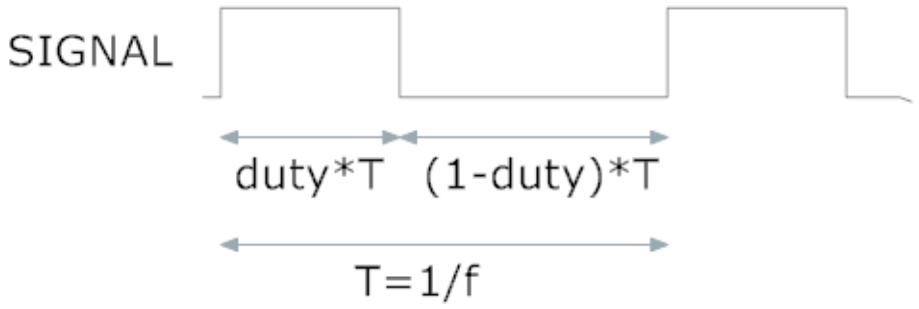
Vaja 3
Delo s časovnikom (timer)

Na enem od pinov vrat port0 (npr. P0.4 – LED0) generiraj pravokotni signal. Frekvenca in duty cycle naj bosta nastavljiva preko dveh globalnih spremenljivk.



Uporaba pinov vzporednih vrat port0:

p0.24 in p0.31:

nista splošna vhodno/izhodna pina

p0.9 in p0.11 (oz. p0.1 in p0.9 na starih sistemih – št.<50000):

izhoda prilagodilnika nivojev zaporednih vrat MAX232, ki vsiljuje svoje stanje
=> pini morajo biti vhodni

p0.12 do p0.15 (tipke) in p0.27 (analogna napetost – potenciometer)

vsiljujejo svoje stanje => pini morajo biti vhodni
tipke lahko odklopimo - stikalo (angl. jumper) J16
potenciometer lahko odklopimo - stikalo (angl. jumper) J20
=> potem so pini lahko izhodni

Maske posameznih pinov (gpio.h):

```
#define P0_0   0x00000001
#define P0_1   0x00000002
…
```

Maske diod LED in tipk (gpio.h):

```
#define LED0  0x00000010
#define LED1  0x00000020
…
#define T0  0x00001000
#define T1 0x00002000
…
```

Časovnik TIMER1 (lpc2138 ima še TIMER0

Registri (glej timer.h):
T1TC – timer counter – števec deljenih impulzov ure VPB
T1PC – prescale counter – števec delilnika impulzov ure VPB
T1TCR – timer control – vpis konst. *counter_enable* **oz** *counter_reset* **sproži oz. ustavi štetje**
T1MRj – primerjalni registri – ko doseže T1TC vrednost v T1MRj (j=0,1,2,3), lahko sprožimo odziv (prekinitev, reset, stop)

void timer1_init(int prescale, int *match, int control, int count);

prescale : ko T1PC == prescale => T1TC = T1TC+1
match : zbirka 4 celih št. - ko T1TC == match[j] => sproži dejanje, določeno v control
control : kombinacija zastavic, ki določa odziv na ujemanje T1TC in match[j] (j=0,1,2,3)
mrji : T1TC == match[j] => sproži prekinitev
mrjr : T1TC == match[j] => resetira števca
mrjs : T1TC == match[j] => ustavi števec
npr. control = mr0i | mr0r : T1TC == match[0] => sproži prekinitev in resetira števec
count : način štetja
timer : šteje impulze ure VPB
counter_rising | capi : šteje pozitivne fronte vh. signala
counter_falling | capi : šteje negativne fronte vh. signala
counter_both | capi : šteje pozitivne in negativne fronte vh. signala
capi = cap0, cap1, cap2, cap3 (eden izmed 4 capture signalov)
pin pin za štetje front signala (parameter count ne sme biti timer)

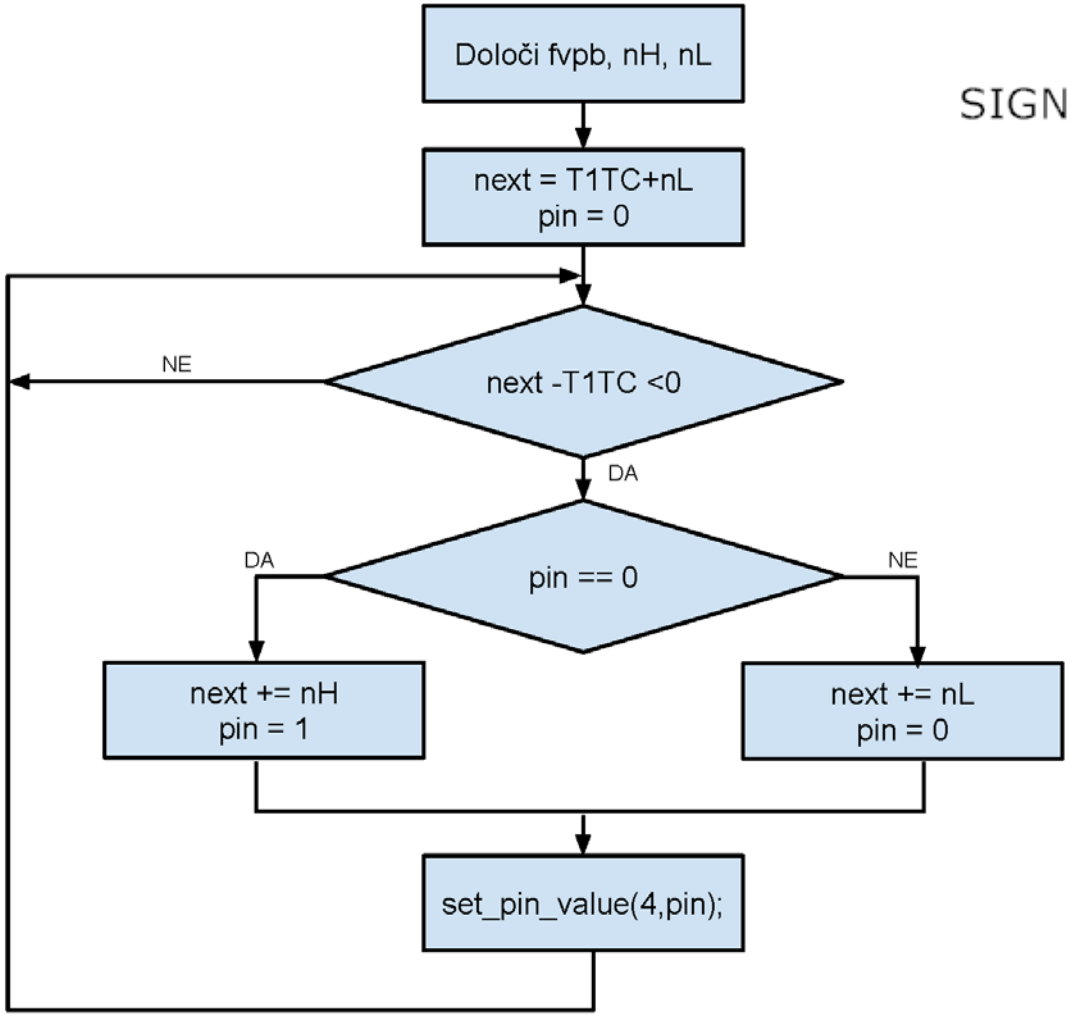
Postopek 1 (polling):

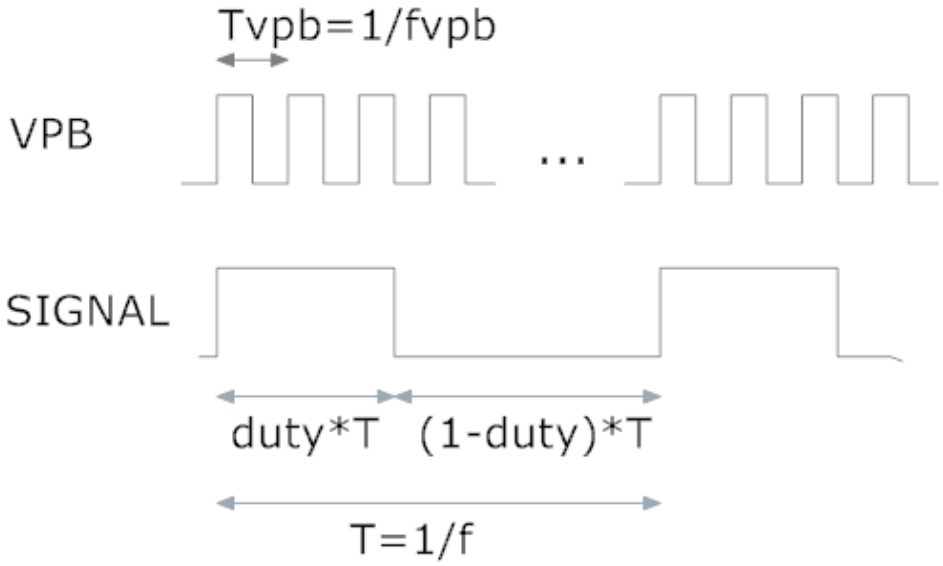
Inicializacija števca (startup.c) :

```
int prescale = 0;
int match[] = { 0, 0, 0, 0 };
int control = 0;        /*  pri štetju se nič ne zgodi tudi ko je T1TC = match[j]  */
int count = timer;      /* štejemo pulze ure VPB (nedeljene) */
int pin = 0;            /* ne štejemo front signala na vhodnem pinu */

timer1_init( prescale, match, control, count ,pin);

/* Zagon števnika: */
T1TCR = counter_enable;
```

Postopek 1 (polling):

Glavni program (main.c):





nH = duty * fvpb / f
nL = (1 - duty) * fvpb / f

Postopek 2 (prekinitve):

void timer1_init(int prescale, int *match, int control, int count);

prescale = 0;	**še vedno štejemo nedeljene impulze ure VPB**
match = {xxx, 0, 0, 0};	**tu bomo nastavljali vrednost, ob kateri se pin spremeni**
control = mr0i | mr0r;	**ko je T1TC = match[0], se sproži prekinitev, števnik pa se postavi na 0**
count = timer;	**še vedno samo štejemo pulze ure VPB**

Vektorski nadzornik prekinitev (VIC):
Možne vrste prekinitev (definicije v vic.h – bitne maske, ki se prepišejo v nadzorni register za VIC) :
wdt, arm_core0, arm_core1, timer0, timer1, uart0, uart1, pwm0, i2c0, spi0, spi1_ssp, pll, rtc, eint0, eint1, eint2, eint3, ad0, i2c1, bod in ad1

void vic_init(int fiq, int irq, voidfuncptr *function, int *interrupt, voidfuncptr def);

fiq	**: prednostne prekinitve – poljubna kombinacija (mask) prekinitev (bitna ALI op. mask)**
irq	**: navadne IRQ prekinitve – poljubna kombinacija (mask) prekinitev (bitna ALI op. mask)**
function	**: polje 16-ih kazalcev na funkcije po prioriteti razvrščenih vektorskih prekinitev IRQ**
interrupt	**: polje 16-ih po prioriteti razvrščenih vektorskih prekinitev IRQ**
def	**: kazalec na funkcijo po prioriteti nerazvrščene prekinitve IRQ**

Postopek 2 (prekinitve):

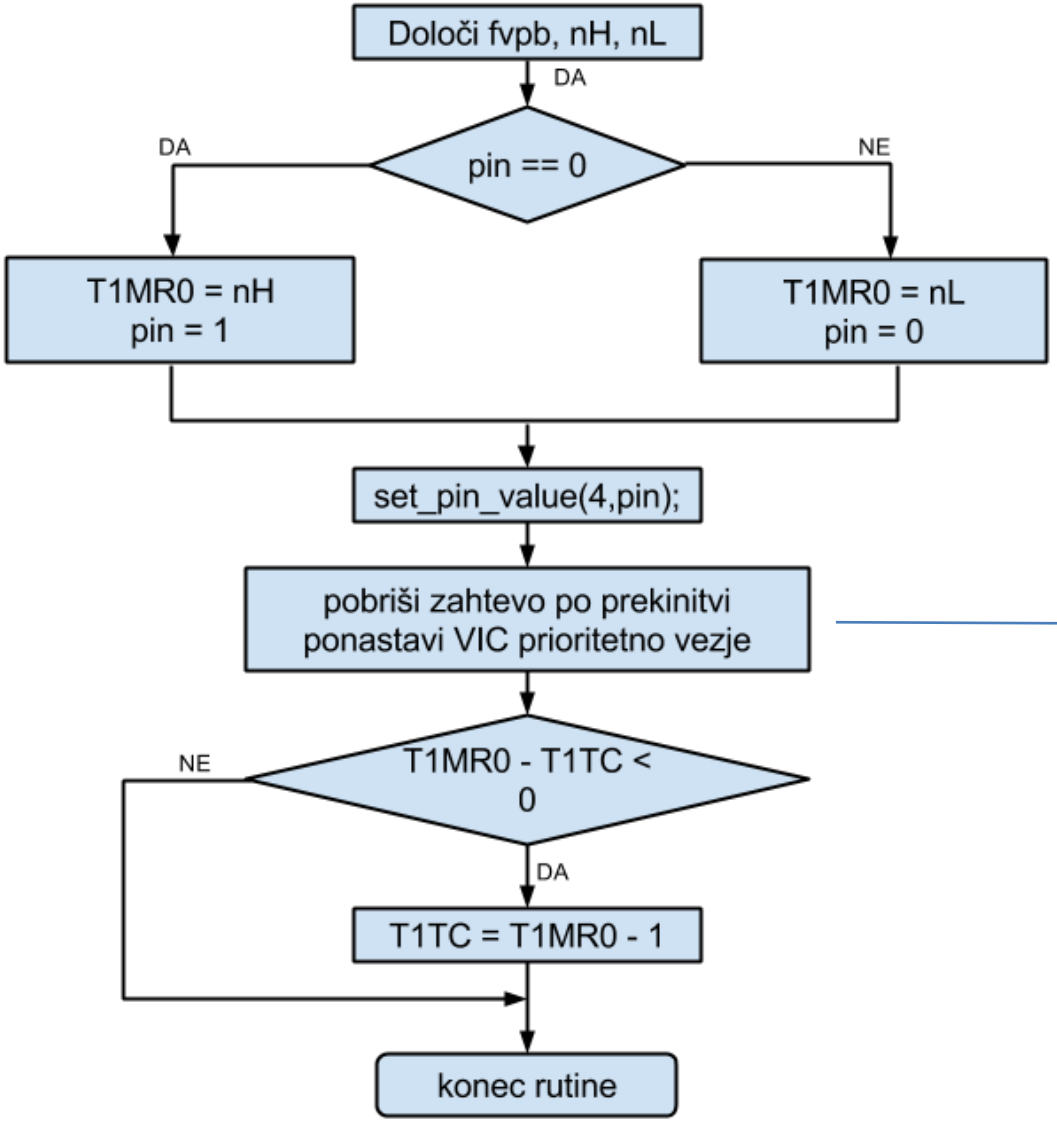
Inicializacija nadzornika prekinitev (startup.c) :

```
int fiq = 0;              // nobena prekinitev ne bo prednostna
int irq = timer1;         // timer naj bo navadna IRQ prekinitev

voidfuncptr function[16] =
{ pulse_train, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

int interrupt[16] =
{ timer1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
```

=> vic_init(…)

Postopek 2 (prekinitve):

pulse_train() **prekinitvena rutina (main.c):**



T1IR=mr0_interrupt;
VICVectAddr=0;