Warsaw University of Technology

Faculty of Mechatronics

Microcontrollers

Lab 01

Report

Blinking LED

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 device using Keil uVision5 software.

The LEDs need to be switched each X ms.

where X = 300 + 01 = 301 ms.

We need two types of loops for the delay between the blinking of the LED. We must prepare a software delay (based on nested loop) and another hardware delay (based on Timer 0).

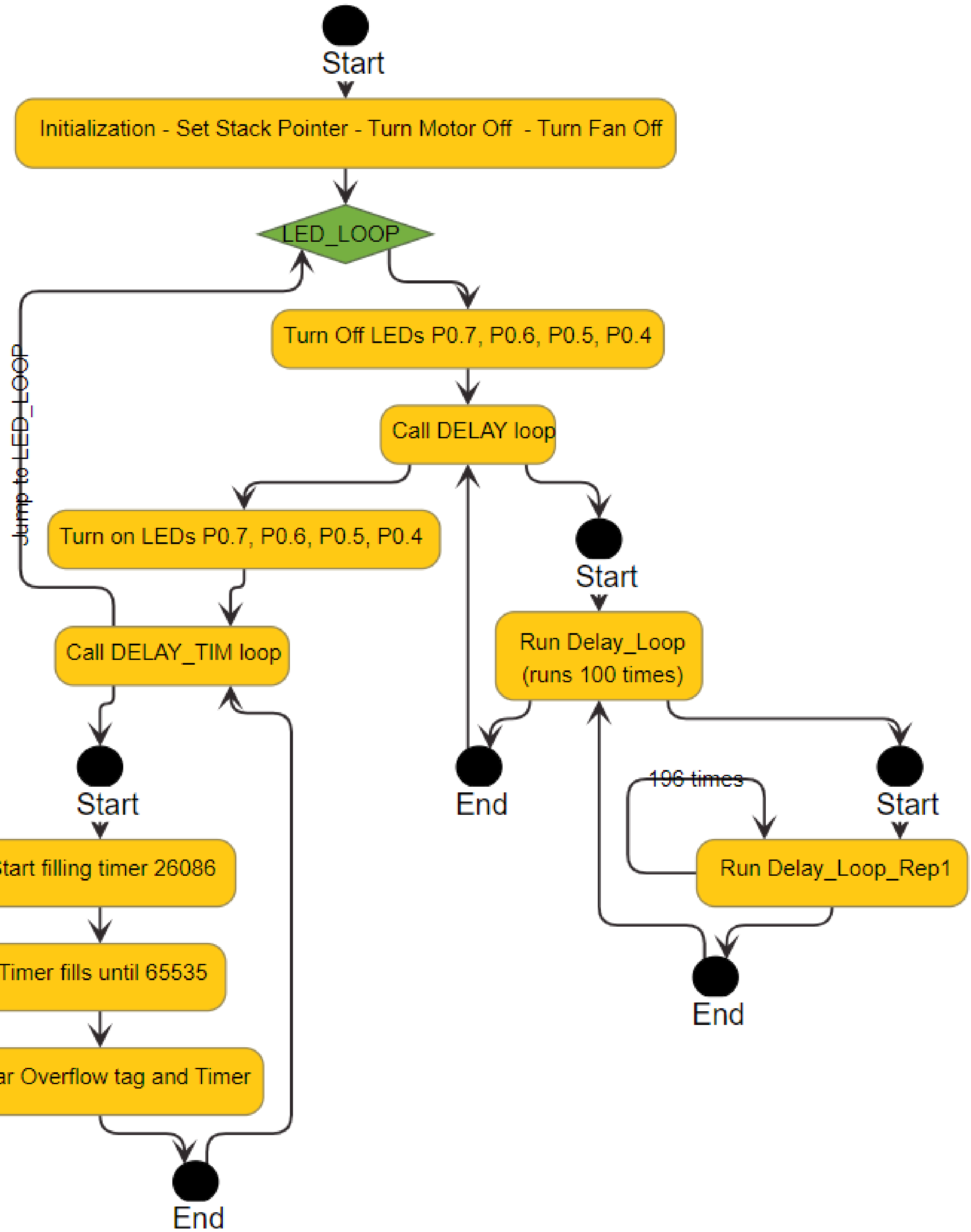
We need to know that,

Overflow value = (2^16) −1 = 65535

1. Result

- The number of milliseconds X and calculations for
    - a) the delay loop from the software delay (including the number of machine cycles per instruction)
      I was able to achieve 307 ms delay, Total no. of machine cycles is 39,502 for the Delay loop.
    - b) the initial value of the Timer 0 counting register from the hardware delay
      Initial count value = TH0*256 + TL0 = 101*256+230 = 26086
      Number of ticks for the timer until overflow = 65535 – 26086 = 39,449

- A flowchart of the algorithm of the application



- File with the program 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 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

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 ;Call Software based loop


        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 ;Call Timer based loop


        JMP LED_LOOP ;Start LED_LOOP again



LOOP:

; main program loop

        JMP LOOP

; ___________________________________________________________

; subroutines and interrupt service routines start here

DELAY:


        MOV R0,#100 ;Assign this value to R0

        DELAY_LOOP: ;Run 100 times

                MOV R1,#196 ;Assign this value to R1

                DELAY_LOOP_REP1: ;Run 196 times

                        DJNZ R1,DELAY_LOOP_REP1 ;wait until above value reached


                DJNZ R0,DELAY_LOOP


        RET
```

```
DELAY_TIM:

	//SETB TF0 ;Bit is addressable, so we can call the execution directly

	MOV TMOD,#00000001B ;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

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

TF_CHK:

	JNB TF0, TF_CHK ;Check if Overflow at TF0

	CLR TF0 ;Clear Overflow tag

	CLR TR0 ;Clear Timer

	RET

; ____________________________________________________________

END						; compiled code ends here
```

- Screenshots from the debugger/simulator, showing the measurements of the periods of both types of delays conducted in the Logic Analyzer window