## PART 1 –UART, BUTTON, SPEECH

Task List

- Remap MUX pins
- Setup UART for proper operations
- Setup button for GPIO
- Enable UART and button interrupts
- Figure out character sending process in interrupt

## HIGH LEVEL

1. Setup stack for supervisor and IRQ
2. Setup GPIO2 clock and UART2 clock
3. Remap mux for P8 by changing register values
4. Setup falling edge detect for GPIO2_1
5. Unmask bit 1 of MIR_clear1 to allow GPIOINT2A
6. Unmask bit 14 of MIR_clear1 clear to allow UART interrupts
7. Change UART mode to A, and setup for 8 bit, one stop, no parity
8. Set DLL and DLH values for desired baud rate
9. Set baud rate mode to 16 in MDR1
10. Switch back from mode A to operational mode
11. Enable THR interrupt and modem interrupt in IER
12. Disable and clear FIFO
13. Enable IRQ input by clearing bit 7

INT_DIRECTOR

1. Save register on stack
2. Test bit 14 of IRQ for UART
    a. If no, the test bit 0 at IRQ1
        i. If yes, then test bit 1 of GPIO2_IRQStatus_0
            1. If yes, then BUTTON_SVC
    b. If yes, then test IIT of IIR_UART
        i. If yes, then TALKER_SVC

BUTTON_SVC

1. Turn off GPIO2_1 interrupt
2. Clear INTC_Control for NEWIRQ
3. Set bit 1 of THR and bit 3 for Modem, change interrupt in IER_UART
4. PASS_ON

TALKER_SVC

1. Read bit 4 of Modem Status Register
    a. If 1 then, read bit 5 of LSR_UART register
        i. If 1 then, SEND_CHAR
    b. If 0 then, branch to NOMODEM

SEND_CHAR

1. Load pointer to character in memory
2. Load pointer to character counter
3. Load character to be sent, increment pointer
4. Load character count
5. Setup pointer to THR register
6. Store character to be sent in THR register
7. Subtract counter value
8. Store counter back in data
    a. If this isn't the last character, PASS_ON
    b. ELSE
        i. Reload character pointer
        ii. Reload character counter
        iii. PASS_ON

NOMODEM

    a. read bit 5 of LSR_UART for TXFIFO
        i. If yes, then MASK INT
            1. PASS_ON
        ii. ELSE
            1. PASS_ON

PASS_ON

1. Restore register
2. SUBS #4 outta there

## LOW LEVEL

1. Setup stack for supervisor and IRQ

3. Setup GPIO2 clock and UART2 clock
    a. **Store 0x02 in CM_PER_GPIO2_CLKCTRL at 0x44E000B0 to enable GPIO2 clock**
    b. **Store 0x02 in CM_PER_UART5_CLKCTRL at 0x44E00038 to enable UART5 clock**
4. Setup falling edge detect for GPIO2_1 and interrupt generation
    a. **Store 0x02 in GPIO2_FALLINGDETECT at 0x4814C000+0x14C using READ, MODIFY, WRITE**

   **b. Store 0x02 in GPIO2_IRQSTATUS_SET_0 at 0x4814C000+0x034**

5. Unmask bit 1 on MIR_CLEAR1 to allow GPIOINT2A, and bit 14 to allow UART5INT
   **a. Store 0x4001 in INTC_MIR_CLEAR1 at 0x482000A8**
6. Remap lcd_data14 as input for UART5_CTSN (Mode 6)
   a. **Read, ORR, write 0x26 in conf_lcd_data14 at 0x44E10000+0x8D8**
7. Remap lcd_data15 as output for UART5_RTSN (Mode 6)
   a. **Read, ORR, write 0x06 in conf_lcd_data13 at 0x44E10000+0x8DC**
8. Remap lcd_data9 as input for UART5_RXD (Mode 4)
   a. **Read, ORR, write 0x24 in conf_lcd_data9 at 0x44E10000+0x8C4**
9. Remap lcd_data8 as output for UART5_TXD (Mode 4)
   a. **Read, ORR, write 0x04 in conf_lcd_data8 at 0x44E10000+0x8C0**
10. Change UART mode to A, and setup for 8 bit, one stop, no parity
    a. **Write 0x83 in UART_LCR at 0x481AA000+0x0C**
11. Set DLL and DLH values for desired baud rate
    a. **Write 0x4E in UART_DLL at 0x481AA000+0x00**
    b. **Write 0x00 in UART_DLH at 0x481AA000+0x04**
12. Set baud rate mode to 16 in MDR1
    a. **Write 0x00 in UART_MDR1 at 0x481AA000+0x20**
13. Switch back from mode A to operational mode
    a. **Write 0x03 in UART_LCR at 0x481AA000+0x0C**
14. Disable and clear FIFO
    a. **Write 0x6 to FCR_UART at 0x481AA000+0x08**
15. Enable IRQ input by clearing bit 7
    **a. BIC #0x80**

INT_DIRECTOR

3. Save register on stack
4. Test bit 14 of IRQ for UART
   a. **Test (0x4000) in INTC_PENDING_IRQ1 at 0x482000B8**
   b. If no, the test bit 0 at IRQ1
      i. **Test (0x01) in INTC_PENDING_IRQ1 at 0x482000B8**
      ii. If yes, then test bit 1 of GPIO2_IRQStatus_0
         1. **Test (0x02) in GPIO_IRQSTATUS_0 at 0x481AC02C**
         2. If yes, then BUTTON_SVC
         3. If NO, PASS_ON
   c. If yes, then test IIT of IIR_UART
      i. **Test (0x01) in IIR_UART at 0x481AA000+0x08**
      ii. If (interrupt present) 0, then TALKER_SVC
5. Else PASS_ON

BUTTON_SVC

```
9.  Turn off GPIO2_1 interrupt
        a.  Store 0x02 in GPIO2_IRQSTATUS at 0x481AC02C
10. Set bit 1 of THR and bit 3 for Modem, change interrupt in IER_UART
        a.  Write 0xA to IER_UART at 0x481AA000+0x04
11. PASS_ON
```

TALKER_SVC

```
2.  Read bit 4 of Modem Status Register
        a.  Test (0x10) in Modem Status Register at 0x481AA000+0x18
        b.  If 1 then, read bit 5 of LSR
                i.   Test (0x20) in LSR_UART at 0x481AA000+0x60
                ii.  If 1 then, send a character
                iii. Test to see if this is the final character
                        1.  If  yes, then Point to first character for next send
                                a.  Disable UART_INT
                                        i.  Reset THR enable bit in the IER_UART
                                            Store 0x0000 in IER_UART at 0x481AA000+0x04
                        2.  PASS_ON
```

SEND_CHAR

```
1.  Load pointer to character in memory
2.  Load pointer to character counter
3.  Load character to be sent, increment pointer
4.  Load character count
5.  Setup pointer to THR register
6.  Store character to be sent in THR register
        a.  STORE register value in THR_UART at 0x481AA000
7.  Subtract counter value
8.  Store counter back in data
        a.  If this isn't the last character, PASS_ON
        b.  ELSE
                i.   Reload character pointer
                ii.  Reload character counter
                iii. PASS_ON
```

NOMODEM

```
    1.  Test Bit 5 of LSR_UART
                iv.  Test (0x20) in LSR_UART at 0x418AA000+0x60
                v.   If yes, then MASK INT
                        1.  Store 0b00 in IER_UART at 0x481AA000+0x04
                        2.  PASS_ON
```

vi. PASS_ON

PASS_ON

16. Restore register
17. Clear INTC_Control for NEWIRQ****
    a. **Store 0x01 at INTC_Control at 0x48200048**
18. SUBS #4 outta there

## PART 2 – TIMER, UART, BUTTON, LED, COUNTDOWN

Task List

- Modify UART program to allow for timer interrupts
- Enable LED after count down
- Have timer trigger the UART speech
- Setup logic to allow to trigger for LED after countdown

# HIGH LEVEL

MAINLINE

1. Enable UART5, TIMER4, GPIO2, and GPIO1 clocks
2. Initialize LEDs on GPIO1 for output
3. Initialize falling_edge_detect and Interrupt on GPIO2_1 for button
4. Initialize INTC for interrupts
5. Initialize TIMER4 register or 1 second interrupts
6. Remap lcd_data14 as input for UART5_CTSN (Mode 6)
7. Remap lcd_data15 as output for UART5_RTSN (Mode 6)
8. Remap lcd_data9 as input for UART5_RXD (Mode 4)
9. Remap lcd_data8 as output for UART5_TXD (Mode 4)
10. Initialize UART5 for interrupts and communication with board
11. Change UART mode to A, and setup for 8 bit, one stop, no parity
12. Set DLL and DLH values for desired baud rate
13. Set baud rate mode to 16 in MDR1
14. Switch back from mode A to operational mode
15. Disable and clear FIFO
16. Enable IRQ input by clearing bit 7

INT_DIRECTOR

1. Test to see if the interrupt came from the UART
    a. If NO
        i. Branch to TCHK
2. Test to see if the interrupt came from ITT of IIR_UART
    a. If yes
        i. Branch to TALKER_SVC

TCHK

1. Test to see if the interrupt came from the clock
    a. If NO,
        i. Branch to BCHK

```
2. Test to see if the overflow for TIMER4 was triggered
    a. If NO
        i. Branch to PASS_ON
3. Reset overflow for timer
4. Turn on UART5 interrupt by enabling THRIT and MODEMTIST
5. Force RTS to active low in UART_MCR Register
6. PASS_ON
```

BCHK

```
1. Test INTC_Pending to see if the interrupt came from the button
    a. If NO
        i. Branch to PASS_ON
2. Check GPIO2_IRQ for interrupt status
    a. If YES
        i. Branch to BUTTON_SVC
    b. If NO
        i. Branch to PASS_ON
```

BUTTON_SVC

```
1. Reset GPIO2 Interrupt request in IRQSTATUS register
2. Start TIMER4
3. PASS_ON
```

TALKER_SVC

```
3. Read bit 4 of Modem Status Register
    a. If 1 then, read bit 5 of LSR_UART register
        i. If 1 then, SEND_CHAR
    b. If 0 then, branch to NOMODEM
```

SEND_CHAR

```
1. Load pointer to character in memory
2. Load pointer to character counter
3. Load character to be sent, increment pointer
4. Load character count
5. Setup pointer to THR register
6. Store character to be sent in THR register
7. Subtract counter value
```

8. Store counter back in data
9. Disable UART Interrupts
10. Reload Timer with new value for 1 second
11. If last character is sent, then turn off timer 4
    a. ELSE
        i. Turn on TIMER4
12. If last character has not been sent then
    a. Branch to PASS_ON
13. Reload character pointer with starting character
14. Reload character count with original value
15. Turn on USER LED0

NOMODEM

    b. read bit 5 of LSR_UART for TXFIFO
        i. If yes, then MASK INT
            1. PASS_ON
        ii. ELSE
            1. PASS_ON

## LOW LEVEL

MAINLINE

1. Enable UART5, TIMER4, GPIO2, and GPIO1 clocks
    a. **Store 0x02 in CM_PER_GPIO2_CLKCTRL at 0x44E000B0 to enable GPIO2 clock**
    b. **Store 0x02 in CM_PER_UART5_CLKCTRL at 0x44E00038 to enable UART5 clock**
    c. **Store 0x02 in CM_PER_GPIO1_CLKCTRL at 0x44E000AC to enable GPIO1 clock**
    d. **Store 0x02 in CM_PER_TIMER4_CLKCTRL at 0x44E00088 to enable TIMER4 clock**
2. Initialize LEDs on GPIO1 for output
    a. **Store 0x01E00000 in GPIO1_CLEARDATAOUT at 0x4804C000+0x190**
    b. **Store 0xFE1FFFFF in GPIO1_OE at 0x4804C000+0x134 using READ, MODIFY, WRITE**
3. Initialize falling_edge_detect and Interrupt on GPIO2_1 for button
    a. **Store 0x02 in GPIO2_FALLINGDETECT at 0x4814C000+0x14C using READ, MODIFY, WRITE**
    b. **Store 0x02 in GPIO2_IRQSTATUS_SET_0 at 0x4814C000+0x034**
4. Initialize INTC for interrupts
    a. **Store 0x2 in INTC_Sysconfig at 0x48200010**
    b. **Store 0x4001 in INTC_MIR_CLEAR1 at 0x482000A8**
    c. **Store 0x10000000 in INTC_MIR_CLEAR2 at 0x482000C8**
5. Initialize TIMER4 register or 1 second interrupts
    a. **Store 0x02 in PRCMCLKSEL_TIMER4 at 0x44E00000 +0x510**
    b. **Store 0x01 to Timer 4 CFG at 0x48044000+0x010**

**c. Store 0x02 to Timer 4 IRQ Enable at 0x48044000 + 0x02C**

**d. Store 0xFFFF8000 in Timer 4 TLDR at 0x48044000 + 0x040**

**e. Store 0xFFFF8000 in Timer 4 TCRR at 0x48044000 + 0x03C**

6. Remap lcd_data14 as input for UART5_CTSN (Mode 6)
   a. **Store 0x2E in conf_lcd_data14 at 0x44E10000+0x8D8**
7. Remap lcd_data15 as output for UART5_RTSN (Mode 6)
   a. **Store 0x0E in conf_lcd_data13 at 0x44E10000+0x8DC**
8. Remap lcd_data9 as input for UART5_RXD (Mode 4)
   a. **Store 0x2C in conf_lcd_data9 at 0x44E10000+0x8C4**
9. Remap lcd_data8 as output for UART5_TXD (Mode 4)
   a. **Store 0x0C in conf_lcd_data8 at 0x44E10000+0x8C0**
10. Change UART mode to A, and setup for 8 bit, one stop, no parity
    a. **Write 0x83 in UART_LCR at 0x481AA000+0x0C**
11. Set DLL and DLH values for desired baud rate
    a. **Write 0x4E in UART_DLL at 0x481AA000+0x00**
    b. **Write 0x00 in UART_DLH at 0x481AA000+0x04**
12. Set baud rate mode to 16 in MDR1
    a. **Write 0x00 in UART_MDR1 at 0x481AA000+0x20 ACCORDING TO DATABOOK, THIS IS DONE AFTER LCR**
13. Switch back from mode A to operational mode
    a. **Write 0x03 in UART_LCR at 0x481AA000+0x0C**
14. Disable and clear FIFO
    a. **Write 0x6 to FCR_UART at 0x481AA000+0x08**
15. Enable IRQ input by clearing bit 7
    **a. BIC #0x80**

INT_DIRECTOR

1. Test to see if the interrupt came from the UART
   a. **Test (0x4000) bit 14 of INTC_PENDING_IRQ1 at 0x482000B8**
   b. If NO
      i. Branch to TCHK
2. Test to see if the interrupt came from ITT of IIR_UART
   a. **Test (0x01) bit 0 of IIR_UART at 0x481AA008**
   b. If yes
      i. Branch to TALKER_SVC

TCHK

1. Test to see if the interrupt came from the clock
   a. **Test bit 28 (0x10000000) of INTC_Pending_IRQ2 at 0x482000D8**
   b. If NO,
      i. Branch to BCHK

2. Test to see if the overflow for TIMER4 was triggered
    a. **Test bit 1 (0x02) of TIMER4_IRQStatus at 0x48044028**
    b. If NO
        i. Branch to PASS_ON
3. Reset overflow for timer
    a. **Store 0x02 in TIMER4_IRQStatus at 0x48044028**
4. Turn on UART5 interrupt by enabling THRIT and MODEMTIST
    a. **Store 0xA in IER_UART at 0x481AA004**
5. Force RTS to active low in UART_MCR Register
    a. **Store 0x2 in MCR_UART at 0x481AA010**
6. PASS_ON

BCHK

1. Test INTC_Pending to see if the interrupt came from the button
    a. **Test bit 0 (0x01) of INTC_PENDING_IRQ1 at 0x482000B8**
    b. If NO
        i. Branch to PASS_ON
2. Check GPIO2_IRQ for interrupt status
    a. **Test bit 1 (0x02) of GPIO2_IRQSTATUS_0 at 0x481AC02C**
    b. If YES
        i. Branch to BUTTON_SVC
    c. If NO
        i. Branch to PASS_ON

BUTTON_SVC

1. Reset GPIO2 Interrupt request in IRQSTATUS register
    a. **Store 0x02 in GPIO2_IRQSTATUS_0 at 0x481AC02C**
2. Start TIMER4
    a. **Store 0x01 in TIMER4_TCLR at 0x48044038**
3. PASS_ON

TALKER_SVC

1. Read bit 4 of Modem Status Register
    a. **Test bit 4 (0x10) of MSR_UART at 0x481AA018**
    b. If 1 then, read bit 5 of LSR_UART register
        i. **Test bit 5 (0x20) of LSR_UART at 0x481AA014**
        ii. If 1 then, SEND_CHAR
    c. If 0 then, branch to NOMODEM

SEND_CHAR

```
    1.  Load pointer to character in memory
    2.  Load pointer to character counter
    3.  Load character to be sent, increment pointer
    4.  Load character count
    5.  Setup pointer to THR register
    6.  Store character to be sent in THR register
            a.  Store character in THR_UART at 0x481AA000
    7.  Subtract counter value
    8.  Store counter back in data
    9.  Disable UART Interrupts
            a.  Store 0x0 in IER_UART at 0x481AA004
    10. Reload Timer with new value for 1 second
            a.  Store 0xFFFF8000 in TIMER_TCRR at 0x4804403C
    11. If last character is sent, then turn off timer 4
            a.  Store 0x00 in TIMER_TCLR at 0x48044038
            b.  ELSE
                    i.  Turn on TIMER4
                    ii. Store 0x01 in TIMER_TLCR at 0x48044038
    12. If last character has not been sent then
            a.  Branch to PASS_ON
    13. Reload character pointer with starting character
    14. Reload character count with original value
    15. Turn on USER LED0
            a.  Store 0x00200000  in GPIO2_SETDATAOUT at 0x4804C000

NOMODEM

        2.  Test Bit 5 of LSR_UART
                    iii. Test (0x20) in LSR_UART at 0x418AA000+0x60
                    iv.  If yes, then MASK INT
                            1.  Store 0b00 in IER_UART at 0x481AA000+0x04
                            2.  PASS_ON
                    v.   PASS_ON

PASS_ON

    1.  Restore register
    2.  Clear INTC_Control for NEWIRQ****
            a.  Store 0x01 at INTC_Control at 0x48200048
    3.  SUBS #4 outta there
```