Clear led procedure
Resets counter to see what led its on
Load base adress of gpio1
Write the value to clear all leds to cleardataout

Low level algorithm

Setup the stack
Move #02 for clock
Load base address for CLOCK1 (0x44E000AC)
Store #02 in base address of CLOCK1
Load value to turn off leds 21-24 (0x01E00000)
Load base address of GPIO1 (0x4804C000)
Add offset for cleardataout to base address (0x190)
store(0x01E00000)to cleardata out for gpio1(0x4804C190)offset added to base address
Add offset for gpio output enable (0x0134) to base gpio1 address
Move #0xFE1FFFFF word to enable as output
Modify (and the bits to clear them)
Store in output enable register
Compare value counter
If over then reset
Load gpio1 data clear address (0x4804C190)

Load clear values 21-24 (0x01E00000)
Write to cleardataout
Else increment counter
load shift by 0x00080000
Then shift LSL by that value

Task for part two:

For part two we need to develop a interrupt procedure that will service a interrupt request from a push button switch connected to GPIO2_1. First time the button is pushed the interrupt procedure will start the led pulsing the second time it is pressed it will stop the leds from pulsing.

High level algorithm:

Setup the stacks for procedures

Turn on the clock

Value to enable clock for a gpio module
Load it to the address of the clock module
Load value to turn off leds 21-24

Get base address for GPIO1
Add offset for cleardata out to base register
Store value to cleardataoutregister

Program GPIO pins 21-24 as outputs
Add output enable register to base address for GPIO1
Read,Modify,Write process

Detect falling edge on gpio2_1 and assert pointerpend1

Initialize INTC

Load led display

Processor IRQ enabled to CPSR

Set the toggle to check state of led
Initialize it to zero

Interrupt LOOP:

Compare flag
If button not pushed go to LIGHTS
If equal go to Interrupt LOOP again
Loop again

Once switch is pressed go to INT_DIRECTOR

INT_DIRECTOR
Push registers
Load adress of INTC_PENDING_IRQ1
Test first but in mir1
If its not from gpioint1a then go to wait loop
Check if bit 2=1
BNE BUTTON_SVC if bit 2=1 button pushed
BEQ PASS_ON if but 2=0 go to wait loop

PASS_ON goes back to mainline

BUTTON_SVC:
Turn off gpio2_1 interrupt
Write to IRQ status register
Load address of int_contrl register
Write value to clear bit
Store to int control register

Load flag
Compare value from flag with 1
If it is one change it to zero and store it
switch was pressed
Turn off the lights

Load address if gpiocleardataout
Write to it and clear the leds
else Change flag to one
Reset register that checks what led its on
Restore register go back to wait loop

Lights procedure
Loop
Check bit position if in range then continue
If over then reset go to Clear
Add to counter for checking if over or not
Shift over to next led
Go to a delay function that has a value we just subtract from
Until all lights have been turned on

Clear led procedure
Resets counter to see what led its on
Load base address of gpio1
Write the value to clear all leds to cleardataout

Low level algorithm

Load r13 with stack1
Point to top of stack add(#0x1000)
Switch to IRQ MODE (CPS #0x12)
Load r13 with stack2
Point to top of stack add(#0x1000)
Switch to SVC mode (CPS #0x13)

Move #02 for clock
Load base address for CLOCK1 (0x44E000AC)
Store #02 in base address of CLOCK1
Move #02 for clock
Load base address for CLOCK2 (0x44E000B0)
Store #02 in base address of CLOCK2

Load value to turn off leds 21-24 (0x01E00000)
Load base address of GPIO1 (0x4804C000)
Add offset for cleardataout to base address (0x190)
store(0x01E00000)to cleardata out for gpio1(0x4804C190)offset added to base address
Add offset for gpio output enable (0x0134) to base gpio1 address
Move #0xFE1FFFFF word to enable as output
Modify (and the bits to clear them)
Store in output enable register

To base address of gpio add (0x14C) falling detect register offset
Copy (0x00000002) for bit 2
Modify (set bit 2)
To base address of gpio2 add (#0x34) the IRQ_STATUS_SET0 offset
Store value for bit 2 to IRQ_STATUS_SET0

Initialize INTC
Load the base address for MIR_CLEAR1
Copy value (#0x01) to unmask INTC INT 1
Store that value to the MIR_CLEAR1 register

Enable Processor IRQ in CPSR
Copy CPSR
Clear bit 7 (#0x80)
Write back to CPSR

Set toggle
Initialize it to zero
Store the value to toggle

Interrupt loop
Load flag value
Compare with 0
BNE to lights to turn them on
Else keep looping

INT_DIRECTOR
 Save register 0-3 on stack and the return address
 Load address of INTC_PENDING_IRQ1 (0x482000B8)

Test first bit in MIR1 (#0x00000001)
If not from GPIOINT1A then go back to wait loop
Else Load (0x481AC02C) for the IRQ_STATUS0 register
Test if bit (#0x00000002) equals 1
BNE to button__SVC
Else go back to wait loop

button_SVC
Copy value (#0x00000002) to turn of interrupt request
Store it in GPIO2_IRQ_STATUS0 register
Load address of INTC_CONTROL register (0x48200048)
Copy value (#01) to clear bit 0
Write to INTC_CONTROL register

Check the flag
Compare with 0

If switch was pressed
Copy and store 0 to flag
Turn off leds, load base address of gpio1 (0x4804C190)and write (0x01E00000) to clear
return to loop to wait for next press

Else copy and store 1 to flag
Return back to loop for next press

Compare value counter
If over then reset
Load gpio1 data clear address (0x4804C190)
Load clear values 21-24 (0x01E00000)
Write to cleardataout
Else increment counter
load shift by 0x00080000
Then shift LSL by that value
Go to next led to light

Design Log

11/21/18

Started to work on the Led portion of the project, I used this image to help me find the correct pins for the four user led's.

Table 8 shows the signals used to control the four LEDs

= Table 8. User LED Control Signals/Pins =

LED	GPIO SIGNAL	PROC PIN
USR0	GPIO1_21	V15
USR1	GPIO1_22	U15
USR2	GPIO1_23	T15
USR3	GPIO1_24	V16

A logic level of "1" will cause the LEDs to turn on.

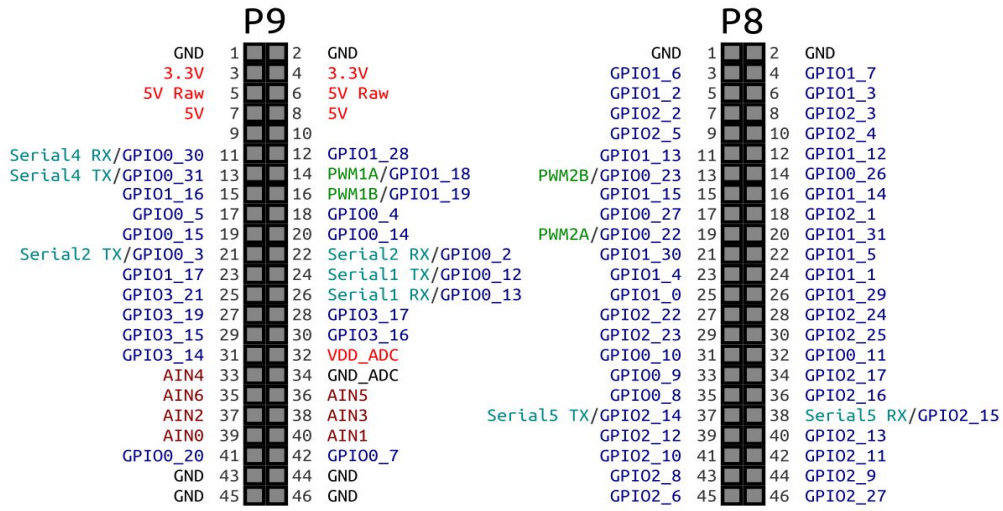
Found all the required offsets to add to the base gpio1 address in the tables at the end of the book.

11/23/18
I started to test my code for the project but no matter what could not get it to work after several hours of debugging it seemed like all my logic was right so I thought it might be issue with the software turns out I built the project and did configure but forgot to load the program and it was loading the Average project 1 file the whole time so now I always make sure to double check every step when I change the code and run it. Also had some trouble when I used a computer that ran ccs version 8 and my other code was on ccs version 6 so just try to find computers with version 6 and not deal with the extra hassle.

11/24/18
Got the leds to work overall following the example from the book the code was similar except added logic to loop thru four leds and not just light one. Tried out different values for the loop delay value to make the leds blink fast or slow.

11/26/18

Wrote my code for part two and asked Alex to see if I was on the right track turns out I was using a single stack and you need two one for the IRQ and one for the SVC. Also In the program from book there was a different MIR used than the one we needed because we needed to use gpio2_1 for the switch and not gpio1 like in the example from the book. Alex helped explain it clearly with the help of a diagram he drew. Used this diagram to find what pin gpio2_1 was connected to on the board.



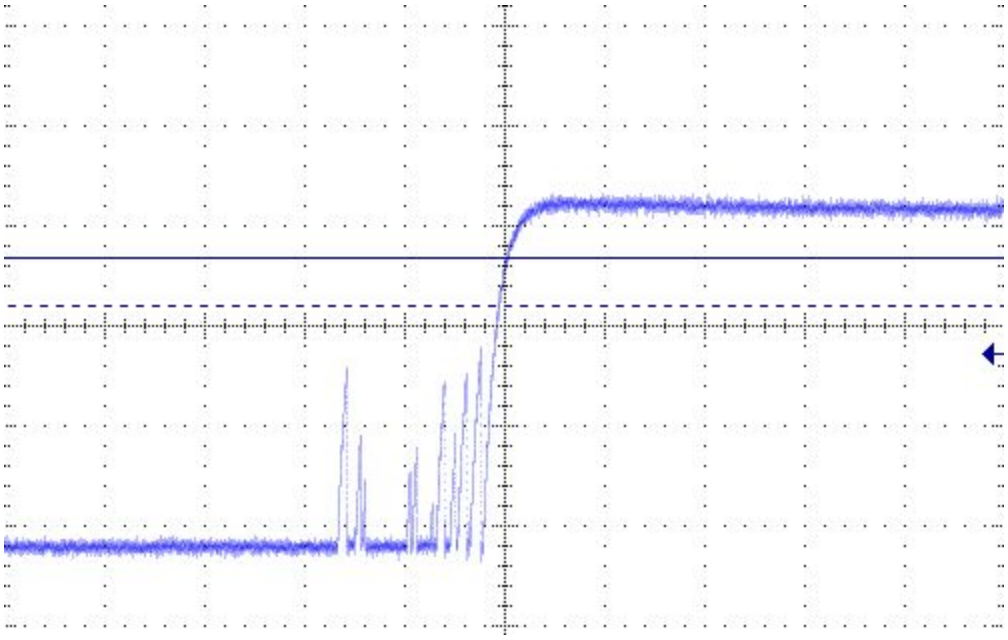
11/28/18

Had some difficulty with the logic in the code I needed a boolean or a flag variable to check when the switch was pressed or not and Alex said to not have a lot of code in the button service procedure because you want the interrupt to be quick in and out and not spending time executing a lot of code in there. Also my led's would not turn off so I modify my code and I added a seperate function in to turn them off if the button was pressed.

11/30/18

Completed the code and went in to test it for the final time and to demo it. When testing it my code seems sluggish and slow when the button was pressed turns out the wire was not connected very well and the station was not one that had the debounced switch

connected to it so it had lag to it when I switched to a station with the debounced switch everything worked perfectly so I did a bit of research into this and found this interesting graph that shows that the when a switch is not debounced.



12/6/18

I started project part three wrote the algorithms for it and started to implement it in ccs. I asked Alex for help on the offsets it was in the CM_DPLL register section of the book.

12/7/18

Spent all day working on the code checking to make sure all my offsets were correct and that my logic worked for the leds.

Part 3

Task

For part three we need to develop a interrupt procedure that will service a interrupt request from a push button switch connected to GPIO2_1. First time the button is

pushed the interrupt procedure will start the led pulsing the second time it is pressed it will stop the leds from pulsing. This time it will be using a timer instead of a delay loop like in part two of the project.

High level algorithm

Setup the stacks for procedures

Turn on the clock

Value to enable clock for a gpio module
Load it to the address of the clock module
Load value to turn off leds 21-24

Get base address for GPIO1
Get base address for GPIO2

Add offset for cleardata out to base register
Store value to cleardataoutregister

Program GPIO pins 21-24 as outputs
Add output enable register to base address for GPIO1
Read,Modify,Write process

Detect falling edge on gpio2_1 and assert pointerpend1

Initialize INTC

Reset and enable timer 4 interrupt

Turn on timer 4 clock

Load led display

Processor IRQ enabled to CPSR

Interrupt LOOP: waits for signal loop forever

Int director procedure

Check if the interrupt came from the timer or the button

Else go to pass on procedure

Button service procedure

Turn off IRQ request on GPIO2_1

Generate new IRQ

Turn off LEDS

Check flag if zero set it equal to 1 then set timer flag to zero go to pass on

Else set flag to zero and go to pass on

Timer interrupt procedure check for timer overflow and go to pass on

Else go to IRQ timer procedure

IRQ timer procedure clear interrupts and generate new irq

Check flag if equal turn off leds

Else keep them rotating

Pass on procedure restores register goes back to mainline
Low level algorithm

Initialize stack1 and stack2

Load r13 with stack1
Point to top of stack add(#0x1000)
Switch to IRQ MODE (CPS #0x12)
Load r13 with stack2
Point to top of stack add(#0x1000)
Switch to SVC mode (CPS #0x13)

Move #02 for clock
Load base address for CLOCK1 (0x44E000AC)
Store #02 in base address of CLOCK1
Move #02 for clock
Load base address for CLOCK2 (0x44E000B0)
Store #02 in base address of CLOCK2

Load value to turn off leds 21-24 (0x01E00000)
Load base address of GPIO1 (0x4804C000)
Add offset for cleardataout to base address (0x190)
store(0x01E00000)to cleardata out for gpio1(0x4804C190)offset added to base address
Add offset for gpio output enable (0x0134) to base gpio1 address
Move #0xFE1FFFFF word to enable as output
Modify (and the bits to clear them)
Store in output enable register

To base address of gpio add (0x14C) falling detect register offset
Copy (0x00000002) for bit 2
Modify (set bit 2)
To base address of gpio2 add (#0x34) the IRQ_STATUS_SET0 offset
Store value for bit 2 to IRQ_STATUS_SET0

Initialize INTC
Load the base address for MIR_CLEAR1
Copy value (#0x01) to unmask INTC INT 1
Store that value to the MIR_CLEAR1 register
Enable gpio interrupt at(0x482000A8)
write(0x01 to unmask)
Enable Processor IRQ in CPSR
Copy CPSR
Clear bit 7 (#0x80)
Write back to CPSR

Enable timer 4
load(0x44E00088) address for cm_per_timer4_clkctrl
Write (0x02)

Initialize timer 4

load(0x48044000) for base address
Write (0x01) to reset
Load timer with (0xFFFF8300)

Interrupt loop

INT_DIRECTOR
Int director procedure

Check for timer test with (0x00000001)
Else check for button(0x481AC02C) and test with 0x00000002

Timer interrupt procedure

Check if its timer interrupt
load(0X482000D8)
Test with (0x1)

Check if its overflow interrupt
load(0x48040028)
Compare with (0x2)

Pass on
Restore registers
Go to mainline

IRQ timer procedure
Load IRQ status(0x480444028)
Write (0x02) to clear interrupt
Load control register(0x48200048)
Enable new IRQ (0x01)

Load flag
Ifequal clear leds
Load value to turn off leds 21-24 (0x01E00000)
Load base address of GPIO1 (0x4804C000)
Add offset for cleardataout to base address (0x190)

If not equal rotate leds

Button service procedure
Load irq status register(0x481AC02C)
Turn of interrrupt (0x00000002)
Int control register(0x48200048)
write(0x01) to clear
Load value to turn off leds 21-24 (0x01E00000)
Load base address of GPIO1 (0x4804C000)
Add offset for cleardataout to base address (0x190)
If flag equals zero set to one