Project 2

## Part 1 Task List

* Identify GPIO connection to USER LEDs on GPIO
* Enable/disable output to 4 BBB LEDs
* Create pattern using delay loop
* About 1 second per LED

### HIGH LEVEL

Setup necessary stack

Turn on GPIO1 by writing #02 to clock

Load ptr to base address of GPI01

Clear LED outputs with base address offset

Set GPIO pins as outputs using Read, Modify, Write

Read values of GPI01\_OE

Modify the values for appropriate LED bits

Write back modified value to GPIO1\_OE

Add offset to GPIO1 base address pointer for SETDATAOUT

Store address to turn on LED0

Call to TIMER Procedure

**TIMER:**

Store used register in the stack

Load value to register for necessary timer delay

Repeat

Subtract 1 from register with timer delay

Until Timer register = 0

Return back to mainline

Clear LED outputs with base address offset

Add offset to GPIO1 base address pointer for SETDATAOUT

Store address to turn on LED1

Call to TIMER Procedure

Clear LED outputs with base address offset

Add offset to GPIO1 base address pointer for SETDATAOUT

Store address to turn on LED2

Call to TIMER Procedure

Clear LED outputs with base address offset

Add offset to GPIO1 base address pointer for SETDATAOUT

Store address to turn on LED3

Call to TIMER Procedure

### LOW LEVEL

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GPI01 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  |  |  |  |  | L4 | L3 | L2 | L1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Function | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Hex | F | | | | E | | | | 1 | | | | F | | | | F | | | | F | | | | F | | | | F | | | |
| OUTY | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Hex | 0 | | | | 1 | | | | E | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | |

Setup stacks

Store 0x02 in CM\_PER\_GPIO1\_CLKCTRL at 0x44E000AC to enable GPIO1 clock

Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190 to set LED output to low

Store 0xFE1FFFFF in GPIO1\_OE at 0x4804C000+0x134 using **READ, MODIFY, WRITE** to enable LEDs

Repeat forever:

**LED0:**

Store 0x00200000 in GPIO1\_SETDATAOUT at 0x4804C000+0x194 to enable LED0

Call TIMER procedure

Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190 to set LED output to low

**LED1:**

Store 0x00400000 in GPIO1\_SETDATAOUT at 0x4804C000+0x194 to enable LED0

Call TIMER procedure

Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190 to set LED output to low

**LED2:**

Store 0x00800000 in GPIO1\_SETDATAOUT at 0x4804C000+0x194 to enable LED0

Call TIMER procedure

Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190 to set LED output to low

**LED3:**

Store 0x01000000 in GPIO1\_SETDATAOUT at 0x4804C000+0x194 to enable LED0

Call TIMER procedure

Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190 to set LED output to low

**TIMER:**

Store used registers on stack

Load 0x0009BC20 into register

Repeat

Subtract 1 from register, update flags

Check for zero flag, and repeat if not zero

Until 0x0009BC20 reaches zero

## PART 2 Task List

* Modify LED code to allow for button interrupts
* Identify offsets for GPIO2\_1
* Identify POINTERPEND and MIR for GPIO2
* Modify startup IRQ, add .global int\_director, add .extern int\_director
* Turn on Falling edge, turn on IRQ request generation for button
* Turn on IRQ in CPSR
* Create INT\_Director to redirect IRQ requests
* Create button service to control LEDs on/off cycle

### High Level

MAINLINE:

1. Setup regular, and IRQ interrupt stack
2. Enable GPIO1 and GPIO2 clocks
3. Set GPIO1\_24 – 21 for LED off (low)
4. Set GPIO1\_24-21 as outputs
5. Setup falling edge detect for GPIO2\_1 and interrupt generation
6. Unmask bit 1 on MIR\_CLEAR1 to allow GPIOINT2A
7. Enable IRQ input by clearing bit 7 in CPSR

INT\_DIRECTOR:

1. Save register
2. Test bit 0 at IRQ1 for button
3. IF NO, then PASS\_ON
4. If YES, then check Bit 1 of GPIO2\_IRQSTATUS\_0
5. If YES, then BUTTON\_SVC, ELSE
6. PASS\_ON

PASS\_ON

1. Restore register saved in INT\_DIRECTOR
2. SUBS PC, LR, #4 outta this shizz

BUTTON\_SVC

1. Turn off GPIO2\_1 interrupt
2. Enable INTC for new interrupt
3. Test LEDSTATUS memory to see if LEDs should be on
4. If ON, Turn off by writing bit to memory
5. If OFF, Turn on by writing bit to memory

IDLE:

1. Check LEDSTATUS in memory to see if LEDs should be on
2. If yes, then run LEDLOOP
3. If NO, then run IDLE

LEDLOOP

1. Run LED1
2. Run LED2
3. Run LED3
4. Run LED4
5. Run IDLE

LED1

Store Register of stack

Store address to turn on LED1

Call to TIMER Procedure

Clear LED outputs with base address offset

Return registers

-------------------------------REPEAT FOR LEDS 1-4 ------------------------------------------

TIMER

Store used register in the stack

Load value to register for necessary timer delay

Repeat

Subtract 1 from register with timer delay

Until Timer register = 0

Return back to Mainline

### LOW LEVEL

**GPIO2\_1 BIT**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| GPI02 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTY | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| Hex | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 2 | | | |

**MIR1**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MIR1 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTY | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Hex | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 1 | | | |

MAINLINE:

1. Setup regular, and IRQ interrupt stack
2. Enable GPIO1 and GPIO2 clocks
   1. Store 0x02 in CM\_PER\_GPIO1\_CLKCTRL at 0x44E000AC to enable GPIO1 clock
   2. Store 0x02 in CM\_PER\_GPIO2\_CLKCTRL at 0x44E000B0 to enable GPIO2 clock
3. Set GPIO1\_24 – 21 for LED off (low)
   1. Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190
4. Set GPIO1\_24-21 as outputs
   1. Store 0xFE1FFFFF in GPIO1\_OE at 0x4804C000+0x134 using **READ, MODIFY, WRITE**
5. Setup falling edge detect for GPIO2\_1 and interrupt generation
   1. Store 0x02 in GPIO2\_FALLINGDETECT at 0x4814C000+0x14C using **READ, MODIFY, WRITE**
   2. Store 0x02 in GPIO2\_IRQSTATUS\_SET\_0 at 0x4814C000+0x034
6. Unmask bit 1 on MIR\_CLEAR1 to allow GPIOINT2A
   1. Store 0x01 in INTC\_MIR\_CLEAR1 at 0x482000A8
7. Enable IRQ input by clearing bit 7 in CPSR
   1. BIC #0x80

INT\_DIRECTOR:

1. Save register
2. Test bit 0 at IRQ1 for button
   1. Test (0x01) in INTC\_PENDING\_IRQ1 at 0x482000B8
3. IF NO, then PASS\_ON
4. If YES, then check Bit 1 of GPIO2\_IRQSTATUS\_0
   1. Test (0x02) in GPIO2\_IRQSTATUS\_0 at 0x481AC02C
5. If YES, then BUTTON\_SVC, ELSE
6. PASS\_ON

BUTTON\_SVC

1. Turn off GPIO2\_1 interrupt
   1. Store 0x02 in GPIO2\_IRQSTATUS\_0 at 0x481AC02C
2. Enable INTC for new interrupt
   1. Store 0x01 in INTC\_CONTROL at 0x48200048
3. Test LEDSTATUS memory to see if LEDs should be on
   1. Test bit 0 of =LEDSTATUS
4. If ON, Turn off by writing bit to memory
   1. Store 0x00 in =LEDSTATUS
5. If OFF, Turn on by writing bit to memory
   1. Store 0x01 in =LEDSTATUS

PASS\_ON

1. Restore register saved in INT\_DIRECTOR
2. SUBS PC, LR, #4 outta this shizz

IDLE:

1. Check LEDSTATUS in memory to see if LEDs should be on
   1. Test Bit 0 of =LEDSTATUS
2. If yes, then run LEDLOOP
3. If NO, then run IDLE

LEDLOOP:

1. Run LED1
2. Run LED2
3. Run LED3
4. Run LED4
5. Run IDLE

LED1

1. Store Register of stack
2. Store address to turn on LED1
   1. Store 0x00200000 in GPIO1\_SETDATAOUT at 0x4804C000+0x194 to enable LED0
3. Call to TIMER Procedure
4. Clear LED outputs with base address offset
   1. Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190
5. Return registers

------------------------------REPEAT FOR LEDS 1-4 (SEE PART 1 FOR BITS) ----------------------------------------

TIMER

1. Store used register in the stack
2. Load value to register for necessary timer delay (0x0009BC20)
3. Repeat
   1. Subtract 1 from register with timer delay
4. Until Timer register = 0
5. Return back to Mainline

## PART 3 Task List

* Find a way to track which LEDs should be on for Timer
* Introduce Timer interrupts ever 1 second
* Keep button press functionality

### High Level

MAINLINE:

1. Setup regular, and IRQ interrupt stack
2. Enable GPIO1, GPIO2 clocks and
3. Set GPIO1\_24 – 21 for LED off (low)
4. Set GPIO1\_24-21 as outputs
5. Setup falling edge detect for GPIO2\_1 and interrupt generation
6. Unmask bit 1 on MIR\_CLEAR1 to allow GPIOINT2A, Unmask Bit 28 for DMTIMER4
7. Enable TIMER4 CLK and set multiplexer for 32kHz clock
8. Initialize timer register for desired count and overflow
9. Enable IRQ input by clearing bit 7 in CPSR

INT\_DIRECTOR:

1. Save register
2. Test bit 0 at IRQ1 for button
3. IF NO, then TCHK
4. If YES, then check Bit 1 of GPIO2\_IRQSTATUS\_0
5. If YES, then BUTTON\_SVC, ELSE
6. PASS\_ON

TCHK

1. Test bit 28 of INTC
2. If NO, then PASS\_ON
3. If YES, then check bit 1 of TIMER4 overflow
4. If NO, then PASS\_ON
5. If YES, reset TIMER4\_IRQ trigger
6. Branch to LED

BUTTON\_SVC

1. Turn off GPIO2\_1 interrupt
2. Load ptr to =LEDSTATUS
3. Test bit 4 of LEDSTATUS
4. IF ON, then turn OFF
5. IF OFF, then turn ON
6. Store new value into =LEDSTATUS
7. Turn on auto reload for timer and start timer
8. Branch LED

LED

1. Load pointer to =LEDSTATUS
2. Test bit 4 of LEDSTATUS
3. IF NO, then PASS ON
4. IF YES, then fall through to BIT0

BIT0:

1. Test bit 0
2. If NO, then jump to BIT1
3. If YES, add 0x01 to =LEDSTATUS
4. Clear LED output
5. Turn on LED2
6. PASS\_ON

BIT1:

1. Test bit 0
2. If NO, then jump to BIT2
3. If YES, add 0x02 to =LEDSTATUS
4. Clear LED output
5. Turn on LED3
6. PASS\_ON

BIT2:

1. Test bit 0
2. If NO, then jump to BIT2
3. If YES, add 0x04 to =LEDSTATUS
4. Clear LED output
5. Turn on LED4
6. PASS\_ON

BIT3:

1. Store #0x11 in LEDSTATUS to turn on LEDs and LED1
2. Clear LED output
3. Turn on LED1
4. PASS\_ON

### LOW LEVEL

**MIR2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MIR1 | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OUTY | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Hex | 1 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 0 | | | | 1 | | | |

|  |  |  |
| --- | --- | --- |
|  | GPIO1 Bit Location | LEDSTATUS Bit Tracking (ON) |
| **LED1** | 0x00200000 | 0x00000001 |
| **LED2** | 0x00400000 | 0x00000002 |
| **LED3** | 0x00800000 | 0x00000004 |
| **LED4** | 0x01000000 | 0x00000008 |
| **LED ON/OFF** | -------------- | 0x0000001X |

MAINLINE:

1. Setup regular, and IRQ interrupt stack
2. Enable GPIO1, GPIO2 clocks
   1. Store 0x02 in CM\_PER\_GPIO1\_CLKCTRL at 0x44E000AC to enable GPIO1 clock
   2. Store 0x02 in CM\_PER\_GPIO2\_CLKCTRL at 0x44E000B0 to enable GPIO2 clock
3. Set GPIO1\_24 – 21 for LED off (low)
   1. Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190
4. Set GPIO1\_24-21 as outputs
   1. Store 0xFE1FFFFF in GPIO1\_OE at 0x4804C000+0x134 using **READ, MODIFY, WRITE**
5. Setup falling edge detect for GPIO2\_1 and interrupt generation
   1. Store 0x02 in GPIO2\_FALLINGDETECT at 0x4814C000+0x14C using **READ, MODIFY, WRITE**
6. Unmask bit 1 on MIR\_CLEAR1 to allow GPIOINT2A, Unmask Bit 28 for DMTIMER4
   1. Store 0x01 in INTC\_MIR\_CLEAR1 at 0x482000A8
7. Enable TIMER4 CLK and set multiplexer for 32kHz clock
   1. Store 0x02 in PRCMCLKSEL\_TIMER4 at 0x44E00000 +0x510
8. Initialize timer register for desired count and overflow
   1. Store 0x01 to Timer 4 CFG at 0x48044000+0x010
   2. Store 0x02 to Timer 4 IRQ Enable at 0x48044000 + 0x02C
   3. Store 0xFFFF8000 in Timer 4 TLDR at 0x48044000 + 0x040
   4. Store 0xFFFF8000 in Timer 4 TCRR at 0x48044000 + 0x03C
9. Enable IRQ input by clearing bit 7 in CPSR
   1. BIC #0x80

INT\_DIRECTOR:

1. Save register
2. Test bit 0 at IRQ1 for button
   1. Test (0x01) in INTC\_PENDING\_IRQ1 at 0x482000B8
3. IF NO, then TCHK
4. If YES, then check Bit 1 of GPIO2\_IRQSTATUS\_0
   1. Test (0x02) in GPIO2\_IRQSTATUS\_0 at 0x481AC02C
5. If YES, then BUTTON\_SVC, ELSE
6. PASS\_ON

PASS\_ON

1. Clear INTC\_CONTROL for new IRQ requests
2. Restore register saved in INT\_DIRECTOR
3. SUBS PC, LR, #4 outta this shizz

TCHK

1. Test bit 28 of INTC
   1. Test (0x10000000) of INTC\_PENDING\_IRQ2 at 0x482000D8
2. If NO, then PASS\_ON
3. If YES, then check bit 1 of TIMER4 overflow
   1. Test (0x02) in Timer4\_IRQStatus at 0x48044000+0x028
4. If NO, then PASS\_ON
5. If YES, reset TIMER4\_IRQ trigger
   1. Store #0x02 in Timer4\_IRQStatus at 0x48044000+0x028
6. Branch to LED

LED

1. Load pointer to =LEDSTATUS
2. Test bit 4 of LEDSTATUS
   1. Test 0x10 of =LEDSTATUS
3. IF NO, then PASS ON
4. IF YES, then fall through to BIT0

BIT0:

1. Test bit 0
   1. Test 0x01 of =LEDSTATUS
2. If NO, then jump to BIT1
3. If YES, add 0x01 to =LEDSTATUS
4. Clear LED output
   1. Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190
5. Turn on LED2
   1. Store 0x0040000 in GPIO1\_SETDATAOUT at 0x04804C000+0x194
6. PASS\_ON

BIT1:

1. Test bit 0
2. If NO, then jump to BIT2
3. If YES, add 0x02 to =LEDSTATUS
4. Clear LED output
   1. Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190
5. Turn on LED3
   1. Store 0x0080000 in GPIO1\_SETDATAOUT at 0x04804C000+0x194
6. PASS\_ON

BIT2:

1. Test bit 0
2. If NO, then jump to BIT2
3. If YES, add 0x04 to =LEDSTATUS
4. Clear LED output
   1. Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190
5. Turn on LED4
   1. Store 0x0100000 in GPIO1\_SETDATAOUT at 0x04804C000+0x194
6. PASS\_ON

BIT3:

1. Store #0x11 in LEDSTATUS to turn on LEDs and LED1
2. Clear LED output
   1. Store 0x01E00000 in GPIO1\_CLEARDATAOUT at 0x4804C000+0x190
3. Turn on LED1
   1. Store 0x0020000 in GPIO1\_SETDATAOUT at 0x04804C000+0x194
4. PASS\_ON