Warsaw University of Technology

Faculty of Mechatronics

Microcontrollers

Lab 02

Report

Timers and Interrupts

Made by:

Harshit Verma (302601)

1. Theory

I this course of this exercise, we must create an application to switch on and off LEDs on ADuC834 by pressing specific button using Keil uVision5 software.

The LEDs should work accordingly:

- Button pressed–turns on LED blinking @ P0.5
- Button pressed–turns off LED blinking @ P0.5
- Button pressed–turns on LED blinking @ P0.6
- Button pressed–turns off LED blinking @ P0.6

Required frequencies of blinking:

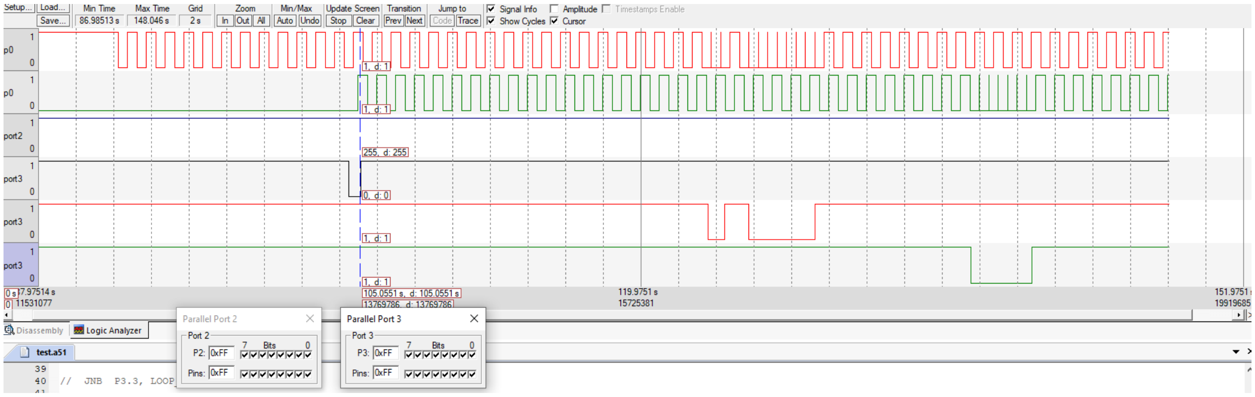
LED @ P0.5 -> 1 Hz (i.e., 0.5 s off and 0.5 s on)

LED @ P0.6 -> (2+Y) Hz,

where Y = 01, therefore, 21 Hz. Since the no. is too small, I assume the value 210 Hz.

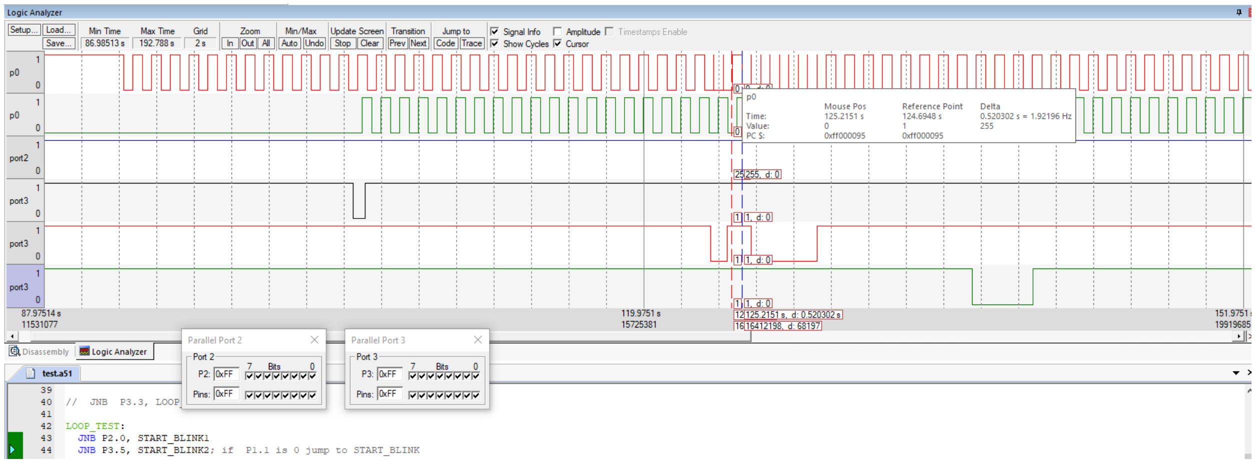
1. Result

- Screenshots from the simulator/debugger showing the input signals on the pins and the related changes of the LEDs frequencies. Mark the blinking frequencies in the logic analyzer using cursors.



```
// JNB P3.3, LOOP
```

Figure -1 Screen representing 210 Hz(105s on each cycle)



p0

	Mouse Pos	Reference Point	Delta
Time:	125.2151 s	124.6948 s	0.520302 s = 1.92196 Hz
Value:	0	1	255
PC $:	0xff000095	0xff000095	

```
// JNB P3.3, LOOP

LOOP_TEST:
  JNB P2.0, START_BLINK1
  JNB P3.5, START_BLINK2; if  P1.1 is 0 jump to START_BLINK
```

Figure-2 Screen representing 1 Hz(0.5s on each cycle)

- Comment on the possibility of mutual blocking of ISRs in your solution. Is it possible to observe such an effect in the logic analyzer?

In this Logic Analyzer, the Timer 0 and Timer 1 can work simultaneously, the one with the higher priority will occur. In the case where the priorities are same, the interrupt which occurs in the code first will be implemented.

- Two versions of the program code –in assembly language and in C language.

  Assembly Code:

```
$NOMOD51              ; suppress pre-definition of 8051 SFR names
                            ; core SFRs (ACC, B, DPL, DPH, PSW, SP)
remain defined

$include (aduc834.h)  ; include ADuC834 symbols

;
-------------------------------------------------------------------
; user symbols here
FAN EQU P1.0                ; P1.0 toggles the fan

;
-------------------------------------------------------------------
CSEG                        ; Code SEGment starts here

ORG 0000H                   ; set ORiGin for subsequent statements
at 0000H
      JMP START             ; jump to START (0060H)
ORG 000BH
      JMP ISR_T0
ORG 001BH
      JMP ISR_T1

ORG 0060H                   ; main program starts here
START:
; initialization starts here
; code between START and LOOP labels is intended
; for a single execution at the beginning of the program
      MOV SP,#7FH           ; set the stack pointer to 0x7F (hex)
      MOV P0,#11111111B                ; turn the motor off
      CLR FAN               ; turn the fan off

      MOV TMOD,#00010001B ;Bit is not addressable, so we assign the
value for the execution we want
      MOV TH0,#0x65 ;High value in hexadeciaml execution
      MOV TL0,#0xE6 ;Low value in hexadecimal execution

//    MOV TH0,#0x7A120 ;High value in hexadeciaml execution
//    MOV TL0,#0x7A120 ;Low value in hexadecimal execution
```

```
      SETB ET0
      SETB ET1
      SETB EA

//    JNB  P3.3, LOOP_TEST2

LOOP_TEST:
  JNB P2.0, START_BLINK1
  JNB P3.5, START_BLINK2; if  P1.1 is 0 jump to START_BLINK
  JNB P3.2, START_OFF1; if  P1.1 is 0 jump to START_BLINK
  JNB P3.3, START_OFF2; if  P1.1 is 0 jump to START_BLINK
      JMP LOOP_TEST

START_BLINK1:
      SETB TR0 ;Bit is addressable, so we can call the execution
directly

      JMP LOOP_TEST

START_BLINK2:
      SETB TR1 ;Bit is addressable, so we can call the execution
directly

      JMP LOOP_TEST

START_OFF1:
      CLR P0.7

      JMP LOOP_TEST

START_OFF2:
    CLR P0.6

      JMP LOOP_TEST

;LED_LOOP:

      ;TURN_OFF:
      ; CLR P0.7  ;CLR put value 0 for this execution by default
      ; CLR P0.6
      ; CLR P0.5
      ; CLR P0.4

;      CALL DELAY

      ; TURN_ON:
```

```
      ; SETB P0.7 ;SETB put value 1 for this execution by default
      ; SETB P0.6
      ; SETB P0.5
      ; SETB P0.4

      ; CALL DELAY_TIM

      ; JNB P1.1, LOOP ;Jump if P1.1 is 0 to Loop



      ; JMP LED_LOOP


;LOOP:
; main program loop
  ; JNB P2.0, LED_LOOP ;Jump if P2.0 is 0 to LED_LOOP
      ;JMP LOOP
; ------------------------------------------------------------------
; subroutines and interrupt service routines start here

ISR_T0:

      CPL P0.7
      RETI

ISR_T1:

      CPL P0.6
      RETI


; ------------------------------------------------------------------
END                           ; compiled code ends here
```

C Language:

```
#include <aduc834.h>

#define FAN P1_0

void main() {
   SP = 0x7F;          // Set stack pointer to 0x7F
   P0 = 0xFF;          // Turn off the motor
   FAN = 0;            // Turn off the fan

   TMOD = 0x11;         // Configure Timer 0 and Timer 1 modes
   TH0 = 0x65;          // Load Timer 0 high byte with 0x65
   TL0 = 0xE6;          // Load Timer 0 low byte with 0xE6
```

```
    ET0 = 1;               // Enable Timer 0 interrupt
    ET1 = 1;              // Enable Timer 1 interrupt
    EA = 1;               // Enable global interrupts

    while(1) {
        if (P2_0 == 0) {   // Check if P2.0 is 0
            TR0 = 1;       // Start Timer 0
        } else if (P3_5 == 0) { // Check if P3.5 is 0
            TR1 = 1;       // Start Timer 1
        } else if (P3_2 == 0) { // Check if P3.2 is 0
            P0_7 = 0;      // Turn off motor
        } else if (P3_3 == 0) { // Check if P3.3 is 0
            P0_6 = 0;      // Turn off motor
        }
    }
}

void ISR_T0() interrupt 1 {
    P0_7 = ~P0_7;          // Toggle P0.7
    // Clear Timer 0 interrupt flag automatically
}

void ISR_T1() interrupt 3 {
    P0_6 = ~P0_6;          // Toggle P0.6
    // Clear Timer 1 interrupt flag automatically
}
```