Digitális technika 2.

BMEVIIIAA06

7. előadás

Assembly programozás

(megszakítások és időzítők)

Láttuk eddig:

- A programjainkat szubrutinokba szervezhetjük
- Minden szubrutint akár több helyről is meghívhatunk, mindig a hívás helyére tér vissza.
- Minden utasítás ideje pontosan ismert, így minden program végrehajtási ideje előre kiszámítható
  - » Kivéve ha pl. ciklusszám függ valamilyen előre nem ismert bejövő adattól
- Ha várakozni akarunk, valami feleslegeset kell csináltatni a processzorral
  - » Ez nem mindig jó mert pl:
    - » A várakozás 100%-ban lefoglalja a processzort
    - » Ha néha van épp mit csináljunk, de várakozni is kell, nehéz kiszámítani mennyit várakozzunk.
    - » Pontosan ugyanannyit szeretnénk várakozni akkor is ha van mit csinálni és akkor is ha épp nincs.

Programmal ellenőrzött készenlét

- Külső eseményre várakozunk (pl: gombnyomás)

```
ciklus:
    BTST PORTA,#9
    BRA NZ, ciklus
```

- Amíg várunk, mást nem tudunk csinálni
- Ha mást is akarunk (több külső esemény), bonyolult lesz a program

```
ciklus:
    BTST PORTA,#9
    BRA Z, egyik_gomb_megnyomva
    ……
    BTST PORTB,#4
    bra Z, masik_gomb_megnyomva
    bra ciklus
```

- A kiszolgálás sorrendje a program sorrendjétől függ
- Ha egy esemény kiszolgálása alatt egy fontosabb esemény is bekövetkezik, azt csak a kiszolgálás után fogjuk észrevenni
  - » Rettentő bonyolult lenne, ha a kiszolgálás közben akarnánk észrevenni

Megoldás: Megszakítás (Interrupt, IT)

- Megszakítás: automatikus szubrutinhívás
  - Egy esemény hatására hívódjon meg automatikusan egy adott szubrutin.
- Természetesen az ilyen szubrutinokra vonatkoznak további követelmények:
  - Egyértelmű kell legyen hol van a belépési pontja
  - Nem változtathat meg semmit, aminek a változására a program többi része nem számít
    - Általában egyik számozott regisztert, de még a flageket sem
  - Meg kell szüntesse az őt kiváltó eseményt
    - Különben újra és újra bekövetkezne

Megszakítás

- Nagyon sok megszakítási forrás lehet
  - Majdnem minden periféria kérhet megszakítást, van ami többfélét is
  - Speciális „belső” események is „megszakítást” váltanak ki (pl. nullával osztás vagy stack error)
- Minden megszakítási forrás egy saját „megszakításkezelő szubrutint”, vagy „megszakítási szubrutint” indít
  - » ISR: interrupt service routine

Vektortábla

- A megszakítási szubrutinok kezdőcímei (belépési pontjai) tetszőleges helyen lehetnek.
  - » Akkor honnan tudja a processzor, hogy hol vannak?
    - » Van egy speciális hely a memóriában, ahol ezek fel vannak írva…
- Vektortábla (IVT: Interrupt Vector Table)
  - Egy táblázat a programmemóriában 0x0004-től 0x00FF-ig
  - Minden sora egy-egy ISR kezdőcíme
  - Ezért nem szoktuk a programot a memória elején kezdeni…

Vektortábla

- A program reset után a 0 címen kezdődik, ott épp elfér egy goto ugró utasítás (2 szó)
- Utána 126 db megszakítás rutin kezdőcímének van hely
- Általában a kisebb típusokon nincs ennyi megszakítás forrás sem, nem mind van használva.
  - » *de pl. van olyan nagyobb típus, ahol ez még nem is elég és ott 1FF-ig tart a vektortábla*

Reset – GOTO Instruction	000000h
Reset – GOTO Address	000002h
Oscillator Fail Trap Vector	000004h
Address Error Trap Vector	
General Hard Trap Vector	
Stack Error Trap Vector	
Math Error Trap Vector	
Reserved	
General Soft Trap Vector	
Reserved	
Interrupt Vector 0	000014h
Interrupt Vector 1	
—	
—	
Interrupt Vector 52	00007Ch
Interrupt Vector 53	00007Eh
Interrupt Vector 54	000080h
—	
—	
Interrupt Vector 116	0000FCh
Interrupt Vector 117	0000FEh

Hogyan kerül a vektortáblába a szubrutinom kezdőcime?

Odatehetjük „kézzel is”, fordítódirektívákkal:

```
.section ivt, code
.org 0x0014 ;első megszakítás (INT0) kezdőcíme
.pword paddr(szubrutin)



.text
szubrutin:
…
```

De nem akarjuk…

Hogyan kerül a vektortáblába a szubrutinom kezdőcime?

Inkább kérjük meg a linkert:

```
.global __INT0Interrupt  ; figyelem, két _ jel (__)


…
.text
…
__INT0Interrupt:
…
```

Csak egy adott nevű szubrutint kell definiálni, és a linker mindent megold.

Hogyan kerül a vektortáblába a szubrutinom kezdőcime?

- Honnan tudom, mi a neve az adott megszakításnak?
- Adatlap 8-2 táblázat:

TABLE 8-2: INTERRUPT VECTOR DETAILS

Interrupt Description	MPLAB® XC16 ISR Name	Vector #	IRQ #	IVT Address	Interrupt Bit Location		
					Flag	Enable	Priority
	Highest Natural Order Priority						
External Interrupt 0	_INT0Interrupt	8	0	000014h	IFS0[0]	IEC0[0]	IPC0[2:0]
Input Capture 1	_IC1Interrupt	9	1	000016h	IFS0[1]	IEC0[1]	IPC0[6:4]
Output Compare 1	_OC1Interrupt	10	2	000018h	IFS0[2]	IEC0[2]	IPC0[10:8]
Timer1	_T1Interrupt	11	3	00001Ah	IFS0[3]	IEC0[3]	IPC0[14:12]
Direct Memory Access 0	_DMA0Interrupt	12	4	00001Ch	IFS0[4]	IEC0[4]	IPC1[2:0]
Input Capture 2	_IC2Interrupt	13	5	00001Eh	IFS0[5]	IEC0[5]	IPC1[6:4]
Output Compare 2	_OC2Interrupt	14	6	000020h	IFS0[6]	IEC0[6]	IPC1[10:8]
Timer2	_T2Interrupt	15	7	000022h	IFS0[7]	IEC0[7]	IPC1[14:12]
Timer3	_T3Interrupt	16	8	000024h	IFS0[8]	IEC0[8]	IPC2[2:0]
SPI1 General	_SPI1Interrupt	17	9	000026h	IFS0[9]	IEC0[9]	IPC2[6:4]
SPI1 Transfer Done	_SPI1TXInterrupt	18	10	000028h	IFS0[10]	IEC0[10]	IPC2[10:8]
UART1 Receiver	_U1RXInterrupt	19	11	00002Ah	IFS0[11]	IEC0[11]	IPC2[14:12]
UART1 Transmitter	_U1TXInterrupt	20	12	00002Ch	IFS0[12]	IEC0[12]	IPC3[2:0]
A/D Converter 1	_ADC1Interrupt	21	13	00002Eh	IFS0[13]	IEC0[13]	IPC3[6:4]

Jelző és vezérlő bitek

- Minden megszakításhoz tartozik:
  - Saját jelzőbit (IF, flag)
    - » 1-essel jelzi, hogy az adott megszakítást kiváltó esemény bekövetkezett-e

TABLE 8-2: INTERRUPT VECTOR DETAILS

Interrupt Description	MPLAB® XC16 ISR Name	Vector #	IRQ #	IVT Address	Interrupt Bit Location		
					Flag	Enable	Priority
Highest Natural Order Priority							
External Interrupt 0	_INT0Interrupt	8	0	000014h	IFS0[0]	IEC0[0]	IPC0[2:0]
Input Capture 1	_IC1Interrupt	9	1	000016h	IFS0[1]	IEC0[1]	IPC0[6:4]
Output Compare 1	_OC1Interrupt	10	2	000018h	IFS0[2]	IEC0[2]	IPC0[10:8]
Timer1	_T1Interrupt	11	3	00001Ah	IFS0[3]	IEC0[3]	IPC0[14:12]
Direct Memory Access 0	_DMA0Interrupt	12	4	00001Ch	IFS0[4]	IEC0[4]	IPC1[2:0]
Input Capture 2	_IC2Interrupt	13	5	00001Eh	IFS0[5]	IEC0[5]	IPC1[6:4]
Output Compare 2	_OC2Interrupt	14	6	000020h	IFS0[6]	IEC0[6]	IPC1[10:8]
Timer2	_T2Interrupt	15	7	000022h	IFS0[7]	IEC0[7]	IPC1[14:12]
Timer3	_T3Interrupt	16	8	000024h	IFS0[8]	IEC0[8]	IPC2[2:0]
SPI1 General	_SPI1Interrupt	17	9	000026h	IFS0[9]	IEC0[9]	IPC2[6:4]
SPI1 Transfer Done	_SPI1TXInterrupt	18	10	000028h	IFS0[10]	IEC0[10]	IPC2[10:8]
UART1 Receiver	_U1RXInterrupt	19	11	00002Ah	IFS0[11]	IEC0[11]	IPC2[14:12]
UART1 Transmitter	_U1TXInterrupt	20	12	00002Ch	IFS0[12]	IEC0[12]	IPC3[2:0]
A/D Converter 1	_ADC1Interrupt	21	13	00002Eh	IFS0[13]	IEC0[13]	IPC3[6:4]

Jelző és vezérlő bitek

– Saját engedélyező bit (IEC, enable control)

» 1-esre állíthatjuk, hogy megengedjük az adott megszakítás szubrutin végrehajtását

» Kezdetben, reset után mindegyik 0.

TABLE 8-2: INTERRUPT VECTOR DETAILS

Interrupt Description	MPLAB® XC16 ISR Name	Vector #	IRQ #	IVT Address	Interrupt Bit Location		
					Flag	Enable	Priority
Highest Natural Order Priority							
External Interrupt 0	_INT0Interrupt	8	0	000014h	IFS0[0]	IEC0[0]	IPC0[2:0]
Input Capture 1	_IC1Interrupt	9	1	000016h	IFS0[1]	IEC0[1]	IPC0[6:4]
Output Compare 1	_OC1Interrupt	10	2	000018h	IFS0[2]	IEC0[2]	IPC0[10:8]
Timer1	_T1Interrupt	11	3	00001Ah	IFS0[3]	IEC0[3]	IPC0[14:12]
Direct Memory Access 0	_DMA0Interrupt	12	4	00001Ch	IFS0[4]	IEC0[4]	IPC1[2:0]
Input Capture 2	_IC2Interrupt	13	5	00001Eh	IFS0[5]	IEC0[5]	IPC1[6:4]
Output Compare 2	_OC2Interrupt	14	6	000020h	IFS0[6]	IEC0[6]	IPC1[10:8]
Timer2	_T2Interrupt	15	7	000022h	IFS0[7]	IEC0[7]	IPC1[14:12]
Timer3	_T3Interrupt	16	8	000024h	IFS0[8]	IEC0[8]	IPC2[2:0]
SPI1 General	_SPI1Interrupt	17	9	000026h	IFS0[9]	IEC0[9]	IPC2[6:4]
SPI1 Transfer Done	_SPI1TXInterrupt	18	10	000028h	IFS0[10]	IEC0[10]	IPC2[10:8]
UART1 Receiver	_U1RXInterrupt	19	11	00002Ah	IFS0[11]	IEC0[11]	IPC2[14:12]
UART1 Transmitter	_U1TXInterrupt	20	12	00002Ch	IFS0[12]	IEC0[12]	IPC3[2:0]
A/D Converter 1	_ADC1Interrupt	21	13	00002Eh	IFS0[13]	IEC0[13]	IPC3[6:4]

Jelző és vezérlő bitek

– Saját prioritás állító bitek (IPC, priority control)

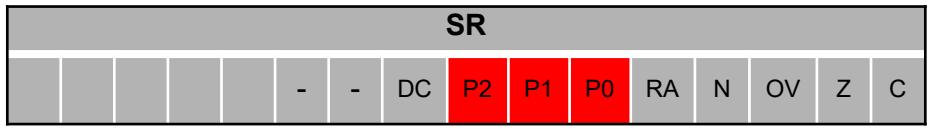
» Minden forráshoz 3 bit tartozik (0-tól 7-ig tartó szám állítható be rajta)

TABLE 8-2: INTERRUPT VECTOR DETAILS

Interrupt Description	MPLAB® XC16 ISR Name	Vector #	IRQ #	IVT Address	Interrupt Bit Location		
					Flag	Enable	Priority
Highest Natural Order Priority							
External Interrupt 0	_INT0Interrupt	8	0	000014h	IFS0[0]	IEC0[0]	IPC0[2:0]
Input Capture 1	_IC1Interrupt	9	1	000016h	IFS0[1]	IEC0[1]	IPC0[6:4]
Output Compare 1	_OC1Interrupt	10	2	000018h	IFS0[2]	IEC0[2]	IPC0[10:8]
Timer1	_T1Interrupt	11	3	00001Ah	IFS0[3]	IEC0[3]	IPC0[14:12]
Direct Memory Access 0	_DMA0Interrupt	12	4	00001Ch	IFS0[4]	IEC0[4]	IPC1[2:0]
Input Capture 2	_IC2Interrupt	13	5	00001Eh	IFS0[5]	IEC0[5]	IPC1[6:4]
Output Compare 2	_OC2Interrupt	14	6	000020h	IFS0[6]	IEC0[6]	IPC1[10:8]
Timer2	_T2Interrupt	15	7	000022h	IFS0[7]	IEC0[7]	IPC1[14:12]
Timer3	_T3Interrupt	16	8	000024h	IFS0[8]	IEC0[8]	IPC2[2:0]
SPI1 General	_SPI1Interrupt	17	9	000026h	IFS0[9]	IEC0[9]	IPC2[6:4]
SPI1 Transfer Done	_SPI1TXInterrupt	18	10	000028h	IFS0[10]	IEC0[10]	IPC2[10:8]
UART1 Receiver	_U1RXInterrupt	19	11	00002Ah	IFS0[11]	IEC0[11]	IPC2[14:12]
UART1 Transmitter	_U1TXInterrupt	20	12	00002Ch	IFS0[12]	IEC0[12]	IPC3[2:0]
A/D Converter 1	_ADC1Interrupt	21	13	00002Eh	IFS0[13]	IEC0[13]	IPC3[6:4]

Prioritás

- Minden megszakításhoz választhatunk egy prioritás szintet (0-7)
- Az éppen futó kódnak is van egy aktuális prioritás szintje
  - » SR-ben ott a 3 bit



- Ha az egyébként engedélyezett és kiváltott megszakítás (IEC=1 és IF=1) **prioritása nagyobb**, mint az aktuális prioritás, akkor jut csak érvényre.

Prioritás

- Kezdetben (reset után):
  - Minden megszakítás beállított prioritása 4-es (0b100)
  - A futó kód prioritása (SR bitjeinek állása): 0-s (0b000)
    - » Így minden megszakítás érvényre tud jutni.
- Ha SR-ben a 3 bitet mind 1-esre (7-re) állítjuk, gyakorlatilag letiltottuk az összes megszakítást
  - » *Csak majdnem az összeset, de ezt majd később…*
- Ha valamelyik megszakítás prioritását 0-ra állítjuk, azzal gyakorlatilag azt a megszakítást tiltottuk le, mert sosem lehet nagyobb a prioritása semminél.

Prioritás

- A prioritás mindig nagyobb kell legyen, mint az aktuális
- Ha egy megszakítás érvénye jut „megemeli” a prioritást
  - » Vagyis még egyszer biztosan nem tud érvényre jutni *(lesz időnk kezelni)*
- A megszakítási szubrutin végén a prioritást vissza kell állítani az előző állapotra
  - » Vagyis azt meg kell jegyezni valahol a megszakításba belépéskor
    - » Jegyezzük meg a stack-en pont úgy, mint a visszatérési címet
      - » Mentsük el SR-t is a megszakításba belépéskor

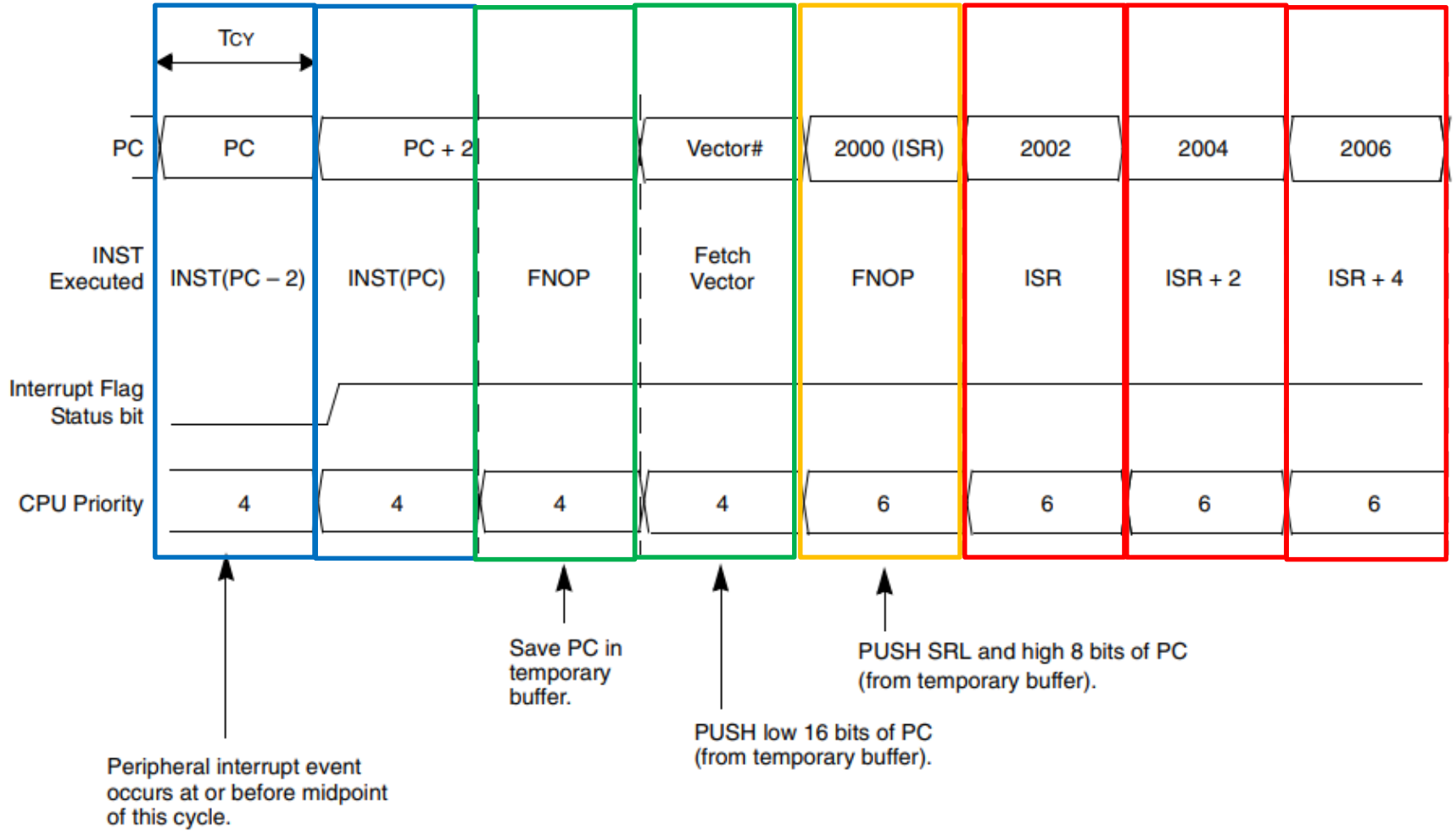
Visszatérés megszakításból

- A megszakításba belépés az SR-t is menti
- A „sima” visszatérés (**`return`** vagy **`retlw`**) nem tölti azt vissza
  - » Kell egy speciális utasítás, ami igen
- Speciális utasítás:

  `retfie`

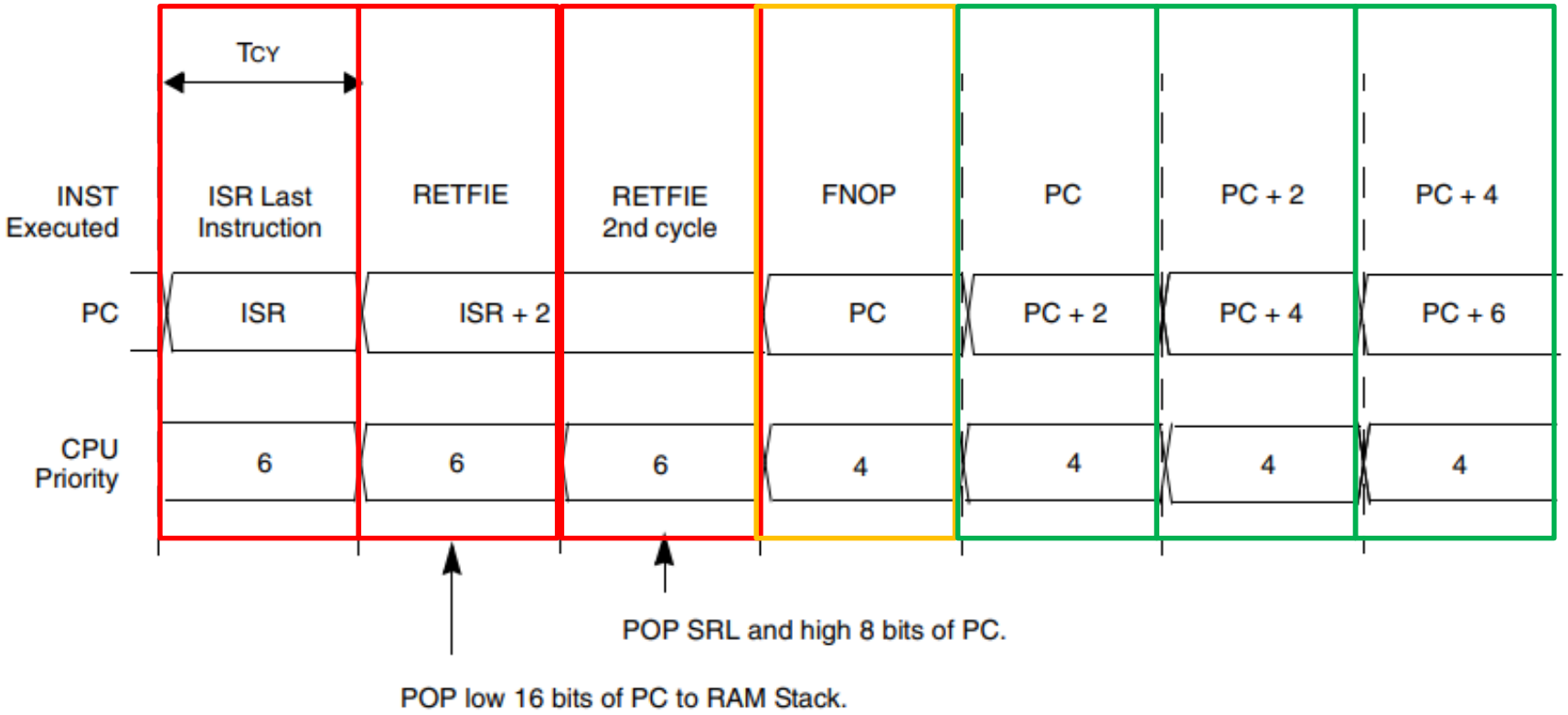
A megszakítási szubrutinok mindig csak retfie-vel térhetnek vissza.

Belépés időzítése



Note: Where FNOP is a forced `NOP` instruction automatically inserted by the CPU.

Visszatérés időzítése



Hogyan írjunk megszakítás szubrutint

- 1. Tegyük jó helyre
  - nevezzük el megfelelően
  - deklaráljuk a címkéjét „.global”-ként
- 2. Mentsünk el benne minden regisztert, amit használunk
- 3. Töröljük a flag-et, ami okozta a megszakítást
- 4. Töltsünk vissza mindent, amit elmentettünk
- 5. Térjünk vissza jól: **`retfie`**

Hogyan lehet letiltani egy megszakítást?

- Minden megszakításnak van:
  - Prioritása: 0:letiltva, 1-7 engedélyezve
    - » Reset után ez 4-es, vagyis engedélyez
  - Saját engedélyező bitje: 0: letiltva, 1: engedélyezve
    - » Reset után ezek 0-k, vagyis minden tiltva van
- Közösen az összes megszakítás tiltható:
  - Globális megszakítás engedélyező bit (INTCON2 regiszter, GIE bit)
    - » Reset után ez 1-es, vagyis engedélyezve van alapból
  - Speciális vezérlő utasítás, amivel adott ideig tiltható minden megszakítás: disi #konstans
    - » Adott utasításciklusnyi időre letiltja az összes megszakítást, majd automatikusan újra engedélyezi

Külső megszakítás bemenet használata

- A lehető legegyszerűbb megszakítás-forrás:
  külső megszakítás bemenet (external interrupt)
- Minden mikrokontrollernek van valahány külső megszakítás-bemenete
  - Olyan speciális port bemenet, amelynek fel- vagy lefutó éle megszakítást képes okozni.
  - A tanult PIC24-nek 5 ilyen van *(hogy konkrétan melyik láb az, majd később…)*
    - » *INT0 - INT4*
      - » *Konkrétan akármelyik láb lehet, mert egy-egy nagy multiplexer segít választani*
- Állítható, hogy felfutó vagy lefutó élre kérjen
  - ITCON2 regiszter INTxEP bit

Külső megszakítás bemenet használata

- Írjunk egy megszakítási szubrutint, amely a külső megszakítás-bemeneten érkező lefutó él hatására a portC 4. bitjének aktuális állapotát invertálja. Állítsuk a megszakítás prioritását a lehető legnagyobbra.
- Lépések:
  1. Írjuk meg a megszakítási szubrutint
  2. Írjuk meg a főprogram megfelelő részét
     1. Állítsuk be a külső megszakítás-bemenetet lefutó élre
     2. Állítsunk prioritást
     3. Engedélyezzük a külső megszakítást

Külső megszakítás bemenet használata

- Megszakítás szubrutin:

Interrupt Description	MPLAB® XC16 ISR Name	Vector #	IRQ #	IVT Address	Interrupt Bit Location		
					Flag	Enable	Priority
Highest Natural Order Priority							
External Interrupt 0	_INT0Interrupt	8	0	000014h	IFS0[0]	IEC0[0]	IPC0[2:0]

```
.global __INT0Interrupt

.text
__INT0Interrupt:
     blcr       IFS0,#0    ;töröljük le a jelzőflag-et!
     btg        LATC,#4   ;invertáljuk a portlábat
retfie                     ;kész vagyunk
```

Külső megszakítás bemenet használata

- Főprogram:

```
main:
        bset    INTCON2,#INT0EP
        mov     #IPC0,w0
        mov     #0x0007,w1    ;0b111
        ior     w1,[w0],[w0]  ;prioritás állítás (több bit egyszerre)
        blcr    IFS0,#0  ;töröljük le a jelzőflag-et, a következő éltől kezdünk
        bset    IEC0,#0  ;engedélyezzük a megszakítást


vegtelen:
        bra vegtelen          ;innentől nincs mit csinálni
```

REGISTER 8-4: INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-1	R-0	R/W-0	U-0	U-0	U-0	U-0	R/W-0
GIE	DISI	SWTRAP	—	—	—	—	AIVTEN
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT4EP	INT3EP	INT2EP	INT1EP	INT0EP
bit 7							bit 0

bit 0 **INT0EP:** External Interrupt 0 Edge Detect Polarity Select bit
1 = Interrupt on negative edge
0 = Interrupt on positive edge

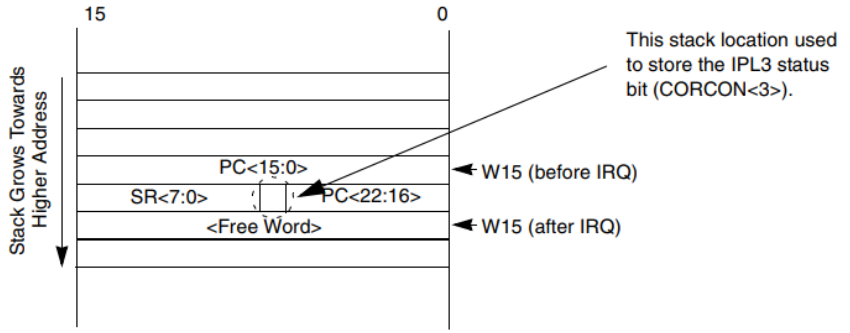
Interrupt Description	MPLAB® XC16 ISR Name	Vector #	IRQ #	IVT Address	Interrupt Bit Location		
					Flag	Enable	Priority
Highest Natural Order Priority							
External Interrupt 0	_INT0Interrupt	8	0	000014h	IFS0[0]	IEC0[0]	IPC0[2:0]

Trap-ek

- A megszakítások (interrupt)
  - mind valamilyen periféria miatt következnek be
  - letilthatók
- A kivételek (trap)
  - Nem letilthatók
  - Általában valamilyen belső esemény miatt következnek be
  - Jellemzően hibajelzésre használhatók

Trap-ek

- A trap-eknek saját prioritásuk van (8-tól 15-ig)
  - Emiatt mindig nagyobb prioritásúak, mint az összes többi megszakítás
    - » *Van egy 4., rejtett prioritás-bit is, ez is elmentődik a stack-be, pont van helye…*
  - Mindig érvényre tudnak jutni
  - Egy trap-et is megszakíthat egy másik
    - » Bizonyos esetekben



- Összesen 8 trap-nek van hely (az első 8) hely a vektortáblában.

Trap-ek

- Összesen 8 trap-nek van hely (az első 8) hely a vektortáblában.
- Összesen 5db van:
  - Kiválasztott órajel leállt
    - Ilyenkor automatikus átkapcsolás történik
  - Nem létező memóriát címeztek
  - Programmemória hiba
    - » Ez kiváltható szoftverből is (általános hardver hiba)
    - » *SGHT: software generated hard trap*
  - Stack overflow
  - Nullával osztás

TABLE 8-1: TRAP VECTOR DETAILS

Trap Description	MPLAB® XC16 ISR Name	Vector #	IVT Address
Oscillator Failure	_OscillatorFail	0	000004h
Address Error	_AddressError	1	000006h
General Hardware Error – ECCBDE	_NVMError	2	000008h
General Hardware Error – SGHT	_NVMError	2	000008h
Stack Error	_StackError	3	00000Ah
Math Error – DIV0ERR	_MathError	4	00000Ch
Reserved	_ReservedTrap5	5	00000Eh
Reserved	_ReservedTrap6	6	000010h
Reserved	_ReservedTrap7	7	000012h

Trap-ek

- A trap-ek egymást meg tudják szakítani (a prioritásuk szerint)
- Vannak olyan kombinációk, amik esetén a program végrehajtás nem lehetséges
  - Pl: Az órajel az előbb szűnt meg, de az ezt kiszolgáló megszakítási szubrutin memóriahibát okoz…
    - » Ilyen esetekben a processzor automatikusan újraindul
    - » *Egészen pontosan: Ha az első 4 trap esetén egy magasabb prioritású trap szubrutinja alacsonyabb prioritású trap-et okozna, az reset.*

Mi történik, ha nem írjuk meg a megszakítás szubrutint?

- Eddig se írtunk egy megszakítási szubrutint sem…
- Mivel nem is engedélyeztünk egy megszakítást sem, legfeljebb a trap-ek tudnak érvényre jutni
  - » Akkor úgyis valami hiba van
- Ha nem írunk meg semmit, a vektortábla akkor is ki lesz töltve egy címmel.
  - Ezen a címen egy reset utasítás lesz.
    - » (és debug-ban egy breakpoint is)
- Ha megírjuk az __DefaultInterrupt szubrutint, akkor helyette az kerül be az üres helyekre.

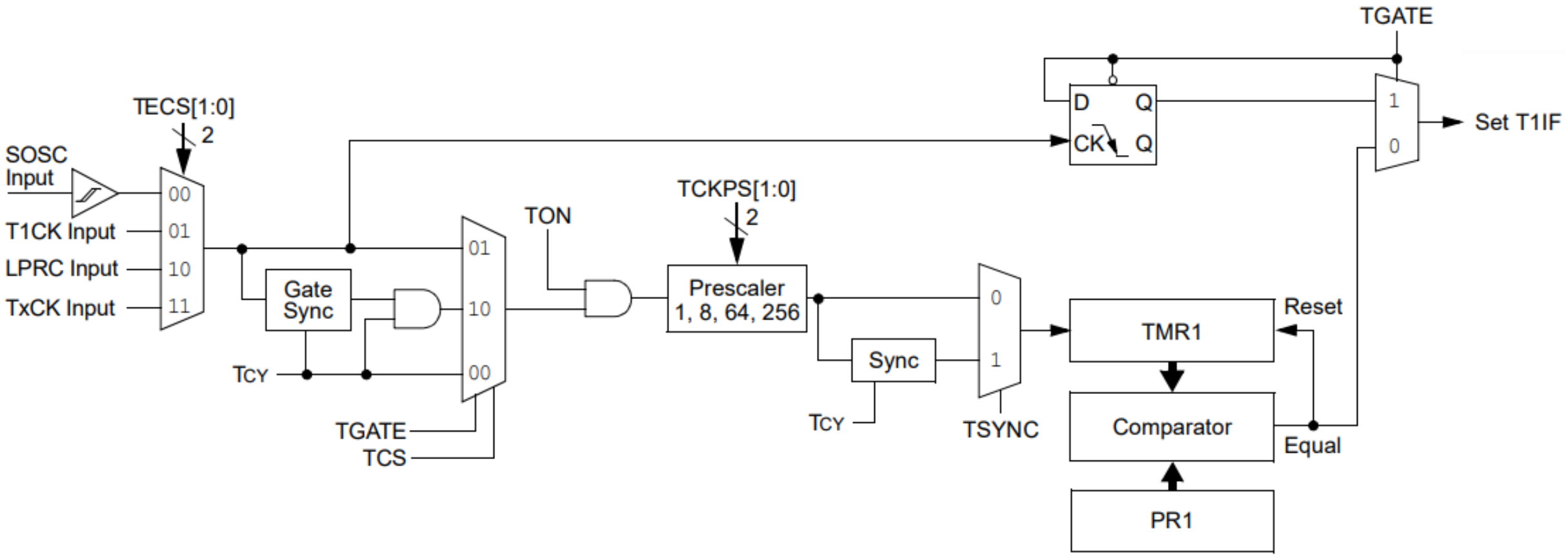
Gyors mentés

- Fontos, hogy a megszakítási szubrutin olyan gyors legyen, amennyire csak lehet.
  - Főleg a legnagyobb prioritású, ami nem engedi a többit futni
- Erre a célra van két speciális utasítás:
  - push.s
  - pop.s
  - Az első 4 W regisztert (W0-W3) egy ciklus alatt elmenti
  - Nem a vermet használja, hanem „árnyékregisztereket”
    - » Ebből csak egy van, egyszerre csak egy megszakítási szubrutin használhatja

Timer

- Az egyik legegyszerűbb és legtöbbet használt periféria
- Általános célú időzítő hardver
- Gyakorlatilag számláló
  - Állítható periódussal
  - Választható órajellel
- Megszakítást képes okozni
- Több, de korlátozott számú van belőle
  - Ebben a mikrokontrollerben 3 db.
    - » és még 10 egyéb, speciális célú időzítő, amit lehet így is használni

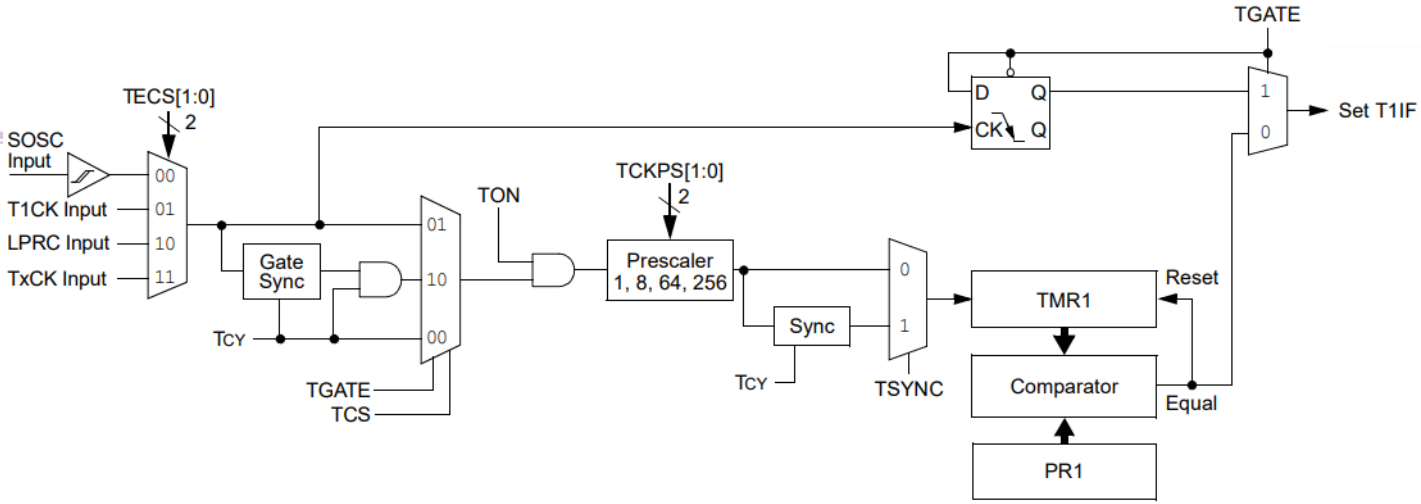
Timer1-2-3



T1CON: (vezérlő regiszter)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
TON	-	TSIDL	-	-	-	TECS1	TECS0	-	TGATE	TCKPS1	TCKPS0	-	TSYNC	TCS	-

Timer1



- 16 bites felfele számláló: TMR1
- 16 bites periódus regiszter: PR1
- Számolhat külső vagy belső órajelet
- Az órajel osztható 8-cal, 64-gyel vagy 256-tal
- Ha a TMR1 elérte a PR1 értékét, lenullázódik és a megszakítás flag (T1IF) 1-re áll.
  - » Figyelem, nem törlődik… → majd a megszakítás rutin letörli
- Használható „kapuzott” (gated) időzítőként:
  - Ilyenkor a számláló külső jellel engedélyezhető
  - Ilyenkor mindig a belső órajelet számolja
  - A külső jel lefutó éle ad megszakítást
    - » A számláló értéke arányos azzal, hogy mennyi ideig volt 1-es a gate bemenet
      - » Időmérésre használható közvetlenül

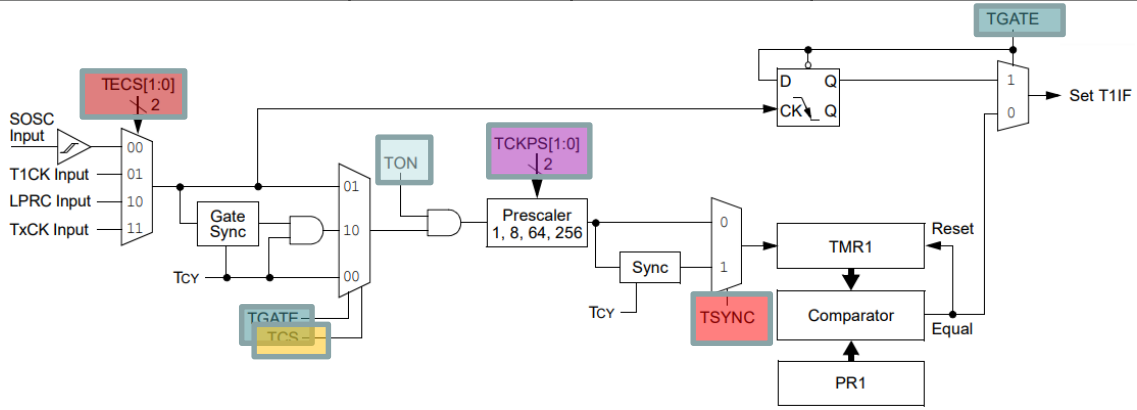
Timer1-2-3

T1CON: (vezérlő regiszter)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
TON	-	TSIDL	-	-	-	TECS1	TECS0	-	TGATE	TCKPS1	TCKPS0	-	TSYNC	TCS	-

TON	Indítás/leállítás	1 = Számláló számol 0 = Számláló áll
TSILD	Működés IDLE módban	1 = A timer IDLE módban leáll 0 = A timer IDLE módban is folytatja a működést
TECS[1:0]	Órajel forrás állítás, ha TCS = 1 Gate forrás állítás, ha TGATE = 1	11 = Közös timer bemenet 10 = Belső 32kHz 01 = T1CK bemenet 00 = Külső 32kHz órakvarc

TGATE	GATE funkció	Csak akkor érvényes, ha TCS = 0 1 = A számláló csak a TECS biteken kiválasztott bemenet 1-es értéke esetén lép a belső órajel hatására 0 = A számláló minden órajelre lép
TKCPS[1:0]	Előosztás	11 = 1 : 256 10 = 1 : 64 01 = 1: 8 00 = 1 : 1
TSYNC	Külső órajel szinkronizáció	1 = A timer mindig az utasításvégrehajtással szinkronban lép (külső bemenetről is) 0 = Nem szinkronizálja a külső órajelet
TCS	Órajel választás	1 = Az órajelet TECS[1:0] bitek értéke határozza meg 0 = Belső órajel (Fosc/2)



Timer használata

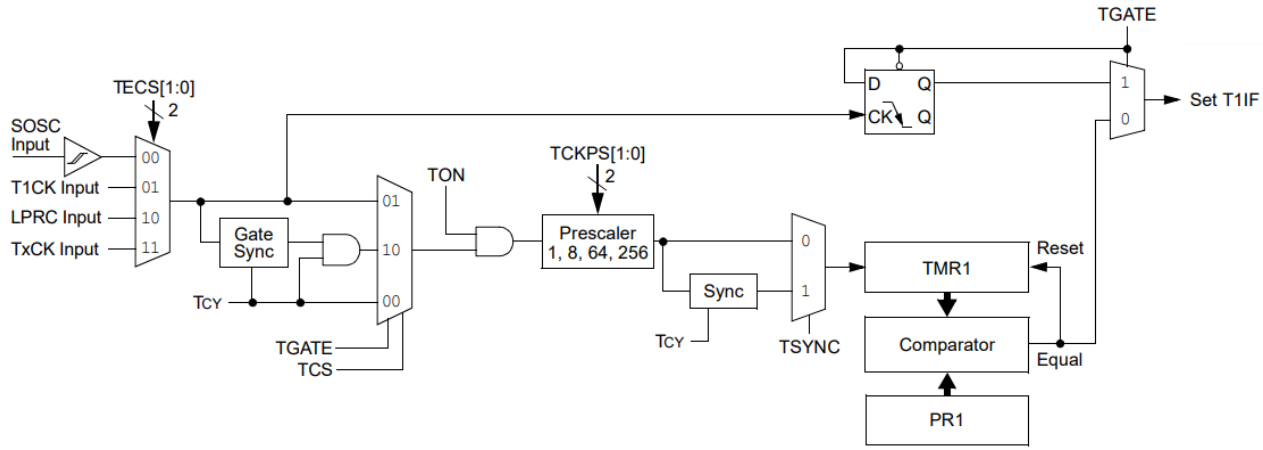
Pl. másodpercenként szeretnénk megszakítást kérni:

- 32MHz-es órajelből → Fcy= 16MHz, Tcy = 62.5ns
- 1 másodperc = 16 000 000 db Tcy, eddig nem számol el 16 biten.
- Válasszuk a 256-os előosztót:

16000000/256 → 62500

PR1 = 62500

T1CON = **0x8030**



TON: bekapcsoljuk **TCKPS**: 256-os előosztó **TCS**: belső órajel

TSIDL: IDLE módban is működünk **TECS**: mindegy **TGATE**: nincs GATE **TSYNC**: nem szikronizálunk

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
TON	-	TSIDL	-	-	-	TECS1	TECS0	-	TGATE	TCKPS1	TCKPS0	-	TSYNC	TCS	-

Timer használata

Állítsuk PR1-et 62500-ra → T1CON-t 0x8030-ra → engedélyezzük a megszakítást. Írjuk meg a megszakításfüggvényt.

```
main:
        mov     #62500,w0
        mov     w0,PR1
        clr     TMR1   ;hátha már valami volt benne
        mov     #0x8030,w0
        mov     w0,T1CON
        bclr    IFS0,#3   ;hívhatjuk #T1IF-nek is
        bset    IEC0,#3

vegtelen:
        bra vegtelen            ;innentől nincs mit csinálni

.global __T1Interrupt
__T1Interrupt:
        bclr    IFS0,#3         ;töröljük az IT-t!
                                ;ide jön, amit másodpercenként csinálni kell
        retfie
```

Interrupt Description	MPLAB® XC16 ISR Name	Vector #	IRQ #	IVT Address	Interrupt Bit Location		
					Flag	Enable	Priority
Highest Natural Order Priority							
External Interrupt 0	_INT0Interrupt	8	0	000014h	IFS0[0]	IEC0[0]	IPC0[2:0]
Input Capture 1	_IC1Interrupt	9	1	000016h	IFS0[1]	IEC0[1]	IPC0[6:4]
Output Compare 1	_OC1Interrupt	10	2	000018h	IFS0[2]	IEC0[2]	IPC0[10:8]
Timer1	_T1Interrupt	11	3	00001Ah	IFS0[3]	IEC0[3]	IPC0[14:12]

Demo: Megszakítás érvényre jutás ideje

Feladat: Külső jel felfutó élének hatására invertáljunk meg egy kimenetet olyan gyorsan, amilyen gyorsan csak tudjuk.

PIC24 (asm):

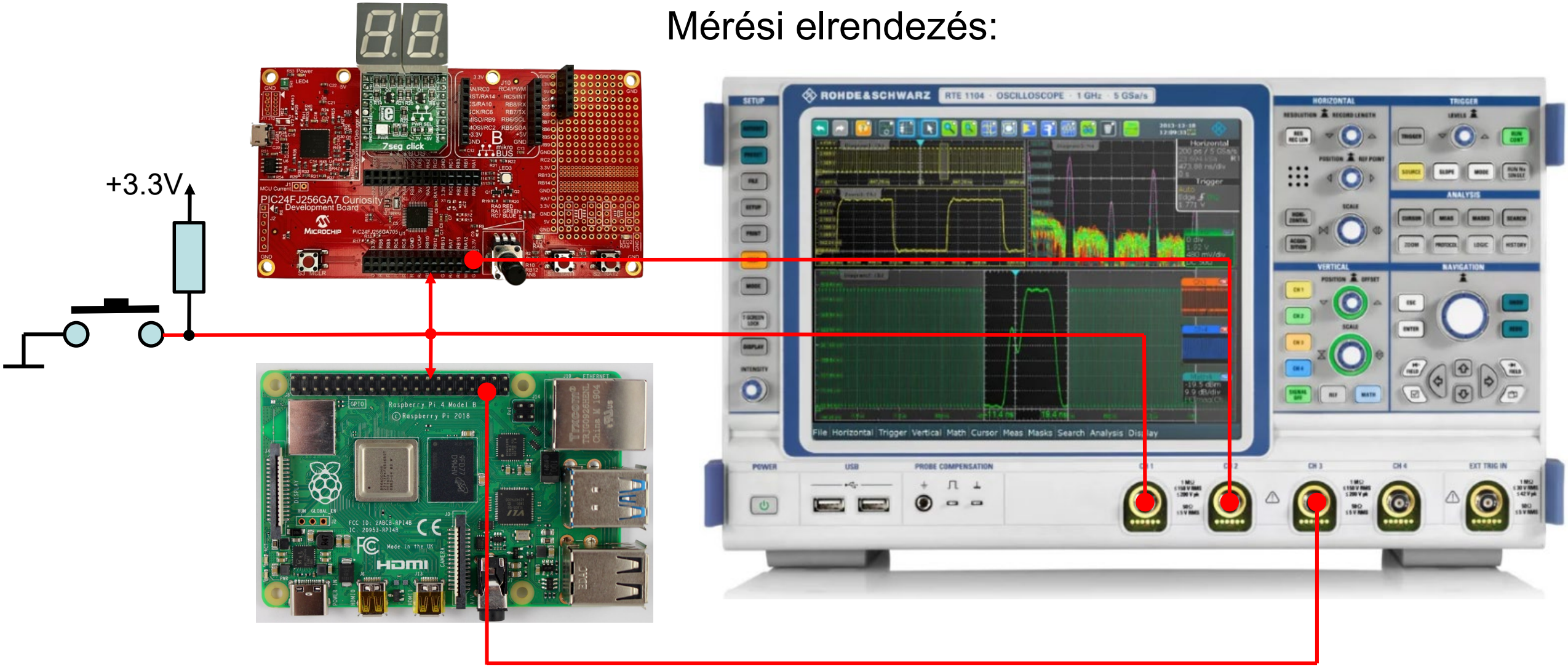
```
.global __reset
.global __INT1Interrupt
.text
__INT1Interrupt:
    btg LATA,#8    ;Kimenet invertál
    bclr IFS1,#4   ;INT1 flag letöröl
    retfie

__reset:
    mov #__SP_init,w15 ; Stack pointer inicializalas
    mov #__SPLIM_init,w0 ;
    mov w0,_SPLIM ; stack limit inicializalas
main:
    bclr ANSELA,#8
    bclr ANSELA,#11; LED és gomb1 digitális portlábak
    bclr TRISA,#8 ;LED1 kimenet
    bset TRISA,#11; GOMB1 bemenet
    mov.b #29,w0
    mov.b wreg,RPINR0H ;GOMB1 legyen az INT1
    bclr IFS1,#4     ;INT1 flag letorol
    bset IEC1,#4     ;INT1 engedélyez
vegtelen:
    bra vegtelen
```

Raspberry pi (C, linux):

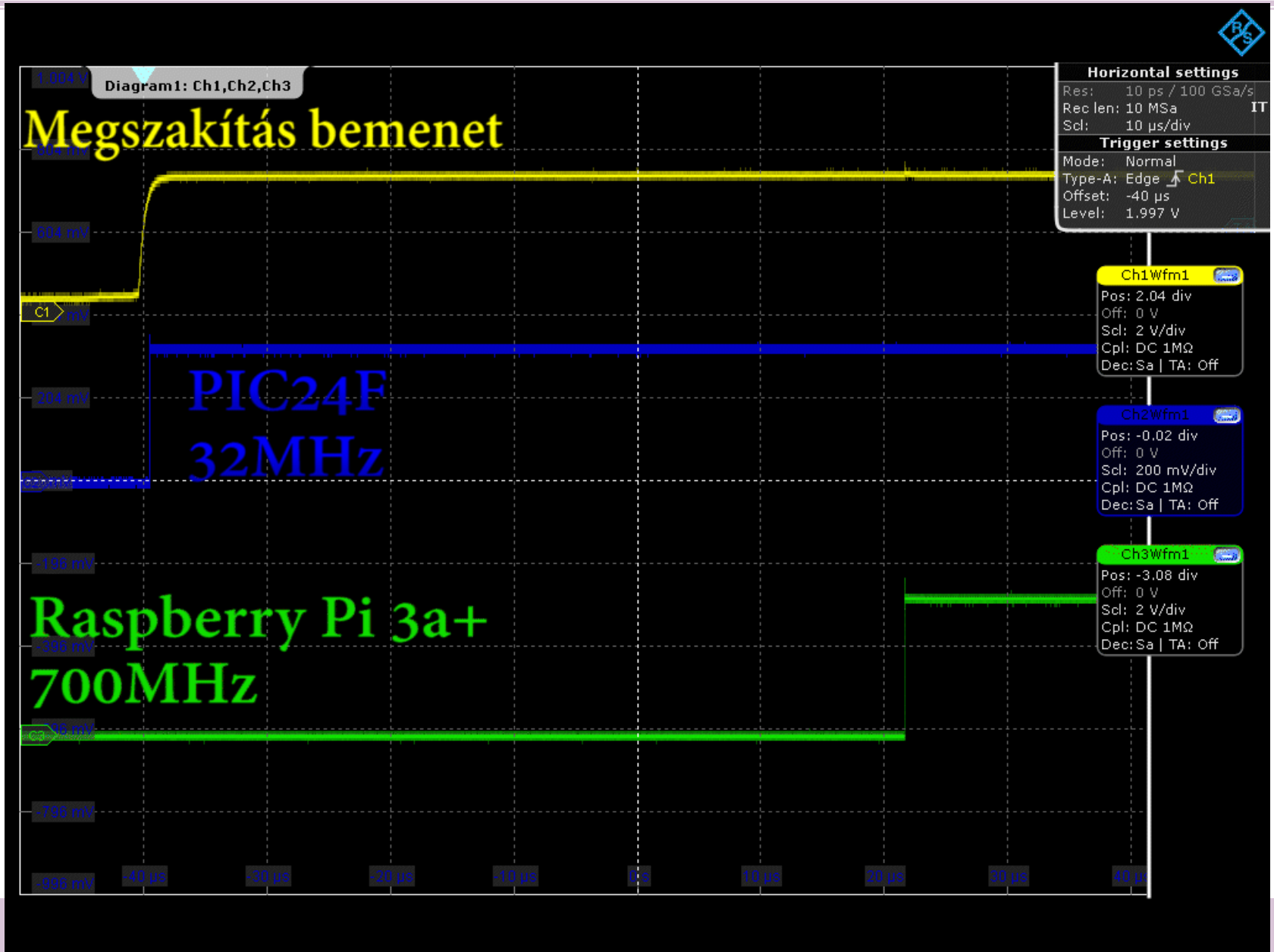
```
#include <unistd.h>
#include <wiringPi.h>

// Gomb, led
#define GOMB 22
#define LED 1

volatile int ledstate;

// Megszakításkezelo
void handler(void) {
        ledstate=!ledstate;
        digitalWrite(LED,ledstate);
}

int main(void) {
        // Init
        wiringPiSetup();
        pinMode(GOMB, INPUT);
        pinMode(LED, OUTPUT);
        ledstate=0;
        digitalWrite(LED,0);
        // Megszakításkezelo regisztralasa
        wiringPiISR(GOMB, INT_EDGE_RISING, &handler);

        // Innentol nincs mit csinálni
        for (;;) {
                sleep(1);
        }
}
```

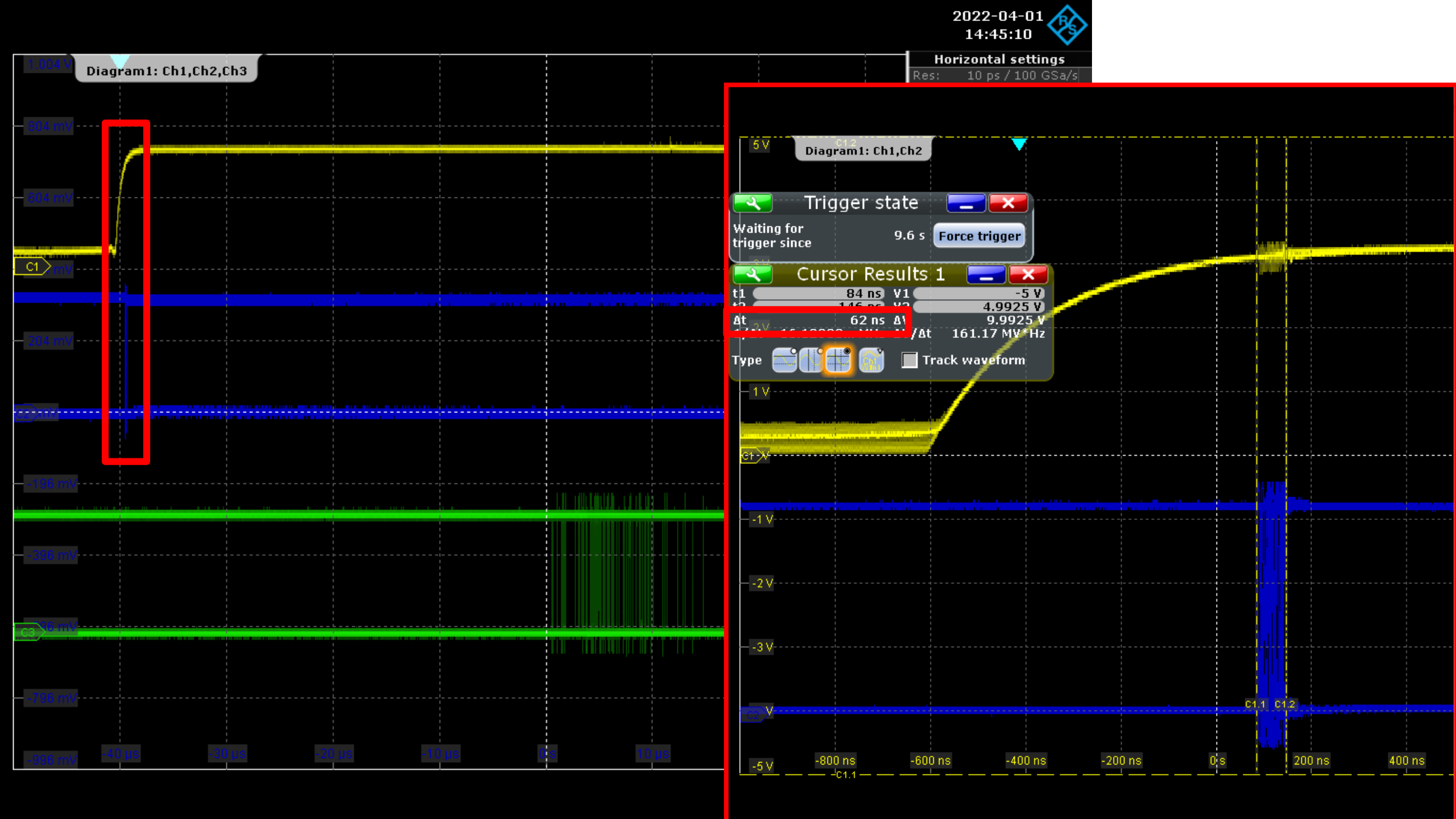
IT válaszidő mérése

Mérési elrendezés:



IT válaszidő mérése




2022-04-01
14:45:10
Horizontal settings
Res: 10 ps / 100 GSa/s
Diagram1: Ch1,Ch2,Ch3
1.004 V
804 mV
604 mV
C1
204 mV
-196 mV
-396 mV
C3
-796 mV
-996 mV
-40 µs
-30 µs
-20 µs
-10 µs
0 s
10 µs
Diagram1: Ch1,Ch2
5 V
Trigger state
Waiting for trigger since
9.6 s
Force trigger
Cursor Results 1
t1 84 ns
V1 -5 V
4.9925 V
Δt 62 ns
9.9925 V
161.17 MV*Hz
Type
Track waveform
1 V
-1 V
-2 V
-3 V
-5 V
C1.1
C1.2
-800 ns
-600 ns
-400 ns
-200 ns
0 s
200 ns
400 ns