#### 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
   1. If no, the test bit 0 at IRQ1
      1. If yes, then test bit 1 of GPIO2\_IRQStatus\_0
         1. If yes, then BUTTON\_SVC
   2. If yes, then test IIT of IIR\_UART
      1. 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
   1. If 1 then, read bit 5 of LSR\_UART register
      1. If 1 then, SEND\_CHAR
   2. 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
   1. If this isn’t the last character, PASS\_ON
   2. ELSE
      1. Reload character pointer
      2. Reload character counter
      3. PASS\_ON

NOMODEM

1. read bit 5 of LSR\_UART for TXFIFO
   * 1. If yes, then MASK INT
        1. PASS\_ON
     2. ELSE
        1. PASS\_ON

PASS\_ON

1. Restore register
2. SUBS #4 outta there

# LOW LEVEL

1. Setup stack for supervisor and IRQ

1. Setup GPIO2 clock and UART2 clock
   1. **Store 0x02 in CM\_PER\_GPIO2\_CLKCTRL at 0x44E000B0 to enable GPIO2 clock**
   2. **Store 0x02 in CM\_PER\_UART5\_CLKCTRL at 0x44E00038 to enable UART5 clock**
2. Setup falling edge detect for GPIO2\_1 and interrupt generation
   1. **Store 0x02 in GPIO2\_FALLINGDETECT at 0x4814C000+0x14C using READ, MODIFY, WRITE**
   2. **Store 0x02 in GPIO2\_IRQSTATUS\_SET\_0 at 0x4814C000+0x034**
3. Unmask bit 1 on MIR\_CLEAR1 to allow GPIOINT2A, and bit 14 to allow UART5INT
   1. **Store 0x4001 in INTC\_MIR\_CLEAR1 at 0x482000A8**
4. Remap lcd\_data14 as input for UART5\_CTSN (Mode 6)
   1. **Read, ORR, write 0x26 in conf\_lcd\_data14 at 0x44E10000+0x8D8**
5. Remap lcd\_data15 as output for UART5\_RTSN (Mode 6)
   1. **Read, ORR, write 0x06 in conf\_lcd\_data13 at 0x44E10000+0x8DC**
6. Remap lcd\_data9 as input for UART5\_RXD (Mode 4)
   1. **Read, ORR, write 0x24 in conf\_lcd\_data9 at 0x44E10000+0x8C4**
7. Remap lcd\_data8 as output for UART5\_TXD (Mode 4)
   1. **Read, ORR, write 0x04 in conf\_lcd\_data8 at 0x44E10000+0x8C0**
8. Change UART mode to A, and setup for 8 bit, one stop, no parity
   1. **Write 0x83 in UART\_LCR at 0x481AA000+0x0C**
9. Set DLL and DLH values for desired baud rate
   1. **Write 0x4E in UART\_DLL at 0x481AA000+0x00**
   2. **Write 0x00 in UART\_DLH at 0x481AA000+0x04**
10. Set baud rate mode to 16 in MDR1
    1. **Write 0x00 in UART\_MDR1 at 0x481AA000+0x20**
11. Switch back from mode A to operational mode
    1. **Write 0x03 in UART\_LCR at 0x481AA000+0x0C**
12. Disable and clear FIFO
    1. **Write 0x6 to FCR\_UART at 0x481AA000+0x08**
13. Enable IRQ input by clearing bit 7
    1. **BIC #0x80**

INT\_DIRECTOR

1. Save register on stack
2. Test bit 14 of IRQ for UART
   1. **Test (0x4000) in INTC\_PENDING\_IRQ1 at 0x482000B8**
   2. If no, the test bit 0 at IRQ1
      1. **Test (0x01) in INTC\_PENDING\_IRQ1 at 0x482000B8**
      2. 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
   3. If yes, then test IIT of IIR\_UART
      1. **Test (0x01) in IIR\_UART at 0x481AA000+0x08**
      2. If (interrupt present) 0, then TALKER\_SVC
3. Else PASS\_ON

BUTTON\_SVC

1. Turn off GPIO2\_1 interrupt
   1. **Store 0x02 in GPIO2\_IRQSTATUS at 0x481AC02C**
2. Set bit 1 of THR and bit 3 for Modem, change interrupt in IER\_UART
   1. **Write 0xA to IER\_UART at 0x481AA000+0x04**
3. PASS\_ON

TALKER\_SVC

1. Read bit 4 of Modem Status Register
   1. **Test (0x10) in Modem Status Register at 0x481AA000+0x18**
   2. If 1 then, read bit 5 of LSR
      1. **Test (0x20) in LSR\_UART at 0x481AA000+0x60**
      2. If 1 then, send a character
      3. Test to see if this is the final character
         1. If yes, then Point to first character for next send
            1. Disable UART\_INT

Reset THR enable bit in the IER\_UART

**Store 0x0000 in IER\_UART at 0x481AA000+0x04**

* + - 1. 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
   1. **STORE register value in THR\_UART at 0x481AA000**
7. Subtract counter value
8. Store counter back in data
   1. If this isn’t the last character, PASS\_ON
   2. ELSE
      1. Reload character pointer
      2. Reload character counter
      3. PASS\_ON

NOMODEM

1. Test Bit 5 of LSR\_UART
   * 1. **Test (0x20) in LSR\_UART at 0x418AA000+0x60**
     2. If yes, then MASK INT
        1. **Store 0b00 in IER\_UART at 0x481AA000+0x04**
        2. PASS\_ON
     3. PASS\_ON

PASS\_ON

1. Restore register
2. Clear INTC\_Control for NEWIRQ\*\*\*\*
   1. **Store 0x01 at INTC\_Control at 0x48200048**
3. 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
   1. If NO
      1. Branch to TCHK
2. Test to see if the interrupt came from ITT of IIR\_UART
   1. If yes
      1. Branch to TALKER\_SVC

TCHK

1. Test to see if the interrupt came from the clock
   1. If NO,
      1. Branch to BCHK
2. Test to see if the overflow for TIMER4 was triggered
   1. If NO
      1. 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
   1. If NO
      1. Branch to PASS\_ON
2. Check GPIO2\_IRQ for interrupt status
   1. If YES
      1. Branch to BUTTON\_SVC
   2. If NO
      1. Branch to PASS\_ON

BUTTON\_SVC

1. Reset GPIO2 Interrupt request in IRQSTATUS register
2. Start TIMER4
3. PASS\_ON

TALKER\_SVC

1. Read bit 4 of Modem Status Register
   1. If 1 then, read bit 5 of LSR\_UART register
      1. If 1 then, SEND\_CHAR
   2. 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
    1. ELSE
       1. Turn on TIMER4
12. If last character has not been sent then
    1. Branch to PASS\_ON
13. Reload character pointer with starting character
14. Reload character count with original value
15. Turn on USER LED0

NOMODEM

1. read bit 5 of LSR\_UART for TXFIFO
   * 1. If yes, then MASK INT
        1. PASS\_ON
     2. ELSE
        1. PASS\_ON

# LOW LEVEL

MAINLINE

1. Enable UART5, TIMER4, GPIO2, and GPIO1 clocks
   1. **Store 0x02 in CM\_PER\_GPIO2\_CLKCTRL at 0x44E000B0 to enable GPIO2 clock**
   2. **Store 0x02 in CM\_PER\_UART5\_CLKCTRL at 0x44E00038 to enable UART5 clock**
   3. **Store 0x02 in CM\_PER\_GPIO1\_CLKCTRL at 0x44E000AC to enable GPIO1 clock**
   4. **Store 0x02 in CM\_PER\_TIMER4\_CLKCTRL at 0x44E00088 to enable TIMER4 clock**
2. Initialize LEDs on GPIO1 for output
   1. **Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190**
   2. **Store 0xFE1FFFFF in GPIO1\_OE at 0x4804C000+0x134 using READ, MODIFY, WRITE**
3. Initialize falling\_edge\_detect and Interrupt on GPIO2\_1 for button
   1. **Store 0x02 in GPIO2\_FALLINGDETECT at 0x4814C000+0x14C using READ, MODIFY, WRITE**
   2. **Store 0x02 in GPIO2\_IRQSTATUS\_SET\_0 at 0x4814C000+0x034**
4. Initialize INTC for interrupts
   1. **Store 0x2 in INTC\_Sysconfig at 0x48200010**
   2. **Store 0x4001 in INTC\_MIR\_CLEAR1 at 0x482000A8**
   3. **Store 0x10000000 in INTC\_MIR\_CLEAR2 at 0x482000C8**
5. Initialize TIMER4 register or 1 second interrupts
   1. **Store 0x02 in PRCMCLKSEL\_TIMER4 at 0x44E00000 +0x510**
   2. **Store 0x01 to Timer 4 CFG at 0x48044000+0x010**
   3. **Store 0x02 to Timer 4 IRQ Enable at 0x48044000 + 0x02C**
   4. **Store 0xFFFF8000 in Timer 4 TLDR at 0x48044000 + 0x040**
   5. **Store 0xFFFF8000 in Timer 4 TCRR at 0x48044000 + 0x03C**
6. Remap lcd\_data14 as input for UART5\_CTSN (Mode 6)
   1. **Store 0x2E in conf\_lcd\_data14 at 0x44E10000+0x8D8**
7. Remap lcd\_data15 as output for UART5\_RTSN (Mode 6)
   1. **Store 0x0E in conf\_lcd\_data13 at 0x44E10000+0x8DC**
8. Remap lcd\_data9 as input for UART5\_RXD (Mode 4)
   1. **Store 0x2C in conf\_lcd\_data9 at 0x44E10000+0x8C4**
9. Remap lcd\_data8 as output for UART5\_TXD (Mode 4)
   1. **Store 0x0C in conf\_lcd\_data8 at 0x44E10000+0x8C0**
10. Change UART mode to A, and setup for 8 bit, one stop, no parity
    1. **Write 0x83 in UART\_LCR at 0x481AA000+0x0C**
11. Set DLL and DLH values for desired baud rate
    1. **Write 0x4E in UART\_DLL at 0x481AA000+0x00**
    2. **Write 0x00 in UART\_DLH at 0x481AA000+0x04**
12. Set baud rate mode to 16 in MDR1
    1. **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
    1. **Write 0x03 in UART\_LCR at 0x481AA000+0x0C**
14. Disable and clear FIFO
    1. **Write 0x6 to FCR\_UART at 0x481AA000+0x08**
15. Enable IRQ input by clearing bit 7
    1. **BIC #0x80**

INT\_DIRECTOR

1. Test to see if the interrupt came from the UART
   1. **Test (0x4000) bit 14 of INTC\_PENDING\_IRQ1 at 0x482000B8**
   2. If NO
      1. Branch to TCHK
2. Test to see if the interrupt came from ITT of IIR\_UART
   1. **Test (0x01) bit 0 of IIR\_UART at 0x481AA008**
   2. If yes
      1. Branch to TALKER\_SVC

TCHK

1. Test to see if the interrupt came from the clock
   1. **Test bit 28 (0x10000000) of INTC\_Pending\_IRQ2 at 0x482000D8**
   2. If NO,
      1. Branch to BCHK
2. Test to see if the overflow for TIMER4 was triggered
   1. **Test bit 1 (0x02) of TIMER4\_IRQStatus at 0x48044028**
   2. If NO
      1. Branch to PASS\_ON
3. Reset overflow for timer
   1. **Store 0x02 in TIMER4\_IRQStatus at 0x48044028**
4. Turn on UART5 interrupt by enabling THRIT and MODEMTIST
   1. **Store 0xA in IER\_UART at 0x481AA004**
5. Force RTS to active low in UART\_MCR Register
   1. **Store 0x2 in MCR\_UART at 0x481AA010**
6. PASS\_ON

BCHK

1. Test INTC\_Pending to see if the interrupt came from the button
   1. **Test bit 0 (0x01) of INTC\_PENDING\_IRQ1 at 0x482000B8**
   2. If NO
      1. Branch to PASS\_ON
2. Check GPIO2\_IRQ for interrupt status
   1. **Test bit 1 (0x02) of GPIO2\_IRQSTATUS\_0 at 0x481AC02C**
   2. If YES
      1. Branch to BUTTON\_SVC
   3. If NO
      1. Branch to PASS\_ON

BUTTON\_SVC

1. Reset GPIO2 Interrupt request in IRQSTATUS register
   1. **Store 0x02 in GPIO2\_IRQSTATUS\_0 at 0x481AC02C**
2. Start TIMER4
   1. **Store 0x01 in TIMER4\_TCLR at 0x48044038**
3. PASS\_ON

TALKER\_SVC

1. Read bit 4 of Modem Status Register
   1. **Test bit 4 (0x10) of MSR\_UART at 0x481AA018**
   2. If 1 then, read bit 5 of LSR\_UART register
      1. **Test bit 5 (0x20) of LSR\_UART at 0x481AA014**
      2. If 1 then, SEND\_CHAR
   3. 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
   1. **Store character in THR\_UART at 0x481AA000**
7. Subtract counter value
8. Store counter back in data
9. Disable UART Interrupts
   1. **Store 0x0 in IER\_UART at 0x481AA004**
10. Reload Timer with new value for 1 second
    1. **Store 0xFFFF8000 in TIMER\_TCRR at 0x4804403C**
11. If last character is sent, then turn off timer 4
    1. **Store 0x00 in TIMER\_TCLR at 0x48044038**
    2. ELSE
       1. Turn on TIMER4
       2. **Store 0x01 in TIMER\_TLCR at 0x48044038**
12. If last character has not been sent then
    1. Branch to PASS\_ON
13. Reload character pointer with starting character
14. Reload character count with original value
15. Turn on USER LED0
    1. **Store 0x00200000 in GPIO2\_SETDATAOUT at 0x4804C000**

NOMODEM

1. Test Bit 5 of LSR\_UART
   * 1. **Test (0x20) in LSR\_UART at 0x418AA000+0x60**
     2. If yes, then MASK INT
        1. **Store 0b00 in IER\_UART at 0x481AA000+0x04**
        2. PASS\_ON
     3. PASS\_ON

PASS\_ON

1. Restore register
2. Clear INTC\_Control for NEWIRQ\*\*\*\*
   1. **Store 0x01 at INTC\_Control at 0x48200048**
3. SUBS #4 outta there