11/30/2017

Project 2

ECE 371

**Phong Nguyen**

Table of Contents

[PART 1 LEDS SEQUENCE 2](#_Toc499941473)

[Part 1 Flowchart 2](#_Toc499941474)

[GPIO1 pins Table 2](#_Toc499941475)

[Part 1 Algorithm 3](#_Toc499941476)

[Low Level algorithm 3](#_Toc499941477)

[Part 1 Assembly code 4](#_Toc499941478)

[PART 2 SETTING UP INTERRUPT 6](#_Toc499941479)

[Part 2 Flow chart 6](#_Toc499941480)

[Interrupt controller unmask 6](#_Toc499941481)

[Part 2 Algorithm 7](#_Toc499941482)

[Assembly code 7](#_Toc499941483)

[PART 3 SETTING UP DTIMER 12](#_Toc499941484)

[Part 3 Flow chart 12](#_Toc499941485)

[Setting up register Timer3 12](#_Toc499941486)

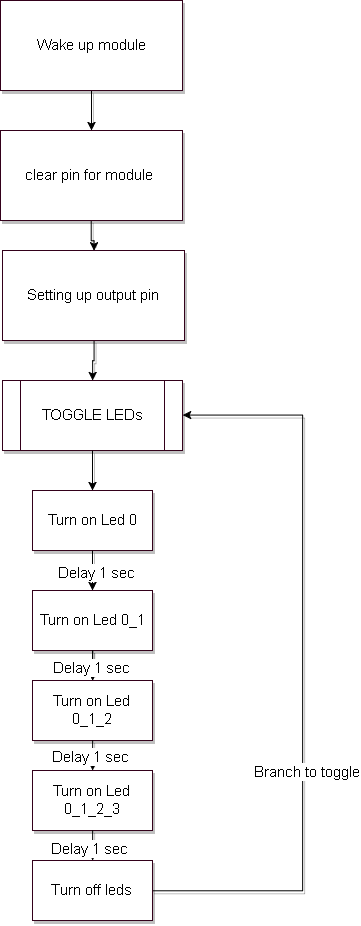
[Part 3 Algorithm 13](#_Toc499941487)

[Low Algorithm Part 3 14](#_Toc499941488)

[Part 3 Assembly code 16](#_Toc499941489)

# PART 1 LEDS SEQUENCE

## Part 1 Flowchart



## GPIO1 pins Table

Setting up table pin for GPIO1 base address 0x4804C000

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GPIO1  Bit# | 31 | 30 | 29 | 28 | 27 | 16 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Function |  | Button |  |  |  |  |  | LED3 | LED2 | LED1 | LED0 |  |  |  |  |  |
| OE in binary | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| OE in hex | F | | | | E | | | | 1 | | | | F | | | |
| CLEAR in hex | 0 | | | | 1 | | | | E | | | | 0 | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GPIO1  Bit# | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Function |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| OE in binary | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| OE in hex | F | | | | F | | | | F | | | | F | | | |
| CLEAR in hex | 0 | | | | 0 | | | | 0 | | | | 0 | | | |

## Part 1 Algorithm

Start up the module

Clear the pin on the module

Setting up leds output pins

Toggle leds

Start the loop:

Adding LED 0 for 1 second

Adding LED 1 so LED 0,1 for 1 second

Adding LED 2 so LED 0,1,2 for 1 second

Adding LED 3 so LED 0,1,2,3 for 1 second

And turn off all LEDs for 1 second

Repeat the loop.

## Low Level algorithm

Note: base address for GPIO1\_control register is 0x4804C000

Wake up module GPIO1 Store a 0x2 in address 0x44E000AC CM\_PER\_GPIO1\_CLKCTRL

Clear pin 21 to 24 by storing 0x01E00000 at the address 0x4804C190 for GPIO1\_CLEARDATAOUT

Setting up pin 21 to 24 to output enable for (0 is enable) by And the value 0xFE1FFFFF with the value taken from the 0x4804C134 of GPIO1\_OE

Toggle leds subroutine

Loop:

Go load 0x00200000 value to turn on LED 0 and send

Call Delay 1 second

Go Load 0x00400000 value to turn on LED 1 and send

Call Delay 1 second

Go load 0x00800000 value to turn on LED2 and send

Call Delay 1 second

Go load 0x01000000 value to turn on LED3 and send

Call Delay 1 second

Clear process:

Set GPIO1 bits 0x01E00000 to GPIO1\_CLEARDATAOUT at 0x4804C190

SET GPIO1 bits to outputs by RMW 0xFE1FFFFF to GPIO1\_EN at 0x4804C134

Send: Write LED value to GPIO1\_SETDATAOUT register at 0x4804C194

Call Delay 1 second

Go to loop

Delay 1 second:

Load value R2 with 0x00400000

Substract R2

Call Delay 1 second again until flag z is set

The branch to the current work

## Part 1 Assembly code

@MODULE SETUP using GPIO1 pin 21-24 for output LEDS

@ 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

@ GPIO1\_ClEARDATAOUt for pin 21-24

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

MOV R1, #0x01E00000 @value to turn off 21-24 LEDS on GPIO1

STR R1,[R0,#0x190] @Write to GPIO1\_CLEARDATAOUT register

@ GPIO1\_OE for pin 21-24

ADD R4,R0,#0x0134 @R4 is GPIO1\_OE register address

MOV R5,#0xFE1FFFFF @ pin 21-24 for output enable (0 is enable)

LDR R6,[R4] @Load current value from the GPIO1\_OE

AND R6,R6,R5 @Modify current value with pin 21-24

STR R6,[R4] @Write back to GPIO\_OE register address

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

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

@LED part for BUTTON

@turn on LED

TURN\_ON:

@TOGGLE LED and wait for interrupt from button

LDR R1,=0x4804C000 @Base address of GPIO1

ADD R3,R1,#0x190 @R3 is address to CLEARDATAOUT of GPIO1

ADD R4,R1,#0x194 @R4 is address to SETDATAOUT of GPIO1

LED\_0:

MOV R6,#0x00200000 @Value to Turn on LED0

STR R6,[R4] @Write to Turn on LED0

BL ONE\_SEC @Go to delay 1 sec

LED\_0\_1:

MOV R6,#0x00600000 @Value to Turn on LED\_0\_1

STR R6,[R4] @Write to Turn on LED\_0\_1

BL ONE\_SEC @Go to delay 1 sec

LED\_0\_1\_2:

MOV R6,#0x00E00000 @Value to Turn on LED\_0\_1\_2

STR R6,[R4] @Write to Turn on LED2

BL ONE\_SEC @Go to delay 1 sec

LED\_0\_1\_2\_3:

MOV R6,#0x01E00000 @Value to Turn on LED\_0\_1\_2\_3

STR R6,[R4] @Write to Turn on LED3

BL ONE\_SEC @Go to delay 1 sec

LEDOFF:

MOV R5,#0x01E00000 @Load value to reset the GPIO1

STR R5,[R3] @Write to CLEARDATAOUT for GPIO1

BL ONE\_SEC @Go to delay 1 sec

B LED\_0 @go back to turn on LED\_0

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

ONE\_SEC:

MOV R2,#0x00400000 @delay 1 sec

LOOP2:

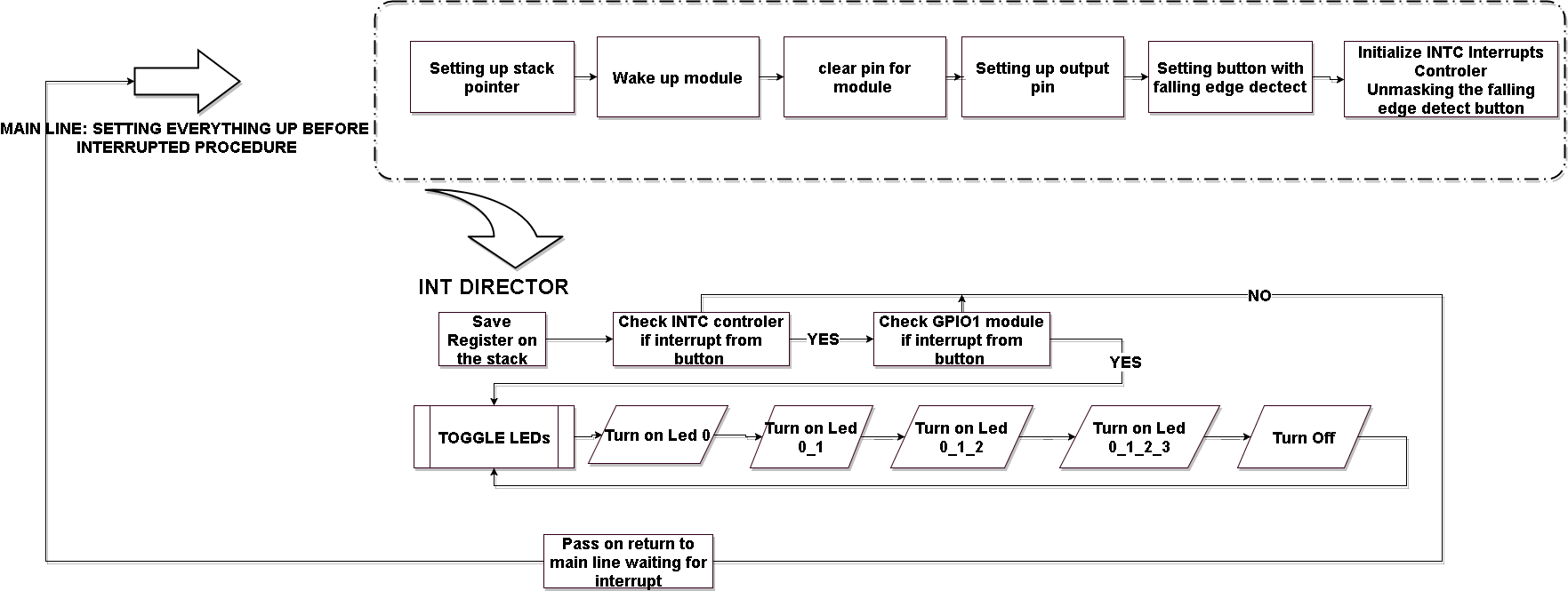
SUBS R2,#1 @Count down

BNE LOOP2 @jump back ONE\_SEC

MOV PC,R14 @after done 1 sec then go to current work

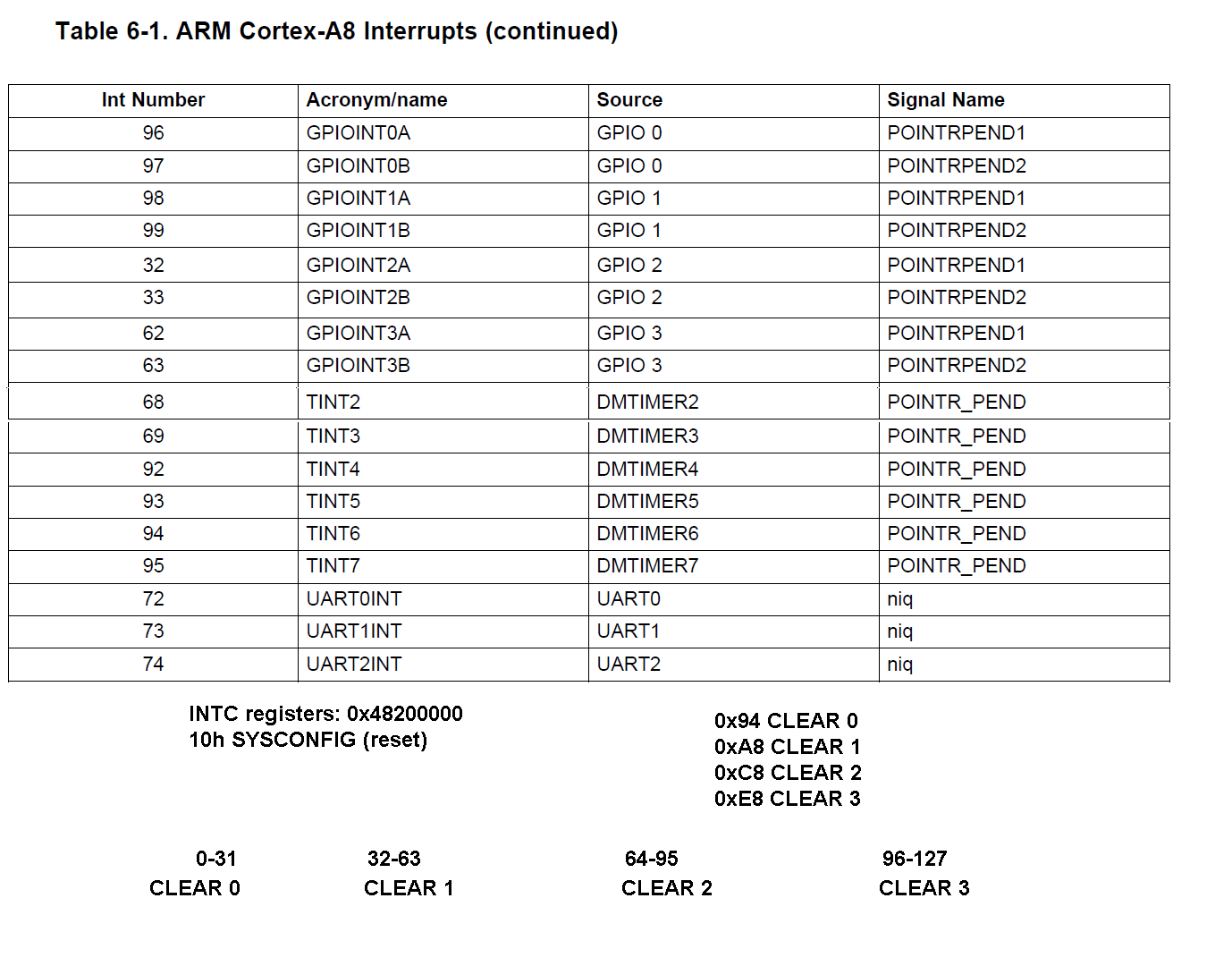
# PART 2 SETTING UP INTERRUPT

## Part 2 Flow chart



## Interrupt controller unmask

When unmasking the POINTRPEND for GPIO1. We have to look up the which pin in the interrupt control that was mask. The table below will show us how the mask of falling edge detect button work. We will use this table to unmask the DTIMER3 in the part 3. I will have separate table template for DTIMER3



We pick INTC\_MIR\_CLEAR3 for GPIO1 because the POINTPEND1 at Int number 98

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Int number | 99 | 98 | 97 | 96 |
| pin | 3 | 2 | 1 | 0 |
| Unmask | 0 | 1 | 0 | 0 |
| HEX | 0x04 | | | |

After unmasking there will be a dead loop in mainline program. When the button is pushed the program will intercept the interrupt signal and go to INT DIRECTOR. From here, it will check if the signal it come from surely from the button. By checking INTC and GPIO1, if the conformation of both are satisfied. The procedure toggle Led will be called. Otherwise it will just pass on and generate another NEWIRQA so can respond to new IRQ.

## Part 2 Algorithm

*Mainline:*

Setting up Stack pointer

Start up the module

Clear the pin on the module

Setting up leds output pins

Setting up button falling edge detect

Initialize INTC interrupts controller Unmasking the falling edge detect button

*INT DIRECTOR*

Check the INTC if yes go to check the module, else if no go to pass\_on

Check the module if yes go to toggle leds, else if no go to pass on

Toggle leds

Start the loop:

Adding LED 0 for 1 second

Adding LED 1 so LED 0,1 for 1 second

Adding LED 2 so LED 0,1,2 for 1 second

Adding LED 3 so LED 0,1,2,3 for 1 second

And turn off all LEDs for 1 second

Repeat the loop.

Pass\_on

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

## Assembly code

@Phong Nguyen

@Project2

@ECE 371

@ Program generate Interrupt to the processor. if the button is pushed the first time procedure

@of this request will light 4 LEDS in sequence, one to the next one for 1 second.

@Then turn all 4 LEDs off using interrupts from TIMER3 clock.

@if the button is pushed again then turn off all 4 LEDS sequence and waiting to for next interrupts push

@button. Program modify the Startup\_ARMCA8 file to intercept IRQ Interrupt Service Procedure

@INT\_Director

.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

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

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

@MODULE SETUP using GPIO1 pin 21-24 for output LEDS

@ 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

@ GPIO1\_ClEARDATAOUt for pin 21-24

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

MOV R1, #0x01E00000 @load value to turn off 21-24 LEDS on GPIO1

STR R1,[R0,#0x190] @Write to GPIO1\_CLEARDATAOUT register

@ GPIO1\_OE for pin 21-24

ADD R4,R0,#0x0134 @R4 is GPIO1\_OE register address

MOV R5,#0xFE1FFFFF @ pin 21-24 for output enable (0 is enable)

LDR R6,[R4] @Load current value from the GPIO1\_OE

AND R6,R6,R5 @Modify current value with pin 21-24

STR R6,[R4] @Write back to GPIO\_OE register address

@BUTTON SETUP using GPIO1 pin 30 for input button failing 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 Controler

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

@ 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

LDR R1,=DISPLAY @GPIO1 0x4804C000 with offset 13C dataout

MOV R2,0x0 @value indicate display off

STR R2,[R1] @Store in display off

B LOOP

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

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

INT\_DIRECTOR:

@ We check IRQ from the button and Timer

@ To check the IRQ we check the INTC interrupt controller and also the IRQ STATUS

@ of both Timer and button ortherwise will call PASS\_ON to get NEWIRQA from INTC

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

@Checking button and signal from INTC

CHECK\_INTC\_BUTTON:

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

LDR R2,[R0] @Load value to check

TST R2,#0x00000004 @pin 2 associate with INTC\_MIR\_CLEAR3 which was unmask

BNE CHECK\_GPIO\_BUTTON @Go to check button in GPIO1 module

B PASS\_ON

@Checking module GPIO1

CHECK\_GPIO\_BUTTON:

LDR R1,=0x4804C02C @Check GPIO IRQ STATUS from the module

LDR R2,[R1] @Load value from module

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 nothing in GPIO1 go to pass on

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

@This section is where the work is taking place from according to the info above

@But first remember to turn off interrupt request for the register

BUTTON\_SVC:

@check if Display is on or of

@turn off IRQ request for GPIO1

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

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

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

@ turn off NEWIRQA bit in INTC\_CONTROL so can respond to new IRQ

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

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

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

@checking display

LDR R1,=DISPLAY @GPIO1 0x4804C000 with offset 13C dataout

LDR R2,[R1] @R2 get value from display

TST R2,#0x1 @if there is a one in display or not

BNE TURN\_OFF @if there is a one then go to turn off

MOV R2,#0x1 @write a 1 in display

STR R2,[R1] @R1

B TURN\_ON @if there is nothing go to turn on

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

@LED part for BUTTON

@turn on LED

TURN\_ON:

@TOGGLE LED and wait for interrupt from button

LDR R1,=0x4804C000 @Base address of GPIO1

ADD R3,R1,#0x190 @R3 is address to CLEARDATAOUT of GPIO1

ADD R4,R1,#0x194 @R4 is address to SETDATAOUT of GPIO1

LED\_0:

MOV R6,#0x00200000 @Value to Turn on LED0

STR R6,[R4] @Write to Turn on LED0

BL ONE\_SEC @Go to delay 1 sec

LED\_0\_1:

MOV R6,#0x00600000 @Value to Turn on LED\_0\_1

STR R6,[R4] @Write to Turn on LED\_0\_1

BL ONE\_SEC @Go to delay 1 sec

LED\_0\_1\_2:

MOV R6,#0x00E00000 @Value to Turn on LED\_0\_1\_2

STR R6,[R4] @Write to Turn on LED2

BL ONE\_SEC @Go to delay 1 sec

LED\_0\_1\_2\_3:

MOV R6,#0x01E00000 @Value to Turn on LED\_0\_1\_2\_3

STR R6,[R4] @Write to Turn on LED3

BL ONE\_SEC @Go to delay 1 sec

LEDOFF:

MOV R5,#0x01E00000 @Load value to reset the GPIO1

STR R5,[R3] @Write to CLEARDATAOUT for GPIO1

BL ONE\_SEC @Go to delay 1 sec

B LED\_0 @go back to turn on LED\_0

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

ONE\_SEC:

MOV R2,#0x00400000 @delay 1 sec

LOOP2:

SUBS R2,#1 @Count down

BNE LOOP2 @jump back ONE\_SEC

MOV PC,R14 @after done 1 sec then go to current work

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

@turn off LEDs and Stop the timer3

TURN\_OFF:

LDR R1,=0x4804C190 @Base address for GPIO1\_CLEARDATAOUT register

MOV R2,#0x01E00000 @Value to turn off pin 21-24

STR R2,[R1] @Write to GPIO1\_CLEARDATAOUT register

B PASS\_ON @jump to pass on

PASS\_ON:

@ turn off NEWIRQA bit in INTC\_CONTROL so can respond to new IRQ

LDR R0,=0x48200048 @ Address of 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

DISPLAY: .word 0x0

STACK1: .rept 1024

.word 0x0000

.endr

STACK2: .rept 1024

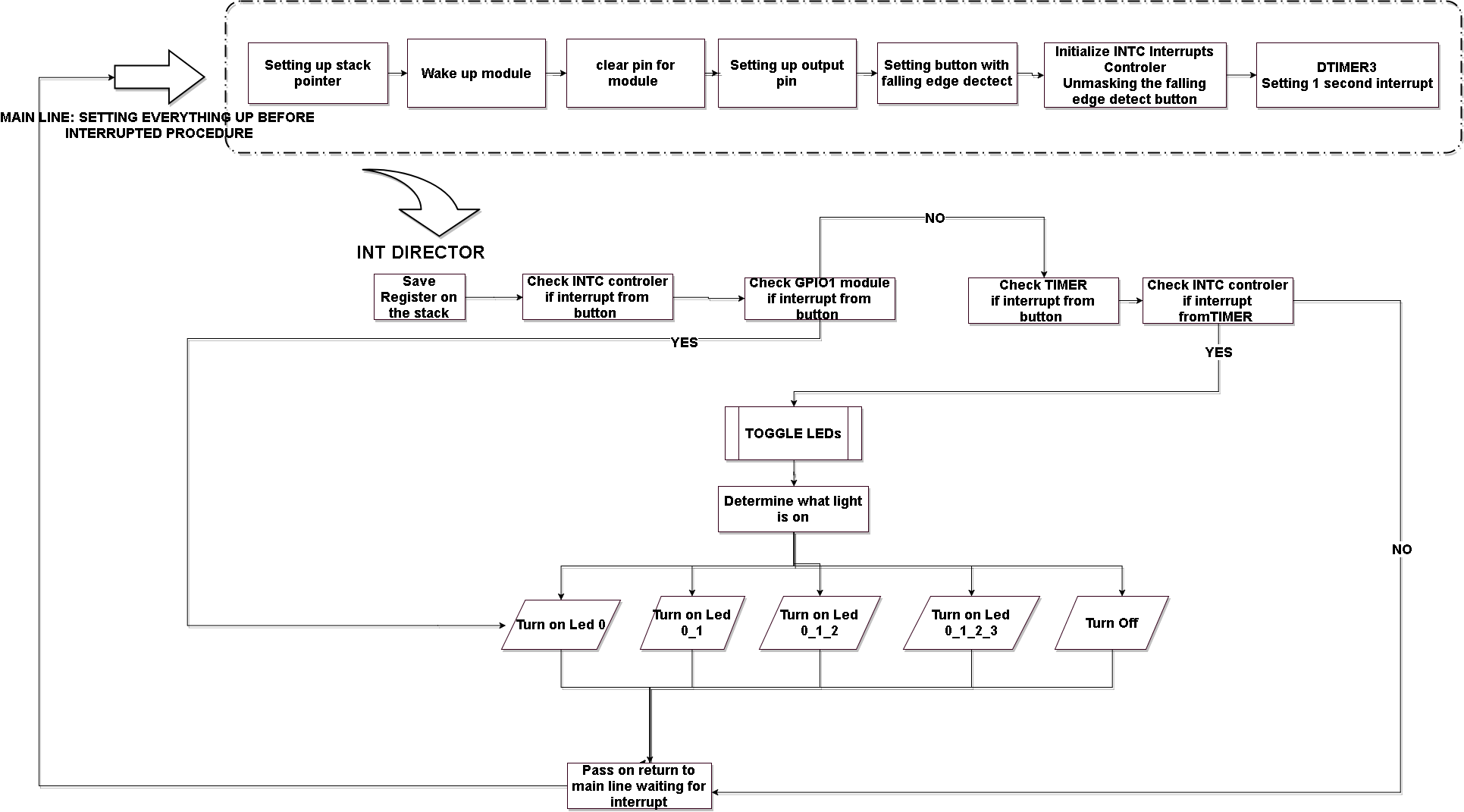
.word 0x0000

.endr

.END

# PART 3 SETTING UP DTIMER

## Part 3 Flow chart

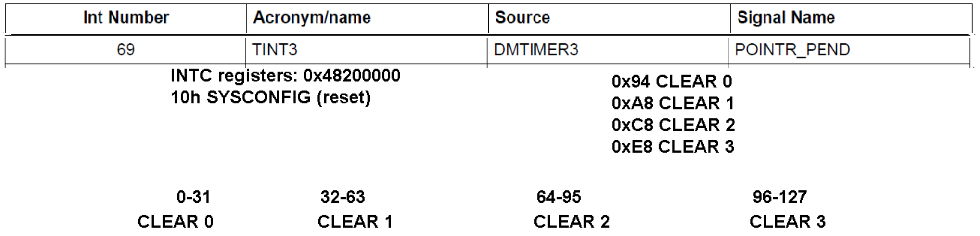


## Setting up register Timer3



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 372768 pulse per second. 372768 in hex would be 0x00008000. To get the desired time, follow the formula:

Desired time (1 second) = 0x1\_0000\_0000 – 0x0000\_8000 = 0xFFFF8000



Base on int number in interrupt controller, we going to unmask this pin on the INTC. Table template unmask the pin

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Int number | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 |
| Pin | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Unmask binary | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| Unmask hex | 2 | | | | 0 | | | |

After unmasking there will be a dead loop in mainline program. When the Timer is overflown the program will intercept the interrupt signal and go to INT DIRECTOR. From here, it will check if the signal it come from the button or timer. By checking INTC and Timer register, if the conformation of both are satisfied. The procedure toggle Led will be called. Otherwise it will just pass on and generate another NEWIRQA so can respond to new IRQ.

## Part 3 Algorithm

*MAINLINE:*

Setting up Stack pointer

Start up the module

Clear the pin on the module

Setting up leds output pins

Setting up button falling edge detect

Setting up DTIMER3

Initialize INTC interrupts controller Unmasking the falling edge detect button

Unmasking the DTIMER overflown interrupt

Dead loop

*INT DIRECTOR*

INT controller check

CHECK\_INTC\_BUTTON if yes go to CHECK\_GPIO\_BUTTON

Otherwise CHECK\_INTC\_TIMER, if yes go to CHECK\_GPIO\_TIMER

else no on both condition go to PASS\_ON

CHECK\_GPIO\_BUTTON

if yes go to BUTTON\_SVC

Else no go to PASS\_ON

CHECK\_GPIO\_TIMER

if yes go to TIMER\_SVC

Else no go to PASS\_ON

BUTTON\_SVC

Turn off the interrupt from the GPIO module

check if the DTIMER is running if yes go to LED\_OFF

Else if DTIMER is not running go to LED\_ON

LED\_ON

Turn on the led 0 and reload the timer go to PASS\_ON

LED\_OFF

Turn off all leds and stop the timer, then go to PASS\_ON

TIMER\_SVC

Determine what led is not turn on

If there is no LED is on, Adding LED 0 and PASS\_ON

If LED1 is not on, Adding LED 1 so LED 0,1 and PASS\_ON

If LED2 is not on, Adding LED 2 so LED 0,1,2 and PASS\_ON

If LED3 is not on, Adding LED 3 so LED 0,1,2,3 and PASS\_ON

If LED3 is on, turn off all LEDs and PASS\_ON

PASS\_ON

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

Return to Supervisor mode

## Low Algorithm Part 3

***MAINLINE***

First check from the INTC\_PENDING second check from IRQ STATUS from the register

Setting up the INT\_DIRECTOR in startup\_ARMCA8.s code to take the execution to INT\_DIRECTOR

In the program

1. Point stack1 for SVC mode and point stack2 for IRQ mode
2. Enable 0x02the clock for CM\_PER\_GPIO1\_CLKCTRL module 0x44E0\_00AC
3. CLEARDATAOUT for GPIO1 load value 0x01E0\_0000
4. GPIO1\_OE load value 0xFE1FFFFF using AND to modify (so don’t touch other pin if they are enable 0)
5. GPIO1\_FALLINGDETECT at pin 30 value 0x4000\_0000 using ORR (ignore other pin just turn on the Button to high)
6. Also enable GPIO1\_IRQSTATUS\_SET\_0 0x4804\_C034 at pin 30 with value 0x4000\_0000
7. INTC 0x4820\_0000, reset the INTC by writing 0x2 for INTC\_CONFIG register
8. Unmask Timer3, INTC INT #69, Write value 0x20 pin 5 at INTC\_MIR\_CLEAR2
9. Unmask Button, INTC INT #98, POINTPEND1 at GPIO1 Write value 0x04 pin 2 at INTC\_MIR\_CLEAR3

10)Enable the CM\_PER\_TIMER3\_CLKCTRL at 0x44E0\_0084 write value 0x2

11) Setting the MUX for CLKSEL\_TIMER3\_CLK to 32 KHz address 0x44E0\_0503 write value 0x2

12) Initialize DMTimer 3 register 0x4804\_2000

13) Reset timer at CFG configuration register 0x4804\_2010 write 0x1

14) Enable overflow interrupt at IRQENABLE\_SET 0x4804\_202C write 0x2

15) Count value for 1 second 0x1\_0000\_0000-0x8000=0xFFFF\_8000 Write 0xFFFF\_8000 to TLDR (reload) offset 0x40

16) Write 0xFFFF\_8000 to TCRR (Count) offset 0x3C

17)Enable IRQ in CPSR clear bit 7 write back in CPSR\_c lower 8 bit in CPSR

18) dead loop

***INT\_DIRECTOR***

CHECK\_INTC\_BUTTON:

1) push register on the stack

2) Reading INTC\_PENDING\_IRQ3 0x4820\_00F8 (Button press or not)

3) test pin2 of INTC\_PENDING\_IRQ3 with 0x0000\_0004 associate with INTC\_MIR\_CLEAR3 which was unmasked

4) if pin 2 of INTC\_PENDING\_IRQ3 is 0 flag Z is set (not button push IRQ) then Branch to CHECK\_INTC\_TIMER

5) if pin 2 of INTC\_PENDING\_IRQ3 is 1 flag Z is clear then 6) if bit 30 is 1 (Z flag is clear) BNE then go to CHECK\_GPIO\_BUTTON else not our IRQ Load INTC\_CONTROL 0x4820\_0048 write #01 to clear bit 0

7) if bit 30 is 0 (Z flag is set) then go to PASS\_ON

CHECK\_INTC\_TIMER:

1. Reading INTC\_PENDING\_IRQ2 0x482000D8 (Timer3 overflow or not)
2. Test pin 5 of INTC\_PENDING\_IRQ2 with 0x0000\_0020 associate with INTC\_MIR\_CLEAR2 which was unmasked
3. If pin 5 of INTC\_PENDING\_IRQ2 is 0 flag Z is set (not Timer) then Branch to PASS\_ON
4. If pin of INTC\_PENDING\_IRQ2 is 1 flag Z is clear, go to CHECK\_GPIO\_TIMER

CHECK\_GPIO\_BUTTON

1)Button is pressed by Test bit 30 0x4000\_0000 value load from GPIO1\_IRQSTATUS\_0 0x4804\_C02C.

2)If pin is 1 then go to BUTTON\_SVC

3)Else If pin is 0 then go to PASS\_ON

CHECK\_GPIO\_TIMER

1. Check if activated by Timer3 by test bit 1 0x2 value load from IRQSTATUS\_TIMER3 0x4804\_2028
2. If timer 3 bit 1 of IRQSTATUS\_TIMER3 is 0 flag z is set (not timer3) then Branch to PASS\_ON
3. If timer 3 bit 1 of IRQSTATUS\_TIMER3 is 1 flag z is clear, then Branch to TIMER\_SVC.

BUTTON\_SVC:

1. Turn off GPIO1\_14 IRQ request by writing 0x4000\_0000 pin 30 at GPIO1\_IRQSTATUS\_0 0x4804\_C02C.
2. Check if DTIMER 3 is running then go to LED\_OFF, else if DTIMER3 is not running LED\_ON
3. Else if GPIO1\_DATAOUT show 4 LED status if none of them is lit

LED\_ON:

1)Turn on LED0,

2)The timer 3 write 0x03 to 0x4804\_2038 TCLR\_TIMER3 (reload and start)

3)Go to PASS\_ON waiting for another interrupt

LED\_OFF:

1)Turn off all LEDs

2)Turn off DTIMER3 write 0x00 to 0x4804\_2038 TCLR\_TIMER3

3)Go to PASS\_ON waiting for another interrupt

TIMER\_SVC:

1)Turn off timer request write 0x2 at 0x4804\_2028 IRQSTATUS\_TIMER3

2)Check if GPIO1\_DATAOUT 0x4804C13C status of 4 LED

* 1. If LED0 is lit, turn on LED1 then PASS\_ON
  2. If LED1 is lit, turn on LED2 then PASS\_ON
  3. If LED2 is lit, turn on LED3 then PASS\_ON
  4. If LED3 is lit, turn off all LED then PASS\_ON
  5. If no LED is lit, turn on LED0 then PASS\_ON

PASS\_ON:

1. At INTC\_CONTROL 0x4820\_0048 write #01 to clear bit 0
2. restore register pop from the Stack R0-R3, LR
3. pass execution to wait LOOP PC=LR - #4

## Part 3 Assembly code

@Phong Nguyen

@Project2

@ECE 371

@ Program generate Interrupt to the processor.

@if the button is pushed the first time procedure

@of this request will light 4 LEDS in sequence, one to the next one for 1 second .

@Then turn all 4 LEDs off using interrupts from TIMER3 clock.

@if the button is pushed again then turn off all 4 LEDS sequence

@and waiting to for next interrupts push button.

@Program modify the Startup\_ARMCA8 file to intercept IRQ Interrupt Service Procedure

@INT\_Director

.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

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

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

@MODULE SETUP using GPIO1 pin 21-24 for output LEDS

@ 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

@ GPIO1\_ClEARDATAOUt for pin 21-24

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

MOV R1, #0x01E00000 @load value to turn off 21-24 LEDS on GPIO1

STR R1,[R0,#0x190] @Write to GPIO1\_CLEARDATAOUT register

@ GPIO1\_OE for pin 21-24

ADD R4,R0,#0x0134 @R4 is GPIO1\_OE register address

MOV R5,#0xFE1FFFFF @ pin 21-24 for output enable (0 is enable)

LDR R6,[R4] @Load current value from the GPIO1\_OE

AND R6,R6,R5 @Modify current value with pin 21-24

STR R6,[R4] @Write back to GPIO\_OE register address

@BUTTON SETUP using GPIO1 pin 30 for input button failling 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,=0xFFFF8000 @Count value 1 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 Controler

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

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

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

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

@ We check IRQ from the button and Timer

@ To check the IRQ we check the INTC interrupt controller and also the IRQ STATUS

@ of both Timer and button ortherwise will call PASS\_ON to get NEWIRQA from INTC

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

@Checking button and signal from INTC

CHECK\_INTC\_BUTTON:

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

LDR R2,[R0] @Load value to check

TST R2,#0x00000004 @pin 2 associate with INTC\_MIR\_CLEAR3 which was unmask

BNE CHECK\_GPIO\_BUTTON @Go to check button in GPIO1 module

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\_GPIO\_TIMER @Go to check timer in GPIO1 module

@other wise not our IRQ signal

B PASS\_ON

@Checking module GPIO1

CHECK\_GPIO\_BUTTON:

LDR R1,=0x4804C02C @Check GPIO IRQ STATUS from the module

LDR R2,[R1] @Load value from module

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 nothing in GPIO1 go to pass on

@Checking module Timer3

CHECK\_GPIO\_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

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

@This section is where the work is taking place from according to the info above

@But first remember to turn off interrupt request for the 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 writing there 1

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

@check if GPIO1\_DATAOUT has LEDs is lit

LDR R1,=0x4804C13C @GPIO1 0x4804C000 with offset 13C dataout

@LDR R2,[R1] @Load value from GPIO\_DATAOUT to check

@TST R2,#0x01E00000 @ test pin 21-24 on GPIO1

@TESTING TIMER RIGHT HERE

LDR R0,=0x48042038 @Address for Timer3 TCLR

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

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

BEQ LED\_ON @if z flag is clear (no led is lit) go LED\_ON

B LED\_OFF @if z flag is set (at least 1 led is lit) go LED\_OFF

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

@LED part for BUTTON

@turn on LED and start auto load the timer3

LED\_ON:

LDR R1,=0x4804C194 @Load address of GPIO1\_SETDATAOUT register

MOV R2,#0x00200000 @Value to turn on LED0 for first sequence

STR R2,[R1] @Write value to turn on the LED0

@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 Timer3 controler

B PASS\_ON @ go to PASS\_ON

@turn off LEDs and Stop the timer3

LED\_OFF:

LDR R1,=0x4804C190 @Base address for GPIO1\_CLEARDATAOUT register

MOV R2,#0x01E00000 @Value to turn off pin 21-24

STR R2,[R1] @Write to GPIO1\_CLEARDATAOUT register

@turn off TIMER

LDR R0,=0x48042038 @Address for Timer3 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

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

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

@LED part for TIMER3

TIMER\_SVC:

@turn off IRQ request for TIMER3

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

MOV R2,#0x02 @value to reset Timer 3 Overflow IRQ request

STR R2,[R0] @Write value to turn off IRQ signal

@TOGGLE LED

LDR R1,=0x4804C000 @Base address of GPIO1

ADD R3,R1,#0x190 @R3 is address to CLEARDATAOUT of GPIO1

ADD R4,R1,#0x194 @R4 is address to SETDATAOUT of GPIO1

LDR R2,[R1,#0x13C] @offset 13C to get the dataout status GPIO1\_DATAOUT

TST R2,#0x01E00000 @check if LEDs is all off

BEQ LED\_0 @Turn LED\_0 on

TST R2,#0x00400000 @Check if LED\_1 is lit

BEQ LED\_0\_1 @Go to turn on LED\_0\_1

TST R2,#0x00800000 @check if LED\_2 is lit

BEQ LED\_0\_1\_2 @Go to turn on LED\_0\_1\_2

TST R2,#0x01000000 @Check if LED0-1-2 is lit

BEQ LED\_0\_1\_2\_3 @Go to turn on LED3

TEQ R2,#0x01E00000 @Check if LED\_0\_1\_2\_3 is lit

BEQ LEDOFF @Go to turn off all LEDs

LED\_0\_1:

MOV R6,#0x00600000 @Value to Turn on LED1

STR R6,[R4] @Write to Turn on LED1

B PASS\_ON @Go to PASS\_ON

LED\_0\_1\_2:

MOV R6,#0x00E00000 @Value to Turn on LED2

STR R6,[R4] @Write to Turn on LED2

B PASS\_ON @Go to PASS\_ON

LED\_0\_1\_2\_3:

MOV R6,#0x01E00000 @Value to Turn on LED3

STR R6,[R4] @Write to Turn on LED3

B PASS\_ON @Go to PASS\_ON

LEDOFF:

MOV R5,#0x01E00000 @Load value to reset the GPIO1

STR R5,[R3] @Write to CLEARDATAOUT for GPIO1

B PASS\_ON @Go to PASS\_ON

@ turn off NEWIRQA bit in INTC\_CONTROL so can respond to new IRQ

PASS\_ON:

LDR R0,=0x48200048 @Address of 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

.END