3/22/2018

Project 2

ECE 372

**Phong Nguyen**

Table of Contents

[PART 1 SETUP I2C (2/25-3/10/2018) 1](#_Toc509526361)

[Initialization for I2C 1](#_Toc509526362)

[Part 1 Algorithm 1](#_Toc509526363)

[Part 1 Low Level algorithm 1](#_Toc509526364)

[Part 1 Assembly code 1](#_Toc509526365)

[PART 2 SETUP IRQ (3/10-3/20/2018) 1](#_Toc509526366)

[Interrupt unmasking 1](#_Toc509526367)

[Sending bytes to LCD 1](#_Toc509526368)

[Part 2 High Algorithm 1](#_Toc509526369)

[Part 2 Low Level algorithm 1](#_Toc509526370)

[Part 2 Assembly code 1](#_Toc509526371)

# PART 1 SETUP I2C (2/25-3/10/2018)

## Initialization for I2C

To map the I2C, 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.



The I2C that is needed for this project is I2C1\_SDA\_MUX3, and I2C1\_SCL\_MUX3. 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 spi0\_d1, spi0\_cs0 change to mode 2.

After mapping and started the I2C clock at CM\_PER\_I2C1\_CLKCTRL, next step is setting the module clock for I2C to communicate with LCD screen. First of all, is setting Prescaler value I2C\_PSC offset 0xB0. Initially, value default for Prescaler is 48MHz, the value that we write to the register will be divider for the Prescaler. In this case, we want a 12 MHz for our I2C so 48/4=12 MHz. We will write 0x3 to register.

Next setting data rate, we want 100kbps by writing value for SCLH and SCLL. Here we need some calculation

First finding period T:

After having period, we can find period for tLow and tHigh

Next we find internal rate clock

The formula that provided in the registers are

So we write 0x35 for 0x4802A035(SCLL) and 0x37 for 0x4802A037(SCLH)

Setting Own Address is the next step, I2C\_OA offset 0xA8. For this register, we simply write 0x00.

Setting Slave Address I2C\_SA offset 0x78. For this register, we write 0x3C.

Setting configuration for I2C\_CON offset 0xA4, in this register there are many feature bits. But we only need bit 10 MST and bit 9 TRX, bit 1 STT, and bit 0 STP. For now, we write the value 0x8600 to the register. we don’t want to start the transmission yet.

Finally, We set the Count register I2C\_CNT this register depend on how many bytes we want to write out to the I2C including instruction and ASCII. In this initialization, I just write to initialize the screen so write 0xC to the register.

The important part of the transmission is the XRDY bit in IRQ\_RAW\_STATUS register. Since the LCD does not talk back to the processor, so we just need to pay attention to this bit for now.

When setting the I2C\_CON register, notice the XRDY is set to 1 indicate the acknowledge and ready to write from I2C. One other thing to look for is AERR Access error bit and BB bit bus busy, AERR bit is also let us know if there is something wrong with the I2C transmission. And BB is indicating if the bus is busy.

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.

## Part 1 Algorithm

High level language:

1. Setting up stack
2. Select MUX for I2C1
3. Turn on I2C1 Clock
4. Setting up I2C1 (prescaler, SCLL, SCLH, Own address, Slave address )

SEND\_INST: (Sending instruction to initialize the LCD)

1. Polling on Bus Busy check if the bus is busy or not
2. START the I2C1, Set the Counter value 10
3. Send instruction through the I2C\_DATA
4. Decrement the counter
5. Delay
6. Clear XRDY for the next write
7. Delay

SECOND PART:

1. Set the I2C\_CNT to 10 character write 0x12

MESSAGE: (Sending message to the LCD to display)

1. Polling on Bus Busy check if the bus is busy or not
2. START the I2C1, Set the Counter value 10
3. Send message through the I2C\_DATA
4. Decrement the counter
5. Delay
6. Clear XRDY for the next write
7. Delay

## 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. I2C Setting up MUX
   1. I2C1\_SDA
      1. 0x44E10958 conf\_spi0\_d1 mode 2: 0x6A
   2. I2C1\_SCL
      1. 0x44E1095C conf\_spi0\_cs0 mode 2: 0x6A
3. Turn on I2C clock CM\_PER\_I2C1\_CLKCTRL
   1. Address 0x44E00048 write 0x02
4. Disable Auto IDLE I2C\_SYSC
   1. Write 0x2 at 0x4802A010
5. Set Prescaler to get 12 MHZ I2C\_PSC
   1. Write 0x3 to 0x4802A0B0
6. Set low time register SCLL
   1. Write 0x35 to 0x4802A0B4
7. Set high time register SCLH
   1. Write 0x37 to 0x4802A0B8
8. Setup Own address I2C\_OA
   1. Write 0x00 to 0x4802A078
9. Setup Slave address I2C\_SA
   1. Write 0x3C to 0x4802A0AC
10. Set the configuration register I2C\_CON
    1. Write 0x8600 to 0x4802A0A4
11. Polling bus busy P\_BB
12. Start the transmission START

SEND\_INST:

1. DELAY
2. Polling XRDY P\_XRDY
3. DELAY

16)load pointer to the array instruction

17) sending data to I2C\_DATA 0x4802A09

18) decrement the counter

19) DELAY

20) clearing XRDY CLEAR\_XRDY

21)DELAY

22)Go to SEND\_INST if counter is not 0

SECOND PART:

23) Write 0x12 to count register 0x4802A098

24)Clear pointer and counter

MESSAGE:

1. DELAY
2. Polling XRDY P\_XRDY
3. DELAY

16)load pointer to the array instruction

17) sending data to I2C\_DATA 0x4802A09

18) decrement the counter

19) DELAY

20) clearing XRDY CLEAR\_XRDY

21)DELAY

22)Go to MESSAGE if counter is not 0

FUNCTIONS PART

P\_BB:

23) test 0x10000 at I2C\_IRQSTATUS\_RAW 0x4802A024

P\_XRDY:

24) TST 0x10 at I2C\_IRQSTATUS\_RAW 0x4802A024\

START:

25) Write 0x8603 to I2C\_CON 0x4802A0A4

START:

26) Write 0x8603 to I2C\_CON 0x4802A0A4

CLEAR\_XRDY:

27) Write 0x10 to I2C\_IRQSTATUS 0x4802A028

DELAY:

28) Decrement 0x00040000 until it reaches 0

## Part 1 Assembly code

@Phong Nguyen

@ECE 372

@project 2

@I2C Initialization and LCD

@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

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

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

@Setting up the I2C1

@----------------------------------------------------------------------------------------------------------------------------------

@Control module base address L4\_WKUP 0x44E10000

@Setting up MUX for I2C1\_SDA

@mode 2: 0010 . AND bit 3,2,0 to 0 and OR bit 1

LDR R0,=0x44E10958 @Address of conf\_spi0\_d1

MOV R2,#0x6A @mode 2 disable Pullup/Pulldown, Rx enable, Slow slew

STR R2, [R0] @store back for register spi0\_sclk switching

@Setting up MUX for I2C1\_SCL

LDR R0,=0x44E1095C @Address of conf\_spi0\_cs0

MOV R2,#0x6A @mode 2 disable Pullup/Pulldown, Rx enable, Slow slew

STR R2, [R0] @store back for register spi0\_sclk switching

@Setting up MUX for I2C1\_SCL

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Clock for I2C1 base address CM\_PER 0x44E00000

LDR R0,=0x44E00048 @Address for CM\_PER\_I2C1\_CLKCTRL

MOV R2,#0x2 @Value to start the clock

STR R2,[R0] @Store value to turn the clock on

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Seting Debug Subsystem I2C\_SYSTEST

LDR R0,=0x44E000BC @Address for I2C\_SYSTEST

LDR R2,[R0] @Load Value from register

AND R2,R2,#0xFFFFBFFF @Value to clear bit 14 for free debug

STR R2,[R0] @Store back to register

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Disable I2C module

LDR R0,=0x44E000A4 @Address for I2C\_CON

LDR R2,[R0] @Load value in address

LDR R4,=0xFFFF7FFF @Value to clear bit 15

BIC R2,R4 @Clear bit 15

STR R2,[R0] @Store back in register

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Reset I2C module

LDR R0,=0x44E00010 @Address for I2C\_SYSC

MOV R2,#0x2 @Value to reset

STR R2,[R0] @Store value in register

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Disable AUTOIDLE in I2C\_SYSC and do a software reset

LDR R0, =0x4802A010 @ Address for I2C\_SYSC

MOV R2,#0x02 @Value to turn off IDLE

STR R2,[R0] @Store value back to register

@------------------------------

@Prescaler value setting up module clock

@initial value of Prescaler 48 MHZ divine by 4 to get 12 MHZ

LDR R0,=0x4802A0B0 @Address of I2C\_PSC

MOV R2,#0x3 @Setting value divine by 4

STR R2,[R0] @Store value divine by 4

@------------------------------

@Setup Low time register SCLL. Base address I2C1: 0x4802A000

LDR R0,=0x4802A0B4 @address for I2C\_SCLL

MOV R2,#0x35 @Value 53 in Dec for SCLL

STR R2,[R0] @Store value 53 into register

@Setup High time register SCLH. Base address I2C1: 0x4802A000

LDR R0,=0x4802A0B8 @address for I2C\_SCLH

MOV R2,#0x37 @value 55 in Dec for SCLH

STR R2,[R0] @Store value 55 into register

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Setup Own address I2C\_0A.

@Make sure clearing bit 9-7

LDR R0,=0x4802A078 @address for I2C\_OA

MOV R2,#0x00 @Value to clearing bit

STR R2,[R0] @Store value to the address

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Setting Count register

LDR R0,=0x4802A098 @Address of I2C\_CNT

MOV R2,#0xA @Value to write 10 characters

STR R2,[R0] @Store count value into register

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Setting slave address

LDR R0,=0x4802A0AC @Address of I2C\_SA

LDR R4,[R0] @load value to modify

MOV R2,#0xC00 @Value to have

AND R4,R4,R2 @Clear bit 9,8,7

STR R4,[R0] @Store value into register to clear bit

MOV R2,#0x3C @write address 0x78 shift left 1 logic 0x0F of slave to the register after clearing

STR R2,[R0] @Store in the register

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Transmission procedure

@Clearing XRDY state

LDR R0,=0x4802A028 @Address of I2C\_IRQSTATUS\_RAW

LDR R2,=0x7FFF @if bit 4 XRDY is set ready to send the data

STR R2,[R0] @Write there to disable the IRQSTATUS

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@configuration register

@bit 15 I2C\_EN module enable, bit 10 MST Master mode, bit 9 TRX Transmitter mode

LDR R0,=0x4802A0A4 @address for I2C\_CON

LDR R2,=0x8600 @Value for bit 9,10,15

STR R2,[R0] @Store value into register

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

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

BL P\_BB @Check BUS if it is busy

BL START @Start the I2C1

MOV R5,#0xA @ Counter value 10

MOV R3, #0x00 @ CLEAR POINTER

SEND\_INST:

BL DELAY @Delay

BL P\_XRDY @POLL until XRDY is set

BL DELAY @Delay after POLL

@STORE instruction

LDR R0,=INST\_DATA

LDRB R1,[R0,R3] @Load current instruction

ADD R3,R3,#0x01 @increment instruction offset value

LDR R0,=0x4802A09C @Address I2C\_DATA

STRB R1,[R0] @Write to register

NOP

BL DELAY @Delay

BL CLEAR\_XRDY @Clear XRDY for the next write

BL DELAY @Delay

SUBS R5,R5,#1 @Decrement the counter

BNE SEND\_INST @ Send more instructions if counter is not 0

BL DELAY

NOP

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

@INTIALIZE The I2C for second Message

@configuration register

@bit 15 I2C\_EN module enable, bit 10 MST Master mode, bit 9 TRX Transmitter mode

LDR R0,=0x4802A0A4 @address for I2C\_CON

LDR R2,=0x8600 @Value for bit 9,10,15

STR R2,[R0] @Store value into register

@Setting Count register

LDR R0,=0x4802A098 @Address of I2C\_CNT

MOV R2,#0x12 @Value to write 10 characters

STR R2,[R0] @Store count value into register

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

@Send MESSAGE

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

BL P\_BB @Check BUS if it is busy

BL START @Start the I2C1

MOV R6,#0x12 @ Counter value 10

MOV R3, #0x00 @ CLEAR POINTER

BL RELOAD

MESSAGE:

BL DELAY @Delay

BL P\_XRDY @POLL until XRDY is set

BL DELAY @Delay after POLL

@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,=0x4802A09C @Address I2C\_DATA

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

NOP

BL DELAY @Delay

BL CLEAR\_XRDY @Clear XRDY for the next write

BL DELAY @Delay

SUBS R6,R6,#1 @Decrement the counter

BNE MESSAGE @ Send more instructions if counter is not 0

BL DELAY

NOP

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

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

@Functions sections

@--------------------------------------------------------------

@When the BUS is free initialize the transmission

P\_BB: @Polling on Busy bus

LDR R0,=0x4802A024 @Address for I2C\_IRQSATUS\_RAW

LDR R2,[R0] @Load value in the address

TST R2,#0x1000 @Test if the bit is clear or set

BNE P\_BB @Keep Polling if the Bus is Busy

MOV PC,R14 @Return

@--------------------------------------------------------------

P\_XRDY: @Polling on XRDY

LDR R0,=0x4802A024 @Address for I2C\_IRQSTATUS\_RAW

LDR R2,[R0] @Load value in the address

TST R2,#0x10 @Test value with bit 4 XRDY

BEQ P\_XRDY @Keep Polling if the XRDY is not set

MOV PC,R14 @Return

@---------------------------------------------------------------

START:

LDR R0,=0x4802A0A4 @Address for I2C\_CON

LDR R2,=0x8603 @Value to Start

STR R2,[R0] @Store value to register

MOV PC,R14 @Return

@--------------------------------------------------------------

STOP:

LDR R0,=0x4802A0A4 @Address for I2C\_CON

LDR R2,=0x8603 @Value to Start

STR R2,[R0] @Store value to register

MOV PC,R14 @Return

@----------------------------------------------------------------

CLEAR\_XRDY:

LDR R0,=0x4802A028 @Address for I2C\_IRQSTATUS

MOV R2,#0x10 @Value to reset the XRDY

STR R2,[R0] @Write value to clear XRDY

MOV PC,R14 @Return

@-------------------------------------------------------------------

DELAY:

LDR R2,=0x00040000 @DELAY value 0.2 s

DELAY\_LOOP:

SUBS R2,R2,#1 @Decrement the value

BNE DELAY\_LOOP @Return

MOV PC,R14

@------------------------------------------------------------------

CHECK:

LDR R0,=0x4802A024 @Load address I2C\_IRQSTATUS\_RAW

LDR R2,[R0] @load value in address

TST R2,#0x4 @testing bit 2

BEQ CHECK

MOV PC,R14

RELOAD:

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

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

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

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

MOV R2,#13 @Reload original number of char in String again

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

MOV PC,R14

.data

.align 4

INST\_DATA:

.byte 0x00,0x38,0x39,0x14,0x78,0x5E,0x6D,0x0C,0x01,0x06

.align 4

.align 4

MESSAGE\_DATA:

.byte 0x50

.ascii "Phong Nguyen"

.align 4

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

CHAR\_COUNT: .word 13 @counter for character to send

STACK1: .rept 1024

.word 0x0000

.endr

STACK2: .rept 1024

.word 0x0000

.endr

.end

# PART 2 SETUP IRQ (3/10-3/20/2018)

## Interrupt unmasking

Unmasking 2 interrupts in INTC register. I2C1INT pin 7 INTC INT #71 and the button GPIO1A is at int number 98. For GPIO1A, the pin calculated was pin 2 at INTC\_MIR\_CLEAR3 register. For I2C1INT, the pin calculated was pin 7 at INTC\_MIR\_CLEAR2. Simply write 0x04 to INTC\_MIR\_CLEAR3 and write 0x400 to INTC\_MIR\_CLEAR2.

## Sending bytes to LCD

There are many IRQ bits in IRQENABLE\_SET, we just need to unmask XRDY bit. By writing 0x80 to enable XRDY\_IE bit 4. When getting the Interrupt from XRDY the process will send the next byte in the array to I2C\_DATA.

## Part 2 High Algorithm

1. High level language:
2. Setting up stack
3. Wake up GPIO1 module
4. Clear pin for GPIO1 module
5. Setting button GPIO1\_30 with falling edge detect
6. Setting up Interrupt controller INTC for GPIO1\_30, I2C1
7. Select MUX for I2C1
8. Turn on I2C1 Clock
9. Setting up I2C1 (prescaler, SCLL, SCLH, Own address, Slave address )

INT\_DIRECTOR

1. Save register on stack (2 STAGEs , STAGE 1 is sending instruction, STAGE 2 is sending message)
2. Check INTC interrupt from the I2C, if not then go to check INTC interrupt from the button
3. If INTC from the I2C, go to check XRDY bit of IRQ\_RAW register I2C1, if is 1 then Check what stage is currently on, 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 check what stage the button doing next BUTTON\_STAGE1 or BUTTON\_STAGE2. If not, go to PASS\_ON

BUTTON\_STAGE1

1. Unmask XRDY\_IE
2. START second transmission

BUTTON\_STAGE2

1. Setting Count register
2. START second transmission

I2C\_SVC\_STAGE1

1. Clearing XRDY
2. Sending Instruction
3. If done sending instruction clear XRDY waiting for button press the second times

I2C\_SVC\_STAGE2

1. Clearing XRDY
2. Sending Message

PASS\_ON

1. Go back to wait loop save registers

## Part 2 Low Level algorithm

1. Setting up STACK
   1. Point to top of stack for service mode 0x10000 CPS #0x12

Point to top of stack for IRQ mode 0x10000 CPS#0x13

1. I2C Setting up MUX
   1. I2C1\_SDA
      1. 0x44E10958 conf\_spi0\_d1 mode 2: 0x6A
   2. I2C1\_SCL
      1. 0x44E1095C conf\_spi0\_cs0 mode 2: 0x6A
2. Turn on I2C clock CM\_PER\_I2C1\_CLKCTRL
   1. Address 0x44E00048 write 0x02
3. INTC interrupt controller 0x48200000
   1. Unmasking GPIOINTA write 0x04 offset 0xE8 MIR\_CLEAR3
   2. Unmasking I2CINT write 0x80 offset 0xC8 MIR\_CLEAR2
4. Disable Auto IDLE I2C\_SYSC
   1. Write 0x2 at 0x4802A010
5. Set Prescaler to get 12 MHZ I2C\_PSC
   1. Write 0x3 to 0x4802A0B0
6. Set low time register SCLL
   1. Write 0x35 to 0x4802A0B4
7. Set high time register SCLH
   1. Write 0x37 to 0x4802A0B8
8. Setup Own address I2C\_OA
   1. Write 0x00 to 0x4802A078
9. Setup Slave address I2C\_SA
   1. Write 0x3C to 0x4802A0AC
10. Set the configuration register I2C\_CON
    1. Write 0x8600 to 0x4802A0A4
11. Turn on GPIO1 CLK
    1. Turn on the GPIO1 at 0x44300AC write 0x02
12. Setting up Falling edge detect on GPIO1\_30
    1. GPIO1\_FALLINGDETECT 0x4804C14C
    2. Pin 30 0x40000000
    3. Setting up GPIO1\_IRQSTATUS

INT\_DIRECTOR:

CHECK\_INTC\_I2C:

1. Test 0x80 INTC\_PENDING\_IRQ2 0x482000D8

CHECK\_INTC\_BUTTON:

1. Test 0x04 INTC\_PENDING\_IRQ3 0x482000F8

CHECK\_I2C:

1. Test 0x10 I2C\_IRQSTATUS\_RAW 0x4802A024
2. Load the stage to check what’s stage to work on

CHECK\_BUTTON:

1. Test 0x40000000 at GPIO IRQ STATUS 0x4804C02C

BUTTON\_SVC:

1. Write 0x40000000 to GPIO1\_IRQSTATUS\_0 0x4804C02C
2. Load STAGE status to see what’s stage the button will do the work

BUTTON\_STAGE1:

1. Write 0x10 to I2C\_IQENABLE\_SET 0x4802A02C
2. START write 0x8603 to I2C\_CON 0x4802A0A4

I2C\_SVC\_STAGE\_1:

1. Clearing IRQ
2. Load INSTRUCTION\_DATA, INSTRUCTION\_POINTER
3. Write data to I2C\_DATA 0x4802A09C

I2C\_SVC\_STAGE\_2:

1. Clearing IRQ
2. Load CHAR\_PTR, CHAR\_COUNT
3. Write data to I2C\_DATA 0x4802A09C

PASS\_ON:

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

## Part 2 Assembly code

@Phong Nguyen

@ECE 372

@project 2 part 2

@I2C Initialization and LCD

@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,#0x80 @Value to unmask I2C1INT pin 7 INTC INT #71

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

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

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

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

@Setting up the I2C1

@------------------------------

@Control module base address L4\_WKUP 0x44E10000

@Setting up MUX for I2C1\_SDA

@mode 2: 0010 . AND bit 3,2,0 to 0 and OR bit 1

LDR R0,=0x44E10958 @Address of conf\_spi0\_d1

MOV R2,#0x6A @mode 2 disable Pullup/Pulldown, Rx enable, Slow slew

STR R2, [R0] @store back for register spi0\_sclk switching

@Setting up MUX for I2C1\_SCL

LDR R0,=0x44E1095C @Address of conf\_spi0\_cs0

MOV R2,#0x6A @mode 2 disable Pullup/Pulldown, Rx enable, Slow slew

STR R2, [R0] @store back for register spi0\_sclk switching

@Setting up MUX for I2C1\_SCL

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Clock for I2C1 base address CM\_PER 0x44E00000

LDR R0,=0x44E00048 @Address for CM\_PER\_I2C1\_CLKCTRL

MOV R2,#0x2 @Value to start the clock

STR R2,[R0] @Store value to turn the clock on

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Seting Debug Subsystem I2C\_SYSTEST

LDR R0,=0x44E000BC @Address for I2C\_SYSTEST

LDR R2,[R0] @Load Value from register

AND R2,R2,#0xFFFFBFFF @Value to clear bit 14 for free debug

STR R2,[R0] @Store back to register

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Disable I2C module

LDR R0,=0x44E000A4 @Address for I2C\_CON

LDR R2,[R0] @Load value in address

LDR R4,=0xFFFF7FFF @Value to clear bit 15

BIC R2,R4 @Clear bit 15

STR R2,[R0] @Store back in register

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Reset I2C module

LDR R0,=0x44E00010 @Address for I2C\_SYSC

MOV R2,#0x2 @Value to reset

STR R2,[R0] @Store value in register

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Disable AUTOIDLE in I2C\_SYSC and do a software reset

LDR R0, =0x4802A010 @ Address for I2C\_SYSC

MOV R2,#0x02

STR R2,[R0] @Store value back to register

@-------------------------------------------------------------------------------------------------------------------------------------------------------

@Prescaler value setting up module clock

@initial value of Prescaler 48 MHZ divine by 4 to get 12 MHZ

LDR R0,=0x4802A0B0 @Address of I2C\_PSC

MOV R2,#0x3 @Setting value divine by 4

STR R2,[R0] @Store value divine by 4

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Setup Low time register SCLL. Base address I2C1: 0x4802A000

LDR R0,=0x4802A0B4 @address for I2C\_SCLL

MOV R2,#0x35 @Value 53 in Dec for SCLL

STR R2,[R0] @Store value 53 into register

@Setup High time register SCLH. Base address I2C1: 0x4802A000

LDR R0,=0x4802A0B8 @address for I2C\_SCLH

MOV R2,#0x37 @value 55 in Dec for SCLH

STR R2,[R0] @Store value 55 into register

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Setup Own address I2C\_0A.

@Make sure clearing bit 9-7

LDR R0,=0x4802A078 @address for I2C\_OA

MOV R2,#0x00 @Value to clearing bit

STR R2,[R0] @Store value to the address

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Setting Count register

LDR R0,=0x4802A098 @Address of I2C\_CNT

MOV R2,#0xD @Value to write 13 characters

STR R2,[R0] @Store count value into register

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Setting slave address

LDR R0,=0x4802A0AC @Address of I2C\_SA

LDR R4,[R0] @load value to modify

MOV R2,#0xC00 @Value to have

AND R4,R4,R2 @Clear bit 9,8,7

STR R4,[R0] @Store value into register to clear bit

MOV R2,#0x3C @write address 0x78 shift left 1 logic 0x0F of slave to the register after clearing

STR R2,[R0] @Store in the register

@-------------------------------------------------------------------------------------------------------------------------------------------------------@Transmission procedure

@Clearing XRDY state

LDR R0,=0x4802A028 @Address of I2C\_IRQSTATUS\_RAW

LDR R2,=0x7FFF @if bit 4 XRDY is set ready to send the data

STR R2,[R0] @Write there to disable the IRQSTATUS

@-------------------------------------------------------------------------------------------------------------------------------------------------------@configuration register

@bit 15 I2C\_EN module enable, bit 10 MST Master mode, bit 9 TRX Transmitter mode

LDR R0,=0x4802A0A4 @address for I2C\_CON

LDR R2,=0x8600 @Value for bit 9,10,15

STR R2,[R0] @Store value into 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\_I2C:

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

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

TST R2,#0x80 @UART2INT pin 10 INTC INT #71

BNE CHECK\_I2C @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 I2C\_IRQ\_RAW

CHECK\_I2C:

LDR R0,=0x4802A024 @Address of I2C\_IRQSTATUS\_RAW

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

@TST R2,#0x1000 @check bit 12 BB . IF Bus is free or busy

@BNE PASS\_ON @if 1 BUS is busy go to PASS\_ON

TST R2,#0x10 @check if bit 4 XRDY . IF XRDY is ready or not

BEQ PASS\_ON @Jump to PASS\_ON if I2C is not set

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

LDR R0,=STAGE @Load to see if the screen complete the instruction

LDRB R1,[R0] @Load value from address

TST R1,#0x1 @Value when the pointer is reach

BEQ I2C\_SVC\_STAGE\_1 @Jump to Servicing I2C if pointer

B I2C\_SVC\_STAGE\_2 @otherwise jump to Servicing I2C if pointer is not 13

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:

NOP

@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

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

LDR R0,=STAGE @Load to see if the screen complete the instruction

LDRB R1,[R0] @Load value from address

TST R1,#0x1 @Value when the pointer is reach

BEQ BUTTON\_STAGE1 @jump to BUTTON\_STAGE1

B BUTTON\_STAGE2 @jump to BUTTON\_STAGE2

BUTTON\_STAGE1:

NOP

@Interrupt enable

@XRDY bit 4 for check if the LCD ready to send next char

LDR R0,=0x4802A02C @address for I2C\_IQENABLE\_SET

MOV R2,#0x10 @value to unmask bit 4

STR R2,[R0] @Store value in register

@START the I2C1 to get acknowledge from I2C1

LDR R0,=0x4802A0A4 @Address for I2C\_CON

LDR R2,=0x8603 @Value to Start

STR R2,[R0] @Store value to register

B PASS\_ON @Go to PASS\_ON

BUTTON\_STAGE2:

NOP

@@@@@@@@@@@@@ Put a hold on I2C1 I

@configuration register

@bit 15 I2C\_EN module enable, bit 10 MST Master mode, bit 9 TRX Transmitter mode

LDR R0,=0x4802A0A4 @address for I2C\_CON

LDR R2,=0x8600 @Value for bit 9,10,15

STR R2,[R0] @Store value into register

@Setting Count register

LDR R0,=0x4802A098 @Address of I2C\_CNT

MOV R2,#0xD @Value to write 10 characters

STR R2,[R0] @Store count value into register

@START the I2C1 to get acknowledge from I2C1

LDR R0,=0x4802A0A4 @Address for I2C\_CON

LDR R2,=0x8603 @Value to Start

STR R2,[R0] @Store value to register

BL RELOAD

B PASS\_ON @Go to PASS\_ON

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

@STAGE 1 sending instruction

I2C\_SVC\_STAGE\_1:

NOP

@CLEARING XRDY interrupt

LDR R0,=0x4802A028 @Address for I2C\_IRQSTATUS

MOV R2,#0x10 @Value to reset the XRDY

STR R2,[R0] @Write value to clear XRDY

@STORE instruction

LDR R0,=INSTRUCTION\_DATA @Load address instruction

LDR R2,=INSTRUCTION\_POINTER @Load Pointer for instruction

LDRB R4,[R2] @Load value pointer

LDRB R1,[R0,R4] @Load current instruction

ADD R4,R4,#0x01 @increment instruction offset value

STRB R4,[R2] @Store back pointer for the next use

LDR R0,=0x4802A09C @Address I2C\_DATA

STRB R1,[R0] @Write to register

@BEFORE go back to PASS ON need to see if need to transfer to next stage

SUBS R4,R4,#0xD @Value counter in R4

BNE PASS\_ON @Go to PASS\_ON if counter is not reach 0xD

LDR R0,=STAGE @Do a change in STAGE

MOV R1,#0x1 @value for stage 2

STRB R1,[R0] @Store value stage 2

@CLEARING XRDY interrupt

LDR R0,=0x4802A028 @Address for I2C\_IRQSTATUS

MOV R2,#0x10 @Value to reset the XRDY

STR R2,[R0] @Write value to clear XRDY

B PASS\_ON

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

@STAGE 2 sending message

I2C\_SVC\_STAGE\_2:

NOP

@CLEARING XRDY interrupt

LDR R0,=0x4802A028 @Address for I2C\_IRQSTATUS

MOV R2,#0x10 @Value to reset the XRDY

STR R2,[R0] @Write value to clear XRDY

@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,=0x4802A09C @Address I2C\_DATA

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

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

STR R3,[R1] @Store value back into CHAR\_COUNT

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

BL RELOAD

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

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

RELOAD:

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

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

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,#13 @Reload original number of char in String again

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

MOV PC,R14

.data

.align 2

STACK1: .rept 1024

.word 0x0000

.endr

STACK2: .rept 1024

.word 0x0000

.endr

.align 4

INSTRUCTION\_DATA:

.byte 0x00,0x38,0x39,0x14,0x78,0x5E,0x6D,0x0C,0x01,0x06,0x80,0x07,0xF

INSTRUCTION\_POINTER:

.byte 0x0

STAGE: .byte 0x0

SET\_CNT: .word 0x0

.align 4

MESSAGE:

.byte 0x50

.ascii "Phong Nguyen"

.align 4

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

CHAR\_COUNT: .word 13 @counter for character to send

.end

I have done this project by myself

Phong Nguyen