Generador de Polsos

Dispositius Programables — Enginyeria de Sistemes TIC

Previs

Tasca prèvia 1

Observeu que el tros de codi de la subrutina i la macro dels codis anteriors són idèntics. Calculeu el nombre de cicles màquina (clk) que es tarda en executar aquest tros de codi en funció del valor que es carrega al registre r19 (paràmetre) en la primera instrucció.

per a cmençar a comptar farem un petit esquema dels bucles amb els temps de clk:

LDI		1	
LDI		1	
LDI		1	
SUBI		1	
	BRNE	2	$(((3\cdot255 - 1)\cdot255 - 1)\cdot$x-1) = cicles
SUBI		1	
	BRNE	2	
SUBI		1	
	BRNE	2	

Tasca prèvia 2

Tenint en compte que el rellotge del microcontrolador és de 16 MHz, calculeu el valor que s'ha de carregar al registre r19 per tal que el temps total que tarda en executar-se aquest tros de codi sigui de 0.5 s.

per a que passin 0,5 segons han de transcórrer 8 milions de cicles, és a dir la meitat de la frecuencia del avr.

$(((3\cdot255 - 1)\cdot255 - 1)\cdot$x-1) $\rightarrow$ 764 · 254 · x $\rightarrow$ 194.000 ·x = 8.000.000

x = (8.000.000/194.000) = 41

per tan, per a que transcorren 0,5 segons haurem de posar el r19 al valor 37.

Tasca prèvia 3

Una manera per crear un to audible amb una sortida digital és generant una ona quadrada de freqüència audible, per exemple fto = 600 Hz. Això ho podem aconseguir commutant entre 0 i 1 una de les potes de sortida de l'Arduino. Comproveu si la subrutina / macro waitabit serveix per poder generar una ona quadrada de fto = 600 Hz en una de les potes de l'Arduino. Si ho considereu oportú, modifiqueu waitabit. Anomeneu a la nova subrutina / macro waitato.

16Mhz / 600hz = 26666 cicles, pero volem un semicicle, llavors seran 13333 cicles per fer

Hem de trobar els valors de r17 i r18 per tal d'aconseguir 600Hz

si tenim r17= 255
r19 el deixem a 2 ja que aquest bucle fa que s'esperi massa
hem de buscar r18

$(((3\cdot255 - 1)\cdot255 - 1)\cdot$x-1)+4 = 13333 cicles = $(((3\cdot255 - 1)\cdot r18 - 1)\cdot$2-1)

r18 = 13333/(1· (3·255-1)) -1 = 16

Tasca prèvia 4

Escriviu el codi assemblador que configura l'Arduino amb la pota 8 i 13 (LED) com a sortida i la pota 7 com a entrada. Aprofiteu els pull-ups interns, que es controlen amb els registres DDRx, PORTx i el bit PUD del registre MCUCR.

bucle 1: 255*3-1
bucle2: (4+B1)(255-1)+3
bucle3: (4+B2)(255-1)+3
DDRB_o = 0x4
DDRD_i = 0x0A

```
main:
        ldi r16, 0xFF
out DDRB_o, r16
ldi r16, 0x00
out DDRD_i, r16
```

Tasca prèvia 5

Escriviu en el fitxer p3-codi1.s un programa en assemblador que generi un to de fto = 600 Hz en la pota 8 de l'Arduino.

per a generar una freqüència de 600Hz en la pota 8 haurem de posar r18 = 16 i el r19 = 2 r17 el deixarem a 255 tal y com hem calculat abans. Per posar la freqüència a la pota 8 haurem de escriure un OUT PORTB_o, r16

```
1  .set DDRB_o , 0x4
2  .equ PORTB_o , 0x5
3  .macro waitabit tot
4          ldi r19,2
5  wait3:
6          ldi r18,16
7  wait2:
8          ldi r17,0xFF
9  wait1:
10         subi r17,0x01
11         brne wait1
12         subi r18,0x01
13         brne wait2
14         subi r19,0x01
15         brne wait3
16 .endm
17 .global main
18 main:
19         ldi r16,0x1
20         out DDRB_o,r16
21 loop:
22         ldi r16,0x00
23         out PORTB_o,r16
24         waitabit 0x1
25         ldi r16,0x1
26         out PORTB_o,r16
27         waitabit 0x1
28         rjmp loop
```

Tasca prèvia 6

En el fitxer segon.s s'ha definit una macro amb un paràmetre.
Proposeu una manera d'aconseguir una funcionalitat similar (ús d'un paràmetre) quan des del programa principal del fitxer primer.s es fa una crida a la subrutina waitabit.

Tasca prèvia 7

Escriviu en el fitxer p3-codi2.s un programa en assemblador que generi un to de fto = 600 Hz en la pota 8 de l'Arduino, i que al mateix temps encengui i apagui el LED a una fLED = 2 Hz.

```
1  DDRB_o = 0x4
2  PORTB_o = 0x5
3  PINB = 0x3
4
5  waitabit:
6          ldi r18,17
7  wait2:
8          ldi r17,0xFF
9  wait1:
10         subi r17,0x01
11         brne wait1
12         subi r18,0x01
13         brne wait2
14         sbi PINB, 0x0
15         subi r20,1
16         brne waitabit
17         ret
18
19
20 .global main
21 main:
22         ldi r23, 0xFF
23         ldi r24, 0x00
24         ldi r16,0x1
25         out DDRB_o,r23
26 loop:
27         ldi r20, 255
28         rcall waitabit
29         ldi r20, 45
30         rcall waitabit
31         SBI PINB, 0x5
32         rjmp loop /* loop forever */
```

en el previ 7 hem de fer que els 600Hz sortin per la pota 8, que aixo ja ho havien aconseguit en previs anteriors. ara el que hem proposat fer es que per a que el port del led s'encengui cada 2Hz hem de activar la sortida d'aquest cada cop que en la pota 8 passen 300Hz, aixi aconsseguirem que per a cada 300Hz que passin a la pota 8, fagi toggle a la pota 13 del led. per tant tindrem 600Hz a la pota 8 i 2Hz a la pota 13 del led.

farem una mascara per que nomes es canvi el bit de la posició de memòria. per fer això carregarem un byte amb el bit que volem canviar per exemple:
LDI r20, 00100000

llegirem el port dentrada de la pota 8 i el multiplicarem per el valor de r20 que es el que habiem carregat el valor del bit que voliem canviar. a continuació farem una OR amb els dos valor, es a dir:
OR r21(on carreguem el valor que hem llegit), r20
i això ens canviara nomes el bit que voliem modificar de la sortida i deixarà tots els bits que hi havien amb el mateix valor. En el codi proposat no esta fet amb mascara ja que al laboratori no va funcionar i ho vam haver de fer amb la instrucció sbi que permet canviar un sol bit. en el següent previ si que ens va funcionar per tant en allà si que surt el mètode de la mascara.

Tasca prèvia 8

Escriviu en el fitxer p3-codi3.s un programa en assemblador per tal que quan a la pota 7 de l'Arduino hi hagi una tensió de 0 V encengui el LED i al cap d'1.5 s l'apagui. Un cop apagat, es torna a comprovar el valor de la pota 7, de manera que si encara hi ha una tensió de 0 V el LED es torna a encendre durant 1.5 s.

```
DDRB_o = 0x4
PORTB_o = 0x5
PINB = 0x3
DDRD_o = 0x0A
PORTD_o = 0x0B
wait600Hz:
        ldi r18,17
wait2:
        ldi r17,0xFF
wait1:
        subi r17,0x01
        brne wait1
        subi r18,0x01
        brne wait2
        subi r20,1
        brne wait600Hz
        ret
.global main
main:
        ldi r16, 0xFF
        ldi r19, 0x00
        ldi r22, 0b10000000
        out DDRB_o,r16
        out DDRD_o, r19

comprovacio_led_pin7:
        in r23, 0x09 /*p0000000 & 10000000*/
        and r23, r22
        brne comprovacio_led_pin7
        rjmp loop1500ms

loop1500ms:
        sbi PINB, 0x5
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 15
        rcall wait600Hz
        SBI PINB, 0x5
        rjmp comprovacio_led_pin7 /* loop forever */
```

El que hi ha en aquest codi del avr es simple, hem de comprovar tota l'estona quin valor hi ha en la pota 7. un cop a la pota 7 hi ha un valor de 0V el bucle loop1500ms s'activa a traves d'un rjmp del bucle de comprovació de la pota 7. un cop aques bucle ha acabat de esperar els 1,5 s que dura el bucle s'acabarà i tornarà a comprovar el pin7.

Això ens diu que si la pota 7 esta tota l'estona a 0V el led estara continuament ences en el bucle de esperar 1,5s i al acabar el bucle gastara 2 cicles imperceptibles a simple vista on el led estarà apagat i despres es tornara a encendre i aixi succesivament, per tant el que veurem si tota l'estona esta a 0V es el led continuament ences amb dos cicles apagat imperceptibles per a l'ull huma cada 1,5s.

En canvi si posem la pota 7 a 0V i posteriorment la posem a 5V el que farà es que esperarà 1,5 segons amb el led ences on no comprobara el valor del pin 7 i per tant estara els 1,5 segons ences fins que acabi el bucle i torni a comprovar si esta a 5V o a 0V i dependrà del valor que es tori a encendre o no.

Tasca prèvia 9

Modifiqueu el fitxer p3-codi3.s, i anomeneu-lo p3-codi4.s, per tal que quan a la pota 7 de l'Arduino hi hagi una tensió de 0 V es generi un to de fto = 600 Hz durant 1.5 s en la pota 8 de l'Arduino.

```
DDRB_o = 0x4
PORTB_o = 0x5
PINB = 0x3
DDRD_o = 0x0A
PORTD_o = 0x0B

wait600Hz:
        ldi r18,17
wait2:
        ldi r17,0xFF
wait1:
        subi r17,0x01
        brne wait1
        subi r18,0x01
        brne wait2
        sbi PINB, 0x0
        subi r20,1
        brne wait600Hz
        ret
.global main
main:
        ldi r16, 0xFF
        ldi r19, 0x00
        ldi r22, 0b10000000
        out DDRB_o,r16
        out DDRD_o, r19
comprovacio_led_pin7:
        in r23, 0x09 /*p0000000 & 10000000*/
        and r23, r22
        brne comprovacio_led_pin7
        rjmp loop1500ms
loop1500ms:
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 255
        rcall wait600Hz
        ldi r20, 15
        rcall wait600Hz
        SBI PINB, 0x5
        rjmp comprovacio_led_pin7 /* loop forever */
```

En aquest previ el que farem es utilitzar el mateix sistema que en l'anterior previ 8 peerò en comptes de modificar la sortida del led el que modificarem serà la sortida 8 per tal que surti una freqüencia de 600Hz. Vam utilitzar el metode de mascara explicat anteriorment per tal de canviar la sortida del pin 8.

Tasques

Tasca 10

Assembleu els fitxers primer.s i segon.s, transferiu-los a l'Arduino i comproveu que no es poden observar diferències en l'efecte que tenen sobre el LED. Des-assembleu els dos programes.
avr−objdump −S nom.elf > nom.disam

Descriviu les diferències entre els dos fitxers .asm.

```
  68:   11 24           eor     r1, r1
  6a:   1f be           out     0x3f, r1        ; 63
  6c:   cf ef           ldi     r28, 0xFF       ; 255
  6e:   d8 e0           ldi     r29, 0x08       ; 8
  70:   de bf           out     0x3e, r29       ; 62
  72:   cd bf           out     0x3d, r28       ; 61
  74:   0e 94 4a 00     call    0x94    ; 0x94 <main>
  78:   0c 94 53 00     jmp     0xa6    ; 0xa6 <_exit>

0000007c <__bad_interrupt>:
  7c:   0c 94 00 00     jmp     0       ; 0x0 <__vectors>

00000080 <waitabit>:
  80:   39 e2           ldi     r19, 0x29       ; 41

00000082 <wait3>:
  82:   2f ef           ldi     r18, 0xFF       ; 255

00000084 <wait2>:
  84:   1f ef           ldi     r17, 0xFF       ; 255

00000086 <wait1>:
  86:   11 50           subi    r17, 0x01       ; 1
  88:   f1 f7           brne    .-4             ; 0x86 <wait1>
  8a:   21 50           subi    r18, 0x01       ; 1
  8c:   d9 f7           brne    .-10            ; 0x84 <wait2>
  8e:   31 50           subi    r19, 0x01       ; 1
  90:   c1 f7           brne    .-16            ; 0x82 <wait3>
  92:   08 95           ret

00000094 <main>:
  94:   0f ef           ldi     r16, 0xFF       ; 255
  96:   04 b9           out     0x04, r16       ; 4

00000098 <loop>:
  98:   00 e0           ldi     r16, 0x00       ; 0
  9a:   05 b9           out     0x05, r16       ; 5
  9c:   f1 df           rcall   .-30            ; 0x80 <waitabit>
  9e:   0f ef           ldi     r16, 0xFF       ; 255
  a0:   05 b9           out     0x05, r16       ; 5
  a2:   ee df           rcall   .-36            ; 0x80 <waitabit>
  a4:   f9 cf           rjmp    .-14            ; 0x98 <loop>

000000a6 <_exit>:
  a6:   f8 94           cli

000000a8 <__stop_program>:
  a8:   ff cf           rjmp    .-2             ; 0xa8 <__stop_program>
eric@eric-pad:~/Escritorio/pract3_lab$
```

```
00000068 <__ctors_end>:
  68:   11 24           eor     r1, r1
  6a:   1f be           out     0x3f, r1        ; 63
  6c:   cf ef           ldi     r28, 0xFF       ; 255
  6e:   d8 e0           ldi     r29, 0x08       ; 8
  70:   de bf           out     0x3e, r29       ; 62
  72:   cd bf           out     0x3d, r28       ; 61
  74:   0e 94 40 00     call    0x80    ; 0x80 <main>
  78:   0c 94 59 00     jmp     0xb2    ; 0xb2 <_exit>

0000007c <__bad_interrupt>:
  7c:   0c 94 00 00     jmp     0       ; 0x0 <__vectors>

00000080 <main>:
  80:   0f ef           ldi     r16, 0xFF       ; 255
  82:   04 b9           out     0x04, r16       ; 4

00000084 <loop>:
  84:   00 e0           ldi     r16, 0x00       ; 0
  86:   05 b9           out     0x05, r16       ; 5
  88:   39 e2           ldi     r19, 0x29       ; 41
  8a:   2f ef           ldi     r18, 0xFF       ; 255
  8c:   1f ef           ldi     r17, 0xFF       ; 255
  8e:   11 50           subi    r17, 0x01       ; 1
  90:   f1 f7           brne    .-4             ; 0x8e <loop+0xa>
  92:   21 50           subi    r18, 0x01       ; 1
  94:   d9 f7           brne    .-10            ; 0x8c <loop+0x8>
  96:   31 50           subi    r19, 0x01       ; 1
  98:   c1 f7           brne    .-16            ; 0x8a <loop+0x6>
  9a:   0f ef           ldi     r16, 0xFF       ; 255
  9c:   05 b9           out     0x05, r16       ; 5
  9e:   39 e2           ldi     r19, 0x29       ; 41
  a0:   2f ef           ldi     r18, 0xFF       ; 255
  a2:   1f ef           ldi     r17, 0xFF       ; 255
  a4:   11 50           subi    r17, 0x01       ; 1
  a6:   f1 f7           brne    .-4             ; 0xa4 <loop+0x20>
  a8:   21 50           subi    r18, 0x01       ; 1
  aa:   d9 f7           brne    .-10            ; 0xa2 <loop+0x1e>
  ac:   31 50           subi    r19, 0x01       ; 1
  ae:   c1 f7           brne    .-16            ; 0xa0 <loop+0x1c>
  b0:   e9 cf           rjmp    .-46            ; 0x84 <loop>

000000b2 <_exit>:
  b2:   f8 94           cli

000000b4 <__stop_program>:
  b4:   ff cf           rjmp    .-2             ; 0xb4 <__stop_program>
eric@eric-pad:~/Escritorio/pract3_lab$
```

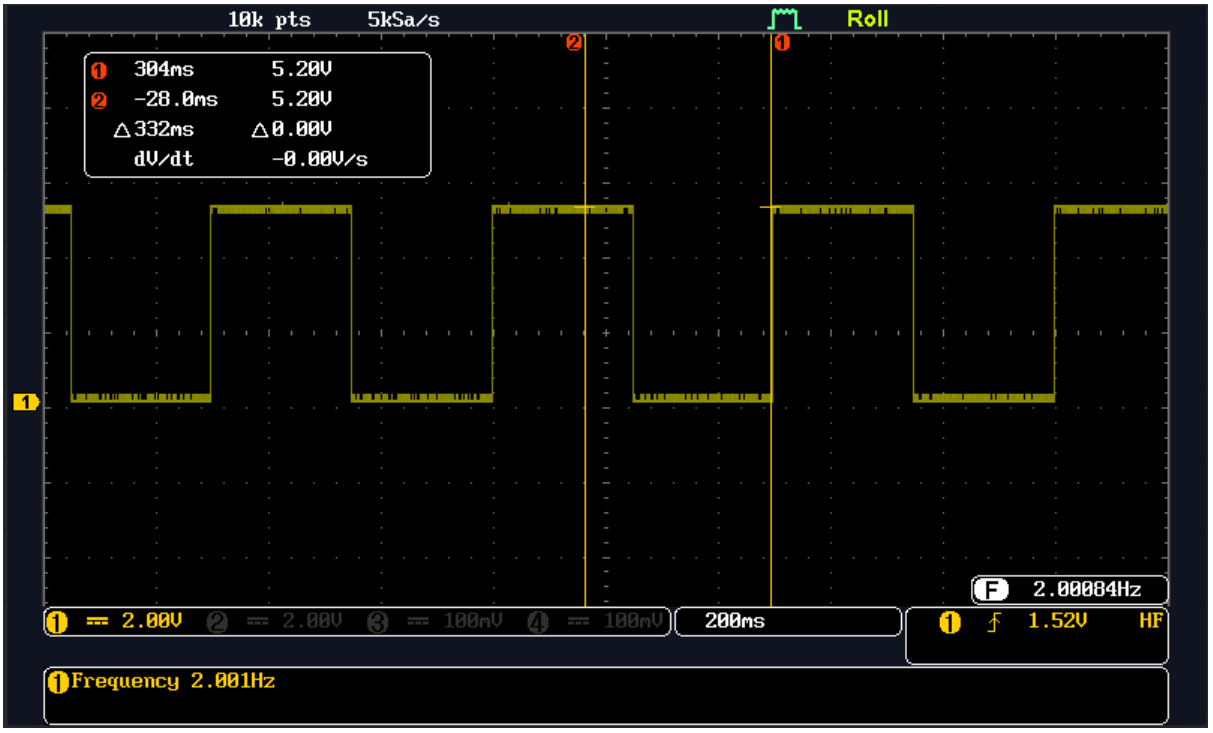
Tasca 11

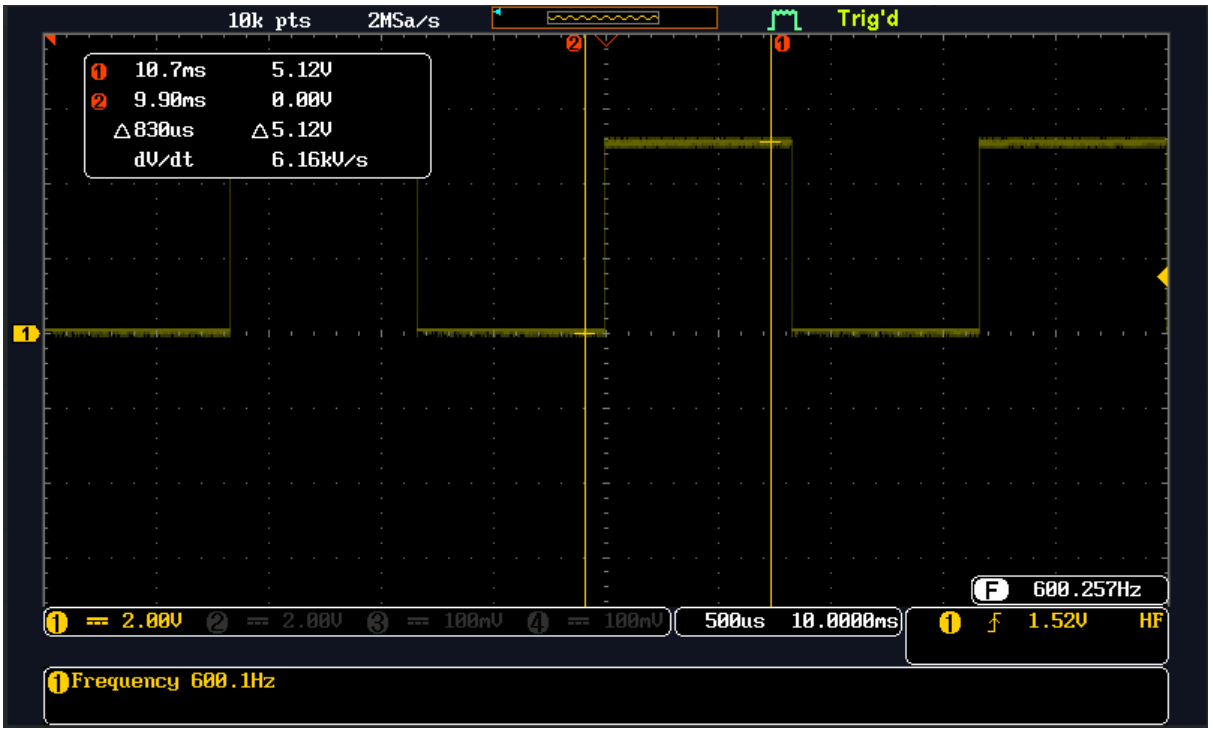
Comproveu el correcte funcionament de p3-codi1.s.

podem veure que el osciloscopi ens mostra el correcte funcionament del codi ja que ens marca que te una fto = 600Hz que es el que voliem

Tasca 12

Comproveu el correcte funcionament de p3-codi2.s.

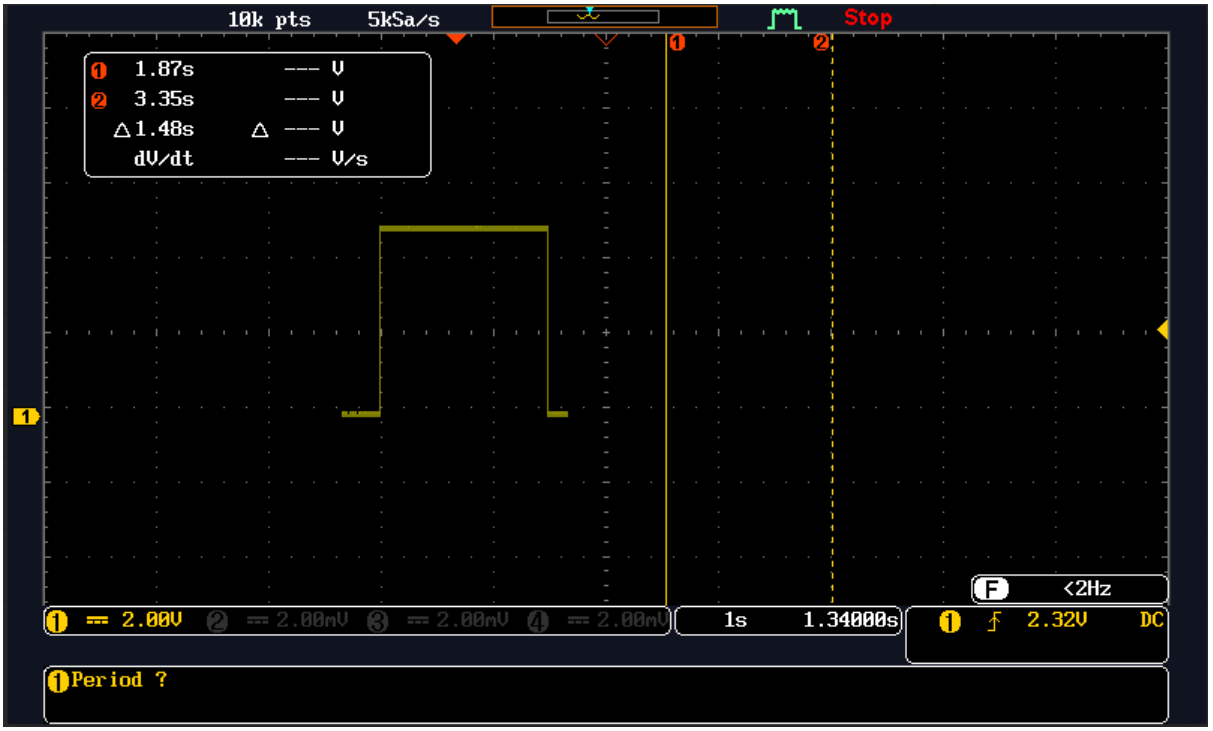




Com podem veure el osciloscopi marca 600Hz i 2Hz en les potes desitjades.

Tasca 13

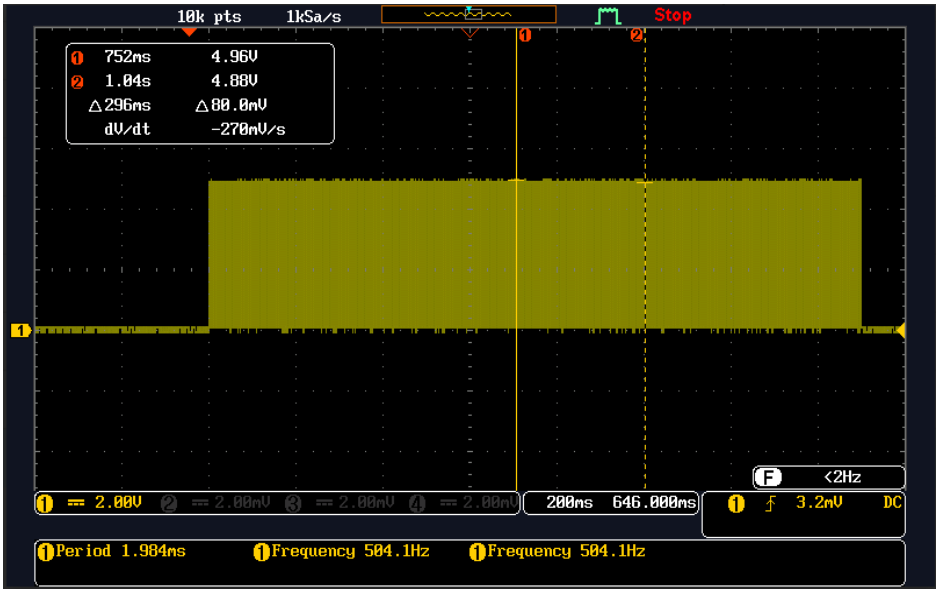
Comproveu el correcte funcionament de p3-codi3.s.



Aqui podem veure com al posar el pin 7 a 0V i seguidament el posem a 5V el que teoricament hauria de fer es esperar amb el led ences 1,5 s i apagar-se perque estara a 5V. I efectivament es el que ens mostra l'osciloscopi, 1,5s exactes.

Tasca 14

Comproveu el correcte funcionament de p3-codi4.s.



El que hi trobem aqui es la mateixa dinamica que en l'anterior, posem el pin7 a 0V i seguidament a 5V i hauriem de veure a la pota 8 els 600Hz que efectivament es el que ens mostra l'osciloscopi.