Design Log

1/15/19

Started to look through the manuals and the beaglebone book to find the right offsets for the uart5 and to understand how it works. The rc8660 had a lot of different features built into it.

1/16/19

Working on the high level algorithm for the uart5 to get it to say a word mainly just focusing on the initialization steps for it by following chapter 5 in the book and looking at the manuals to help set everything up correctly.

1/17/19

Finished up my low level algorithm and started writing code in assembly had some issues with opening a new file from ccs.

1/21/19

Was at home converting all my code from low level to assembly using notepad and halfway thru i went to take a break when i came back my computer went to sleep im thinking that shouldn't be a problem when i open it the majority of my work was unsaved even though i thought i hit control-s lesson learned never leave for a break without saving everything to 100 emails,drive,docs anything and everything, lost a lot of valuable time going back and retyping everything i lost due to my dumb mistakes.

1/23/19

There was some issue with the hardware at first i thought it was my code but Alex helped us by telling us that the “file load error” was not our code but the hardware and the uart so we had to unplug everything turn it off and then load the program then redo the cycle and plug the uart back into the connections and that did the trick although sometimes my system would just freeze up and not work at all in that case i switched computers to a different one and then everything worked again.

1/24/19

I got the talker to talk and and to say a message when you press the button on gpio2_1 Messed around and found ways to change the voice,volume and speed with the rc8660. Tried to get it to play a song using the example in the manual for the rc8660 but could not get it to make any noise when you use the chords not sure why but decided to first finish the project before spending time on the extra credit

2/7/19

Had a lot of trouble implementing the leds at first I thought maybe I was using mixing the cleartosend and cleardataout signals up. I also used the led code from last term because I thought maybe i forgot how to use leds correctly but the leds still would not light up.

2/8/19

Spent several hours trying different ways of making leds light. I got the leds to turn on right away as soon as you push the button but then they would not turn off at all. My leds were supposed to turn off in the button service which is when you press the button to start the countdown the leds would turnoff and then supposedly light up when it hit “ 0 blastoff” on the uart.

2/14/19

Alex took a look at my demo and saw that the timer was not working at all basically the light for the TXD signal is supposed to flash every time for every number when the timer works but mine only did it once that meant the timer didn't work like i thought it did. That's why i spent so much time trying to get my leds to work with the timer when it wasn't working at all. Starting to go thru my timer code sections to see if i'm missing something.

2/15/19

Went back to my old timer code from 371 and double checked all my offsets and sure enough there were one or two values that were off and needed to be fixed. Got the timer to work because the led started to blink on the TX signal light on the uart everytime a word was sent and so right after that the leds turned on and worked as supposed to because it was my timer causing me problems all along not the leds.

Task for part 1 :

Develop a program that sends a basic message to the rc8860 evaluation board on a interrupt basis when a button on gpio2_1 is pressed.

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 clocks

Initialize the gpio2_1 to detect falling edge

Setup the gpio2_1 for the IRQ enable

Setup the INTC controller for gpio2_1

Setup INTC for uart5

Setup INTC for sysconfig

Enable IRQ in CPSR

Map the uart5 pins to beaglebone the txd,rxd,cts,rts

Turn on uart5 clock module

Initialize uart5 by setting up the baud rate enable DLL and DLH access

Initialize the uart5 tx and rx registers clear bit and enable THR

Turn off TX_FIFO and disable FIFO control register

Enable IRQ interrupt

Endless wait loop to wait for interrupt

Int director procedure

Save registers

Check if interrupt came from from uart5

If not from uart5 go to button check

Else it was from uart5 go to TALKER_SVC section

Button check read INTC_PENDING_IRQ3 register

Check if it was gpio2_1 if not restore registers and return to wait loop

Else it was gpio2_1 go to BUTTON_SVC procedure

BUTTON_SVC procedure turn of IRQ request from gpio2_1

Generate new IRQ and enable THR and the MSR interrupt restore register and return

TALKER_SVC procedure
Save registers

Check the CTS signal and check to see if THR is empty

Enable THR and MSR interrupts

Get a CHAR from the message and increment pointer by one

Variable named CHAR_PTR to keep track everytime char sent over decrement by one

Check the value to see if its the last char after the decrement

If the value less than zero counter disable THR interrupt return to mainline

Else value greater than zero go send next character by restoring register and returning to mainline

Low level Algorithm

Setup stacks

Turn on clocks

Load base address for gpio1 for the led to light so (0x44E000AC)
Load base address for gpio2 for button (0x44E000B0) write 0x02
Move #02 for clock
Base address for uart5(0x44E00000) plus offset(0x38) so (0x44E00038) write 0x02

Falling detect gpio2_1
Write 0x00000002 to falling detect register +base address at (0x481AC14C)
Store 0x00000002 to IRQ status register at (0x481AC034)

INTC initialization
Write #2 to INTC_SYSCONFIG (0x48200010)

Enable gpio interrupt for gpio2_1 and uart5 interrupt
Address of INTC_MIR_CLEAR1 (0x482000A8) write 0x01 to unmask
Address of INTC_MIR_CLEAR1(0x482000A8) write 0x4000 for uart5 interrupt

MAP uart5 txd,rxd,cts and rts to pins on beaglebone black p8 connector
Change pin 31 on p8 to mode 6 by writing
#6 to control module base(0x44E10000)+0x8D8 for offset pin31
Change pin 32 on p8 to mode 6 by writing
#6 to control module base(0x44E10000) +0x8DC for offset pin32
Change pin 37 on p8 to mode 4 by writing
#4 to control module base(0x44E10000) +0x8C0 for offset pin37
Change pin 38 on p8 to mode 4 by writing
#4 to control module base(0x44E10000) +0x8C4 for offset pin38

Initialize uart5 baud rate
DLL and DLH access writing 0x83 to (0x481AA00C)
Since 0x481AA000 is base and offset for LCR is 0x0C

Division by 16 mode write 0x00 to 0x481AA020 uart base(0x481AA000) + MDR1 Register offset (0x20)

Toggle the LCR register bit 7 base address write 0x03 to the uart +LCR offset (0x0481AA00C)

Enable THR and Modem bit by writing to IER_UART interrupt enable register at (0x481AA004) write 0x0A to it

Turn off fifo by writing 0x00 to uartbase(0x481AA000) plus offset for fifo control register which is (0x08) so (0x481AA008)

Enable processor IRQ write 0x80 to clear bit 7

Loop wait for interrupt

INT_DIRECTOR

Check if interrupt came from uart5 by testing INT46
Test with #0x4000 at base address plus int pending irq1 (0x482000B8)

Else go to button_check

Load address for IIR_uart register (0x481AA008)
Test bit (0x1)
BEQ to TALKER_SVC section to send char
Else go back to wait loop by restoring registers

BUTTON_CHECK
Check if its from button by testing bit 2 (0x00000002)
At the base address of gpio2 + irq status register (0x481AC02C)
IF button pressed go to BUTTON_SVC
Else load control register (0x48200048) and clear bit 0 (0x01)
And return to wait loop by restoring registers

BUTTON_SVC routine
Load gpio2_1 irq register (0x481AC02C)
Move value (0x00000002) to turn off interrupt request
Store it at (0x481AC02C)
Load INT_CONTROL register(0x48200048)
Clear bit 0 (0x01)
Load uart5 base + IER_UART register (0x481AA004)
Enable THR MODEM interrupts by writing (0xA) to that address
Restore register return to wait loop

TLKR_SVC routine
Load base adress of uart + MSR register (0x481AA018)
Test it with (0x10) to check for CTS signal
Not equal go to THR_CHECK
Else load base for uart +LSR register (0x481AA014)
Check with (0x20) if not equal to go to wait loop
Else go to CHAR_SEND

THR_CHECK routine
Load adress of uart5 + LSR register (0x481AA014)
Test bit (0x20) IF not equal go to wait loop
Else load uart5 base + IER_UART register (0x481AA004)
Store (0x2) to mask THR
Branch to wait loop

CHAR_SEND routine
Load uart5 base address (0x481AA000)
Load pointer for char (ptr_for_char)
Get the value and increment pointer to next
Check if its the last one if it is go to MESSAGE_OVER
Compare with 0x0D the padding between ascii strings
If not equal go to interrupt clear
Else go to wait loop

MESSAGE_OVER routine
Load base of uart5+ IER_UART register (0x481AA004)
Clear it with value (0x0) to disable interrupt
Load pointer for char
Return to wait loop

INTERRUPT_CLEAR routine
Load base of uart5+ IER_UART register (0x481AA004)
Clear it with value (0x0) to disable interrupt
Return to wait loop

.data make the stacks and pointer for message

Part two of the project

Once we got part one working our objective for this part was to add in a timer that would countdown from 10-0 and to light up the leds on the beaglebone to simulate a rocket launch.

High Level Algorithm:

Setup the stack for procedures

Turn on the clocks

Initialize the gpio2_1 to detect falling edge

Setup the gpio2_1 for the IRQ enable

Initialize the leds on gpio1

Setup the INTC controller for gpio2_1

Setup INTC for uart5

Setup INTC for sysconfig

Setup INTC for timer4

Enable IRQ in CPSR

Map the uart5 pins to beaglebone the txd,rxd,cts,rts

Turn on uart5 clock module

Initialize uart5 by setting up the baud rate enable DLL and DLH access

Initialize the uart5 tx and rx registers clear bit and enable THR

Turn off TX_FIFO and disable FIFO control register

Enable IRQ interrupt

Endless wait loop to wait for interrupt

Int director procedure

Save registers

Check if interrupt came from from uart5

If not from uart5 go to button check

Else it was from uart5 go to TALKER_SVC section

BUTTON_CHECK read INTC_PENDING_IRQ3 register

Check if it was from timer if it was go to TCHK

Check if it was gpio2_1 if not restore registers and return to wait loop

Else it was gpio2_1 go to BUTTON_SVC procedure

Else restore register and return

BUTTON_SVC procedure turn of IRQ request from gpio2_1

Generate new IRQ and enable THR and the MSR interrupt

Write to IER register

Enable timer IRQ

Load gpio1 base and cleardatout

Turn off led’s

TCHK procedure

Check if its timer irq if not go to INTERRUPT to enable new interrupt

Else Check if its the timer if it is go to TIMER_SVC

Else restore registers and return to wait loop

TIMER_SVC procedure

Clear interrupts enable THR and MODEM interrupts

Write to IER register

Restore registers and return

TALKER_SVC procedure

Check the CTS signal and check to see if THR is empty

Go to THR_CHECK

Else go to CHAR_SEND when ready to send

Restore registers and return

THR_CHECK routine

Load LSR register check bit five

Mask interrupt in IER return to wait loop

CHAR_SEND routine

Load pointer and load char and increment pointer
Compare with either null or if the message is over

It null go to next piece to send PIECE_DONE routine

Else go to MESSAGE_OVER routine

MESSAGE_OVER routine

Disable interrupts

If message over light leds

Reset pointers and return to wait loop

PIECE_DONE routine

Disable interrupts load the timer with value

Load the TLDR and TCRR registers and turn on timer

Restore registers and return

Low Level Algorithm

Setup stacks

Turn on clocks

Load base address for gpio1 for the led to light so (0x44E000AC)
Load base address for gpio2 for button (0x44E000B0) write 0x02
Move #02 for clock
Base address for uart5(0x44E00000) plus offset(0x38) so (0x44E00038) write 0x02

Falling detect gpio2_1
Write 0x00000002 to falling detect register +base address at (0x481AC14C)
Store 0x00000002 to IRQ status register at (0x481AC034)

LED initialization
Load base gpio address + cleardataoutregister (0x4804C190)
Mov value to turn on leds (0x01E00000)
Load gpio1 base + outputenable register (0x4804C134)
Store (0xFE1FFFFF) to enable as output

INTC initialization
Write #2 to INTC_SYSCONFIG (0x48200010)

Enable gpio interrupt for gpio2_1 and uart5 interrupt
Address of INTC_MIR_CLEAR1 (0x482000A8) write 0x01 to unmask
Address of INTC_MIR_CLEAR1(0x482000A8) write 0x4000 for uart5 interrupt

MAP uart5 txd,rxd,cts and rts to pins on beaglebone black p8 connector
Change pin 31 on p8 to mode 6 by writing
#6 to control module base(0x44E10000)+0x8D8 for offset pin31
Change pin 32 on p8 to mode 6 by writing
#6 to control module base(0x44E10000) +0x8DC for offset pin32
Change pin 37 on p8 to mode 4 by writing
#4 to control module base(0x44E10000) +0x8C0 for offset pin37
Change pin 38 on p8 to mode 4 by writing
#4 to control module base(0x44E10000) +0x8C4 for offset pin38

Initialize uart5 baud rate
DLL and DLH access writing 0x83 to (0x481AA00C)
Since 0x481AA000 is base and offset for LCR is 0x0C

Division by 16 mode write 0x00 to 0x481AA020 uart base(0x481AA000) + MDR1 Register offset (0x20)

Toggle the LCR register bit 7 base address write 0x03 to the uart +LCR offset (0x0481AA00C)

Enable THR and Modem bit by writing to IER_UART interrupt enable register at (0x481AA004) write 0x0A to it

Turn off fifo by writing 0x00 to uartbase(0x481AA000) plus offset for fifo control register which is (0x08) so (0x481AA008)

Enable processor IRQ write 0x80 to clear bit 7

Loop wait for interrupt

INT_DIRECTOR

Check if interrupt came from uart5 by testing INT46

Test with #0x4000 at base address plus int pending irq1 (0x482000B8)

Else go to button_check
Load address for IIR_uart register (0x481AA008)
Test bit (0x1)
If not button go to TCHK
Else test gpio2_1 irq (0x481AC02C) test with (0x00000002)
If it is button go to BUTTON_SVC
BEQ to TALKER_SVC section to send char
Else go back to wait loop by restoring registers

BUTTON_CHECK
Check if its from button by testing bit 2 (0x00000002)
At the base address of gpio2 + irq status register (0x481AC02C)
If its not button go to TCHK
IF button pressed go to BUTTON_SVC
Else load control register (0x48200048) and clear bit 0 (0x01)
And return to wait loop by restoring registers

BUTTON_SVC routine
Load gpio2_1 irq register (0x481AC02C)
Move value (0x00000002) to turn off interrupt request
Store it at (0x481AC02C)
Load INT_CONTROL register(0x48200048)
Clear bit 0 (0x01)
Load uart5 base + IER_UART register (0x481AA004)
Enable THR MODEM interrupts by writing (0xA) to that address
Turn off leds by loading address (0x4804C190) gpio1 + cleardataout
And word to turn off (0x01E00000)
Store word to cleadataout register
Restore register return to wait loop

TCHK routine
Test timer interrupt (0x482000D8) with (0x10000000)
Else go to CONTINUE to clear interrupt (0x48044048)
And compare with (0x1) then return to wait loop

TIMER_SVC routine
Check for overflow at (0x48044028)

Compare with (0x2) if not equal go to TIMER_SVC
Load uart5 base+ ier register (0x481AA004)
Enable thr,modem interrupts by writing (0xA) to it
Restore registers and return to wait loop

TLKR_SVC routine
Load base adress of uart5 + MSR register (0x481AA018)
Test it with (0x10) to check for CTS signal
Not equal go to THR_CHECK
Else load base for uart +LSR register (0x481AA014)
Check with (0x20) if not equal to go to wait loop
Else go to CHAR_SEND
If message over go to TURNON to light leds

THR_CHECK routine
Load adress of uart5 + LSR register (0x481AA014)
Test bit (0x20) IF not equal go to wait loop
Else load uart5 base + IER_UART register (0x481AA004)
Store (0x2) to mask THR
Branch to wait loop

CHAR_SEND routine
Load uart5 base address (0x481AA000)
Load pointer for char (ptr_for_char)
Get the value and increment pointer to next
Check if its the last one if it is go to MESSAGE_OVER
Compare with 0x0D the padding between ascii strings
If not equal go to interrupt clear
Else go to wait loop

PIECE_DONE routine
Load uart5 base address (0x481AA000)
Disable uart with 0x0 at IER register (0x4) offset
Load timer4 base address (0x48044000)
Load count value (0xFFFF8000)
Write that to TLDR register at (0x40 offset)
And to TCRR register offset (3C)
Load value to turn on timer 0x01
Store it at offset 0x38

MESSAGE_OVER routine
Load base of uart5+ IER_UART register (0x481AA004)
Clear it with value (0x0) to disable interrupt
Load pointer for char
Return to wait loop

INTERRUPT routine
Load base of uart5+ IER_UART register (0x481AA004)
Clear it with value (0x0) to disable interrupt
Load delay value for timer (0xFFFF8000)
Store it into TCRR (0x4804403C) and TLDR (0x48044040)
Load TCLR register (0x48044038) write (0x01) to it
Go back to wait loop

.data make the stacks and pointer for message