Preliminary work

EE 447: Lab #4

General Purpose Timer Modules

Berkay İPEK

2304814 – Sec.2

Question 1)

The Timer0A module that is being used in this question in the mode of Count-Down Periodic. Therefore, it continues to count forever. Also, it is using pre-scale factor of 4. (16MHz / 4 = 4MHz)

It should be noted that I changed Startup.s file by importing my Handler subroutine. Changes are follows:

Pulse.s | Startup.s | programming_directive.s

```
325                     PROC
326                     EXPORT  DebugMon_Handler            [WEAK]
327                     B       .
328                     ENDP
329     PendSV_Handler  PROC
330                     EXPORT  PendSV_Handler              [WEAK]
331                     B       .
332                     ENDP
333     SysTick_Handler PROC
334                     EXPORT  SysTick_Handler             [WEAK]
335                     B       .
336                     ENDP
337                     IMPORT  My_Timer0A_Handler
338     Timer0A_Handler PROC
339                     EXPORT  Timer0A_Handler         [WEAK]
340                     B       My_Timer0A_Handler
341                     ENDP
342     IntDefaultHandler\
343                     PROC
344
345                     EXPORT  GPIOPortA_Handler           [WEAK]
346                     EXPORT  GPIOPortB_Handler           [WEAK]
347                     EXPORT  GPIOPortC_Handler           [WEAK]
348                     EXPORT  GPIOPortD_Handler           [WEAK]
349                     EXPORT  GPIOPortE_Handler           [WEAK]
350                     EXPORT  UART0_Handler               [WEAK]
351                     EXPORT  UART1_Handler               [WEAK]
352                     EXPORT  SSI0_Handler                [WEAK]
353                     EXPORT  I2C0_Handler                [WEAK]
354                     EXPORT  PWM0Fault_Handler           [WEAK]
```

Moreover, I commented undesired exporting handler part. Then, I added my My_Timer0A_Handler as follows:

```
1   ; Pulse.s
2   ; Routine for creating a pulse train using interrupts
3   ; This uses Channel 0, and a 1MHz Timer Clock (_TAPR = 15 )
4   ; Uses Timer0A to create pulse train on PF2
5
6   ;Nested Vector Interrupt Controller registers
7   NVIC_EN0_INT19      EQU 0x00080000 ; Interrupt 19 enable
8   NVIC_EN0            EQU 0xE000E100 ; IRQ 0 to 31 Set Enable Register
9   NVIC_PRI4           EQU 0xE000E410 ; IRQ 16 to 19 Priority Register
10
11  ; 16/32 Timer Registers
12  TIMER0_CFG          EQU 0x40030000
13  TIMER0_TAMR         EQU 0x40030004
14  TIMER0_CTL          EQU 0x4003000C
15  TIMER0_IMR          EQU 0x40030018
16  TIMER0_RIS          EQU 0x4003001C ; Timer Interrupt Status
17  TIMER0_ICR          EQU 0x40030024 ; Timer Interrupt Clear
18  TIMER0_TAILR        EQU 0x40030028 ; Timer interval
19  TIMER0_TAPR         EQU 0x40030038
20  TIMER0_TAR          EQU 0x40030048 ; Timer register
21
22  ;GPIO Registers
23  GPIO_PORTF_DATA     EQU 0x40025010 ; Access BIT2
24  GPIO_PORTF_DIR      EQU 0x40025400 ; Port Direction
25  GPIO_PORTF_AFSEL    EQU 0x40025420 ; Alt Function enable
26  GPIO_PORTF_DEN      EQU 0x4002551C ; Digital Enable
27  GPIO_PORTF_AMSEL    EQU 0x40025528 ; Analog enable
28  GPIO_PORTF_PCTL     EQU 0x4002552C ; Alternate Functions
29
30  ;System Registers
31  SYSCTL_RCGCGPIO     EQU 0x400FE608 ; GPIO Gate Control
32  SYSCTL_RCGCTIMER    EQU 0x400FE604 ; GPTM Gate Control
33
34  ;--------------------------------------------------
35  LOW                 EQU 0x00000028 ;4x (40us=1us * (0x28))
36  HIGH                EQU 0x0000000A ;1x (10us=1us * (0x0A))
37  ;FULL CYCLE SHOULD BE 50.000 NANO SECOND
38  ;WILL BE TURNED ON EVERY 10us = 10.000 NANO SECOND
39  ;WILL BE TURNED OFF EVERY 40us = 40.000 NANO SECOND
40  ;THIS INTERRPUT WILL BE CALLED EACH 1US=1.000 nanosecond
41  ;--------------------------------------------------
42
43              AREA    routines, CODE, READONLY
44              THUMB
45              EXPORT  My_Timer0A_Handler
46              EXPORT  PULSE_INIT
47
48  ;--------------------------------------------------
49  My_Timer0A_Handler  PROC
50                      LDR     R1,=TIMER0_TAILR;When isr is entered, check the region
51                      LDR     R2,[R1]
52                      CMP     R2,HIGH
53                      BEQ     TurnOff ;If previous one is High, then go into Low mode
54                      CMP     R2,LOW
55                      BEQ     TurnOn  ;If previous one is Low, then go into High mode
56
57  TurnOff             LDR     R2,=LOW
58                      STR     R2,[R1]
59                      LDR     R1,=GPIO_PORTF_DATA
60                      MOV     R2,#0x00
61                      STR     R2,[R1]
62                      LDR     R1, =TIMER0_ICR ;clear interrupt
63                      MOV     R0, #0x01
64                      STR     R0, [R1]
65                      BX      LR
66
67  TurnOn              LDR     R2,=HIGH
68                      STR     R2,[R1]
69                      LDR     R1,=GPIO_PORTF_DATA
70                      MOV     R2,#0x04
71                      STR     R2,[R1]
72                      LDR     R1, =TIMER0_ICR ;clear interrupt
73                      MOV     R0, #0x01
74                      STR     R0, [R1]
75                      BX      LR
76
77                      ENDP
```

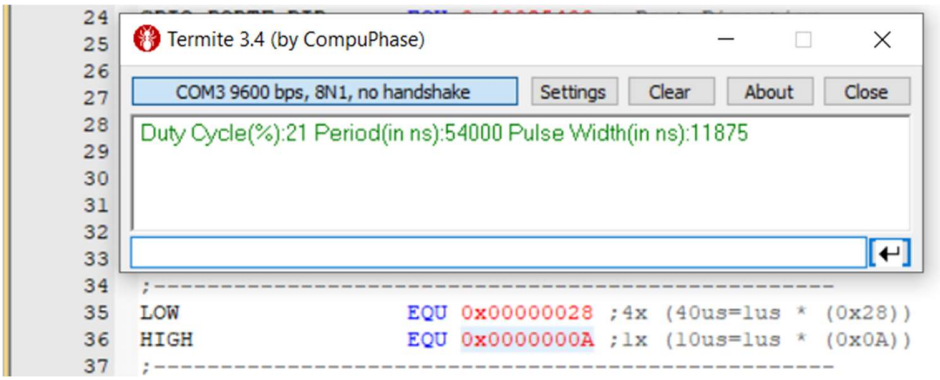
```
78  ;-------------------------------------------------
79
80  PULSE_INIT  PROC
81              LDR R1, =SYSCTL_RCGCGPIO ; start GPIO clock
82              LDR R0, [R1]
83              ORR R0, R0, #0x20 ; set bit 5 for port F
84              STR R0, [R1]
85              NOP ; allow clock to settle
86              NOP
87              NOP
88              LDR R1, =GPIO_PORTF_DIR ; set direction of PF2
89              LDR R0, [R1]
90              ORR R0, R0, #0x04 ; set bit2 for output
91              STR R0, [R1]
92              LDR R1, =GPIO_PORTF_AFSEL ; regular port function
93              LDR R0, [R1]
94              BIC R0, R0, #0x04
95              STR R0, [R1]
96              LDR R1, =GPIO_PORTF_PCTL ; no alternate function
97              LDR R0, [R1]
98              BIC R0, R0, #0x00000F00
99              STR R0, [R1]
100             LDR R1, =GPIO_PORTF_AMSEL ; disable analog
101             MOV R0, #0
102             STR R0, [R1]
103             LDR R1, =GPIO_PORTF_DEN ; enable port digital
104             LDR R0, [R1]
105             ORR R0, R0, #0x04
106             STR R0, [R1]
107
108             LDR R1, =SYSCTL_RCGCTIMER ; Start Timer0
109             LDR R2, [R1]
110             ORR R2, R2, #0x01
111             STR R2, [R1]
112             NOP ; allow clock to settle
113             NOP
114             NOP
115             LDR R1, =TIMER0_CTL ; disable timer during setup
116             LDR R2, [R1]
117             BIC R2, R2, #0x01
118             STR R2, [R1]
119             LDR R1, =TIMER0_CFG ; set 16 bit mode
120             MOV R2, #0x04
121             STR R2, [R1]
122             LDR R1, =TIMER0_TAMR
123             MOV R2, #0x02 ; set to periodic, count down
124             STR R2, [R1]
125             LDR R1, =TIMER0_TAILR ; initialize match clocks
126             LDR R2, =LOW
127             STR R2, [R1]
128             LDR R1, =TIMER0_TAPR
129             MOV R2, #15 ; divide clock by 16 to
130             STR R2, [R1] ; get 1us clocks
131             LDR R1, =TIMER0_IMR ; enable timeout interrupt
132             MOV R2, #0x01
133             STR R2, [R1]
134 ; Configure interrupt priorities
135 ; Timer0A is interrupt #19.
136 ; Interrupts 16-19 are handled by NVIC register PRI4.
137 ; Interrupt 19 is controlled by bits 31:29 of PRI4.
138 ; set NVIC interrupt 19 to priority 2
139             LDR R1, =NVIC_PRI4
140             LDR R2, [R1]
141             AND R2, R2, #0x00FFFFFF ; clear interrupt 19 priority
142             ORR R2, R2, #0x40000000 ; set interrupt 19 priority to 2
143             STR R2, [R1]
144 ; NVIC has to be enabled
145 ; Interrupts 0-31 are handled by NVIC register EN0
146 ; Interrupt 19 is controlled by bit 19
147 ; enable interrupt 19 in NVIC
148             LDR R1, =NVIC_EN0
149             MOVT R2, #0x08 ; set bit 19 to enable interrupt 19
150             STR R2, [R1]
151 ; Enable timer
152             LDR R1, =TIMER0_CTL
153             LDR R2, [R1]
154             ORR R2, R2, #0x03 ; set bit0 to enable
```

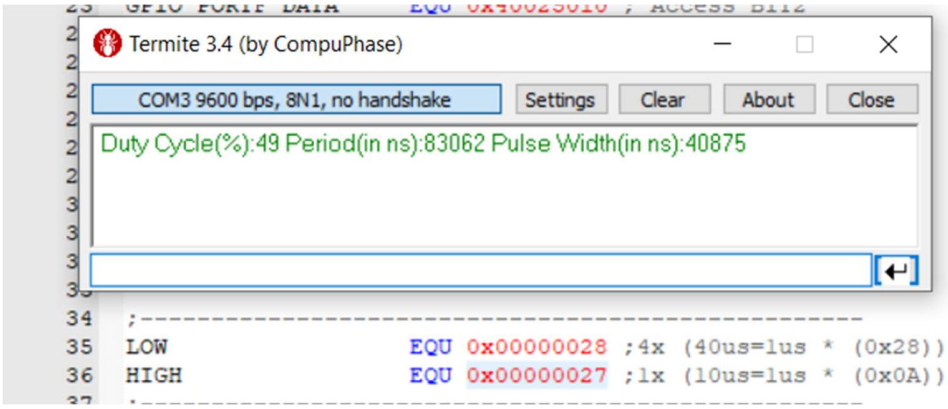
```
155                 STR R2, [R1] ; and bit 1 to stall on debug
156                 BX LR ; return
157                 ENDP
158                 END
```

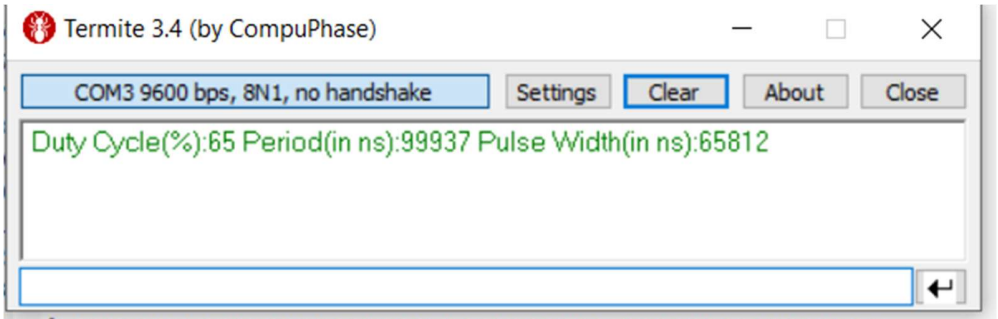
Selection of LOW and HIGH values are explained in code above.

Question 2)

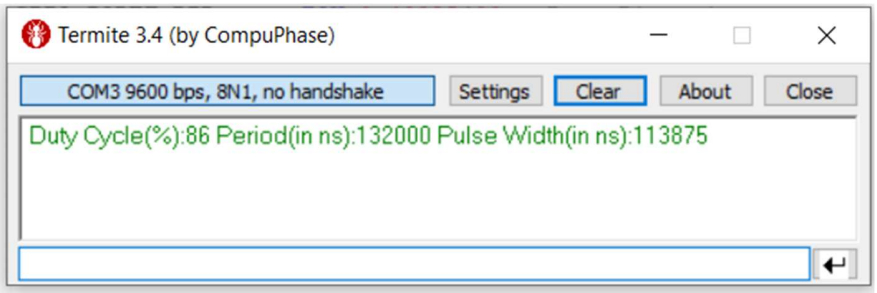
In this question, I used PB4 as timer module. Results are not perfect. I couldn't figure out why it is not perfect. However, results are quite similar, which is satisfying. In my opinion, it is about taking captured time in my program. Results with different LOW and HIGH values are:








```
;-------------------------------------------------
LOW                     EQU 0x00000020 ;4x (40us=1us * (0x28))
HIGH                    EQU 0x00000040 ;1x (10us=1us * (0x0A))
;-------------------------------------------------
```




```
24
25
26
27
28
29
30
31
32
33
34 ;-------------------------------------------------
35 LOW                     EQU 0x00000010 ;4x (40us=1us * (0x28))
36 HIGH                    EQU 0x00000070 ;1x (10us=1us * (0x0A))
37 ;-------------------------------------------------
```

Let’s move to analyze my code:

Main: (B . == loop B loop)

```
1                       AREA        main, READONLY, CODE
2                       THUMB
3                       IMPORT      PULSE_INIT
4                       IMPORT      DETECT_INIT
5                       IMPORT      RESULT
6                       EXPORT      __main
7
8   __main              PROC
9                       BL          PULSE_INIT
10                      BL          DETECT_INIT
11                      BL          RESULT
12                      B           .
13
14
15                      ALIGN
16                      ENDP
17                      END
18
```

PULSE_INIT file is the same as the one in the precious question.

DETECT_INIT:

```
1   ; Timer channel registers for TIMER1:
2   TIMER1_CFG              EQU 0x40031000 ; Configuration Register
3   TIMER1_TAMR             EQU 0x40031004 ; Mode Register
4   TIMER1_CTL              EQU 0x4003100C ; Control Register
5   TIMER1_ICR              EQU 0x40031024 ; Interrupt Clear Register
6   TIMER1_TAILR            EQU 0x40031028 ; Interval Load Register
7   TIMER1_TAPR             EQU 0x40031038 ; Prescaling Divider
8   ; Timer Gate Control
9   SYSCTL_RCGCTIMER        EQU 0x400FE604 ; Timer Clock Gating
10  ;GPIO Registers for Port B
11  ;Port B base 0x40005000
12  GPIO_PORTB_DIR          EQU 0x40005400 ; Port Direction
13  GPIO_PORTB_AFSEL        EQU 0x40005420 ; Alt Function enable
14  GPIO_PORTB_DEN          EQU 0x4000551C ; Digital Enable
15  GPIO_PORTB_AMSEL        EQU 0x40005528 ; Analog enable
16  GPIO_PORTB_PCTL         EQU 0x4000552C ; Alternate Functions
17  ;GPIO Gate Control Register
18  SYSCTL_RCGCGPIO         EQU 0x400FE608
19  ; Setup Port B for signal input
20  ; set direction of PB4
21
22
23                          AREA    routinea, CODE, READONLY
24                          THUMB
25
26                          EXPORT  DETECT_INIT
27
28  DETECT_INIT             PROC
29                          LDR R1, =SYSCTL_RCGCGPIO ; start GPIO clock
30                          LDR R0, [R1]
31                          ORR R0, R0, #0x02 ; set Port B
32                          STR R0, [R1]
33                          NOP ; allow clock to settle
34                          NOP
35                          NOP
36                          LDR R1, =GPIO_PORTB_DIR
37                          LDR R0, [R1]
38                          BIC R0, R0, #0x10 ; clear bit 4 for input
39                          STR R0, [R1]
40
41  ; enable alternate function
42                          LDR R1, =GPIO_PORTB_AFSEL
43                          LDR R0, [R1]
44                          ORR R0, R0, #0x10 ; set bit4 for alternate fuction on PB4
45                          STR R0, [R1]
46  ; set alternate function to T1CCP0 (7)
47
48                          LDR R1, =GPIO_PORTB_PCTL
49                          LDR R0, [R1]
50                          ORR R0, R0, #0x00070000 ; set bits of PCTL to 7
51                          STR R0, [R1] ; to enable T1CCP0 on PB4
52  ; disable analog
53                          LDR R1, =GPIO_PORTB_AMSEL
54                          MOV R0, #0 ; clear AMSEL to diable analog
55                          STR R0, [R1]
56  ; enable analog
57                          LDR R1, =GPIO_PORTB_DEN
58                          MOV R0, #0x10 ; set AFSEL to enable analog
59                          STR R0, [R1]
60
61  ; Start Timer 0 clock
62                          LDR R1, =SYSCTL_RCGCTIMER
63                          LDR R2, [R1] ; Start timer 0
64                          ORR R2, R2, #0x02 ; Timer module = bit position (0)
65                          STR R2, [R1]
66                          NOP
67                          NOP
68                          NOP ; allow clock to settle
69
70  ; disable timer during setup
71                          LDR R1, =TIMER1_CTL
72                          LDR R2, [R1]
73                          BIC R2, R2, #0x01 ; clear bit 0 to disable Timer 0
74                          STR R2, [R1]
75
76  ; set to 16bit Timer Mode
77                          LDR R1, =TIMER1_CFG
```

D:\OKUL\ee_4un_1i\Lab-447\LabFive\QuestionTwo\DETECT_INIT.s

```
                        MOV R2, #0x04 ; set bits 2:0 to 0x04 for 16bit timer
                        STR R2, [R1]
; set for edge time and capture mode
                        LDR R1, =TIMER1_TAMR
                        MOV R2, #0x07 ; set bit2 to 0x01 for Edge Time Mode,
                        STR R2, [R1] ; set bits 1:0 to 0x03 for Capture Mode

; set edge detection to both
                        LDR R1, =TIMER1_CTL
                        LDR R2, [R1]
                        ORR R2, R2, #0x0C ; set bits 3:2 to 0x03
                        STR R2, [R1]
; set start value
                        LDR     R1, =TIMER1_TAILR ; counter counts down,
                        MOV     R0, #0xFFFF ; MAX value
                        STR     R0, [R1]

                        LDR     R1,=TIMER1_TAPR
                        MOV     R0,#0xFF    ;PreScaling
                        STR     R0,[R1]

; Enable timer
                        LDR     R1, =TIMER1_CTL
                        LDR     R2, [R1] ;
                        ORR     R2, R2, #0x03 ; set bit 0 to enable
                        STR     R2, [R1]
                        BX      LR              ;returning
; Now use this data, with other measured data to compute
; period, pulse width, duty cycle, frequency,...

```

These are just configuration settings. There is nothing interesting. Interesting part is in the subroutine, RESULTS.

RESULTS:

```
1   TIMER1_RIS          EQU 0x4003101C ; Raw interrupt Status
2   GPIO_PORTB_DATA     EQU 0x40005040
3   TIMER1_TAR          EQU 0x40031048 ; Counter Register
4   TIMER1_ICR          EQU 0x40031024 ; Interrupt Clear Register
5
6
7                       AREA        writing, DATA, READONLY
8                       THUMB
9   DutyCycle           DCB         "   Duty Cycle(%):"
10                      DCB     0x04
11  Pulse               DCB         "   Pulse Width(in ns):"
12                      DCB     0x04
13  Period              DCB         "   Period(in ns):"
14                      DCB     0x04
15
16
17                      AREA        main, READONLY, CODE
18                      THUMB
19                      IMPORT      CONVRT
20                      IMPORT      OutStr
21                      EXPORT      RESULT
22
23  RESULT              PROC
24                      MOV     R2,#0   ;This register will be the flag which decides # of edge that
    captured
25                      LDR     R7,=TIMER1_TAR
26                      LDR     R8,=TIMER1_ICR
27                      LDR     R9,=GPIO_PORTB_DATA
28                      LDR     R10,=TIMER1_RIS
29  loop                LDR     R0,[R10]
30                      ANDS    R0,#0x4 ;Seperating CAERIS bit,
31                      BEQ     loop ; if there is no captured time, then iterate
32                      LDRH    R3, [R7] ; Get timer register value
33                      MOV     R0,#0x04
34                      STR     R0,[R8] ;Clear ICR
35                      CMP     R2,#0
36                      BEQ     first   ;That means it can be first edge (First edge is in R4)
37                      CMP     R2,#1
38                      BEQ     second  ;That means it is the second edge (Second edge is in R5)
39                      CMP     R2,#2
40                      BEQ     third   ;That means it is the third edge (Third edge is in R6)
41
42  ;That means taking necessary part is finished. Move to calculation part
43  ;WHEN R2=3, IT WILL GO INTO CALCULATION PART
44
45  ;-------------------------------------------------------------------------------------;
46  ;-------------------CALCULATION--------------------------PART-------------------------;
47  ;-------------------------------------------------------------------------------------;
48
49  ;-----------------CONSTANTS (TO BE ABLE TO MULT BY 62.5-------------------------;
50                      MOV     R0,#10
51                      MOV     R1,#625
52
53
54  ;-----------------CALCULATING INTERVALS-------------------------|
55                      SUB     R7,R4,R6 ;PERIOD (FIRST EDGE - THIRD EDGE) [IN CYCLE UNIT, NOT IN ns]
56                      SUB     R8,R4,R5 ;PULSE WIDTH (FIRST EDGE- SECOND EDGE) [IN CYCLE UNIT, NOT IN ns]
57
58  ;-----------------DUTY-CYCLE-------------------------|
59                      MUL     R9,R8,R0 ; Pulse Width *=10
60                      MUL     R9,R0   ;Pulse Width *=10 ( At the end Pulse Width *=100)
61                      UDIV    R9,R7   ; Pulse Width*100 / PERIOD = DUTY CYCLE
62                      LDR     R5,=DutyCycle
63                      BL      OutStr
64                      MOV     R4,R9
65                      BL      CONVRT
66
67  ;-----------------PERIOD-------------------------|
68
69                      MUL     R7,R1   ;PERIOD *= 625
70                      UDIV    R7,R0   ;PERIOD /= 10 (NOW WE GOT PERIOD IN UNITS OF NANOSECOND)
71                      LDR     R5,=Period
72                      BL      OutStr
73                      MOV     R4,R7
74                      BL      CONVRT
75  ;-----------------PULSE-WIDTH-------------------------|
76                      MUL     R8,R1   ;PULSE_WIDTH *= 625
```

```
 77                      UDIV    R8,R0   ;PULSE_WIDTH /= 10 (NOW WE GOT PERIOD IN UNITS OF NANOSECOND)
 78                      LDR     R5,=Pulse
 79                      BL      OutStr
 80                      MOV     R4,R8
 81                      BL      CONVRT
 82
 83  ;-----------------FINISHING-SUBROUTINE-------------------------|
 84                      BX      LR
 85
 86  ;-----------------CATCHING FIRST POS EDGE--------------------------|
 87  first               LDR     R0,[R9]
 88                      CMP     R0,#0x10
 89                      BNE     loop    ;CHECKING WHETHER IT IS POS EDGE OR NOT
 90                      MOV     R4,R3   ;LOAD THE CAPTURED TIME TO R4
 91                      ADD     R2,#1   ;CHANGE FLAG SO THAT IT CAN GO INTO THE SECOND LOOP WHEN THERE
     IS AN EDGE
 92                      B       loop
 93
 94  second              MOV     R5,R3   ;LOAD THE CAPTURED TIME TO R5
 95                      ADD     R2,#1   ;CHANGE FLAG SO THAT IT CAN GO INTO THE THIRD LOOP WHEN THERE IS
     AN EDGE
 96                      B       loop
 97
 98  third               MOV     R6,R3       ;LOAD THE CAPTURED TIME TO R6
 99                      ADD     R2,#1   ;CHANGE FLAG SO THAT IT CAN GO INTO THE CALCULATION LOOP WHEN
     THERE IS AN EDGE
100                      B       loop
101                      ENDP
```