Phil Nevins
ECE 372
Design Project 1

Final Write-Up

This project was very interesting. I ran into quite a few issues , that ended up not being too difficult to fix. It gave me a lot of practice debugging and going line by line through my code, while watching the registers and values in memory to ensure the code is working properly. I also learned that it is a good idea to utilize memory locations, pointers and incrementation instead of having a bunch of different functions for your program, as this was the main issue with the save state not working properly.

The debouncing is an important thing to know because when we are out on the job and have to produce perfect operational products, we will need to debounce every mechanical switch we end up using to ensure the products operate properly.

The timer part was confusing at first, but after thorough research, I was able to figure out how the MIR blocks are setup and how to unmask #69 for timer3. Once I figured this out, it was smooth sailing for the most part.

Design Log

Entry Date: 1/22/2023 (Part 1)

Today, I am reviewing the algorithms and code from Chapter 5 of the ECE 371 manual, pages 241 – 247. I understand it now. You need to initialize the timer# you want to use. In part 1 of the project, we are tasked with making LED3 blink using timer3.

I am also tasked with using timer3, so we need to do the following, in addition to what we did in Design Project 1, Part 2 (cyclone eye):

We need to calculate the value we need for a 2 second delay on the timer. To do this, we use the equation from the 371 manual, which says that TLDR = 0x10000000 – 0x00008000, the 0x00008000 is what you subtract for one second. So, we multiply: 0x00008000 * 2 = 0x00010000. Therefore, to find our 2 second delay value for the timer, we have TLDR = 0x10000000 – 0x00010000 = 0xFFFF000

Inside Initialize INTC:

- Store the value to reset INTC (0x2)
- Write this value (0x2) to the INTC config register (0x48200010)
- Unmask INTC INT 69, timer3 interrupt

Note: the INTC has 128 lines, numbered 0-127 ... but when these are mapped to the control registers for the masks, they're broken down into four 32-bit banks. 0-31 is MIR0, 32-63 is MIR1, 64-95 is MIR2, and 96-127 is MIR3. So, for 68 we want to write to the 4th bit of MIR2, which is a control word of 0x8. For line 69 (the one we need), we use the 5th bit for MIR2, or 0x20

The rest of INTC is the same as we used in Design Project 1, Part 2.

The next thing we need to do is turn on the Timer3CLK

We do this by storing the value to enable timer3 CLK (#0x2), then write this to the CM_PER_TIMER3_CLKCTRL address (0x44E00084). This address is found in the CLKCTRL section of the TI-Manual, on page 1282.
The base address is 0x44E00000 (derived from comparing the 371 manual to the TI manual for timer2, and subtracting the timer2s offset to get the base address), and the offset for CM_PER_TIMER3_CLKCTRL is 84h.

Next is to select the 32Khz CLK for timer 3. We do this by sending 0x02 to the CLKSEL_TIMER3_CLK address (0x44E0050C). This address is found in the CLKSEL_TIMER3_CLK section of the TI-Manual, on page 1379.
The base address is 0x44E00500 (derived from comparing the 371 manual to the TI manual for timer2 and subtracting timer2s offset to get the base address), and the offset for CLK_SEL_TIMER3_CLK is Ch.

The final thing is to initialize the Timer3 registers, with count, overflow and interrupt generation. The base address for the Timer3 registers is (0x48042000). This is derived from the timer2 base address of 48040000. Timer3 is 0x00002000 higher than timer2, which is where we get the 0x48042000 from. The following was derived from the 371 manual, page 245. You need to write 0x1 to Timer3 CFG register (0x48042010), then 0x2 to timer3 IRQENABLE_SET (0x4804202C). Next is to load the delay value we calculated above (0xFFFF0000) and store it in the timer3 TLDR load register and the Timer3 TCRR count register.

Now Timer3 is initialized.

Next we will write the high-level algorithm because the rest of the program is from Design Project 1, Part 2 (Single LED).

High Level Algorithm USR LED3 Blinking with timer3

MAINLINE
Set up stacks for Supervisor and IRQ mode
Turn on GPIO1 clock
Set up to detect a falling edge on GPIO1_3 and generate interrupt
Initialize INTC–Reset INTC and enable timer3 interrupt on INTC #69, button interrupt on #98
Turn on Timer3 CLK and set functional clock input mux for 32.786 KHz clock
Initialize timer registers for count and overflow generation

Enable IRQ input by clearing bit 7 in CPSR
Set GPIO24 as an output

INT_DIRECTOR
Save registers
Check if interrupt is from GPIO1_3
If YES, go to Service Button
    ELSE check for timer3 interrupt
    IF NOT timer3 interrupt, restore registers and return to wait loop
    ELSE check if timer3 overflow interrupt
    IF NOT timer3 overflow interrupt, restore registers and return to wait loop
    ELSE, go to LED3ON

PASS_ON
Turn off NEWIRQA bit in INTC Control
Restore registers
Return to wait loop

BUTTON_SVC
Turn off GPIO1_3 and INTC interrupt
Turn on LED
Start timer3 and set for auto reload
Enable INTC for new interrupt
Restore registers and return to wait loop

LED3ON
Reset Timer3 Overflow IRQ request
Test if LED3 is on
IF ON
    GO TO LED3OFF
ELSE turn on LED3
GO TO BACK

LED3OFF
Turn off LED3

BACK
Enable new IRQ response in INTC_CONTROL
Restore registers and return to wait loop

Low Level Algorithm USR LED3 Blinking with timer3

Initialize Stack1 pointer for SVC Mode
Turn on IRQ Mode using CPS #0x12
Initialize Stack2 pointer for IRQ Mode
Turn on SVC Mode using CPS #0x13

Load R5 with #0x01

Value to enable CLK for GPIO module
ADDR OF CM_PER_GPIO1_CLKCTRL Register (0x44E000AC)
Write #02 to register
Base ADDR for GPIO1 Registers (0x4804C000)

ADDR of GPIO1_FALLINGDETECT Register (0x4804C14C)
Load value for Bit 3 (GPIO1_3)
Read GPIO1_FALLINGDETECT register
Modify (set bit 3)
Write back
Addr of GPIO1_IRQSTATUS_SET_0 Register (0x4804C034)
Enable GPIO1_3 request on POINTRPEND1

Base Addr for INTC (0x48200000)
Value to reset INTC (0x02)
Write to INTC Config Register (0x48200010)
Unmask INTC INT 69, Timer3 interrupt (#0x20)
Write to INTC_MIR_CLEAR_2 register (0x482000C8)
Value to unmask INTC INT 98, GPIONT1A (#0x04)
Write to INTC_MIR_CLEAR3 Register ((0x482000E8)

Value to enable Timer3 CLK (0x02)
Addr of CM_PER_TIMER3_CLKCTRL (0x44E00083)
Turn on
Addr of CLKSEL_TIMER3_CLK Register (0x44E0050C)
Select 32 KHz CLK for Timer3

Base addr for timer3 registers (0x48042000)
Value to reset timer3 (0x01)
Write to Timer3 CFG register (0x48042010)
Value to enable overflow interrupt (0x02)
Write to timer3 IRQENABLE_SET (0x4804202C)
Count value for 2 seconds (0xFFFF0000)
Timer3 TLDR load register (Reload value) ((0x48042040)
Write to Timer3 TCRR count register (0x4804203C)

Load word to program GPIO1_21-24 to output (0xFE1FFFFF)
Addr of GPIO1_OE Register (0x4804C134)
Read GPIO1_OE Register
Modify word read in with R0
Write back to GPIO1_OE Register

Copy CPSR to R3
Clear bit 7 (0x80)

Write back to CPSR

Load word to target GPIO1_21-24 (0x1E00000)
Load addr of GPIO1_CLEARDATAOUT (0x4804C190)
Write to GPIO1_CLEARDATAOUT (This turns LED1-4 OFF)

Wait Loop that waits for IRQ Interrupt Request

INT_DIRECTOR Procedure
Push registers onto stack
Addr of INTC-PENDING_IRQ3 (0x482000F8)
Read INTC-PENDING_IRQ3
Test bit 2
Not from GPIOINT1A, check if Timer3, Else
Load addr of GPIO1_IRQSTATUS_0 (0x4804C02C)
Read Status register to see if button press
Check if bit 3 = 1
If bit 3 = 1 button is pressed, Branch to BUTTON_SVC
Else, Go back. INTC_CONTROL register (0x48200048)
Value of clear bit 0
Write to INTC_CONTROL register
Restore Registers
Pass execution to wait LOOP for now

TCHK
Addr of INTC_PENDING_IRQ2 register (0x482000D8)
Read value
Check if the interrupt from Timer3 (bit 5) (0x20)
No go to PASS_ON, Yes check overflow
Addr of Timer3 IRQSTATUS register (0x48042028)
Read Value
Check bit 1
If overflow, go LED3ON

PASS_ON Procedure
Addr of INTC_CONTROL register (0x48200048)
Value to clear bit 0 (0x01)
Write to INTC_CONTROL register
Restore Registers
Pass execution to wait LOOP for now

BUTTON_SVC Procedure
Value to turn off GPIO1_3 & INTC Interrupt request (0x08)
Write to GPIO1_IRQSTATUS_0 Register (0x4804C02C)

Load addr of GPIO1_SETDATAOUT Register (0x4804C194)
Load value to turn on GPIO1_24 (0x01000000)
Write to GPIO1_SETDATAOUT Register (0x4804C194)

Load value to auto reload timer and start (0x03)
Addr of Timer3 TCLR Register (0x48042038)
Write to TCLR Register

ADDR of INTC_CONTROL Register (0x48200048)
Value to clear bit 0 (0x01)
Write to INTC_CONTROL Register
Restore Registers
Pass execution to wait loop for now

LED3ON
Load addr of Timer3 IRQSTATUS register (0x48042028)
Value to reset Timer2 Overflow IRQ request (0x02)
Write

Base addr for GPIO1 (0x4804C000)
Read value from GPIO1_DATAOUT (0x4804C13C)
Test if bit 24 = 1 (0x01000000)
Load value to set or clear bit 24 (turn on or off LED)
IF LED on, go to LED3OFF
IF LED OFF, turn on with GPIO1_SETDATAOUT (0x4804C194)
Back to wait loop

LED3OFF
Turn LED off with GPIO1_CLEARDATAOUT (0x4804C190)

BACK
Addr of INTC_CONTROL register (0x48200048)
to enable new IRQ response in INTC (0x01)
Write
Restore Registers
Return from IRQ interrupt

Now we will convert the low-level algorithm to code.

After debugging, we have a working program. One issue that arose was using the wrong address, that was a result of a typo.

Entry 1/24/2022

Now we are going to be working on Part 2 & 3, since Part 3 is just adding the debouncing function from the textbook. We are tasked with converting our Cyclone Eye program from Design Project 1, Part 2, to be used with timer3 instead of a delay loop. We must also have the function of the button follow this: one press starts it, the next stops it, the next starts it where it left off. For this, we will need to save the state (which LED is on or what the next one will be).

To achieve the "save state" function, we will be using a flag to designate which LED was on/off, so when we turn them back on, the right one will turn on.

This will be a basic flag function, where inside each LED on/off function, it will set a register to a hex #. Then before we enter the LED functions, there will be a compare function that determines which LED function to re-enter when the button is pressed to turn the LEDs back on. This should be a simple addition.

Update at end of day: I ran into two separate issues. The "saved state" was ahead one step. Meaning, LED0 is turned off, then when its turned back on, LED1 would be turned on. It would not re-start at LED0. The other issue was the program would get stuck in whatever USR LED was on when it was turned off.

Entry Date: 1/30/2023

The debouncing aspect is just a simple set of code to turn on Aux Function CLK, Bit 18 and CLK, through CMP_PER_GPIO_CLKCTR (0xx44E000AC). Then enable debounce on GPIO1_3 (0x4804C150) via bit 3. Then write the debounce interval (31 microseconds) to the GPIO1_DEBOUNCING_TIME (0x4804C154). **Considering this, we will write the algorithms for the cyclone eye w/ save state and debouncing together.**

On the 24th, I continued attempts at getting my Design Project 1, Part 2 code to operate as intended, with the "save state" ability, and could not get it to work properly. So, I decided to re-write my entire program. Originally, I wrote is using separate functions for LED#on / LED#off. I believe this is the root cause of the issue. The way I am re-writing this code today will be using an LED on and LED off function, that increments pointers to the LED sequence that is held in memory.

High Level Algorithm Cyclone Eye with save state and debouncing

MAINLINE
Set up stacks for Supervisor and IRQ mode
Turn on GPIO1 clock
Set up to detect a falling edge on GPIO1_3 and generate interrupt
Initialize INTC–Reset INTC and enable timer3 interrupt on INTC #69, button interrupt on #98
Turn on Timer3 CLK and set functional clock input mux for 32.786 KHz clock
Initialize timer registers for count and overflow generation
Enable debounce and load with proper time

Enable IRQ input by clearing bit 7 in CPSR
Set GPIO21-24 as an output
Turn all LEDs off (initialize state)

INT_DIRECTOR / TCHK
Save registers
Check if interrupt is from GPIO1_3
If YES, go to Service Button
    ELSE check for timer3 interrupt
    IF NOT timer3 interrupt, restore registers and return to wait loop
    ELSE check if timer3 overflow interrupt
    IF NOT timer3 overflow interrupt, restore registers and return to wait loop
    ELSE, go to LEDFunction

PASS_ON
Turn off Timer3 IRQSTATUS
Turn off NEWIRQA bit in INTC Control
Restore registers
Return to wait loop

BUTTON_SVC
Turn off GPIO1_3 and INTC interrupt
Load pointer to LED_Flag in memory
Load value from LED_FLAG to use
Test if LED_Flag is off
IF ON
    GO TO LEDFunction_ON
ELSE GO TO LEDFunction_OFF

LEDFunction_OFF
Set LED_Flag to off
Turn off Timer3
Load pointer to buffer value
Load pointer to Current_State value
Load value from buffer
Store buffer in current state
Turn all LEDs off
GO TO BACK

LEDFunction_ON
Set LED_Flag to on
Load pointer to USRLEDCYCLE value
Load pointer to Current_State value
Increment USRLEDCYCLE pointer by Current State value

Turn on LED using new USRLEDCYCLE value in memory
Auto reload and start Timer3

BACK
Enable new IRQ response in INTC_CONTROL
Restore registers and return to wait loop

LEDFunction
Reset Timer3 Overflow IRQ Request
Load pointer to USRLEDCYCLE
Load value in Buffer to increment
Load value of USRLEDCYCLE + BUFFER
Turn off LED associated with above address
Test BUFFER
    IF Buffer = 20, reset value to 0
    ELSE (Buffer > 20), increment Buffer by #04
Store Buffer in memory
Turn on Next LED (USRLEDCYCLE + BUFFER)
Turn off NEWIRQA bit in INTC Control
Restore registers and return to wait loop

Low Level Algorithm Cyclone Eye with save state and debouncing

Initialize Stack1 pointer for SVC Mode
Turn on IRQ Mode using CPS #0x12
Initialize Stack2 pointer for IRQ Mode
Turn on SVC Mode using CPS #0x13
Load R5 with #0x01

Value to enable CLK for GPIO module
ADDR OF CM_PER_GPIO1_CLKCTRL Register (0x44E000AC)
Write #02 to register
Base ADDR for GPIO1 Registers (0x4804C000)

ADDR of GPIO1_FALLINGDETECT Register (0x4804C14C)
Load value for Bit 3 (GPIO1_3)
Read GPIO1_FALLINGDETECT register
Modify (set bit 3)
Write back
Addr of GPIO1_IRQSTATUS_SET_0 Register (0x4804C034)
Enable GPIO1_3 request on POINTRPEND1

Base Addr for INTC (0x48200000)

Value to reset INTC (0x02)
Write to INTC Config Register (0x48200010)
Unmask INTC INT 69, Timer3 interrupt (#0x20)
Write to INTC_MIR_CLEAR_2 register (0x482000C8)
Value to unmask INTC INT 98, GPIONT1A (#0x04)
Write to INTC_MIR_CLEAR3 Register ((0x482000E8)

Value to enable Timer3 CLK (0x02)
Addr of CM_PER_TIMER3_CLKCTRL (0x44E00083)
Turn on
Addr of CLKSEL_TIMER3_CLK Register (0x44E0050C)
Select 32 KHz CLK for Timer3

Base addr for timer3 registers (0x48042000)
Value to reset timer3 (0x01)
Write to Timer3 CFG register (0x48042010)
Value to enable overflow interrupt (0x02)
Write to timer3 IRQENABLE_SET (0x4804202C)
Count value for 2 seconds (0xFFFF0000)
Timer3 TLDR load register (Reload value) ((0x48042040)
Write to Timer3 TCRR count register (0x4804203C)

Base addr for GPIO1 (0x4804C000)
Addr of CM_PER_GPIO1_CLKCTRL (0x44E000AC)
Value to turn on Aux Funct CLK, bit 18 and CLK (0x00040002)
Write value to CMP_PER_GPIO_CLKCTRL
Addr of GPIO1_DEBOUNCABLE (0x4804C150)
Load value of GPIO1 for bit 3 (0x08)
Enable GPIO1_3 debounce
Addr of GPIO1_DEBOUNCING TIME (0x4804C154)
Value for 31 Micro-Seconds debounce interval (#0xA0)
Write to GPIO1_DEBOUNCING TIME

Load word to program GPIO1_21-24 to output (0xFE1FFFFF)
Addr of GPIO1_OE Register (0x4804C134)
Read GPIO1_OE Register
Modify word read in with R0
Write back to GPIO1_OE Register

Copy CPSR to R3
Clear bit 7 (0x80)
Write back to CPSR

Load word to target GPIO1_21-24 (0x1E00000)
Load addr of GPIO1_CLEARDATAOUT (0x4804C190)
Write to GPIO1_CLEARDATAOUT (This turns LED1-4 OFF)

Wait Loop that waits for IRQ Interrupt Request

INT_DIRECTOR Procedure
Push registers onto stack
Addr of INTC-PENDING_IRQ3 (0x482000F8)
Read INTC-PENDING_IRQ3
Test bit 2
Not from GPIOINT1A, check if Timer3, Else
Load addr of GPIO1_IRQSTATUS_0 (0x4804C02C)
Read Status register to see if button press
Check if bit 3 = 1
If bit 3 = 1 button is pressed, Branch to BUTTON_SVC
Else, Go back. INTC_CONTROL register (0x48200048)
Value of clear bit 0
Write to INTC_CONTROL register
Restore Registers
Pass execution to wait LOOP for now

TCHK
Addr of INTC_PENDING_IRQ2 register (0x482000D8)
Read value
Check if the interrupt from Timer3 (bit 5)
No go to PASS_ON, Yes check overflow
Addr of Timer3 IRQSTATUS register (0x48042028)
Read Value
Check bit 1
If overflow, go LEDFunction

PASS_ON Procedure
Value to turn Timer3 off (0x02)
Load addr of IRQSTATUS Timer3 (0x48042028)
Write to IRQSTATUS Timer3
Addr of INTC_CONTROL register (0x48200048)
Value to clear bit 0 (0x01)
Write to INTC_CONTROL register
Restore Registers
Pass execution to wait LOOP for now

BUTTON_SVC Procedure
Value turns off GPIO1_3 Interrupt Request (0x08)
Write to GPIO1_IRQSTATUS_0 register
Load pointer to LED_Flag
Load value from LED_Flag

Compare LED_Flag value to 0
Branch if equal go to LEDFunction_ON
Branch if not equal go to LEDFunction_OFF

LEDFunction_OFF
@Value to change LED_Flag state & Turn off Timer3 (0x00)
Write to LED_Flag
Load addr to DMTIMER3_TCLR (0x48042038)
Write to DMTIMER3_TCLR to turn timer3 off
Load pointer to BUFFER
Load pointer to Current_State
Load value from BUFFER
Store BUFFER value into Current_State
Load addr of GPIO1_CLEARDATAOUT (0x4804C190)
Value to clear bits 24-21 (0x01E00000)
Write to GPIO1_CLEARDATAOUT
Branch to BACK

LEDFunction_ON:
Value to change LED_Flag state (0x01)
Write to LED_Flag
Load pointer to USRLEDCYCLE
Load pointer to Current_State
Load value from Current_State into R2
Add Current_State offset to base addr in USRLEDCYCLE
Load GPIO1_SETDATAOUT addr (0x4804C194)
Write to GPIO1_SETDATAOUTR register
Load Value into auto realod and start Timer3 (0x03)
Load addr for Timer3 TCLR register (0x48042038)
Write to Timer3 TCLR register

BACK
Addr of INTC_CONTROL register (0x48200048)
to enable new IRQ response in INTC (0x01)
Write
Restore Registers
Return from IRQ interrupt

LEDFunction
Load addr of Timer3 IRQSTATUS register (0x48042028)
Value to reset Timer3 Overflow IRQ request (0x02)
Write
Load pointer to USRLEDCYCLE (R0)
Load pointer to BUFFER (R3)
Load value in BUFFER to increment
Load addr of GPIO1_CLEARDATAOUT (0x4804C190)

Load value of the sum of BUFFER and USERLEDCYCLE
Write to GPIO1_CLEARDATAOUT to turn off LED
Compare value of BUFFER to 20
If BUFFER = #20 reset value to 0
If BUFFER > #20, increment BUFFER by #04
Store new BUFFER value in BUFFER
Load addr of GPIO1_SETDATAOUT (0x4804C194)
Load R4 with BUFFER + USRLEDCYCLE
Write to GPIO1_SETDATAOUT to turn on next LED

Addr of INTC_CONTROL register (0x48200048)
Value to clear bit 0 (0x01)
Write to INTC_CONTROL register

Restore Registers
Pass execution to wait LOOP for now

Array Declarations
USRLEDCYCLE: .word 0x01000000, 0x00800000, 0x00400000, 0x00200000, 0x00400000, 0x00800000
Current_State: .word 0x0
BUFFER: .word 0x0
LED_Flag: .word 0x0

```
@Phil Nevins
@ECE 371 Microprocessor
@Design Project 2, Part 2
@This program will use a pushbutton to trigger an interrupt and cycle an LED
@The program will do this exactly: push button, LED Cyclone on, push button,
@LED cyclone off, push button, LED cyclone on...The cyclone will start
@where it was interrupted at
@Program uses R0-R5

.text
.global _start
.global INT_DIRECTOR

_start:

LDR R13, =STACK1          @Point to base of STACK1 for SVC mode
ADD R13, R13, #0x1000     @Point to top of STACK1
CPS #0x12                       @Switch to IRQ mode
LDR R13, =STACK2          @Point to IRQ STACK2
ADD R13, R13, #0x1000     @Point to top of STACK2
CPS #0x13                       @Back to SVC mode

@Turn on GPIO1 CLK
MOV R0, #0x02             @Value to enable CLK for GPIO module
LDR R1, =0x44E000AC       @ADDR OF CM_PER_GPIO1_CLKCTRL Register
```

```
STR R0, [R1]              @Write #02 to register
LDR R0, =0x4804C000       @Base ADDR for GPIO1 Registers

@Detect Falling Edge on GPIO1_3 and eable to assert POINTRPEND1
ADD R1, R0, #0x14C        @R1 = ADDR of GPIO1_FALLINGDETECT Register
MOV R2, #0x00000008       @Load value for Bit 3 (GPIO1_3)
LDR R3, [R1]              @Read GPIO1_FALLINGDETECT register
ORR R3, R3, R2                  @Modify (set bit 3)
STR R3, [R1]              @Write back
ADD R1, R0, #0x34         @Addr of GPIO1_IRQSTATUS_SET_0 Register
STR R2, [R1]              @Enable GPIO1_3 request on POINTRPEND1

@Initialize INTC
LDR R1, =0x48200000       @Base Addr for INTC
MOV R2, #0x2              @Value to reset INTC
STR R2, [R1,#0x10]        @Write to INTC Config Register
MOV R2, #0x20             @Unmask INTC INT 69, Timer3 interrupt
STR R2, [R1, #0xC8]       @Write to INTC_MIR_CLEAR_2 register
MOV R2, #0x04             @Value to unmask INTC INT 98, GPIONT1A
STR R2, [R1, #0xE8]       @Write to INTC_MIR_CLEAR3 Register

@Turn on Timer3 CLK
MOV R2, #0x2              @Value to enable Timer3 CLK
LDR R1, =0x44E00084       @Addr of CM_PER_TIMER3_CLKCTRL
STR R2, [R1]              @Turn on
LDR R1, =0x44E0050C       @Addr of CLKSEL_TIMER3_CLK Register**
STR R2, [R1]              @Select 32 KHz CLK for Timer3

@Initiliaze Timer3 Registers, with count, overflow, interrupt generation
LDR R1, =0x48042000       @Base addr for timer3 registers
MOV R2, #0x1              @Value to reset timer3
STR R2, [R1, #0x10]       @Write to Timer3 CFG register
MOV R2, #0x2              @Value to enable overflow interrupt
STR R2, [R1, #0x2C]       @Write to timer3 IRQENABLE_SET
LDR R2, =0xFFFF0000       @Count value for 2 seconds
STR R2, [R1, #0x40]       @Timer3 TLDR load register (Reload value)
STR R2, [R1, #0x3C]       @Write to Timer3 TCRR count register

@Turn on GPIO_1 AUX Functional CLK, Enable DEBOUNCE on GPIO1_3 and Set Time
LDR R0,      =0x4804C000              @Base addr for GPIO1
LDR R1,      =0x44E000AC              @Addr of CM_PER_GPIO1_CLKCTRL
LDR R2,      =0x00040002              @Value to turn on Aux Funct CLK, bit 18 and
CLK
STR R2,      [R1]                     @Write value to CMP_PER_GPIO_CLKCTRL
ADD R1,      R0,    #0x0150                 @Addr of GPIO1_DEBOUNCABLE
MOV R2,      #0x00000008              @Load value of GPIO1 for bit 3
STR R2,      [R1]                     @Enable GPIO1_3 debounce
ADD R1,      R0,    #0x154            @Addr of GPIO1_DEBOUNCING TIME
MOV R2,      #0xA0                    @Value for 31 Micro-Seconds debounce interval
STR R2,      [R1]                     @Write to GPIO1_DEBOUNCING TIME

@Program GPIO1_21-24 as output
LDR R0, =0xFE1FFFFF             @Load word to program GPIO1_21-24 to output
LDR R1, =0x4804C134             @Addr of GPIO1_OE Register
LDR R2, [R1]                    @Read GPIO1_OE Register
```

```
AND R2, R2, R0                          @Modify word read in with R0
STR R2, [R1]                    @Write back to GPIO1_OE Register

@Make sure processor IRQ enabled in CPSR
MRS R3, CPSR                @Copy CPSR to R3
BIC R3, #0x80               @Clear bit 7
MSR CPSR_c, R3                  @Write back to CPSR

@Turn all LEDs off
MOV R0, #0x1E00000          @Load word to target GPIO1_21-24
LDR R1, =0x4804C190         @Load addr of GPIO1_CLEARDATAOUT
STR R0, [R1]                @Write to GPIO1_CLEARDATAOUT (This turns LED1-4 OFF)

@Wait for interrupt
WaitLoop: NOP
              B WaitLoop

INT_DIRECTOR:
              STMFD SP!,{R0-R4,LR}              @Push registers onto stack
              LDR R0,=0x482000F8                @Addr of INTC-PENDING_IRQ3
              LDR R1,[R0]                           @Read INTC-PENDING_IRQ3
              TST R1,#0x00000004                @Test bit 2
              BEQ TCHK                              @Not from GPIOINT1A, check if
Timer3, Else
              LDR R0,=0x4804C02C                @Load addr of GPIO1_IRQSTATUS_0
              LDR R1,[R0]                           @Read Status register to see if
button press
              TST R1,#0x00000008                @Check if bit 3 = 1
              BNE BUTTON_SVC                        @If bit 3 = 1 button is pressed
service it
              LDR R0,=0x48200048                @Else, Go back. INTC_CONTROL register
              MOV R1,#01                            @Value of clear bit 0
              STR R1,[R0]                           @Write to INTC_CONTROL register
              LDMFD SP!,{R0-R4,LR}              @Restore Registers
              SUBS PC,LR,#4                     @Pass execution to wait LOOP for now

TCHK:
              LDR R1,=0x482000D8                @Addr of INTC_PENDING_IRQ2 register
              LDR R0,[R1]                           @Read value
              TST R0,#0x20                      @Check if the interrupt from Timer3
(bit 5)
              BEQ PASS_ON                           @No return, Yes check overflow
              LDR R1,=0x48042028                @Addr of Timer3 IRQSTATUS register
              LDR R0,[R1]                           @Read Value
              TST R0,#0x2                           @Check bit 1
              BNE LEDFunction                       @If overflow, go LEDFunction

PASS_ON:
              MOV R1,#0x02                      @Value to turn Timer3 off
              LDR R0,=0x48042028                @Load addr of IRQSTATUS Timer3
              STR R1,[R0]                           @Write to IRQSTATUS Timer3

@turn off NEWIRQA bit in INTC_CONTROL, so processor can respond to new IRQ
              LDR R0,=0x48200048                @Addr of INTC_CONTROL register
              MOV R1,#01                            @Value to clear bit 0
```

```
            STR R1,[R0]                         @Write to INTC_CONTROL register

            LDMFD SP!,{R0-R4,LR}          @Restore Registers
            SUBS PC,LR,#4                 @Pass execution to wait LOOP for now

BUTTON_SVC:
            MOV R1,#0x00000008            @Value turns off GPIO1_3 Interrupt
Request
            STR R1,[R0]                         @Write to GPIO1_IRQSTATUS_0
register
            LDR R2,=LED_Flag              @Load pointer to LED_Flag
            LDR R3,[R2]                         @Load value from LED_Flag
            CMP R3,#0x00                  @Compare LED_Flag value to 0
            BEQ LEDFunction_ON            @Branch if equal go to LEDFunction_ON
            BNE LEDFunction_OFF           @Branch if not equal go to
LEDFunction_OFF

LEDFunction_OFF:
            MOV R4,#0x00                  @Value to change LED_Flag state & Turn
off Timer3
            STR R4,[R2]                         @Write to LED_Flag
            LDR R2,=0x48042038            @Load addr to DMTIMER3_TCLR
            STR R4,[R2]                         @Write to DMTIMER3_TCLR to turn
timer3 off
            LDR R2,=BUFFER                      @Load pointer to BUFFER
            LDR R5,=Current_State         @Load pointer to Current_State
            LDR R3,[R2]                         @Load value from BUFFER
            STR R3,[R5]                         @Store BUFFER value into
Current_State
            LDR R0,=0x4804C190            @Load addr of GPIO1_CLEARDATAOUT
            MOV R1,#0x01E00000            @Value to clear bits 24-21
            STR R1,[R0]                         @Write to GPIO1_CLEARDATAOUT
            B BACK                              @Branch to BACK

LEDFunction_ON:
            MOV R4,#0x01                  @Value to change LED_Flag state
            STR R4,[R2]                         @Write to LED_Flag
            LDR R0,=USRLEDCYCLE           @Load pointer to USRLEDCYCLE
            LDR R1,=Current_State         @Load pointer to Current_State
            LDR R2,[R1]                         @Load value from Current_State
into R2
            LDR R3,[R0,R2]                      @Add Current_State offset to
base addr in USRLEDCYCLE
            LDR R4,=0x4804C194            @Load GPIO1_SETDATAOUT addr
            STR R3,[R4]                         @Write to GPIO1_SETDATAOUTR
register
            MOV R3,#0x3                         @Load Value into auto realod and
start Timer3
            LDR R4,=0x48042038            @Load addr for Timer3 TCLR register
            STR R3,[R4]                         @Write to Timer3 TCLR register

BACK:
            LDR R0,=0x48200048            @Addr of INTC_CONTROL register
            MOV R1,#0x01                  @Value to enable new IRQ response in
INTC
```

```
            STR R1,[R0]                         @Write
            LDMFD SP!,{R0-R4,LR}            @Restore Registers
            SUBS PC,LR,#4                   @Return from IRQ interrupt

LEDFunction:
            LDR R1,=0x48042028              @Load addr of Timer3 IRQSTATUS register
            MOV R2,#0x2                         @Value to reset Timer3 Overflow
IRQ request
            STR R2,[R1]                         @Write
            LDR R0,=USRLEDCYCLE             @Load pointer to USRLEDCYCLE
            LDR R1,=BUFFER                      @Load pointer to BUFFER
            LDR R3,[R1]                         @Load value in BUFFER to
increment
            LDR R2,=0x4804C190              @Load addr of GPIO1_CLEARDATAOUT
            LDR R4,[R0,R3]                      @Load value of the sum of R3 and
R0
            STR R4,[R2]                         @Write to GPIO1_CLEARDATAOUT to
turn off LED
            CMP R3,#20                          @Compare value of R3 to 20
            MOVEQ R3,#0x00                      @If R3 = #20 reset value to 0
            ADDMI R3,R3,#04                     @If R3 > #20, increment R3 by
#04
            STR R3,[R1]                         @Store in BUFFER
            LDR R2,=0x4804C194              @Load addr of GPIO1_SETDATAOUT
            LDR R4,[R0,R3]                      @Load R4 with R3 + R0
            STR R4,[R2]                         @Write to GPIO1_SETDATAOUT to
turn on next LED

            LDR R0,=0x48200048              @Addr of INTC_CONTROL register
            MOV R1,#01                          @Value to clear bit 0
            STR R1,[R0]                         @Write to INTC_CONTROL register

            LDMFD SP!,{R0-R4,LR}            @Restore Registers
            SUBS PC,LR,#4                   @Pass execution to wait LOOP for now

.data
.align 2

USRLEDCYCLE: .word 0x01000000, 0x00800000, 0x00400000, 0x00200000, 0x00400000,
0x00800000
Current_State:  .word 0x0
BUFFER:             .word 0x0
LED_Flag:                   .word 0x0



STACK1:
      .rept 1024
      .word 0x0000
      .endr

STACK2:
      .rept 1024
      .word 0x0000
      .endr
```

```
.END
```

By signing this statement, I affirm that I did not give any help to any other person, did not receive any help from any other person, except TA and Instructor and did not obtain any information from the Internet or other sources.

Signature __Philip A Nevins_ 2/5/2023