Zeynepnur ŞAHİNEL
2305399 09.01.2021

EE447 EXPERIMENT 4 PRELIMINARY WORK

Part 1

In which mode is TIMER0A used?

Periodic Mode. TAMR is set to 0x02. (Count Down)

Calculation of low and high values:

In the PULSE_INIT, clk is divided to 16 through TAPR. Therefore it becomes 1 Mhz meaning timer is increased by each 1 microseconds. To achieve 20 kHz frequency, 50 microsec should be provided.

50 usec/1 usec gives 50 as a period. (in decimal) Therefore, HIGH value should 50/5=10 (A in hex) and LOW value should be 10*4=40 (28 in hex)

```
;-------------------------------------------------
LOW                             EQU   0x28 ;40
HIGH                            EQU   0x0A ;10
;-------------------------------------------------
```

Figure 1. Low and High values to ensure 20% duty cycle

```
                        AREA MAIN, CODE, READONLY, ALIGN=2
                        THUMB
                        EXTERN      PULSE_INIT
                        EXPORT __main

__main          PROC

                        BL PULSE_INIT

                        MOV R6, #0

Final           B   Final

                        ENDP

                        ALIGN
                        END
```

Figure 2. Main Code for Part 1

```
;-------------------------------------------------
My_Timer0A_Handler          PROC

                        LDR R1,=TIMER0_TAILR
                        MOV R0, #0
                        LDR R2,=GPIO_PORTF_DATA
```

```
                    MOV R3,#0

                    CMP R6, #0
                    BEQ High
                    BNE  Low

High                MOV R0, #HIGH
                    MOV R6, #1
                    MOV R3, #0x04
                    B    final

Low                 MOV R0, #LOW
                    MOV R6, #0
                    MOV R3, #0
                    B    final


final           STR R0, [R1]
                STR R3, [R2]
                LDR R1,=TIMER0_ICR
                MOV R0, #0x01
                STR  R0, [R1]


                    BX   LR
                    ENDP
;-------------------------------------------------
```

Figure 3. My_Timer0A_Handler ISR code

Since 20kHz can not be observed with naked eye, Debug is used to observation. Below Figure shows the TAILR register value for 20% cycle.

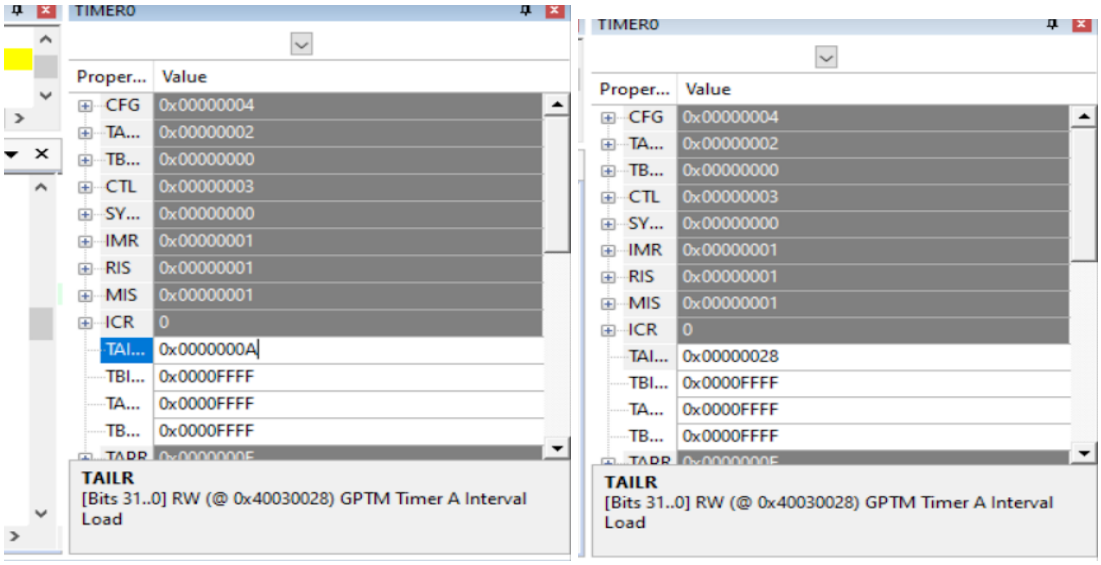

Figure 4. Timer Register values of TIMER0A

Zeynepnur ŞAHİNEL
2305399 09.01.2021

Part 2

In this part, pulse.s is included directly to the code. Then another Timer is created, for this purpose TIMER3 is chosen, which is created thorugh PB2. Pb2 is assigned as input and it takes values of pulse train to achieve this PF2 is connected with a jumper to PB2. Moreover, Convrt.s (from Exp1) and OutStr.s subroutines added to the code

```
;Timer_Init.s
; 16/31 Timer Registers
TIMER3_CFG EQU 0x40033000
TIMER3_TAMR EQU 0x40033004
TIMER3_CTL EQU 0x4003300C
TIMER3_IMR EQU 0x40033018
TIMER3_RIS EQU 0x4003301C ; Timer Interrupt Status
TIMER3_ICR EQU 0x40033024 ; Timer Interrupt Clear
TIMER3_TAILR EQU 0x40033028 ; Timer interval
TIMER3_TAPR EQU 0x40033038
TIMER3_TAR EQU 0x40033048 ; Timer register

;GPIO Registers B ile degistir!
GPIO_PORTB_DATA EQU 0x40005010 ; Access BIT2
GPIO_PORTB_DIR EQU 0x40005400 ; Port Direction
GPIO_PORTB_AFSEL EQU 0x40005420 ; Alt Function enable
GPIO_PORTB_DEN EQU 0x4000551C ; Digital Enable
GPIO_PORTB_AMSEL EQU 0x40005528 ; Analog enable
GPIO_PORTB_PCTL EQU 0x4000552C ; Alternate Functions

; System Registers
SYSCTL_RCGCGPIO EQU 0x400FE608 ; GPIO Gate Control
SYSCTL_RCGCTIMER EQU 0x400FE604 ; GPTM Gate Control
COUNTER_MAX_VAL EQU          0xFFFFFFFF

                                AREA Timer3_init, CODE, READONLY
                                THUMB
                                EXPORT Timer_init


Timer_init  PROC
                    LDR R1 ,=SYSCTL_RCGCGPIO ; start GPIO clock
                    LDR R0 , [R1]
                    ORR R0 , R0 , #0x02 ; set bit [1] for port B
                    STR R0 , [R1]
                    NOP ; allow clock to settle
                    NOP
                    NOP

                    LDR R1 , =GPIO_PORTB_DIR ; set direction of PB2
                    LDR R0 , [R1]
                    BIC R0 , R0 , #0x04 ; clear bit 2 for input
```

```
STR R0 , [R1]

LDR R1 , =GPIO_PORTB_AFSEL ; regular port function
LDR R0 , [R1]
ORR R0 , R0 , #0x04
STR R0 , [R1]

LDR R1 , =GPIO_PORTB_PCTL ; Timer3A alternate func select
LDR R0 , [R1]
ORR R0 , R0 , #0x00000700
STR R0 , [R1]

LDR R1 , =GPIO_PORTB_AMSEL ; disable analog
MOV R0 , #0
STR R0 , [R1]

LDR R1 , =GPIO_PORTB_DEN ; enable port digital
LDR R0 , [R1]
ORR R0 , R0 , #0x04
STR R0 , [R1]

LDR R1 , =SYSCTL_RCGCTIMER ; Start Timer3
LDR R2 , [R1]
ORR R2 , R2 , #0x08 ;1000
STR R2 , [R1]
NOP ; alow clock to settle
NOP
NOP

LDR R1 , =TIMER3_CTL ; disable timer during setup
BIC R2 , R2 , #0x08
STR R2 , [R1]

LDR R1 , =TIMER3_CFG ; set 16 bit mode
MOV R2 , #0x04
STR R2 , [R1]

LDR R1 , =TIMER3_TAMR
MOV R2 , #0x17 ;
STR R2 , [R1]

LDR R1 , =TIMER3_CTL ; initialize match clocks
LDR R2,[R1]
ORR R2,R2,#0x0C
STR R2 , [R1]

LDR R1 , =TIMER3_TAILR
MOV R2 ,#COUNTER_MAX_VAL
STR R2 , [R1]
```

```
                    LDR R1 , =TIMER3_TAPR
                    MOV R2 , #15 ; divide clock by 16 to
                    STR R2 , [R1] ; get 1 us clocks

                    LDR R1 , =TIMER3_IMR ; disable timeout interrupt
                    MOV R2 , #0x00
                    STR R2 , [R1]
; Enable timer
                    LDR R1 , =TIMER3_CTL
                    LDR R2 , [ R1 ]
                    ORR R2 , R2 , #0x03 ; set bit 0 to enable
                    STR R2 , [ R1 ] ;
                    BX LR ; return
                    ENDP
```

Figure 5. Timer_Init.s code

```
;*****************************************************************
; Program_Directives.s
;*****************************************************************
LOW EQU 0x00000028 ;
FIRST EQU 0x20000400
; 16/31 Timer Registers
TIMER3_CFG EQU 0x40033000
TIMER3_TAMR EQU 0x40033004
TIMER3_CTL EQU 0x4003300C
TIMER3_IMR EQU 0x40033018
TIMER3_RIS EQU 0x4003301C ; Timer Interrupt Status
TIMER3_ICR EQU 0x40033024 ; Timer Interrupt Clear
TIMER3_TAILR EQU 0x40033028 ; Timer interval
TIMER3_TAPR EQU 0x40033038
TIMER3_TAR EQU 0x40033048 ; Timer register

;GPIO Registers B access
GPIO_PORTB_DATA EQU 0x40005010 ; Access BIT2
;*****************************************************************
; Program section
;*****************************************************************
;LABEL              DIRECTIVE  VALUE                COMMENT
                    AREA       main, READONLY, CODE
                    THUMB
                    EXTERN          PULSE_INIT
                    EXTERN          Timer_init
                    EXTERN          OutStr
                    EXTERN          Convrt
                    EXPORT     __main; Make available

__main         PROC
```

```
                BL PULSE_INIT
                BL Timer_init
                MOV R8,#LOW


start           MOV R4,#0x00            ;counter for edges

read_flag   LDR R1,=TIMER3_RIS      ;polling for RIS
                LDR R0,[R1]
                STR R0,[R4]
                CMP R0,#0x04            ;if capture flag is high
                BNE read_flag
                BEQ flag_high           ;go to flag_high

flag_high
                ADDS R4,#1              ;add counter by 1
                LDR R1,=TIMER3_ICR      ;clear capture flag
                MOV R0,#0xFF
                STR R0,[R1]

                LDR R1,=TIMER3_TAR  ;loading current TAR value
                LDR R2,[R1]

                CMP R4,#1
                BEQ first_val       ;first edge value

                CMP R4,#2
                BEQ second_val          ;second edge value

                CMP R4,#3
                BEQ third_val       ;third edge value

                CMP R4,#4               ;when counter is 4 go to exit
                BEQ display

first_val  MOV R12,R2                   ;copying TAR value at the first edge
            B read_flag

second_val MOV R6,R2                    ;copying TAR value at the second edge
            B read_flag

third_val MOV R10,R2                    ;copying TAR value at the third edge
            B read_flag

display
                SUB R6,R12              ; led is high i.e. pulse width
                LDR R5,=FIRST
                MOV R4,R6
                BL Convrt
```

```
                    BL OutStr

                    SUB R11,R10,R12     ;R11 holds the period value
                    LDR R5,=FIRST
                    MOV R4,R11
                    BL Convrt
                    BL OutStr

                    MOV R0,#100
                    MUL R6,R6,R0
                    UDIV R6,R11                 ; duty cycle
                    LDR R5,=FIRST
                    MOV R4,R6
                    BL Convrt
                    BL OutStr

                    B exit                      ;to increase readability

exit            NOP

                    B start

                    ENDP
;*****************************************************************
; End of the program  section
;*****************************************************************
;LABEL     DIRECTIVE      VALUE                    COMMENT
                    ALIGN
                    END
```

Figure 6. Program Directive Code

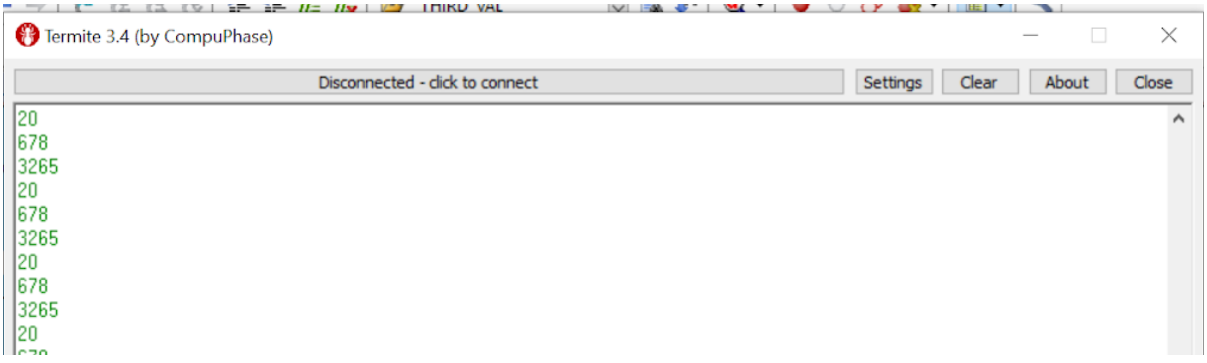

Figure 7. Termite Output

In Fig. 7 pulse width is 678, period is 3265 and the duty cycle is 20.
678 is calculated by substracting the second edge TAR value from the first edge TAR value.
3265 is calculated by sybstracting the thirs edge TAR value from the first edge TAR value.

Zeynepnur ŞAHİNEL
2305399 09.01.2021

Duty cycle is calculated by dividing the pwm cycle to period, and multipled with 100. Here, I couln't convert the pulse width and period values into seconds. Termit prints only the TAR current value differences.

```
;*****************************************************************
; Convrt.s
; Def: Converts max 32-bit hex number's decimal equivalent to ascii
; Writes the ascii result sstarting from [R5] address
;*****************************************************************
; Constants

ASCII       EQU             0x030

;*****************************************************************
; Program section
;*****************************************************************
;LABEL          DIRECTIVE  VALUE            COMMENT
               AREA convert , READONLY, CODE
               THUMB
               EXPORT Convrt ; Reference external subroutine

Convrt         PROC
               ;LDR              R5,=FIRST
               PUSH        {R0,R1,R2,R3,R4,R5,R6,R7,R8}

               MOV               R1,#0x0A
               MOV         R6,#0

start          UDIV        R2, R4,R1 ;divide by ten, div->R2
               MUL               R3, R2,R1 ;multiply div*10 -> R3
               SUB               R0,R4,R3; Least significant digit -> R0
               ADD               R0,#ASCII ;convert digits to ascii
               PUSH        {R0}
               MOV               R4,R2
               CMP         R2,#0
               ADD               R6,#0x01 ; counts how many decimal digits
exists in the number
               BNE               start
               ;ascii equivalent of digits are pushed to stack, then write to location
of R5
write_reg      SUB         R6,#0x01
               POP         {R7}
               STRB  R7,[R5],#1
               CMP         R6,#0x00
               BNE         write_reg
               MOV         R0,#0x0D
               STRB  R0,[R5],#1
               MOV         R0,#0x04
               STRB  R0,[R5]
```

```
POP    {R0,R1,R2,R3,R4,R5,R6,R7,R8}
BX LR
ENDP

ALIGN
END
```

Figure 8. Convrt.s code