2/3/2018

Project 1

ECE 371

**Phong Nguyen**

Table of Contents

[PART 1 LEDS SEQUENCE 2](#_Toc505461525)

[Part 1 Flowchart 2](#_Toc505461526)

[GPIO1 pins Table 2](#_Toc505461527)

[Part 1 Algorithm 3](#_Toc505461528)

[Low Level algorithm 3](#_Toc505461529)

[Part 1 Assembly code 4](#_Toc505461530)

[PART 2 SETTING UP INTERRUPT 6](#_Toc505461531)

[Part 2 Flow chart 6](#_Toc505461532)

[Interrupt controller unmask 6](#_Toc505461533)

[Part 2 Algorithm 7](#_Toc505461534)

[Assembly code 7](#_Toc505461535)

[PART 3 SETTING UP DTIMER 12](#_Toc505461536)

[Part 3 Flow chart 12](#_Toc505461537)

[Setting up register Timer3 12](#_Toc505461538)

[Part 3 Algorithm 13](#_Toc505461539)

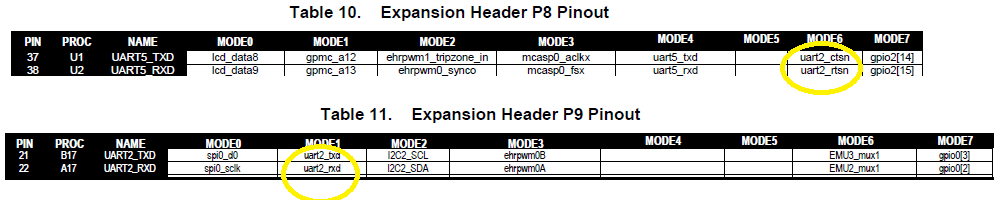
[Low Algorithm Part 3 14](#_Toc505461540)

[Part 3 Assembly code 16](#_Toc505461541)

# PART 1 SETUP UART2

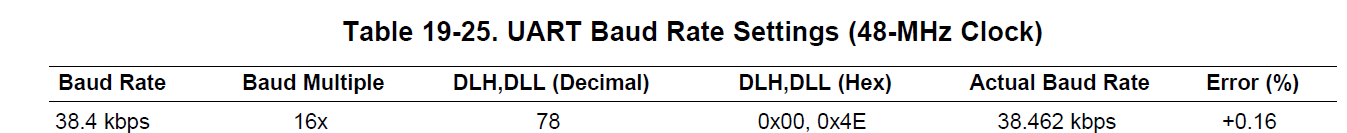
## Initialization for UART2

To map the UART2, I have to know the pins available to the Beagle Bones Black P9 and P8 connector by changing the MUX that select the signal go to the pads.



In table 10 and table 11 above, The UART that is needed for this project is TxD, RxD, CTS, RTS. The default of the MUX when initialized it will begin in MODE0. Essentially in the manual ARM355x, the register name that use to map will be named in MODE0. So all we need to look up LCD\_data8, lcd\_data9 change to mode 6 and spi0\_d0, and sp0\_sclk change to mode 1.

After mapping an and started the UART2 clock, next step is setting the desired Baud rate for UART to communicate with RC8660 Talker through RS-232C. This is a bit complex, there is a requirement for switching mode from operational mode UART to configuration mode A to change the DHL and DLL. The clock is divided by the value written to the DLH and DLL register and the result is then divided by further by 13 or 16. So we all we need to do is change the mode write desired baud rate value into DHL and DLL. In the table 19-25 in the manual, the value that we want is 38.4 Kbps. As we can see we need to write 0x00 for DLH and 0x4E for DLL.



After writing value for DLH and DLL and set up the UART 16x mode by writing 0x00 to the MDR1 register. We need to switch back to operational mode.

Now for the Interrupt part, there are 2 main interrupt signal will be generated from the UART. They are THR interrupts and MODEM status interrupts. Initialized these 2 will be for the part of sending character through RS-232C.

For part of the project, there is no requirement for FIFO. So we also need to disable FIFO.

## Interrupt unmasking

Unmasking 2 interrupts in INTC register. UART2INT is at int number 74 and the button GPIO1A is at int number 98. For GPIO1A, the pin calculated was pin 2 at INTC\_MIR\_CLEAR3 register. For UART2INT, the pin calculated was pin 10 at INTC\_MIR\_CLEAR2. Simply write 0x04 to INTC\_MIR\_CLEAR3 and write 0x400 to INTC\_MIR\_CLEAR2.

## Sending bytes to talker

There are 2 important signal we need to check in order to send character to talker. First is Clear to send (CTS#) active low, we need to look at MODEM register (MSR) bit 4. Second is Transmit Holding Register (THR), we need to look at Line Control Register (LSR) bit 5

Now we need to focus MSR bit 4 and LSR bit 5 and when is the Last char in the String. Making a truth table will be easier to keep track what is the next step to do after sending character. Because of MSR bit 4 is the signal come from Talker so we need to priority check this bit.

|  |  |  |  |
| --- | --- | --- | --- |
| MSR bit 4 | LSR bit 5 | Last character | Next step |
| 0 | 0 | x | Go to PASS ON |
| 0 | 1 | x | **Masking THR interrupt** and go to PASS ON |
| 1 | 0 | X | Go to PASS ON |
| 1 | 1 | No | **Sending a character** and go to PASS ON |
| 1 | 1 | Yes | **Sending a character** and **Set the Pointer to first Char** then go to PASS ON |

## Part 1 Algorithm

High level language:

1. Setting up stack
2. Wake up GPIO1 module
3. Clear pin for GPIO1 module
4. Setting button GPIO1\_30 with falling edge detect
5. Setting up Interrupt controller INTC for GPIO1\_30 and UART2
6. Select MUX for UART2
7. Turn on UART2 Clock
8. Setting up UART2 (desired baud rate, 8 data bits, one stop bit, and no parity, disable FIFO and enable interrupt enable register IER\_UART)

INT\_DIRECTOR

1. Save register on stack
2. Check INTC interrupt from the UART, if not then go to check INTC interrupt from the button
3. If INTC from the UART, go to check IIR bit of IIR\_UART register UART2, if is 1 then go to TLKR\_SVC to send the character, if all is ready. If not, go to PASS\_ON
4. If INTC from the GPIO1, go to check GPIO1\_IRQSTATUS\_0 register, if is 1 then go to BUTTON\_SVS. If not , go to PASS\_ON

BUTTON\_SVC

1. Turn off INTC request from GPIO1
2. And NEWIRQ bit in INTC
3. Enable UART5 interrupt, writing a word to the IER\_UART2 register at offset 0x04. Setting bit 1 for THR transmit Holding Register, and Setting bit 3 enable MODEM change Interrupt.

TALKER\_SVC

1. Check if CTS# is Low (MSR-bit 4==1) **and** THR is empty (LSR-bit 5==1) go to send a char at UART2 THR. If it’s the last character reset pointer to the first character. Disable UART INTERRUPT clearing bit 1 and 3 in UART\_IER
2. If CTS# is low (MSR-bit4==1) **and** THR is not empty (LSR-bit5==0) go to PASS ON
3. If CTS# is high (MSR-bit4==0) **and** THR is empty (LSR-bit==1) go disable THR interrupt by resetting the THR enable bit in UART\_IER to prevent spinning, go to PASS ON
4. Check if CTS# is high (MSR-bit 4==0) **and** THR is not empty (LSR-bit 5==0) Go to PASS ON

PASS ON

1. Turn off NEWIRQA bit in INTC\_CONTROL so can respond to new IRQ

## Part 1 Low Level algorithm

1. Setting up STACK
   1. Point to top of stack for service mode 0x10000 CPS #0x12
      * 1. Point to top of stack for IRQ mode 0x10000 CPS#0x13
2. Turn on GPIO1 CLK
   1. Turn on the GPIO at 0x44300AC write 0x02
3. Setting up Falling edge detect on GPIO1\_30
   1. GPIO1\_FALLINGDETECT 0x4804C14C
   2. Pin 30 0x40000000
   3. Setting up GPIO1\_IRQSTATUS
4. INTC interrupt controller 0x48200000
   1. Unmasking GPIOINTA write 0x04 offset 0xE8 MIR\_CLEAR3
   2. Unmasking UARTINT write 0x400 offset 0xC8 MIR\_CLEAR2
5. UART2 Setting up
   1. RxD UART2 mapping R-M-W
      1. 0x44E10950 spi0\_sclk mode 1: 001
   2. TxD UART2 mapping R-M-W
      1. 0x44E10954 spi0\_d0 mode 1: 001
   3. CTSN UART2 mapping R-M-W
      1. 0x44E108C0 lcd\_data8 mode 6: 110
   4. RTSN UART2 mapping R-M-W
      1. 0x44E108C4 lcd\_data9 mode 6: 110
6. Turn on UART2 clock CM\_PER\_UART2\_CLKCTRL
   1. Address 0x44E00070 write 0x02
7. Switching to configuration mode A
   1. Write 0x83 at 0x4802400C LCR register
   2. Setting DLL 0x48024000 value 0x4E
   3. Setting DLH 0x48024004 value 0x00
   4. Setting UART 16x mode write 0x0 for 0x48024020
8. Switch back to Operational mode write 0x7E to 0x48024008 FCR register
9. Clearing FIFO
   1. Write 0x06 to 0x48024008
10. CPSR IRQ enable
    1. Bit clear 7 CPSR\_c

INT\_DIRECTOR:

1. Check INTC UART2 0x482000D8 pin 10 0x400
2. Check INTC Button 0x482000F8 pin 2 0x04
3. Check UART 0x48024008 jump tp TALKER\_SVC
4. Check Button 0x4804C02C IRQ\_STATUS jump to BUTTON\_SVC

BUTTON\_SVC:

1. Turn off interrupt 0x4804C02C write 0x40000000 pin 30
2. Enable IER\_UART 0x48024004 write 0x000A
3. Go to PASS ON

TALKER\_SVC:

1. Check UART\_MSR 0x48024000 bit 4 jump to NO\_CTS if there is 0
2. Check UART\_LSR 0x48024014 bit 5 jump to SEND\_CHAR if there is 1
3. ELSE PASS ON

NO\_CTS

1. Unmasking THR interrupt 0x48024014 and modem status register 0x48024004
2. PASS ON

SEND\_CHAR

1. Turn on THR interrupt and modem status register
2. Load the pointer and load count value
3. Write char byte to THR register 0x8024000
4. Keep sending until counter is less than 0
5. Store counter and return PASSON

PASS\_ON:

1. INTC\_CONTROL 0x48200048 clear bit 0 and set bit 1
2. Return to mainline.

## Part 1 Assembly code

@Setting up STACK

**.text**

**.global** \_start

**.global** INT\_DIRECTOR

**\_start:**

LDR R13,=STACK1 @Point to base of STACK for SVC mode

ADD R13, R13, #0x1000 @point to top of the STACK

CPS #0x12

LDR R13,=STACK2 @Point to base of STACK for IRQ mode

ADD R13, R13, #0x1000 @Point to top of the STACK

CPS #0x13 @Back to SVC mode

@===========================================================================

@===========================================================================

@ Turn on GPIO1 CLCK

MOV R0,#0x02 @value to turn on GPIO1\_CLK

LDR R1,=0x44E000AC @Load the address for the CM\_PER\_GPIO1\_CLKCTRL

STR R0,[R1] @ Write value to the register

@BUTTON SETUP using GPIO1 pin 30 for input button falling edge detect

@also enable its IRQ

@ Detect falling edge on GPIO1\_30

LDR R0,=0x4804C000 @base address for GPIO1 register

ADD R1,R0,#0x14C @R1 is is GPIO1\_FALLINGDETECT register

MOV R2,#0x40000000 @Load value pin GPIO1\_30 push button

LDR R3,[R1] @Load current value of GPIO1\_FALLINGDETECT register

ORR R3,R3,R2 @Modify the pin GPIO\_30 push button

STR R3, [R1] @Store GPIO\_30 push button for GPIO1\_FALLINGDETECT

@ GPIO1\_IRQSTATUS  on pin 30 R2 0x40000000 sending interrupts to POINTPEND1

ADD R1,R0,#0x34 @R1 address of GPIO1\_IRQSTATUS\_SET\_0 register

STR R2,[R1] @Store pin 30 for GPIO1\_IRQSTATUS\_SET\_0

@ Initialize INTC Interrupts Controller

LDR R1,=0x48200000 @R1 loading base address for interrupts controller

MOV R2,#0x2 @Value to reset the INTC\_CONFIG register

STR R2,[R1,#0x10] @write value to reset the INTC CONFIG

MOV R2,#0x04 @ Value to unmask GPIOINTA , pin 2 INTC INT #98

STR R2,[R1,#0xE8] @Write value to INTC\_MIR\_CLEAR3 register

MOV R2,#0x400 @value to unmask UART2INT pin 10 INTC INT #74

STR R2,[R1,#0xC8] @Write value to INTC\_MIR\_CLEAR2 register

@===================================================================================

@===================================================================================

@===================================================================================

@UART2 Setting up Section

@Mapping the UART2 TxD,RxD,CTS, and RTS using MUX.

@0x44E10000 base address for Control Module

@mode 1 : 001. need to AND 0xFFFFFFF9 for 0s at bit 2 and 3 . OR 0x1 for 1 last bit

@mode 6: 110. need to OR 0x110 for 1s at bit 2 and 3. AND 0xFFFFFFF1 for 0 last bit

@RxD UART2 mapping

LDR R0,=0x44E10950 @adding offset spi0\_sclk in mode 1 for RxD UART2

LDR R2,[R0] @Loading current value spi0\_sclk

AND R2,R2,#0xFFFFFFF9 @AND for 0s at bit 2 and 3

ORR R2,R2,#0x1 @OR for 1 last bit

STR R2, [R1] @store back for register spi0\_sclk switching mode 1

@TxD UART2 mapping

LDR R1,=0x44E10954 @spi0\_d0 in mode 1 for TxD UART2

LDR R3,[R1] @Loading current value spi0\_d0

AND R3,R3,#0xFFFFFFF9 @AND for 0s at bit 2 and 3

ORR R3,R3,#0x1 @OR for 1 last bit

STR R3, [R1] @store back for register spi0\_d0 switching mode 1

@CTSN UART2 mapping

LDR R0,=0x44E108C0 @Lcd\_data8 in mode 6 for CTSN UART2

LDR R2,[R0] @Loading current value lcd\_data8

AND R2,R2,#0xFFFFFFFE @AND 0 for last bit

ORR R2,R2,#0x6 @OR 1s for bit 3 and 2

STR R2, [R0] @store back for register lcd\_data8 switching mode 6

@RTSN UART2 mapping

LDR R1,=0x44E108C4 @Lcd\_data9 in mode 6 for RTSN UART2

LDR R3,[R1] @Loading current value lcd\_data9

AND R3,R3,#0xFFFFFFFE @AND 0 for last bit

ORR R3,R3,#0x6 @OR 1s for bit 3 and 2

STR R3, [R1] @store back for register lcd\_data9 switching mode 6

@============================================================================

@Turn on UART2 clock CM\_PER\_UART2\_CLKCTRL

MOV R2,#0x02 @value to turn on the clock

LDR R1,=0x44E00070 @base address for UART2 clock with offset 0x70

STR R2,[R1] @Store value to turn on the clock

@============================================================================

@ Setting up Baud rate, 8 data bits, 1 stop bit, no parity, disable FIFO and enable interrupt IER\_UART

@ UART2 base address register 0x4802\_4000

@0x48024000 base address UART2

@Switch to Configuration mode A in UART\_LCR offset 0x0C “bit 7 is 0x1”

@and 8 data  bits , one stop bit, no parity. bit 3 is 0, bit 2 is 0, and 1:0 will both 1

LDR R0,=0x4802400C @offset of LCR register

MOV R2,#0x83 @load value of LCR into R2

STR R2,[R0] @store back in LCR register

@Set 38.4 kbps Baud rate and a 16x divisor, DLL offset 0x00 write 0x00, DLH offset 0x04 write 0x4E

@Set DLL

LDR R1,=0x48024000 @Address for DLL

MOV R3,#0x4E @write value 0x00 to DLL

STR R3,[R1] @write 0x00 to DLL offset 0x0

@Set DLH

LDR R0,=0x48024004 @address for DLH offset 0x04

MOV R2,#0x00 @write value 0x4E to DLH

STR R2,[R0] @Store value 0x4E for DLH

@Mode Definition 1 Register MDR offset 0x20 need to write 0x0 for UART 16x mode

LDR R1,=0x48024020 @R1 address MDR

MOV R3,#0x0 @value for 16x mode

STR R3,[R1] @store value in MDR register

@Change back Operational mode  in UART\_LCR  offset 0x0C  “bit 7 is 0x0”

LDR R0,=0x4802400C @0xC offset of LCR register

LDR R2,[R0] @load value of LCR into R2

AND R2,R2,#0x7F @write 0 to bit 7

STR R2,[R0] @store back in LCR register

@Clearing FIFO in FCR register offset at 0x8 bit 1 RX\_FIFO, bit 2 TX\_FIFO, bit 0 FIFO\_EN.

LDR R1,=0x48024008 @R1 address if FCR register

MOV R3,#0x06 @value bit 0 ==0, bit 1==1 and bit 2==1  so 110==0x6

STR R3,[R1] @store value for FCR register

@===================================================================================

@==================================================================================

@ Make sure Processor CPSR IRQ enable

MRS R3,CPSR @ Copy CPSR to R3

BIC R3,#0x80 @Clear bit 7

MSR CPSR\_c, R3 @Write back to CPSR

@wait for interrupts

**LOOP:** NOP

B LOOP

@==================================================================================

@==================================================================================

**INT\_DIRECTOR:**

@Save the register on the stack

STMFD SP!, {R0-R3,LR} @Push register on the stack

@First check UART IRQ then check IRQ from button

**CHECK\_INTC\_UART:**

LDR R0,=0x482000D8 @INTC\_PENDING\_IRQ2 for UART2 INTC

LDR R2,[R0] @load value of PENDING\_IRQ2

TST R2,#0x400 @UART2INT pin 10 INTC INT #74

BNE CHECK\_UART @Go to check IRQ in UART2

**CHECK\_INTC\_BUTTON:**

LDR R0,=0x482000F8 @Address of INTC\_PENDING\_IRQ3 check button

LDR R2,[R0] @Load value to check

TST R2,#0x04 @Button pin 2 INTC INT #98

BNE CHECK\_BUTTON @Go to check button in GPIO1

@Checking UART IRQ using IIR\_UART offset 0x8

**CHECK\_UART:**

LDR R0,=0x48024008 @IIR\_UART2 base address

LDR R2,[R0] @Load value of IIR\_UART2

TST R2,#0x1 @check bit 0 IT\_PENDING

BEQ TALKER\_SVC @if 1 no IRQ, else 0 there is an IRQ

B PASS\_ON @if 1 then go to PASS\_ON

**CHECK\_BUTTON:**

LDR R0,=0x4804C02C @Check GPIO IRQ STATUS

LDR R2,[R0] @Load value from GPIO

TST R2,#0x40000000 @Testing if GPIO1\_30 IRQ register is 1 or 0

BNE BUTTON\_SVC @if z flag is clear Go to button service

B PASS\_ON @z flag is set go to PASS\_ON

@===================================================================================

@Service section remember to turn off interrupt request for register

**BUTTON\_SVC:**

@turn off IRQ request for GPIO1

LDR R1,=0x4804C02C @GPIO1\_IRQSTATUS\_0 address

MOV R2,#0x40000000 @turn off GPIO1\_IRQSTATUS at pin 30 by write a 1

STR R2,[R1] @Writing value to turn IRQ off GPIO1\_IRQ\_RAW\_0

@enable Interrupt Enable Register IER\_UART at offset 0x04

@bit 3 is 1 for MODEM Status Change MSC

@bit 1 is 1 for Transmit Holding Register THR the rest is all 0s for reset value

LDR R1,=0x48024004 @R1 address of IER\_UART

MOV R2,#0x000A @1010 == 0xA value

STR R2,[R1] @Store value into R1 IER\_UART

B PASS\_ON @Go to PASS\_ON

**TALKER\_SVC:**

@There are 2 main registers take part in making decision whether to send character or pass on

@Modem status register MSR bit 4 (Clear to send CTS#)

@Line status register LSR bit 5 (Transmit holding register THR)

LDR R0,=0x48024000 @Address UART\_MSR register

ADD R1,R0,#0x18

LDR R2,[R1] @value to check bit 4 clear to send CTS#

TST R2,#0x10 @check if CTS# is currently asserted

BEQ NO\_CTS @go to NO\_CTS

LDR R0,=0x48024014 @Address UART\_LSR register

LDR R2,[R0] @value to check bit 5 Transmit holding register THR

TST R2,#0x20 @check if LSR bit 5 is currently asserted

BNE SEND\_CHAR @Go to SEND\_CHAR if LSR bit 5 is 1

B PASS\_ON @otherwise if LSR bit 5 is 0 go to PASS\_ON

@@@@@@@@@@@@@@@

**NO\_CTS:**

LDR R0,=0x48024014 @Address UART\_LSR to check bit 5 for THR

LDR R2,[R0] @Load value to check bit 5  UART\_LSR

TST R2, #0x20 @Check if LSR bit 5 is currently asserted

BEQ PASS\_ON @if LSR bit 5 is 0 go to PASS\_ON

@otherwise if LSR bit 5 is 1 disable THR interrupt and modem status register interrupt to avoid Spinning

LDR R0,=0x48024004 @Address IER\_UART to clear bit 1 and 3

LDR R2,[R0] @Load value to R-M-W

AND R2,R2,#0x5 @Clear bit 1 and 3

STR R2,[R0] @Write back to change bit 1 and 3

B PASS\_ON @Go to PASS\_ON after clear

@@@@@@@@@@@@@@@

**SEND\_CHAR:**

@Turn on the THR interrupt and modem status register interrupt

LDR R0,=0x48024004 @address for IER\_UART

MOV R2,#0x0A @Set bit 3 = MODEM STATUS, bit 1 = THR interrupt

STR R2, [R0] @Write to IER\_UART

@Send character

LDR R0,=CHAR\_PTR @R0 = address of pointer store

LDR R1,=CHAR\_COUNT @R1 = address of count store location

LDR R2,[R0] @R2=Address of desired character in text string

LDR R3,[R1] @R3=current character count value

LDRB R4,[R2],#1 @Read char to send from string

STR R2,[R0] @Incremented address store BACK in CHAR\_PTR

LDR R5,=0x48024000 @Load Transmit holding register to send char

STRB R4,[R5] @Send char to Transmit buffer

SUBS R3,R3,#1 @Decrement Character count value by 1

STR R3,[R1]

BPL PASS\_ON @Greater than or equal zero, more Characters go back

LDR R2,=MESSAGE @else if done then reload. Get address of starting string

STR R2,[R0] @Write value reload back to address of pointer

MOV R2,#54 @Reload original number of char in String again

STR R2,[R1] @Write to address of counter for the next message

**PASS\_ON:**

LDR R0,=0x48200048 @Address if INTC\_CONTROL register

MOV R1,#0x1 @Value to clear bit 0

STR R1,[R0] @Write to INTC\_CONTROL register

LDMFD SP!, {R0-R3,LR} @restore register

SUBS PC,LR,#4 @return from the IRQ

**.data**

**.align** 2

**STACK1:** .rept 1024

**.word** 0x0000

.endr

**STACK2:** .rept 1024

**.word** 0x0000

.endr

**MESSAGE:** **.byte** 0x0D

**.ascii** "Your blood pressure is 120 over 70. Your pulse is 54."

**.byte** 0x0D

**.align** 2

**CHAR\_PTR:** **.word** MESSAGE @Pointer to next character

**CHAR\_COUNT:** **.word** 54 @counter for character to send

.end

# PART 2 SETUP TIMER

## Setting up register Timer3



As, I test the String send to talker takes about 5 seconds to complete message “Your blood pressure is 120 over 70. Your pulse is 54.”. So adding up we need to set timer around 15 seconds.

Using Timer3 TCLR register to control the clock. TLDR will be the register to reload the value. The DTIMER will generate interrupt when it overflows. We chose 32.768 KHZ clock for DTIMER3. It means 32768 pulse per second. For 15 seconds 32768 x 15=491520 in hex would be 0x00078000. To get the desired time, follow the formula:

Desired time (15 seconds) = 0x1\_0000\_0000 – 0x0007\_8000 = **0xFFF88000**

## Part 2 Algorithm

1. High level language:
2. Setting up stack
3. Wake up GPIO1 module
4. Clear pin for GPIO1 module
5. Setting up DTIMER3 and load value for 15 secs
6. Setting button GPIO1\_30 with falling edge detect
7. Setting up Interrupt controller INTC for GPIO1\_30, UART2, and DTIMER3
8. Select MUX for UART2
9. Turn on UART2 Clock
10. Setting up UART2 (desired baud rate, 8 data bits, one stop bit, and no parity, disable FIFO and enable interrupt enable register IER\_UART)

INT\_DIRECTOR

1. Save register on stack
2. Check INTC interrupt from the UART, if not then go to check INTC interrupt from the button
3. If INTC from the UART, go to check IIR bit of IIR\_UART register UART2, if is 1 then go to TLKR\_SVC to send the character, if all is ready. If not, go to PASS\_ON
4. If INTC from the GPIO1, go to check GPIO1\_IRQSTATUS\_0 register, if is 1 then go to BUTTON\_SVS. If not, go to PASS\_ON
5. If INTC from the TIMER3, go to check IRQSTATUS\_TIMER3 register, if 1 then go to TIMER\_SVC. If not, go to PASS ON.

BUTTON\_SVC

1. Turn off INTC request from GPIO1
2. Check if Timer is running if not then go to TIMER\_ON otherwise TIMER\_OFF

TIMER\_ON

1. Load value timer to start and reload the timer and go to PASS\_ON

TIMER\_OFF

1. Load value to turn off timer and go to PASS\_ON

TIMER\_SVC

1. Turn off IRQ request for Timer
2. Load value to enable MODEM STATUS Change and Transmit Holding register
3. Go to PASS\_ON

TALKER\_SVC

1. Check if CTS# is Low (MSR-bit 4==1) **and** THR is empty (LSR-bit 5==1) go to send a char at UART2 THR. If it’s the last character reset pointer to the first character. Disable UART INTERRUPT clearing bit 1 and 3 in UART\_IER
2. If CTS# is low (MSR-bit4==1) **and** THR is not empty (LSR-bit5==0) go to PASS ON
3. If CTS# is high (MSR-bit4==0) **and** THR is empty (LSR-bit==1) go disable THR interrupt by resetting the THR enable bit in UART\_IER to prevent spinning, go to PASS ON
4. Check if CTS# is high (MSR-bit 4==0) **and** THR is not empty (LSR-bit 5==0) Go to PASS ON
5. After sending Character reach the end always reload the Timer and pointer to the start for the next IRQ button.

PASS ON

1. Turn off NEWIRQA bit in INTC\_CONTROL so can respond to new IRQ

## Part 2 Low Level algorithm

1. Setting up STACK
   1. Point to top of stack for service mode 0x10000 CPS #0x12
      1. Point to top of stack for IRQ mode 0x10000 CPS#0x13
2. Turn on GPIO1 CLK
   1. Turn on the GPIO at 0x44300AC write 0x02
3. Setting up Falling edge detect on GPIO1\_30
   1. GPIO1\_FALLINGDETECT 0x4804C14C
   2. Pin 30 0x40000000
   3. Setting up GPIO1\_IRQSTATUS
4. Timer Setup DTIMER3 32.768 Khz clock turn on clock write 0x2 to 0x44E00084
5. And MUX CLKSEL\_TIMER3\_CLK 0x44E0050C
6. Reset the timer DTIMER3 and Enable Overflow interrupt write 0x2 to 0x4804202C
7. INTC interrupt controller 0x48200000
   1. Unmasking GPIOINTA write 0x04 offset 0xE8 MIR\_CLEAR3
   2. Unmasking UARTINT write 0x400 offset 0xC8 MIR\_CLEAR2
   3. Unmasking DTIMER3 write 0x20 offset 0xC8 MIR\_CLEAR2
8. UART2 Setting up
   1. RxD UART2 mapping R-M-W
      1. 0x44E10950 spi0\_sclk mode 1: 001
   2. TxD UART2 mapping R-M-W
      1. 0x44E10954 spi0\_d0 mode 1: 001
   3. CTSN UART2 mapping R-M-W
      1. 0x44E108C0 lcd\_data8 mode 6: 110
   4. RTSN UART2 mapping R-M-W
      1. 0x44E108C4 lcd\_data9 mode 6: 110
9. Turn on UART2 clock CM\_PER\_UART2\_CLKCTRL
   1. Address 0x44E00070 write 0x02
10. Switching to configuration mode A
    1. Write 0x83 at 0x4802400C LCR register
    2. Setting DLL 0x48024000 value 0x4E
    3. Setting DLH 0x48024004 value 0x00
    4. Setting UART 16x mode write 0x0 for 0x48024020
11. Switch back to Operational mode write 0x7E to 0x48024008 FCR register
12. Clearing FIFO
    1. Write 0x06 to 0x48024008
13. CPSR IRQ enable
    1. Bit clear 7 CPSR\_c

INT\_DIRECTOR:

1. Check INTC UART2 0x482000D8 pin 10 0x400
2. Check INTC Button 0x482000F8 pin 2 0x04
3. Check INTC Timer 0x482000D8 pin 5 0x020
4. Check UART 0x48024008 jump tp TALKER\_SVC
5. Check Button 0x4804C02C GPIO1 IRQ jump to BUTTON\_SVC
6. Check Timer 0x48042028 IRQSTATUS\_TIMER

BUTTON\_SVC:

1. Turn off interrupt 0x4804C02C write 0x40000000 pin 30
2. Check if timer is running 0x48042038 jump to TIMER\_ON if its 1 otherwise TIMER\_OFF if its 0
3. Go to PASS ON

TIMER\_ON

1. Reset IER\_UART 0x48024004
2. Turn on TIMER3 write 0x03 to start and reload to 0x48042038 TIMER TCLR
3. Go to PASS\_ON

TIMER\_OFF:

1. Turn off the timer write 0x00 at 0x48042038 TIMER TCLR

TIMER\_SVC:

1. Turn of IRQ request for TIMER3
2. Enable IRQ for MODEM STATUS and Transmit Holding register write 0x0A to 0xd48024004 IER\_UART

TALKER\_SVC:

1. Check UART\_MSR 0x48024000 bit 4 jump to NO\_CTS if there is 0
2. Check UART\_LSR 0x48024014 bit 5 jump to SEND\_CHAR if there is 1
3. ELSE PASS ON

NO\_CTS

1. Unmasking THR interrupt 0x48024014 and modem status register 0x48024004
2. PASS ON

SEND\_CHAR

1. Turn on THR interrupt and modem status register
2. Load the pointer and load count value
3. Write char byte to THR register 0x8024000
4. Keep sending until counter is less than 0
5. Reload the TIMER write 0x03 for 0x48042038
6. Store counter and return PASSON

PASS\_ON:

1. INTC\_CONTROL 0x48200048 clear bit 0 and set bit 1
2. Return to mainline.

## Part 2 Assembly code

@Setting up STACK

**.text**

**.global** \_start

**.global** INT\_DIRECTOR

**\_start:**

LDR R13,=STACK1 @Point to base of STACK for SVC mode

ADD R13, R13, #0x1000 @point to top of the STACK

CPS #0x12

LDR R13,=STACK2 @Point to base of STACK for IRQ mode

ADD R13, R13, #0x1000 @Point to top of the STACK

CPS #0x13 @Back to SVC mode

@===========================================================================

@===========================================================================

@ Turn on GPIO1 CLCK

MOV R0,#0x02 @value to turn on GPIO1\_CLK

LDR R1,=0x44E000AC @Load the address for the CM\_PER\_GPIO1\_CLKCTRL

STR R0,[R1] @ Write value to the register

@BUTTON SETUP using GPIO1 pin 30 for input button falling edge detect

@also enable its IRQ

@ Detect falling edge on GPIO1\_30

LDR R0,=0x4804C000 @base address for GPIO1 register

ADD R1,R0,#0x14C @R1 is is GPIO1\_FALLINGDETECT register

MOV R2,#0x40000000 @Load value pin GPIO1\_30 push button

LDR R3,[R1] @Load current value of GPIO1\_FALLINGDETECT register

ORR R3,R3,R2 @Modify the pin GPIO\_30 push button

STR R3, [R1] @Store GPIO\_30 push button for GPIO1\_FALLINGDETECT

@ GPIO1\_IRQSTATUS  on pin 30 R2 0x40000000 sending interrupts to POINTPEND1

ADD R1,R0,#0x34 @R1 address of GPIO1\_IRQSTATUS\_SET\_0 register

STR R2,[R1] @Store pin 30 for GPIO1\_IRQSTATUS\_SET\_0

@TIMER SETUP using DTIMER3 32.768 KHz clock

@also enable its IRQ

MOV R2, #0x2 @Value to turn on Timer3

LDR R1,=0x44E00084 @Address for CM\_PER\_TIMER3\_CLKCTRL

STR R2,[R1] @Write the value to turn on the TIMER3 clock

LDR R1,=0x44E0050C @Address for Mux CLKSEL\_TIMER3\_CLK to 32.768 hz

STR R2,[R1] @write the value to the Mux

@Timer 3 register with count and interrupts generation

LDR R1,=0x48042000 @Base address for Timer3 register

MOV R2,#0x1 @Value to reset register DMTIMER3

STR R2,[R1,#0x10] @Write value to reset at CFG configuration register

@Timer3 IRQENABLE\_SET

MOV R2,#0x2 @Value to Enable Overflow Interrupt

STR R2,[R1,#0x2C] @Write value to the IRQ register

@Load and reload DTIMER3

LDR R2,=0xFFF88000 @Count value 10 Second for the Timer

STR R2,[R1,#0x40] @Timer3 TLDR register (to reload)

STR R2,[R1,#0x3C] @Timer3 TCRR register (to count)

@ Initialize INTC Interrupts Controller

LDR R1,=0x48200000 @R1 loading base address for interrupts controller

MOV R2,#0x2 @Value to reset the INTC\_CONFIG register

STR R2,[R1,#0x10] @write value to reset the INTC CONFIG

MOV R2,#0x04 @ Value to unmask GPIOINTA , pin 2 INTC INT #98

STR R2,[R1,#0xE8] @Write value to INTC\_MIR\_CLEAR3 register

MOV R2,#0x400 @value to unmask UART2INT pin 10 INTC INT #74

STR R2,[R1,#0xC8] @Write value to INTC\_MIR\_CLEAR2 register

MOV R2,#0x20 @ value to unmask Timer3 pin 5 INTC INT #69

STR R2,[R1,#0xC8] @Write value to INTC\_MIR\_CLEAR2 register

@===================================================================================

@===================================================================================

@===================================================================================

@UART2 Setting up Section

@Mapping the UART2 TxD,RxD,CTS, and RTS using MUX.

@0x44E10000 base address for Control Module

@mode 1 : 001. need to AND 0xFFFFFFF9 for 0s at bit 2 and 3 . OR 0x1 for 1 last bit

@mode 6: 110. need to OR 0x110 for 1s at bit 2 and 3. AND 0xFFFFFFF1 for 0 last bit

@RxD UART2 mapping

LDR R0,=0x44E10950 @adding offset spi0\_sclk in mode 1 for RxD UART2

LDR R2,[R0] @Loading current value spi0\_sclk

AND R2,R2,#0xFFFFFFF9 @AND for 0s at bit 2 and 3

ORR R2,R2,#0x1 @OR for 1 last bit

STR R2, [R1] @store back for register spi0\_sclk switching mode 1

@TxD UART2 mapping

LDR R1,=0x44E10954 @spi0\_d0 in mode 1 for TxD UART2

LDR R3,[R1] @Loading current value spi0\_d0

AND R3,R3,#0xFFFFFFF9 @AND for 0s at bit 2 and 3

ORR R3,R3,#0x1 @OR for 1 last bit

STR R3, [R1] @store back for register spi0\_d0 switching mode 1

@CTSN UART2 mapping

LDR R0,=0x44E108C0 @Lcd\_data8 in mode 6 for CTSN UART2

LDR R2,[R0] @Loading current value lcd\_data8

AND R2,R2,#0xFFFFFFFE @AND 0 for last bit

ORR R2,R2,#0x6 @OR 1s for bit 3 and 2

STR R2, [R0] @store back for register lcd\_data8 switching mode 6

@RTSN UART2 mapping

LDR R1,=0x44E108C4 @Lcd\_data9 in mode 6 for RTSN UART2

LDR R3,[R1] @Loading current value lcd\_data9

AND R3,R3,#0xFFFFFFFE @AND 0 for last bit

ORR R3,R3,#0x6 @OR 1s for bit 3 and 2

STR R3, [R1] @store back for register lcd\_data9 switching mode 6

@============================================================================

@Turn on UART2 clock CM\_PER\_UART2\_CLKCTRL

MOV R2,#0x02 @value to turn on the clock

LDR R1,=0x44E00070 @base address for UART2 clock with offset 0x70

STR R2,[R1] @Store value to turn on the clock

@============================================================================

@ Setting up Baud rate, 8 data bits, 1 stop bit, no parity, disable FIFO and enable interrupt IER\_UART

@ UART2 base address register 0x4802\_4000

@0x48024000 base address UART2

@Switch to Configuration mode A in UART\_LCR offset 0x0C “bit 7 is 0x1”

@and 8 data  bits , one stop bit, no parity. bit 3 is 0, bit 2 is 0, and 1:0 will both 1

LDR R0,=0x4802400C @offset of LCR register

MOV R2,#0x83 @load value of LCR into R2

STR R2,[R0] @store back in LCR register

@Set 38.4 kbps Baud rate and a 16x divisor, DLL offset 0x00 write 0x00, DLH offset 0x04 write 0x4E

@Set DLL

LDR R1,=0x48024000 @Address for DLL

MOV R3,#0x4E @write value 0x00 to DLL

STR R3,[R1] @write 0x00 to DLL offset 0x0

@Set DLH

LDR R0,=0x48024004 @address for DLH offset 0x04

MOV R2,#0x00 @write value 0x4E to DLH

STR R2,[R0] @Store value 0x4E for DLH

@Mode Definition 1 Register MDR offset 0x20 need to write 0x0 for UART 16x mode

LDR R1,=0x48024020 @R1 address MDR

MOV R3,#0x0 @value for 16x mode

STR R3,[R1] @store value in MDR register

@Change back Operational mode  in UART\_LCR  offset 0x0C  “bit 7 is 0x0”

LDR R0,=0x4802400C @0xC offset of LCR register

LDR R2,[R0] @load value of LCR into R2

AND R2,R2,#0x7F @write 0 to bit 7

STR R2,[R0] @store back in LCR register

@Clearing FIFO in FCR register offset at 0x8 bit 1 RX\_FIFO, bit 2 TX\_FIFO, bit 0 FIFO\_EN.

LDR R1,=0x48024008 @R1 address if FCR register

MOV R3,#0x06 @value bit 0 ==0, bit 1==1 and bit 2==1  so 110==0x6

STR R3,[R1] @store value for FCR register

@===================================================================================

@==================================================================================

@ Make sure Processor CPSR IRQ enable

MRS R3,CPSR @ Copy CPSR to R3

BIC R3,#0x80 @Clear bit 7

MSR CPSR\_c, R3 @Write back to CPSR

@wait for interrupts

**LOOP:** NOP

B LOOP

@==================================================================================

@==================================================================================

**INT\_DIRECTOR:**

@Save the register on the stack

STMFD SP!, {R0-R3,LR} @Push register on the stack

@First check UART IRQ then check IRQ from button

**CHECK\_INTC\_UART:**

LDR R0,=0x482000D8 @INTC\_PENDING\_IRQ2 for UART2 INTC

LDR R2,[R0] @load value of PENDING\_IRQ2

TST R2,#0x400 @UART2INT pin 10 INTC INT #74

BNE CHECK\_UART @Go to check IRQ in UART2

**CHECK\_INTC\_BUTTON:**

LDR R0,=0x482000F8 @Address of INTC\_PENDING\_IRQ3 check button

LDR R2,[R0] @Load value to check

TST R2,#0x04 @Button pin 2 INTC INT #98

BNE CHECK\_BUTTON @Go to check button in GPIO1

**CHECK\_INTC\_TIMER:**

LDR R1,=0x482000D8 @reading value from INTC\_PENDING\_2

LDR R2,[R1] @load value into R2 to test

TST R2,#0x00000020 @pin 5 associate with INTC\_MIR\_ClEAR2 which was unmask

BNE CHECK\_TIMER @Go to check timer in GPIO1 module

@Checking UART IRQ using IIR\_UART offset 0x8

**CHECK\_UART:**

LDR R0,=0x48024008 @IIR\_UART2 base address

LDR R2,[R0] @Load value of IIR\_UART2

TST R2,#0x1 @check bit 0 IT\_PENDING

BEQ TALKER\_SVC @if 1 no IRQ, else 0 there is an IRQ

B PASS\_ON @if 1 then go to PASS\_ON

**CHECK\_BUTTON:**

LDR R0,=0x4804C02C @Check GPIO IRQ STATUS

LDR R2,[R0] @Load value from GPIO

TST R2,#0x40000000 @Testing if GPIO1\_30 IRQ register is 1 or 0

BNE BUTTON\_SVC @if z flag is clear Go to button service

B PASS\_ON @z flag is set go to PASS\_ON

@Checking module Timer3

**CHECK\_TIMER:**

LDR R1,=0x48042028 @ Address for IRQSTATUS\_TIMER3

LDR R2,[R1] @Load Value to test from the register

TST R2, #0x02 @ Test if bit 1 is 0 or 1

BNE TIMER\_SVC @if z flag is clear Go to Timer service

B PASS\_ON @if z Flag is set go to Pass on

@==================================================================================

@===================================================================================

@Service section remember to turn off interrupt request for register

**BUTTON\_SVC:**

@turn off IRQ request for GPIO1

LDR R1,=0x4804C02C @GPIO1\_IRQSTATUS\_0 address

MOV R2,#0x40000000 @turn off GPIO1\_IRQSTATUS at pin 30 by write a 1

STR R2,[R1] @Writing value to turn IRQ off GPIO1\_IRQ\_RAW\_0

@check if the timmer is running

LDR R0,=0x48042038 @Address for Timer3 TCLR

LDR R2,[R0] @0x3 start and reload the timer

TST R2,#0x03 @check if timer is running and auto reload

BEQ TIMER\_ON @if z flag is clear go to turn timer on

B TIMER\_OFF @if z flag is set go to timer off

**TIMER\_ON:**

@first time runing

@enable Interrupt Enable Register IER\_UART at offset 0x04

@bit 3 is 1 for MODEM Status Change MSC

@bit 1 is 1 for Transmit Holding Register THR the rest is all 0s for reset value

LDR R1,=0x48024004 @R1 address of IER\_UART

MOV R2,#0x000A @1010 == 0xA value

STR R2,[R1] @Store value into R1 IER\_UART

@@@@@@@@@@@@@@@@@@

@Turn on Timer

MOV R2,#0x03 @Load value to auto reload timer and start

LDR R0,=0x48042038 @Address for Timer3 TCLR

STR R2,[R0] @Store auto reload in Timer controller

B PASS\_ON @go to PASS\_ON

**TIMER\_OFF:**

@turn off Timer

LDR R0,=0x48042038 @Address for Timer TCLR

MOV R2,#0x00 @value to turn off the timer

STR R2,[R0] @Write to timer register TCLR

B PASS\_ON @go to PASS\_ON

@=========================================================================================

**TIMER\_SVC:**

@Turn off IRQ request for Timer3

LDR R0,=0x48042028 @Load IRQSTATUS\_TIMER3 register

MOV R2,#0x02 @Value to turn off the timer

STR R2,[R0] @Write to timer register TCLR

@enable Interrupt Enable Register IER\_UART at offset 0x04

@bit 3 is 1 for MODEM Status Change MSC

@bit 1 is 1 for Transmit Holding Register THR the rest is all 0s for reset value

LDR R1,=0x48024004 @R1 address of IER\_UART

MOV R2,#0x000A @1010 == 0xA value

STR R2,[R1] @Store value into R1 IER\_UART

B PASS\_ON @Go to PASS\_ON

**TALKER\_SVC:**

@There are 2 main registers take part in making decision whether to send character or pass on

@Modem status register MSR bit 4 (Clear to send CTS#)

@Line status register LSR bit 5 (Transmit holding register THR)

LDR R0,=0x48024000 @Address UART\_MSR register

ADD R1,R0,#0x18

LDR R2,[R1] @value to check bit 4 clear to send CTS#

TST R2,#0x10 @check if CTS# is currently asserted

BEQ NO\_CTS @go to NO\_CTS

LDR R0,=0x48024014 @Address UART\_LSR register

LDR R2,[R0] @value to check bit 5 Transmit holding register THR

TST R2,#0x20 @check if LSR bit 5 is currently asserted

BNE SEND\_CHAR @Go to SEND\_CHAR if LSR bit 5 is 1

B PASS\_ON @otherwise if LSR bit 5 is 0 go to PASS\_ON

@@@@@@@@@@@@@@@

**NO\_CTS:**

LDR R0,=0x48024014 @Address UART\_LSR to check bit 5 for THR

LDR R2,[R0] @Load value to check bit 5  UART\_LSR

TST R2, #0x20 @Check if LSR bit 5 is currently asserted

BEQ PASS\_ON @if LSR bit 5 is 0 go to PASS\_ON

@otherwise if LSR bit 5 is 1 disable THR interrupt and modem status register interrupt to avoid Spinning

LDR R0,=0x48024004 @Address IER\_UART to clear bit 1 and 3

LDR R2,[R0] @Load value to R-M-W

AND R2,R2,#0x5 @Clear bit 1 and 3

STR R2,[R0] @Write back to change bit 1 and 3

B PASS\_ON @Go to PASS\_ON after clear

@@@@@@@@@@@@@@@

**SEND\_CHAR:**

@Turn on the THR interrupt and modem status register interrupt

LDR R0,=0x48024004 @address for IER\_UART

MOV R2,#0x0A @Set bit 3 = MODEM STATUS, bit 1 = THR interrupt

STR R2, [R0] @Write to IER\_UART

@Send character

LDR R0,=CHAR\_PTR @R0 = address of pointer store

LDR R1,=CHAR\_COUNT @R1 = address of count store location

LDR R2,[R0] @R2=Address of desired character in text string

LDR R3,[R1] @R3=current character count value

LDRB R4,[R2],#1 @Read char to send from string

STR R2,[R0] @Incremented address store BACK in CHAR\_PTR

LDR R5,=0x48024000 @Load Transmit holding register to send char

STRB R4,[R5] @Send char to Transmit buffer

SUBS R3,R3,#1 @Decrement Character count value by 1

STR R3,[R1]

BPL PASS\_ON @Greater than or equal zero, more Characters go back

LDR R2,=MESSAGE @else if done then reload. Get address of starting string

STR R2,[R0] @Write value reload back to address of pointer

MOV R2,#54 @Reload original number of char in String again

STR R2,[R1] @Write to address of counter for the next message

LDR R0,=0x48024010 @MCR register to turn off interrupt

LDRB R1,[R0] @Read current value of register

BIC R1,R1,#0x08 @Clear bit 3 to disable UART IRQ

STRB R1,[R0] @Write byte to register

@Turn on Timer IRQ

MOV R2,#0x03 @Load value to auto reload timer and start

LDR R0,=0x48042038 @Address for Timer3 TCLR

STR R2,[R0] @Store auto reload in Timer controller

**PASS\_ON:**

LDR R0,=0x48200048 @Address if INTC\_CONTROL register

MOV R1,#0x1 @Value to clear bit 0

STR R1,[R0] @Write to INTC\_CONTROL register

LDMFD SP!, {R0-R3,LR} @restore register

SUBS PC,LR,#4 @return from the IRQ

**.data**

**.align** 2

**STACK1:** .rept 1024

**.word** 0x0000

.endr

**STACK2:** .rept 1024

**.word** 0x0000

.endr

**MESSAGE:** **.byte** 0x0D

**.ascii** "Your blood pressure is 120 over 70. Your pulse is 54."

**.byte** 0x0D

**.align** 2

**CHAR\_PTR:** **.word** MESSAGE @Pointer to next character

**CHAR\_COUNT:** **.word** 54 @counter for character to send

.end