EE-712 Embedded system design

Supporting document for Assembly programming

January 5, 2016

1 Register configurations for GPIO Ports:

There are two on-chip buses that connect the core to the peripherals. The Advanced Peripheral Bus (APB) bus is the legacy bus. The Advanced High-Performance Bus (AHB) bus provides better back-to-back access performance than the APB bus. APB can be used for this Labwork.

Following registers have to be configured to use a port as GPIO.

1. **GPIO Data (GPIODATA):** The GPIODATA register is the data register. In software control mode, values written in the GPIODATA register are transferred onto the GPIO port pins if the respective pins have been configured as outputs through the GPIO Direction (GPIODIR) register. The GPIO ports allow for the modification of individual bits in the GPIO Data (GPIODATA) register by using bits [9:2] of the address bus as a mask.

   For example, GPIODATA Port F (APB) base: 0x40025000. To use all bits in this port we should unmask bits 9 to 2 of the address bus thus making the GPIODATA Port F address as 0x400253FC.

2. **GPIO Direction (GPIODIR):** The GPIODIR register is the data direction register. Setting a bit in the GPIODIR register configures the corresponding pin to be an output, while clearing a bit configures the corresponding pin to be an input. Port pins 7 to 0 are mapped to 7:0 of this register.

   For example, GPIO_DIR register address for Port F(APB) is 0x40025400(with offset). So, to configure pins 0, 4 as inputs and remaining as outputs we need to send #0x0E to the above address. Likewise, other port's DIR register addresses can be found on page 661 of *tm4c123gh6pm.pdf*

3. **GPIO Digital Enable (GPIODEN):** To use the pin as a digital input or output, the corresponding GPIODEN bit must be set. Port pins 7 to 0

are mapped to 7:0 of this register. All GPIO DEN register addresses can be found on page 681 of above mentioned pdf.

4. **GPIO Analog Mode Select (GPIOAMSEL):** Port pins 7 to 0 are mapped to 7:0 of this register. To disable analog function of the pin, enable isolation, and to make the pin capable of digital functions as specified, 0 has to be written to it. All GPIO AMSEL register addresses can be found on page 685 of above mentioned pdf.

5. **GPIO Alternate Function Select (GPIOAFSEL):** The GPIOAFSEL register is the mode control select register. If a bit is clear, the pin is used as a GPIO and is controlled by the GPIO registers. Setting a bit in this register configures the corresponding GPIO line to be controlled by an associated peripheral.
   Port pins 7 to 0 are mapped to 7:0 of this register. In this lab assignment, we will use only GPIO functionality. So, 0 has to be send to the corresponding pins. All GPIO AMSEL register addresses can be found on page 669 of above mentioned pdf.

6. **GPIO Port Control (GPIOPCTL):** The GPIOPCTL register is used in conjunction with the GPIOAFSEL register and selects the specific peripheral signal for each GPIO pin when using the alternate function mode. Most bits in the GPIOAFSEL register are cleared on reset, therefore most GPIO pins are configured as GPIOs by default. All GPIO PCTL register addresses can be found on page 687 of above mentioned pdf.

   Each pin is assigned with 4 bits in this register. Bits 31 to 28 correspond to pin7, bits 27 to 24 are to pin6 and so on. So, to use a pin as GPIO, 4 bits associated to it are to be cleared.

7. **GPIO Run Mode Clock Gating Control (RCGCGPIO):** The RCGCGPIO register provides software the capability to enable and disable GPIO modules in Run mode. When enabled, a module is provided a clock and accesses to module registers are allowed. When disabled, the clock is disabled to save power and accesses to module registers generate a bus fault.

   Address of RCGCGPIO register is 0x400FE608. Bits 0 to 5 are mapped to ports A to F. 1 to corresponding port bit enables clock for that port. Note that each GPIO module clock must be enabled before the registers can be programmed. There must be a delay of 3 system clocks after the GPIO module clock is enabled before any GPIO module registers are accessed.

8. **GPIO Lock (GPIOLOCK) & GPIO Commit (GPIOCR) :** The purpose of GPIOCR is to provide a layer of protection against accidental

programming of critical hardware signals including the GPIO pins that can function as JTAG/SWD signals and the NMI signal. In order to use a pin as an input pin, we have to unlock this pin. Writing 0x4C4F434B to the GPIOLOCK register unlocks the GPIOCR register.
For example, PORT-F has a base address of 0x40025000 and GPIOLOCK has an offset 0x520. Thus we have to write 0x4C4F434B at location 0x40025520. The GPIOLOCK register enables write access to the GPIOCR register. GPIOCR has an offset of 0x524 and we have to write 1 in the last bit of this register, i.e. at location 0x40025524, to unlock PF0.

9. **GPIO Pull-Up Select (GPIOPUR):** The GPIOPUR register is the pull-up control register. When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled. Write access to this register is protected with the GPIOCR register.
   Bits 7:0 of this register are mapped to port pins 7 to 0. When using a port pin as input, it has to be configured with pull-ups.

2 Setting up Assembly project in CCS:

1. Go to Project → New CCS project. Select all the information same as mentioned in **CCS Installation.pdf**(only for this window).
2. After the project is created, Delete main.c file from the project.
3. Right click on the project→Show Build Settings→Advanced Options→Assembler Options
   Check→ Keep the Generated Assembly language file
   Check→ Generate listing file
4. Right click on the project→ New→ File t→ Give filename as filename.asm
   Write your code in the asm file.
5. Build the project
6. Load .out file to the target to see the output.

3 Sample code with comments

Below program makes different color LEDs on based on the switches SW1, SW2. Switches and LEDs are connected to port F.

```
 .thumb
        .text
        .align  2
GPIO_PORTF_DATA_R   .field  0x400253FC,32
GPIO_PORTF_DIR_R    .field  0x40025400,32
GPIO_PORTF_AFSEL_R  .field  0x40025420,32
```

```
GPIO_PORTF_PUR_R    .field 0x40025510,32
GPIO_PORTF_DEN_R    .field 0x4002551C,32
GPIO_PORTF_LOCK_R   .field 0x40025520,32
GPIO_PORTF_CR_R     .field 0x40025524,32
GPIO_PORTF_AMSEL_R  .field 0x40025528,32
GPIO_PORTF_PCTL_R   .field 0x4002552C,32
GPIO_LOCK_KEY       .field 0x4C4F434B,32   ; Unlocks the GPIO_CR register
RED         .equ 0x02
BLUE        .equ 0x04
GREEN       .equ 0x08
SW1         .equ 0x10                      ; on the left side of the Launchpad board
SW2         .equ 0x01                      ; on the right side of the Launchpad board
SYSCTL_RCGCGPIO_R   .field   0x400FE608,32

       .global main

main:   .asmfunc
    BL   PortF_Init                        ; initialize input and output pins of Port F
loop
    LDR R0, FIFTHSEC                       ; R0 = FIFTHSEC (delay 0.2 second)
    BL   delay                             ; delay at least (3*R0) cycles
    BL   PortF_Input                       ; read all of the switches on Port F
    CMP R0, #0x01                          ; R0 == 0x01?
    BEQ sw1pressed                         ; if so, switch 1 pressed
    CMP R0, #0x10                          ; R0 == 0x10?
    BEQ sw2pressed                         ; if so, switch 2 pressed
    CMP R0, #0x00                          ; R0 == 0x00?
    BEQ bothpressed                        ; if so, both switches pressed
    CMP R0, #0x11                          ; R0 == 0x11?
    BEQ nopressed                          ; if so, neither switch pressed
                                           ; if none of the above, unexpected return va
    MOV R0, #(RED+GREEN+BLUE)              ; R0 = (RED|GREEN|BLUE) (all LEDs on)
    BL   PortF_Output                      ; turn all of the LEDs on
    B    loop
sw1pressed
    MOV R0, #BLUE                          ; R0 = BLUE (blue LED on)
    BL   PortF_Output                      ; turn the blue LED on
    B    loop
sw2pressed
    MOV R0, #RED                           ; R0 = RED (red LED on)
    BL   PortF_Output                      ; turn the red LED on
    B    loop
bothpressed
    MOV R0, #GREEN                         ; R0 = GREEN (green LED on)
    BL   PortF_Output                      ; turn the green LED on
    B    loop
```

```
nopressed
    MOV R0, #0                       ; R0 = 0 (no LEDs on)
    BL  PortF_Output                 ; turn all of the LEDs off
    B   loop
    .endasmfunc
;------------delay------------
; Delay function for testing, which delays about 3*count cycles.
; Input: R0  count
; Output: none
ONESEC              .field 5333333,32      ; approximately 1s delay at ~16 MHz cl
QUARTERSEC          .field 1333333,32      ; approximately 0.25s delay at ~16 MHz
FIFTHSEC            .field 1066666,32      ; approximately 0.2s delay at ~16 MHz
delay:  .asmfunc
    SUBS R0, R0, #1                  ; R0 = R0 - 1 (count = count - 1)
    BNE delay                        ; if count (R0) != 0, skip to 'delay'
    BX  LR                           ; return
    .endasmfunc

;------------PortF_Init------------
; Initialize GPIO Port F for negative logic switches on PF0 and
; PF4 as the Launchpad is wired.  Weak internal pull-up
; resistors are enabled, and the NMI functionality on PF0 is
; disabled.  Make the RGB LED's pins outputs.
; Input: none
; Output: none
; Modifies: R0, R1, R2
PortF_Init:  .asmfunc
    LDR R1, SYSCTL_RCGCGPIO_R        ; 1) activate clock for Port F
    LDR R0, [R1]
    ORR R0, R0, #0x20                ; set bit 5 to turn on clock
    STR R0, [R1]
    NOP
    NOP                              ; allow time for clock to finish
    LDR R1, GPIO_PORTF_LOCK_R        ; 2) unlock the lock register
    LDR R0, GPIO_LOCK_KEY              ; unlock GPIO Port F Commit Register
    STR R0, [R1]
    LDR R1, GPIO_PORTF_CR_R          ; enable commit for Port F
    MOV R0, #0xFF                    ; 1 means allow access
    STR R0, [R1]
    LDR R1, GPIO_PORTF_AMSEL_R       ; 3) disable analog functionality
    MOV R0, #0                       ; 0 means analog is off
    STR R0, [R1]
    LDR R1, GPIO_PORTF_PCTL_R        ; 4) configure as GPIO
    MOV R0, #0x00000000              ; 0 means configure Port F as GPIO
    STR R0, [R1]
    LDR R1, GPIO_PORTF_DIR_R         ; 5) set direction register
    MOV R0,#0x0E                      ; PF0 and PF7-4 input, PF3-1 output
    STR R0, [R1]
    LDR R1, GPIO_PORTF_AFSEL_R        ; 6) regular port function
    MOV R0, #0                        ; 0 means disable alternate function
    STR R0, [R1]
    LDR R1, GPIO_PORTF_PUR_R          ; pull-up resistors for PF4,PF0
    MOV R0, #0x11                     ; enable weak pull-up on PF0 and PF4
    STR R0, [R1]
    LDR R1, GPIO_PORTF_DEN_R          ; 7) enable Port F digital port
    MOV R0, #0xFF                     ; 1 means enable digital I/O
    STR R0, [R1]
    BX  LR
    .endasmfunc


;------------PortF_Input------------
; Read and return the status of the switches.
; Input: none
; Output: R0  0x01 if only Switch 1 is pressed
;         R0  0x10 if only Switch 2 is pressed
;         R0  0x00 if both switches are pressed
;         R0  0x11 if no switches are pressed
; Modifies: R1
PortF_Input:  .asmfunc
    LDR R1, GPIO_PORTF_DATA_R  ; pointer to Port F data
    LDR R0, [R1]               ; read all of Port F
    AND R0,R0,#0x11            ; just the input pins PF0 and PF4
    BX  LR                     ; return R0 with inputs
    .endasmfunc

;------------PortF_Output------------
; Set the output state of PF3-1.
; Input: R0  new state of PF
; Output: none
; Modifies: R1
PortF_Output:  .asmfunc
    LDR R1, GPIO_PORTF_DATA_R  ; pointer to Port F data
    STR R0, [R1]               ; write to PF3-1
    BX  LR
    .endasmfunc

    .end                               ; end of file
```

4 References

1. *tm4c123gh6pm.pdf*
2. TivaC examples by Valvano*Examples*