

Digitális technika 2 harmadik laborgyakorlat

A foglalkozás célja:

A már megismert mikrokontrolleres fejlesztői környezetben és a már megismert mérőpanelen hardver perifériák kezelése:

- A kijelző egység kezelése SPI perifériával
- Időzítés megvalósítása TIMER segítségével
- Időmérés megszakítással
- A LED fényerejének szabályozása OC eszköz PWM üzemmódjával

A feladatok megvalósítása során előre elkészített projektet kell betölteni (3_Labor.X), amely tartalmazza a megvalósítandó feladatok vázát, valamint egy külön assembly állományban (init_send.s) a használandó hardver elemek (portok) kezdeti értékadását megvalósító szubrutint (**Disp_init**) és egy konverter rutint (**Disp_conv**) amely a paraméterként kapott bináris számhoz (0...99 közötti érték) előállítja a 7 szegmens kijelzőre kiküldendő bitmintát.

A perifériaelemek bemeneteinek és kimeneteinek portlábhoz rendelése

A PIC24 a belső perifériák portlábhoz rendelésére a PPS (Peripheral Pin Select) rendszert használja, így a legtöbb periféria majdnem az összes portlábon rendelkezésre állhat, akár egyszerre is. A PPS hardver lényegében minden portláb és minden periféria bemenetén elhelyezett egy-egy multiplexerből és a multiplexerek vezérlő bitjeit meghajtó vezérlő regiszterekből áll. A logika az, hogy mindig a „bemeneti pontokon” adhatjuk meg, hogy az adott helyre mi van kapcsolva, mert így nem köthetünk össze sosem két kimenetet, viszont több különböző bemenetre bármikor ráköthetjük ugyanazt a jelet. Ilyen „bemeneti pontjai” egyrészt a perifériák bemeneteinek, másrészt pedig a portlábak meghajtó áramköreinek vannak, ezért a PPS-t vezérlő regiszterek két csoportra oszthatók: a portlábakhoz tartozó regiszterekre, és a perifériákhoz tartozó regiszterekre.

Az egyes portlábakhoz tartozó PPS regiszterek a RPORxx, néven érhetők el. Minden ilyen 16 bites regiszterben az alsó és a felső helyiértékű 8-8 bit külön-külön két egymást követő port beállítását tartalmazza. Ezekbe a regiszterekbe beírt számtól az adott portlábon megjelenő kimeneti funkció függ. Minden belső periféria minden kimeneti funkciója rendelkezik egy funkciókóddal, amely az adott eszköz adatlapjában olvasható. A megfelelő RPORxx regiszter, megfelelő helyiértékű bájtyára beírt, megfelelő funkciókód az adott periféria, adott kimenetét, az adott portra kapcsolja.

Fontos megjegyezni, hogy a PPS-ben kiválasztott funkció teljesen átveszi az irányítást a kiválasztott portláb felett, és ha szükséges, akkor automatikusan kimenetre is állítja azt az eredeti beállítástól függetlenül. Tehát az adott RPOR regiszter beírása után a portlábhoz tartozó LAT és TRIS regiszterekben lévő bitek módosítása hatástalan. A portlábhoz tartozó PORT regiszter olvasásával viszont továbbra is olvasható az adott portlábon megjelenő logikai érték (amit természetesen már a periféria kimenete befolyásol). A megfelelő RPOR regiszter 0 értéke (a 0-s funkciókód) mindig az általános célú portláb, vagyis ha ez van a regiszterben, akkor azt a portot a megszokott módon vezérelhetjük a LAT és TRIS bitjeivel.

Az egyes perifériákhoz tartozó PPS regiszterek az RPINRxx néven érhetők el. Itt is minden regiszter két különböző periféria bemenet beállítását tartalmazza a regiszter alsó és felső 8-8 bitjén, a kimenetekhez hasonló módon. Az adott regiszter alsó vagy felső 8 bites részébe az adott portláb RP-vel vagy RPI-vel kezdődő sorszámát beírva az adott portláb bemenete kapcsolódik a kiválasztott perifériára. Fontos, hogy ebben az esetben a periféria közvetlenül megkapja a portlábon található értéket, azonban a portláb felett nem tudja átvenni az irányítást, azt továbbra is vezérelhetjük a TRIS

bitjével kimenetre, és ilyenkor akár szoftveresen meghajthatjuk a periféria bemenetét (és persze szembe is hajthatunk az erre a portra kötött bemeneti eszközzel). Ezért fontos tehát, hogy a ténylegesen bemenetre használt portlábakat véletlenül se állítsuk kimenetre a TRIS regiszterük segítségével, akkor se, ha egyébként azt már hozzárendeltük egy periféria bemenetéhez.

Az mérés 1. feladatának megoldásához csak az SPI1 (Serial Peripheral Interface) periféria bitsoros adatkimenetére (SDO, vagy MOSI) és órajel kimenetére (SCK) kimenetére van szükség, méghozzá úgy, hogy ezek az RB14-on és RB15-ön jelenjenek meg.

Az RB14 és RB15-ös portlábak az RP14 és RP15 elnevezésűek a PPS-ben, ahogy az az alábbi táblázatban látszik:

TABLE 5: COMPLETE PIN FUNCTION DESCRIPTIONS (PIC24FJ256GA705 UQFN)

Pin	Function	Pin	Function
1	C1INC/C2INC/C3INC/TMPRN/ RP9 /SDA1/T1CK/CTED4/PMD3/RB9	25	AN4/C1INB/ RP2 /SDA2/CTED13/RB2
2	RP22 /PMA1/PMALH/RC6	26	AN5/C1INA/ RP3 /SCL2/CTED8/RB3
3	RP23 /PMA0/PMALL/RC7	27	AN10/ RP16 /PMBE1/RC0
4	RP24 /PMA5/RC8	28	AN11/ RP17 /PMA15/PMCS2/RC1
5	RP25 /CTED7/PMA6/RC9	29	AN12/ RP18 /PMACK1/RC2
6	Vss	30	Vdd
7	VCAP	31	Vss
8	RPI29 /RA11	32	RPI31 /RA13
9	PGD2/ RP10 /OCM1C/CTED11/PMD2/RB10	33	OSCI/CLKI/C1IND/RA2
10	PGC2/REF1/ RP11 /CTED9/PMD1/RB11	34	OSCO/CLKO/C2IND/RA3
11	AN8/LVDIN/ RP12 /PMD0/RB12	35	TDO/PMA8/RA8
12	AN7/C1INC/ RP13 /OCM1D/CTPLS/PMRD/PMWR/RB13	36	SOSCI/ RP4 /RB4
13	TMS/ RP28 /PMA2/PMALU/RA10	37	SOSCO/PWRLCLK/RA4
14	TCK/PMA7/RA7	38	TDI/PMA9/RA9
15	CVREF/AN6/C3INB/ RP14 /CTED5/PMWR/PMENB/RB14	39	AN13/ RP19 /PMBE0/RC3
16	AN9/C3INA/ RP15 /CTED6/PMA14/PMCS/PMCS1/RB15	40	RP20 /PMA4/RC4
17	AVss	41	RP21 /PMA3/RC5
18	AVdd	42	Vss
19	MCLR	43	Vdd
20	RPI30 /RA12	44	RPI32 /RA14
21	VREF+/CVREF+/AN0/C3INC/ RP26 /CTED1/RA0	45	PGD3/ RP5 /ASDA1/OCM1E/PMD7/RB5
22	VREF-/CVREF-/AN1/C3IND/ RP27 /CTED2/RA1	46	PGC3/ RP6 /ASCL1/OCM1F/PMD6/RB6
23	PGD1/AN2/CTCMP/C2INB/ RP0 /RB0	47	RP7 /OCM1A/CTED3/PMD5/INT0/RB7
24	PGC1/AN1-AN3/C2INA/ RP1 /CTED12/RB1	48	RP8 /SCL1/OCM1B/CTED10/PMD4/RB8

Legend: **RPn** and **RPin** represent remappable pins for Peripheral Pin Select (PPS) functions.

Ezek beállításához tartozó konkrét regiszter az **RPOR7**:

REGISTER 11-39: RPOR7: PERIPHERAL PIN SELECT OUTPUT REGISTER 7

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP15R5	RP15R4	RP15R3	RP15R2	RP15R1	RP15R0
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP14R5	RP14R4	RP14R3	RP14R2	RP14R1	RP14R0
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-14 **Unimplemented:** Read as '0'

bit 13-8 **RP15R[5:0]:** RP15 Output Pin Mapping bits

Peripheral Output Number n is assigned to pin, RP15 (see Table 11-7 for peripheral function numbers).

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RP14R[5:0]:** RP14 Output Pin Mapping bits

Peripheral Output Number n is assigned to pin, RP14 (see Table 11-7 for peripheral function numbers).

Ahogy az az adatlapban látszik, az RPOR7 felső bájta az RP15 az alsó bájta az RP14 funkciója.

Az SPI1-es periféria MOSI (vagy SDO) lába, azaz adat kimenete, a 7-es számú funkciókóddal érhető el. Az SPI1 SCK kimenet funkciókódja (amennyiben az SPI master módban van, akkor az órajelet ő adja, vagyis ez kell) pedig a 8-as szám (SCK1OUT).

Az SS jelet nem szeretnénk közvetlenül az SPI hardverrel kapcsolni (azt is lehetne), ezért azt nem rendeljük hozzá egyetlen lábhoz sem.

Hasonlóan az SPI1 SDI vagy MISO lábát sem szeretnénk használni a kijelző visszaolvasására.

TABLE 11-7: SELECTABLE OUTPUT SOURCES (MAPS FUNCTION TO OUTPUT)

Output Function Number	Function	Output Name
0	None (Pin Disabled)	—
1	C1OUT	Comparator 1 Output
2	C2OUT	Comparator 2 Output
3	U1TX	UART1 Transmit
4	U1RTS	UART1 Request-to-Send
5	U2TX	UART2 Transmit
6	U2RTS	UART2 Request-to-Send
7	SDO1	SPI1 Data Output
8	SCK1OUT	SPI1 Clock Output
9	SS1OUT	SPI1 Slave Select Output
10	SDO2	SPI2 Data Output
11	SCK2OUT	SPI2 Clock Output
12	SS2OUT	SPI2 Slave Select Output
13	OC1	Output Compare 1
14	OC2	Output Compare 2
15	OC3	Output Compare 3
16	OCM2A	CCP2A Output Compare
17	OCM2B	CCP2B Output Compare
18	OCM3A	CCP3A Output Compare
19	OCM3B	CCP3B Output Compare
20	OCM4A	CCP4A Output Compare
21	OCM4B	CCP4B Output Compare
22	Reserved	—
23	SDO3	SPI3 Data Output
24	SCK3OUT	SPI3 Clock Output
25	SS3OUT	SPI3 Slave Select Output
26	C3OUT	Comparator 3 Output
27	PWRGT	RTCC Power Control
28	REFO	Reference Clock Output
29	CLC1OUT	CLC1 Output
30	CLC2OUT	CLC2 Output
31	RTCC	RTCC Clock Output

A kijelző meghajtásához szükséges kimenetek beállításához tehát a 0x0807 számot kell az RPOR7 regiszterbe beírunk.

A 4. feladatban a zöld LED vezérlésére lesz szükség, ami az RA1 portlábra van kötve. Ennek a PPS-ben RP27 a neve.

Az RP27 funkcióját az RPOR13 regiszterben állíthatjuk be:

REGISTER 11-45: RPOR13: PERIPHERAL PIN SELECT OUTPUT REGISTER 13

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP27R5	RP27R4	RP27R3	RP27R2	RP27R1	RP27R0
bit 15							
							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	RP26R5	RP26R4	RP26R3	RP26R2	RP26R1	RP26R0
bit 7							
							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15-14 **Unimplemented:** Read as '0'

bit 13-8 **RP27R[5:0]:** RP27 Output Pin Mapping bits
Peripheral Output Number n is assigned to pin, RP27 (see Table 11-7 for peripheral function numbers).

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **RP26R[5:0]:** RP26 Output Pin Mapping bits
Peripheral Output Number n is assigned to pin, RP26 (see Table 11-7 for peripheral function numbers).

Ha a LED fényerejének változtatásához az OC1 perifériát szeretnénk használni, akkor a 13-as funkciót kell válasszuk.

Ahhoz, hogy a zöld LED fényerejét az OC1 határozza meg, az RPOR13 regiszter felső 8 bitjébe kell 13-at írunk.

Most az alsó bájt (RP26) nem lesz használva, ezért a regiszter alsó bitjeit nyugodtan 0-ra állíthatjuk, vagyis az alábbi módon is beírhatjuk:

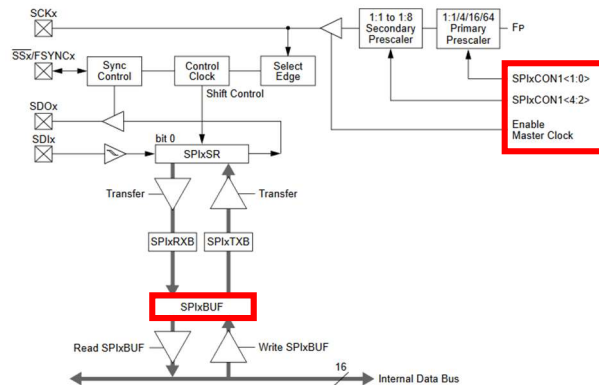
```
mov #0x0D00,w0      ; 0D: felső bájt 13, 00: alsó bájt 0
mov w0,RPOR13
```

Mivel csak egy 16 bites regiszter felső bájtját akarjuk módosítani, ezért célszerű az alsó bájtját meghagyni. Ehhez bájtosan kell adatot mozgatni. A bájtos mozgatás tetszőleges címre csak W0-ból lehetséges az alábbi módon (a processzorhoz tartozó include fájl minden regiszter alsó és felső bájtjának címét is definiálja: RPOR13H a felső, RPOR13L az alsó bájtja ugyanannak a regiszternek):

```
mov #13,w0           ; most muszáj w0-t használjuk
mov.b WREG,RPOR13H   ; itt a WREG szó lehet csak
```

Az SPI1 periféria programozása

Az SPI (Serial Peripheral Interface) periféria lényegében egy shift regiszterből és hozzá tartozó vezérlő áramkörökből áll. A shift regiszter soros adat kimenetének elnevezése általában SDO (Serial Data Out), de mikrokontrolleres környezetben találkozhatunk a MOSI (Master Out, Slave Input) elnevezéssel is. A shift regiszternek soros adatbemenete (SDI, vagy MISO) is van, azt jelenleg nem használjuk.



A shift regiszter párhuzamos ki/bemenete írható/olvasható az **SPI1BUFL** regiszteren keresztül. Ebbe a regiszterbe kell írunk a kívinni kívánt adatot.

Az órajel, ami a shift regisztert lépteti (jellemző elnevezése SCK), pedig származhat külső eszközből (ebben az esetben slave SPI módról beszélünk) vagy adhatjuk mi is a mikrokontrollerből (ez a master SPI mód).

Az SPI-nek nagyon sok egyéb működési módja is van, amit nem fogunk használni a mérésen, ezért itt csak azokat a regisztereket foglaljuk össze, amelyeket használni kell. A többit nyugodtan a reset után kapott alaphelyzetében hagyhatjuk.

Az SPI1-hez a **SPI1CON1L** vezérlő regiszter és az **SPI1STATL** állapotregiszter tartozik, ezek bitjeinek magyarázata az alábbi:

SPI1CON1L: vezérlőregiszter alapfunkciókhoz (hagyományos SPI mód)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPIEN	-	SPISIDL	0	0	MODE16	SMP	CKE	SSEN	CKP	MSTEN	0	0	MCLKEN	0	ENHBUF

SPIEN	SPI engedélyezés	1 = SPI engedélyezve 0 = SPI tiltva	SSEN	Slave select láb használata (csak slave módban!)	1 = Slave módban az SS bemenet 0 értéke esetén működik csak az SPI 0 = Az SS bemenet nincs használva, az SPI folyamatosan működik
SPISIDL	Működés IDLE módban	1 = Az SPI IDLE módban leáll 0 = Az SPI IDLE módban is működik	CKP	Órajel polaritás választás	1 = Az órajel aktív állapota alacsony, inaktív állapota magas. 0 = Az órajel aktív állapota a magas, inaktív állapota az alacsony szint.
MODE16	16/8 bites adat mód	1 = a küldött adat 16 bites 0 = a küldött adat 8 bites	MSTEN	Üzem mód	1 = Master 0 = Slave
SMP	Bemenet mintavételezés ideje	1 = A bemenet mintavételezése a kimenet változtatásakor történik 0 = A bemenet mintavételezése két kimenet változás között középen történik.	MCLKEN	Órajel forrás választás	1 = Az SPI órajele az órajel kimenet (REFO) 0 = Az SPI órajele a belső órajelből lesz leosztva (a BRG-vel)
CKE	Órajel él választás	1 = A kimenet az órajel aktívba váltásakor érvényes 0 = A kimenet az órajel inaktívba váltásakor érvényes	ENHBUF	Bővített buffer mód	1 = Az adó és vevő bufferek (32 bájtos) FIFO sorok 0 = Az adó és vevő buffer egyetlen regiszter

A mérésen a kijelzőn megjelenő tartalom 16 bit, ezt egyszerre kiküldhetjük a hardverrel. Ezért célszerű 16 bites módban használni az SPI-t, vagyis a **MODE16** bitet az SPI1CON1L-ben **állítsuk 1-esbe**.

A bemenettel kapcsolatos dolgok nem lényegesek, hiszen a bemenetet nem szeretnénk használni. A slave select (SS) lábat is programból fogjuk vezérelni, ezért arra sincs szükség. Master módot kell választanunk, vagyis **MSTEN=1**.

Az egyszerű buffer mód elegendő nekünk, ezért **ENHBUF=0**.

A kijelző az órajel felfutó élének pillanatában tekinti érvényesnek a bemenetét (azaz az SPI kimenetét), ezért célszerű azt választani, amikor az órajel aktív állapota magas (**CKP=0**), a kimenet tehát az órajel aktívba váltásakor tekinthető érvényesnek (vagyis **CKE=1**)

Az SPI master módban való használatához az órajel frekvenciáját is meg kell határozni. Erre a célra egyrészt az MCLKEN bit szolgál. Ezt azonban nem állíthatjuk 1-be, hiszen a kijelző maximum 5MHz-es órajellel működtethető (lásd. 2. laborgyakorlat útmutatójában) és ilyenkor a 16MHz-es utasításciklussal megegyező frekvenciájú órajelet választanánk. Ha az MCLKEN=0, akkor az órajel leosztását az SPI1BRGL állítja be.

Az SPI1BRGL értéke az alábbi képlettel számítható ki:

$$SPI1BRGL = \frac{8MHz}{F_{SPI}} - 1$$

Esetünkben SPI1BRGL=1 értéke is már 4MHz-es órajelet ad, ami megfelelően lassú a kijelző számára.

Fontos, hogy az SPIEN bitet csak utoljára állítsuk be, amikor már minden üzemmódot beállítottunk.

Az SPI használata közben az állapotát az állapotregiszterben lévő bitekből kérdezhetjük le. Az állapotregiszter bitjei az alábbiak:

SPI1STATL: állapotregiszter az alapfunkciókhoz

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
U-0	U-0	U-0	R/C-0	R-0	U-0	U-0	R-0	R-0	R/C-0	R-1	U-0	R-1	U-0	R-0	R-0
-	-	-	FRMERR	SPIBUSY	-	-	SPITUR	SRMT	SPIROV	SPIRBE	-	SPITBE	-	SPITBF	SPIRBF

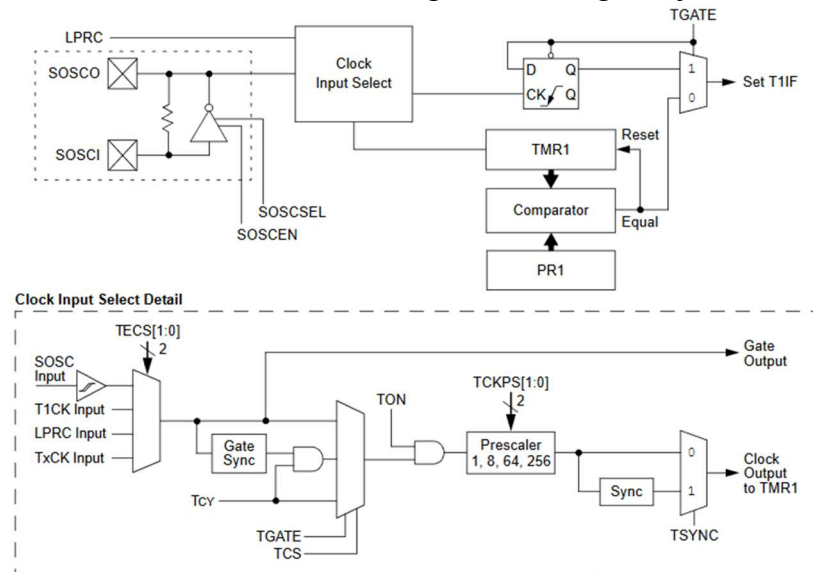
FRMERR	Kerethiba	SPI keret hiba történt (csak slave módban)	SPIRBE	Vételi buffer üres	1 = A vételi buffer üres 0 = A vételi buffer nem üres (olvasható)
SPIBUSY	SPI foglalt	1 = Adatátvitel folyamatban 0 = A periféria épp áll	SPITBE	Adó buffer üres	1 = Az adó buffer üres (írható) 0 = Az adó bufferben van valami
SPITUR	Adatelfogyás hiba	1 = Nem volt mit küldeni, amikor olvastak (csak slave módban) 0 = nem történt ilyen hiba	SPITBF	Adó buffer tele	1 = Az adó buffer tele 0 = Az adó buffer nincs tele
SRMT	Adás vége jelzés	1 = Az utolsó adat is kiküldésre került a bufferből 0 = Adatküldés van folyamatban	SPIRBF	Vételi buffer tele	1 = A vételi buffer tele van 0 = A vételi buffer nincs tele
SPIROV	Túlfutás	1 = Adatvesztés történt, mert adat érkezett, de a vételi bufferben nem volt hely 0 = nem volt ilyen hiba			

Amennyiben azt szeretnénk tudni, hogy az összes beírt bit „odaért-e”, vagyis kiküldésre került-e már, akkor az SPI1STATL regiszter SRMT bitjét kell figyeljük.

A TIMER1 programozása

Időzítőként a Timer1 perifériát használjuk. Ennek a számlálója a TMR1 regiszterben, a periódusa (amelyet elérve újraindul) a PR1 regiszterben található.

A timer1 működésének beállítását a TMR1CON regiszterben végezhetjük el.



A helyes működéshez most belső órajelre van szükségünk (nincs a rendszerórajelen kívül külső órajel csatlakoztatva sehova a panelon), ezért **TCS=0**-t kell válasszunk.

A Gate funkciót nem szeretnénk használni, ezért **TGATE=0**. Ilyenkor TECS bitek értéke tetszőleges lehet, hiszen nem használjuk. **Ekkor a timer bejövő órajele 16MHz lesz, ebből számolhatunk tovább.**

Az előosztót értelemszerűen a szükséges időhöz kell kiválasztani úgy, hogy a periódusregiszter (PR1) értéke még beférjen a 16 bites tartományba, viszont ne is legyen túlságosan kicsi sem.

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/leállitás		1 = Számláló számol 0 = Számláló áll			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					
TSILD	Működés IDLE módban		1 = A timer IDLE módban leáll 0 = A timer IDLE módban is folytatja a működést			TKCPS[1:0]		Előosztás		11 = 1 : 256 10 = 1 : 64 01 = 1 : 8 00 = 1 : 1					
TECS[1:0]	órajel forrás állítás, ha TCS = 1 Gate forrás állítás, ha TGATE = 1		11 = Közös timer bemenet 10 = Belső 32kHz 01 = TICK bemenet 00 = Külső 32kHz órákvarc			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)					

Ha a timert megszakításkérésre használjuk, ne feledkezzünk meg a megszakítás engedélyezéséről, és a megszakítás érvényre jutása esetén a megszakítási jelzés törléséről sem. A Timer1 periféria megszakítását az **IEC0** regiszter **#T1IE** bitjének (3. bit) 1-be írásával engedélyezhetjük. A megszakítási jelzést az **IFS0** regiszter **#T1IF** bitjének (3. bit) 0-ra állításával törölhetjük.

Az Output Compare Timer programozása PWM üzemmódra

Az Output compare 1 (OC1) modult használjuk majd egy LED fényerejének állítására. A fényerő állítására leginkább megfelelő üzemmód a PWM mód. Az OC1 periféria működését az **OC1CON1** és az **OC1CON2** vezérlő regiszterek értéke határozza meg az alábbiak szerint:

Az OC1CON1 regiszterben mindössze két dolgot kell beállítsunk:

1. Az órajelet válasszuk a rendszerórajel (16MHz) értékére, vagyis az **OCTSEL** bitek **mindegyikét állítsuk 1-esre**.
2. Az OC módját válasszuk a PWM módra, ehhez az **OCM** bitek értékét állítsuk az **110 értékre**.

Az összes többi bit értéke 0 maradhat.

REGISTER 15-1: OC1CON1: OUTPUT COMPARE x CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2 ⁽²⁾	ENFLT1 ⁽²⁾
bit 15							bit 8
R/W-0	HSC/R/W-0	HSC/R/W-0	HSC/R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ENFLT0 ⁽²⁾	OCFLT2 ^(2,3)	OCFLT1 ^(2,4)	OCFLT0 ^(2,4)	TRIGMODE	OCM2 ⁽¹⁾	OCM1 ⁽¹⁾	OCM0 ⁽¹⁾
bit 7							bit 0

Az egyes bitek jelentése az alábbiakban olvasható (az eredeti adatlap szerint).

bit 13	OCSIDL: Output Compare x Stop in Idle Mode Control bit 1 = Output Compare x halts in CPU Idle mode 0 = Output Compare x continues to operate in CPU Idle mode
bit 12-10	OCTSEL[2:0]: Output Compare x Timer Select bits 111 = Peripheral clock (Fcy) 110 = Reserved 101 = Reserved 100 = Timer1 clock (only synchronous clock is supported) 011 = Unimplemented 010 = Unimplemented 001 = Timer3 clock 000 = Timer2 clock
bit 9	ENFLT2: Fault Input 2 Enable bit ⁽²⁾ 1 = Fault 2 (Comparator 1/2/3 out) is enabled ⁽³⁾ 0 = Fault 2 is disabled
bit 8	ENFLT1: Fault Input 1 Enable bit ⁽²⁾ 1 = Fault 1 (OCFB pin) is enabled ⁽⁴⁾ 0 = Fault 1 is disabled
bit 7	ENFLT0: Fault Input 0 Enable bit ⁽²⁾ 1 = Fault 0 (OCFA pin) is enabled ⁽⁴⁾ 0 = Fault 0 is disabled
bit 6	OCFLT2: Output Compare x PWM Fault 2 (Comparator 1/2/3) Condition Status bit ^(2,3) 1 = PWM Fault 2 has occurred 0 = No PWM Fault 2 has occurred
bit 5	OCFLT1: Output Compare x PWM Fault 1 (OCFB pin) Condition Status bit ^(2,4) 1 = PWM Fault 1 has occurred 0 = No PWM Fault 1 has occurred
bit 4	OCFLT0: PWM Fault 0 (OCFA pin) Condition Status bit ^(2,4) 1 = PWM Fault 0 has occurred 0 = No PWM Fault 0 has occurred

bit 3 **TRIGMODE**: Trigger Status Mode Select bit
 1 = TRIGSTAT (OCxCON2[6]) is cleared when OCxRS = OCxTMR or in software
 0 = TRIGSTAT is only cleared by software

bit 2-0 **OCM[2:0]**: Output Compare x Mode Select bits⁽¹⁾
 111 = Center-Aligned PWM mode on OCx⁽²⁾
 110 = Edge-Aligned PWM mode on OCx⁽²⁾
 101 = Double-Compare Continuous Pulse mode: Initializes the OCx pin low; toggles the OCx state continuously on alternate matches of OCxR and OCxRS
 100 = Double-Compare Single-Shot mode: Initializes the OCx pin low; toggles the OCx state on matches of OCxR and OCxRS for one cycle
 011 = Single Compare Continuous Pulse mode: Compare events continuously toggle the OCx pin
 010 = Single Compare Single-Shot mode: Initializes OCx pin high; compare event forces the OCx pin low
 001 = Single Compare Single-Shot mode: Initializes OCx pin low; compare event forces the OCx pin high
 000 = Output compare channel is disabled

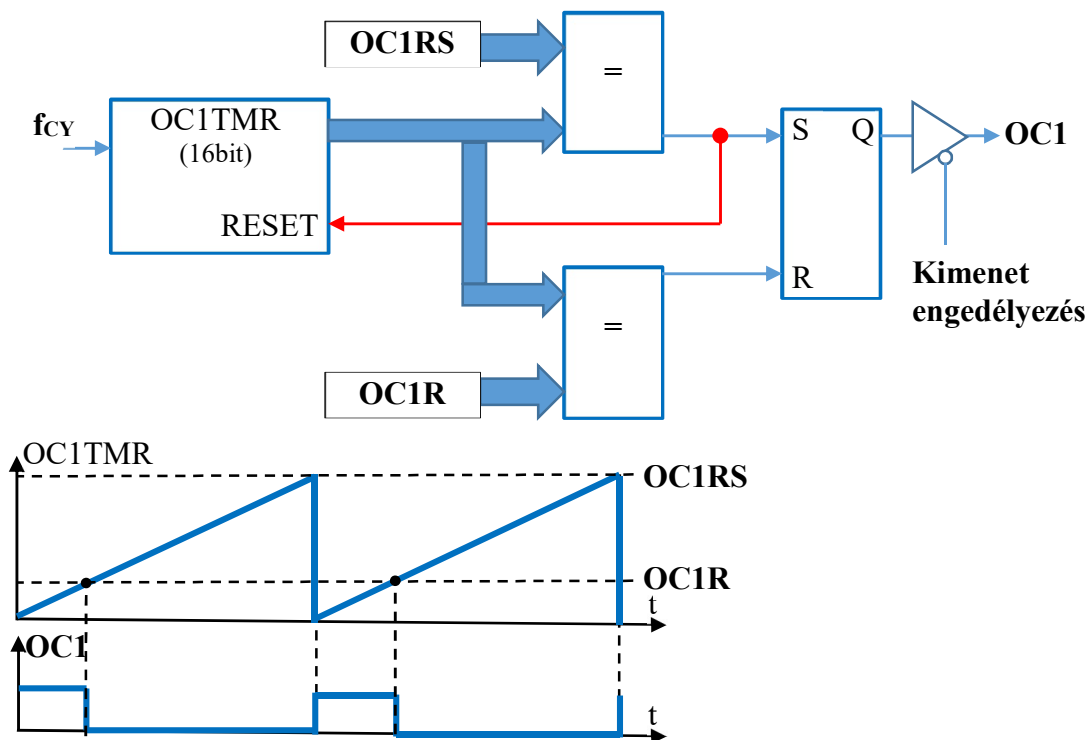
Az **OC1CON2** regiszterben az egyetlen beállítandó adat a szinkronizációs forrás, ami a SYNCSEL bitekkel választható ki. A PWM módhoz az kell, hogy a modul saját magát indítsa újra, azaz a szinkronizációs forrás az adott modul kimenete legyen, vagyis az „OCx sync output”. Ezért az **OC1CON2** alsó 5 bitjét 1-es értékűre kell állítani, a többi bit maradhat 0.

REGISTER 15-2: OC1CON2: OUTPUT COMPARE x CONTROL REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
FLTMD	FLTOUT	FLTTRIEN	OCINV	—	DCB1 ⁽³⁾	DCB0 ⁽³⁾	OC32
bit 15							bit 8
R/W-0	HS/R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0
OCTRIG	TRIGSTAT	OCTRIS	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0
bit 7							bit 0

bit 15	FLTMD: Fault Mode Select bit 1 = Fault mode is maintained until the Fault source is removed and the corresponding OCFLT0 bit is cleared in software 0 = Fault mode is maintained until the Fault source is removed and a new PWM period starts
bit 14	FLTOUT: Fault Out bit 1 = PWM output is driven high on a Fault 0 = PWM output is driven low on a Fault
bit 13	FLTTRIEN: Fault Output State Select bit 1 = Pin is forced to an output on a Fault condition 0 = Pin I/O condition is unaffected by a Fault
bit 12	OCINV: OCMP Invert bit 1 = OCx output is inverted 0 = OCx output is not inverted
bit 11	Unimplemented: Read as '0'
bit 10-9	DCB[1:0]: PWM Duty Cycle Least Significant bits ⁽³⁾ 11 = Delays OCx falling edge by ¾ of the instruction cycle 10 = Delays OCx falling edge by ½ of the instruction cycle 01 = Delays OCx falling edge by ¼ of the instruction cycle 00 = OCx falling edge occurs at the start of the instruction cycle
bit 8	OC32: Cascade Two OC Modules Enable bit (32-bit operation) 1 = Cascade module operation is enabled 0 = Cascade module operation is disabled
bit 7	OCTRIG: OCx Trigger/Sync Select bit 1 = Triggers OCx from the source designated by the SYNCSELx bits 0 = Synchronizes OCx with the source designated by the SYNCSELx bits
bit 6	TRIGSTAT: Timer Trigger Status bit 1 = Timer source has been triggered and is running 0 = Timer source has not been triggered and is being held clear
bit 5	OCTRIS: OCx Output Pin Direction Select bit 1 = OCx pin is tri-stated 0 = Output Compare Peripheral x is connected to an OCx pin
bit 4-0	SYNCSEL[4:0]: Trigger/Synchronization Source Selection bits 11111 = OCx Sync out ⁽¹⁾ 11110 = OCTRIG1 pin 11101 = OCTRIG2 pin 11100 = CTMU trigger ⁽²⁾ 11011 = A/D interrupt ⁽²⁾ 11010 = CMP3 trigger ⁽²⁾ 11001 = CMP2 trigger ⁽²⁾ 11000 = CMP1 trigger ⁽²⁾ 10111 = Not used 10110 = M CCP4 IC/OC interrupt 10101 = M CCP3 IC/OC interrupt 10100 = M CCP2 IC/OC interrupt 10011 = M CCP1 IC/OC interrupt 10010 = IC3 interrupt ⁽²⁾ 10001 = IC2 interrupt ⁽²⁾ 10000 = IC1 interrupt ⁽²⁾ 01111 = Not used 01110 = Not used 01101 = Timer3 match event 01100 = Timer2 match event (default) 01011 = Timer1 match event 01010 = Not used 01001 = Not used 01000 = Not used 00111 = M CCP4 Sync/Trigger out 00110 = M CCP3 Sync/Trigger out 00101 = M CCP2 Sync/Trigger out 00100 = M CCP1 Sync/Trigger out 00011 = Not used 00010 = OC3 Sync/Trigger out ⁽¹⁾ 00001 = OC1 Sync/Trigger out ⁽¹⁾ 00000 = Off, Free-Running mode with no synchronization and rollover at FFFFh

Miután a két vezérlő regisztert megfelelően beállítottuk az OC1 működése az alábbi ábra szerinti:



Az **OC1TMR** regiszter folyamatosan felfele számol, ameddig el nem éri az **OC1RS** értékét. Ekkor újraindul, és a periféria kimenete (amit ugye előzőleg a PPS segítségével a megfelelő helyre kiveztünk) 1-esre változik. A számlálás közben, amikor az **OC1TMR** értéke eléri az **OC1R** regiszter értékét, akkor a kimenet 0-ba változik.

Ilyen módon tehát a kiadódó négyszögjel frekvenciáját az **OC1RS** határozza meg (nagyobb szám, kisebb frekvencia). Az **OC1R** és az **OC1RS** aránya pedig a kitöltési tényezőt határozza meg, azaz annak arányát, ameddig 1-es értékű a kimenet a teljes periódushoz viszonyítva.

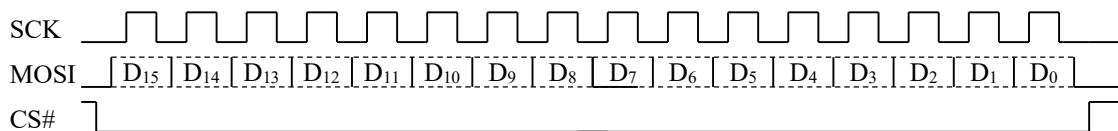
A LED fényerejének beállításához a frekvencia lényegében tetszőleges lehet, hiszen a legkisebb beállítható frekvencia (amikor az **OC1RS**=0xFFFF) is csak 244 Hz lesz, vagyis azt sem látjuk „villogni”. Emiatt célszerű a lehetséges teljes tartományt megadni az **OC1RS**-ben.

Ha pl. 0-tól 100-ig szeretnénk a fényerőt változtatni, akkor célszerű **OC1RS**-t 100-ra választani. Ekkor a kiadódó PWM jel frekvenciája 160 kHz lesz.

(A LED fényereje nem lineárisan változik a kitöltési tényezővel. A látványosabb megjelenés érdekében a mérőpanelen a fényerőt célszerű 0% ... 10% tartományban változtatni. Ha az OC1RS regiszterbe 1000 értéket töltünk, a századmásodpercben megadott reakcióidő ebben a tartományban változtatja majd a kitöltési tényezőt.)

1. Kijelző egység kezelése SPI perifériával

- Töltse le a Villamoskari oktatási portálról (edu.vik.bme.hu) a „3_labor.X.zip” projektet, majd tömörítse ki az asztalra. Indítsa el az MPLAB X IDE v5.50 fejlesztői környezetet, majd „File→Open Project...” funkcióval töltse be ezt a projektet.
- Nyissa meg a „main.s” forráskódot. **A projektben található init_send.s állomány tartalmát NE módosítsa!**
- Írja meg az SPI eszközt inicializáló szubrutint (**Init_SPI**). A szubrutinban
 - a. Rendelje hozzá az SPI1 eszközt a kijelző megfelelő bemeneteihez
 - Az SPI1 órajel kimenete (CLK) legyen RB15 (SCK)
 - Az SPI1 SCO kimenete legyen RB14 (MOSI)
 - b. Az eszköz legyen master
 - c. Állítsa 16 bites üzemmódra
 - d. Engedélyezze a működését
 - e. Az SPI1 órajele legyen 4 MHz.
- Írja meg a **Send_SPI** szubrutint, amely a w0 regiszterben kapott értéket beírja az SPI eszköz bufferébe és megvárja, amíg az eszköz továbbítja azt. Ne feledkezzen meg a CS# jel megfelelő kezeléséről!



- Írjon szubrutint (**Button2_press**), amely akkor tér vissza, amikor éppen lenyomták a BTN2 nyomógombot (1 → 0 átmenet), amely az RA12 port bemeneten található (PORTA,#12).
- Írja meg a főprogramot, amely a BTN2 gomb megnyomására számlál a kijelzőn 0 és 10 között. A gomb lenyomásának észlelésére használja a **Button2_press** szubrutint.
- Próbálja ki az elkészített kódot. Állítsa át konfigurációt szimulátorról hardver eszközre.


(Dashboard ablak – Properties funkció



Connected Hardware Tool: Starter Kits (PKOB)-SN: BUR.....)

- Fordítsa le és töltse be az eszközbe a programot.
- Dokumentálja a megoldását. (Kódrészlet)


2. Kijelző léptetése *TIMER* használatával

- Írja meg az *TIMER1* eszközt inicializáló szubrutint (**Init_Timer**).
 - a. Határozza meg a szükséges periódus értéket (*PR1*) úgy, hogy a Timer századmásodpercenként kérjen megszakítást
 - b. Határozza meg a Timer vezérlő regiszterébe (**T1CON**) töltendő értéket
 - Belső órajelet használunk
 - Az előosztás értéke a kiszámított periódus értékkel századmásodpercenkénti megszakításkérést tesz lehetővé
 - A számláló számol
 - c. Törölje a számláló aktuális értékét (**TMR1**)
- Írja meg a megszakítási szubrutint (**__T1Interrupt**), amely a **szam** változó értékét növeli eggyel.
- Írjon szubrutint (**DelayN10ms**), amely a $N \cdot 10$ ms-t vár. A szubrutin az aktuális **N** értéket a **w1** regiszterben kapja. A szubrutin engedélyezze a *TIMER1* megszakítását (**IEC0, #T1IE**), és a megszakítás rutin által növelt **szam** változó felhasználásával várjon a megadott ideig. A szubrutin visszatérése előtt tiltsa le a *TIMER1* megszakítást. A szubrutin nem ronthatja el a regiszterek értékét.
- Írja meg a főprogramot, amely a kijelzőn az értéket a mérésvezető által megadott tartományban (legfeljebb 00 ... 99 értékek között) lépteti a mérésvezető által megadott századmásodpercenként. A két kijelzés között eltelt időt **DelayN10ms** szubrutin felhasználásával valósítsa meg. A megjelenítéshez használja az előző feladatban elkészített **Send_SPI** szubrutint.
- Próbálja ki az elkészített kódot. A konfiguráció hardver eszközre legyen állítva.
(Dashboard ablak – Properties funkció 
Connected Hardware Tool: Starter Kits (PKOB)-SN: BUR.....)
- Fordítsa le és töltse be az eszközbe a programot.
- Dokumentálja a megoldását. (Kódrészlet)

3. Reakcióidő mérő

Készítsünk reakcióidő mérőt, amely századmásodperc felbontással méri meg egy led felgyulladását és egy nyomógomb megnyomása között eltelt időt.

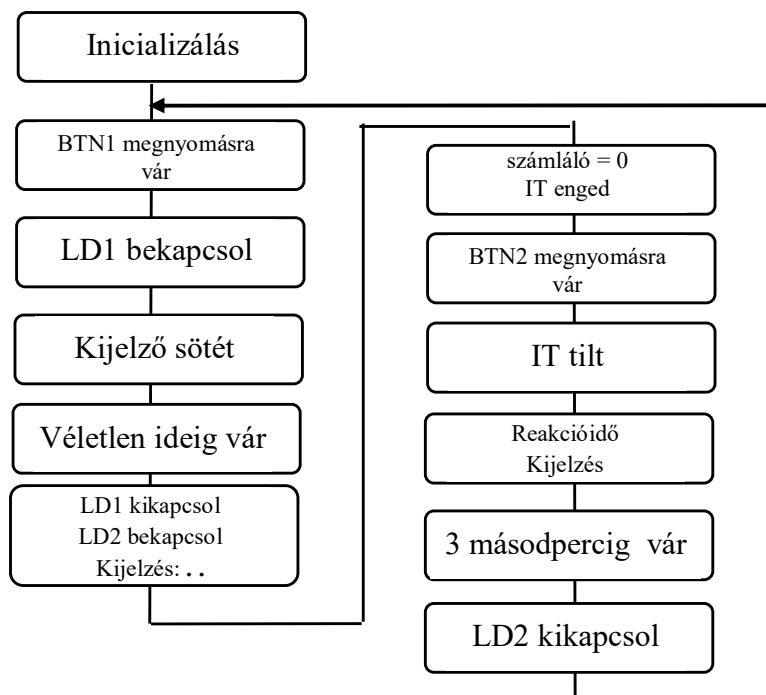
- A kijelzőn induláskor a " - . - . " kijelzés látszik.
 - A BTN1 nyomógomb megnyomására az LD1 led kigyullad, a kijelző sötét lesz.
 - Ezt követően 1 ... 4 másodpercen belül felgyullad az LD2 led és kialszik az LD1 led.
(Tipp: a TMR1 számláló pillanatnyi értékének kiolvasásával „véletlen” értéket kaphatunk. Ebből kell 100 ... 400 közötti értéket előállítani, és a századmásodperces számláló felhasználásával ennyi ideig várakozni.)
 - A kijelzőn a " . . " kijelzés látszik
 - Ekkor kell megnyomni a BTN2 nyomógombot.
 - A gomb megnyomásakor a kijelzőn megjelenik a led kigyulladását és a gomb megnyomása között eltelt idő századmásodpercben (00 ... 99).
 - Ha az idő nagyobb volt, mint 1 mp, a kijelzőn a " - - " kijelzés látszik.
 - 3 mp múlva az LD2 led kialszik. Ekkor lehet a BTN1 gomb megnyomásával újabb reakcióidőt mérni.
- Írja meg a főprogramot, amely a fenti algoritmust valósítja meg.
 - A programban használja az előző feladatok során elkészített szubrutinokat és megszakítást.
 - Próbálja ki az elkészített kódot. A konfiguráció hardver eszközre legyen állítva.

(Dashboard ablak – Properties funkció )

Connected Hardware Tool: Starter Kits (PKOB)-SN: BUR.....)

- Fordítsa le és töltsse be az eszközbe a programot.
- Dokumentálja a megoldását. (Kódrészlet)

Az elkészítendő program folyamatábrája



4. LED fényerejének módosítása

Egészítsük ki az előző programot úgy, hogy a **zöld LED** fényereje a reakcióidő kijelzését követő 3 másodpercben legyen arányos magával a reakcióidővel. Minél kisebb a reakcióidő, annál halványabb a kijelzés. A fényerő szabályozásához használjuk az Output Compare Timer PWM üzemmódját.

- Írja meg az OC1 eszközt inicializáló szubrutint (**Init_PWM**).
 - a. Kösse össze az OC1 kimenetét a zöld LED-del (RA1)
 - b. A trigger a saját kimenet (**OC1CON2**)
 - c. Üzem mód: PWM, belső órajel (**OC1CON1**)
 - d. A periódus 1000 (**OC1RS**)
 - e. Kezdetben sötét a LED (**OC1R**)
- Módosítsa a 3. feladatban elkészített programot úgy, hogy a reakcióidő kijelzésekor 3 másodpercig a LED fényereje arányos legyen a reakcióidővel (a kitöltési tényező 0% ... 10% között változzon). 3 másodperc után a LED ne világítson tovább.
- Próbálja ki az elkészített kódot. A konfiguráció hardver eszközre legyen állítva.

(Dashboard ablak – Properties funkció



Connected Hardware Tool: Starter Kits (PKOB)-SN: BUR.....)

- Fordítsa le és töltse be az eszközbe a programot.
- Dokumentálja a megoldását. (Kódrészlet)

A mérés során használt programváz:

```
.global __reset      ; kötelezően exportálandó címke, ettől kerül a reset vektor
                     ; helyére a kódunk és nem lesz semmi más a memóriában
.extern Disp_init    ; Máshol megírt szubrutin a hardver inicializálására
.extern Disp_conv     ; Máshol megírt szubrutin
                     ; Be: w0 - 0 ... 99 közötti érték
                     ; Ki: w0 - a = seg kijelz?r kiviend? bitminta

.bss
    szam: .space 2    ; itt számol 10ms-ként az IT

.text
__reset:
;kötelező startup kód, ha stack hivatkozás történik ezelőtt, az reset
    mov #__SP_init,w15    ; Stack pointer inicializálás
    mov #__SPLIM_init,w0  ; 
    mov w0,_SPLIM         ; stack limit inicializálás

    call Disp_init

;-----
; 1. feladat - Inicializálás
;   call Init_SPI
; ...

;-----
; 2. feladat - Inicializálás
;   call Init_Timer
; IT engedélyezés
; ...

;-----
; 4. feladat - Inicializálás
;   call Init_PWM
; ...

main:
;-----
; 1. feladat
; BTN2 lenyomásra vár
; szam növelés
; kijelzés Send_SPI rutinnal
;

    bra main

;-----
; 2. feladat
; Számlálás Timerrel
;

    bra main

;-----
; 3. - 4. feladat
; Reakcióidő mérés
;

    bra main
;-----
```

```
;A mérésen készitendo szubrutinok helye:

;-----
.global __T1Interrupt
; 2. feladat: Timer Interrupt rutin
__T1Interrupt:

; ide jön, amit századmásodpercenként csinálni kell

retfie

;-----
; 1. feladat: SPI1 inicializálás
; SCK: RB15, MOSI:RB14
; Master, 16 bit mód, 4 Mhz órajel
; Ront: w0
Init_SPI:

return

;-----
; 1. feladat: SPI küldés
; CS# állítás! (LATC,#9)
; BE: w0
; Ront: -
Send_SPI:

return

;-----
;1. feladat: BTN2 (PORTA,#12) lenyomására várakozás
;BE: -
;KI: -
;Ront: -
Button2_press:

return

;-----
; 2. feladat: Timer1 inicializálás
; IT kérés 10 ms
; Ront: w0
Init_Timer:

return

;-----
;2. feladat: DelayN10ms N*10 ms várakozás
;BE: w1 = N
;KI: -
;Ront: -
DelayN10ms:

return

;-----
; 4. feladat: PWM inicializálás
; OC1 kimenet PORTA 1-re (zöld LED)
; Kezdetben a LED nem világít
; Ront: w0
Init_PWM:

return

;-----
.end
```

Ellenőrző kérdések

Az alábbi ellenőrző kérdések megválaszolásához nézze át az előadáson elhangzott anyagokat, tanulmányozza a mérési útmutatót és a méréshez tartozó kiegészítő tananyagot.

- Mi a lényeges különbség a szoftveres időzítés és a hardveres időzítés között, ha a megszakítások is engedélyezettek?
- Milyen hardveregységre van szükség egy időzítő megvalósításához?
- Mi történik egy megszakítás érvényre jutásakor?
- Mire kell vigyáznunk egy megszakítási szubrutin megírása során?
- Hogyan történik impulzusszélesség modulált jel előállítása a mikrokontroller OC1 perifériája segítségével?
- Mire használható a PWM? Soroljon fel legalább 3 alkalmazást.
- Milyen előnnyel jár a hardveres SPI periféria alkalmazása a tisztán szoftveres (előző mérési feladat) megvalósításhoz képest?
- Sorolja fel, hogy az SPI interfész milyen jeleire van szükségünk a kijelző vezérléséhez?
- Az SPI1CON1L regiszterbe a 0x8520 adatot írtuk. Jelölje meg, hogy az alábbi állítások közül melyik igaz illetve melyik hamis:
 - Az SPI eszköz tiltva van
 - Az SPI eszköz MASTER módban van
 - 8 bites adatátvitelt valósít meg
- Adja meg milyen előosztást és periódus értéket kell megadni a Timer-nek, ha az órajelünk periódusideje 62.5 ns és 10 milliszekundumonként szeretnénk megszakítást kérni.