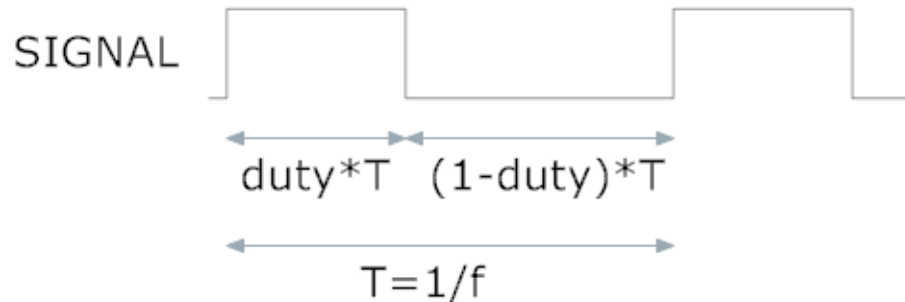


# **Excercise 2**

## **Timer**

**Write a program that toggles one of the LED pins (e.g. LED0 on pin P0.4) to produce a square signal.**



**Additional features: frequency ( $f$ ) and duty cycle ( $\text{duty}$ ) should be adjustable with keys and displayed on the LCD.**

**T0, T1 : used to adjust the frequency from 1-10 Hz ( with 1Hz step)**

**T2, T3: used to adjust the duty cycle from 1-99 % ( with 1% step )**

## ***TIMER1( lpc2138 also has TIMER0)***

**Registers ( see timer.h ):**

**T1TC – timer counter – counts divided peripheral bus (VPB) clock transitions**

**T1PC – prescale counter – counts VPB clock transitions**

**T1TCR – timer control – writing *counter\_enable* (*counter\_reset*) starts/stops the counter**

**T1MRj – match registers – when T1TC reaches a value of any T1MRj ( j=0,1,2,3 ), a predefined action can be performed (trigger interrupt, reset the counter, stop the counter)**

**void timer1\_init(int prescale, int \*match, int control, int count, int pin);**

**prescale    : when T1PC == prescale => T1TC = T1TC+1**

**match       : an array with 4 integers for match registers: when T1TC == match[j] => perform an action  
e.g. Int match[4]={100,200,0,0}**

**control     : configure the actions for match events (T1TC == T1MRj for j=0,1,2,3 )**

**mrji        : T1TC == T1MRj => trigger interrupt**

**mrjr        : T1TC == T1MRj => reset the counter**

**mrjs        : T1TC == T1MRj => stop the counter**

**e.g. control = mr0i | mr0r => when T1TC==T1MR0 => trigger interrupt and reset the counter**

**count       : count mode**

**timer                : count VPB clock transitions**

**counter\_rising | capi    : count positive transition of preconfigured capture pin**

**counter\_falling | capi   : count negative transition of preconfigured capture pin**

**counter\_both | capi     : count all transition of preconfigured capture pin**

**capi = cap0, cap1, cap2, cap3 (bitmasks for 4 available capture signals )**

**pin          : which pin to use as capture signal**

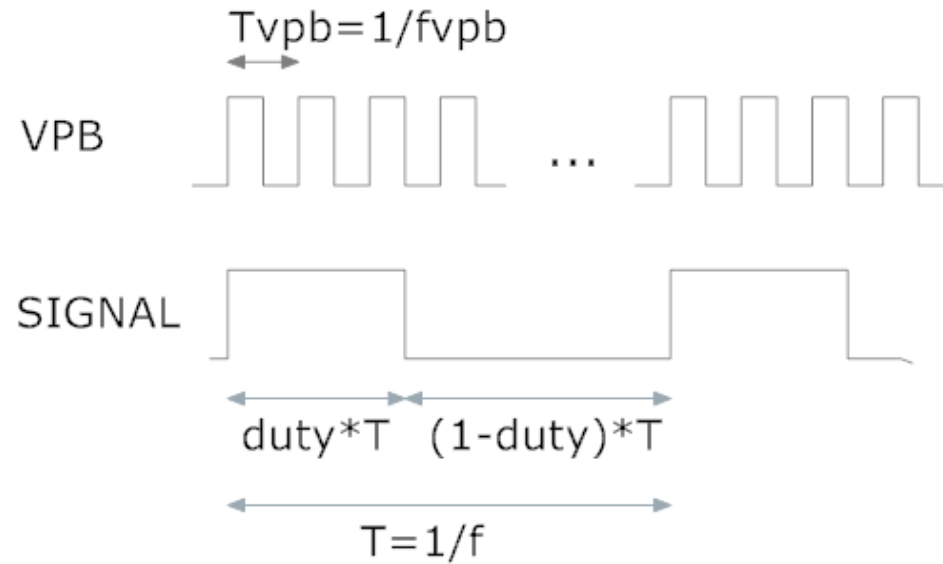
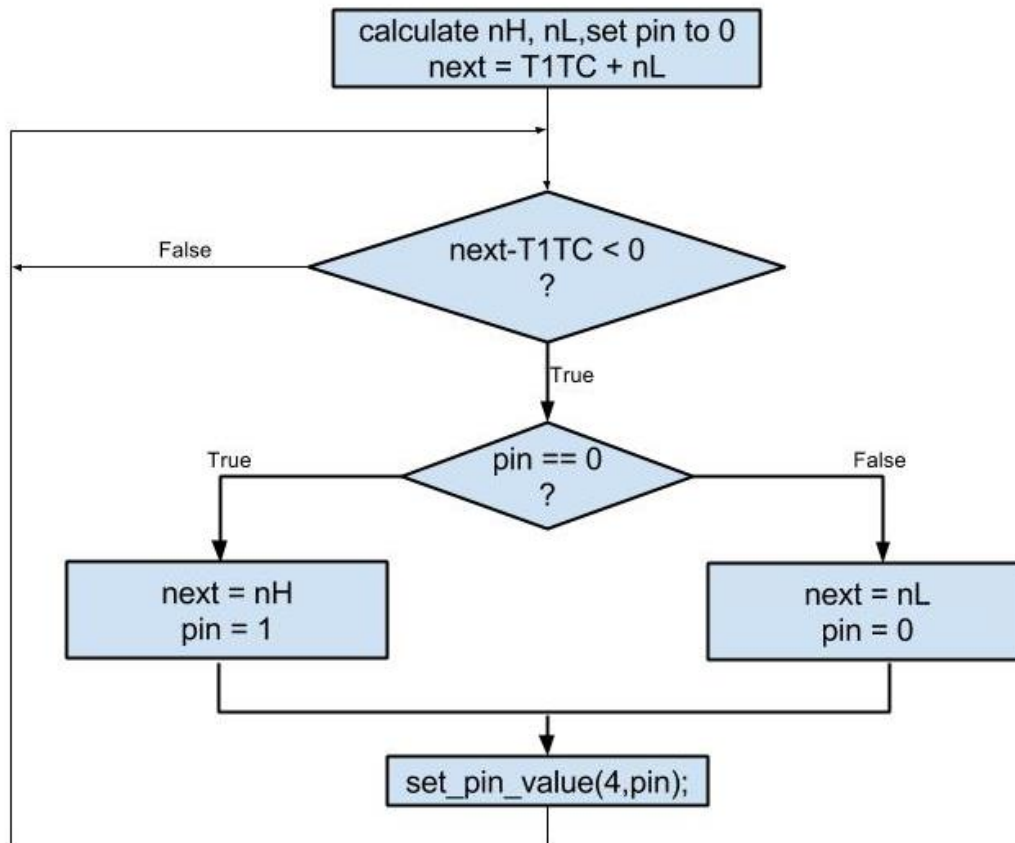
First approach ( polling ):

Initialize the timer ( startup.c ) :

```
int prescale = 0;  
int match[] = { 0, 0, 0, 0 };  
int control = 0;      /* we dont want any special action when T1TC = match[j] */  
int count = timer;    /* timer1 will be free flowing counter of vpb clock */  
int pin = 0;          /* no capture pin */  
  
timer1_init( prescale, match, control, count ,pin);  
  
/* To start the timer: */  
T1TCR = counter_enable;
```

First approach ( polling ):

Main program ( main.c ):



$$nH = duty * f_{vpb} / f$$
$$nL = (1 - duty) * f_{vpb} / f$$

**Usefull functions:**

**Initialize the microcontroler (see init.h, init.c):**

**void init(int clock\_mhz, int div,int input,int output,int value);**

**clock\_mhz: cpu clock frequency fclk in mHz (12,24,36,48, or 60)**

**div: peripheral bus frequency divider**

**cclk: fvpb = fclk**

**cclk\_2: fvpb = fclk/2**

**cclk\_4: fvpb = fclk/4**

**input, output: bit masks for input and output pins**

**use P0\_xx macros (defined in gpio.h) to construct bit masks, e.g.:**

**input = P0\_12 | P0\_13**

**output = P0\_4 | P0\_5**

**value: initial output pins value**

**You only need to call this function once.**

**Usefull functions:**

**GPIO functions (see gpio.h, gpio.c):**

**void set\_pin\_value(int pin\_num, int pin\_value);**

**Write digital value to the output pin with pin number pin\_num (0,1,...15).**

**int get\_pin\_value(int pin\_num);**

**Read digital value of the input pin with pin number pin\_num (0,1,...15).**

**void wait(int delay\_us);**

**a delay function (waits for delay\_us microseconds)**

**Usefull functions:**

**Writing to the LCD (see main.c):**

**void lcd\_driver();**

**Call this function to refresh the display. Use lcd\_string pointer to write text to LCD buffer.**

**For example to write text to LCD:**

**strcpy(lcd\_string, "Hello"); // strcpy() is part of standard string.h library  
lcd\_driver();**