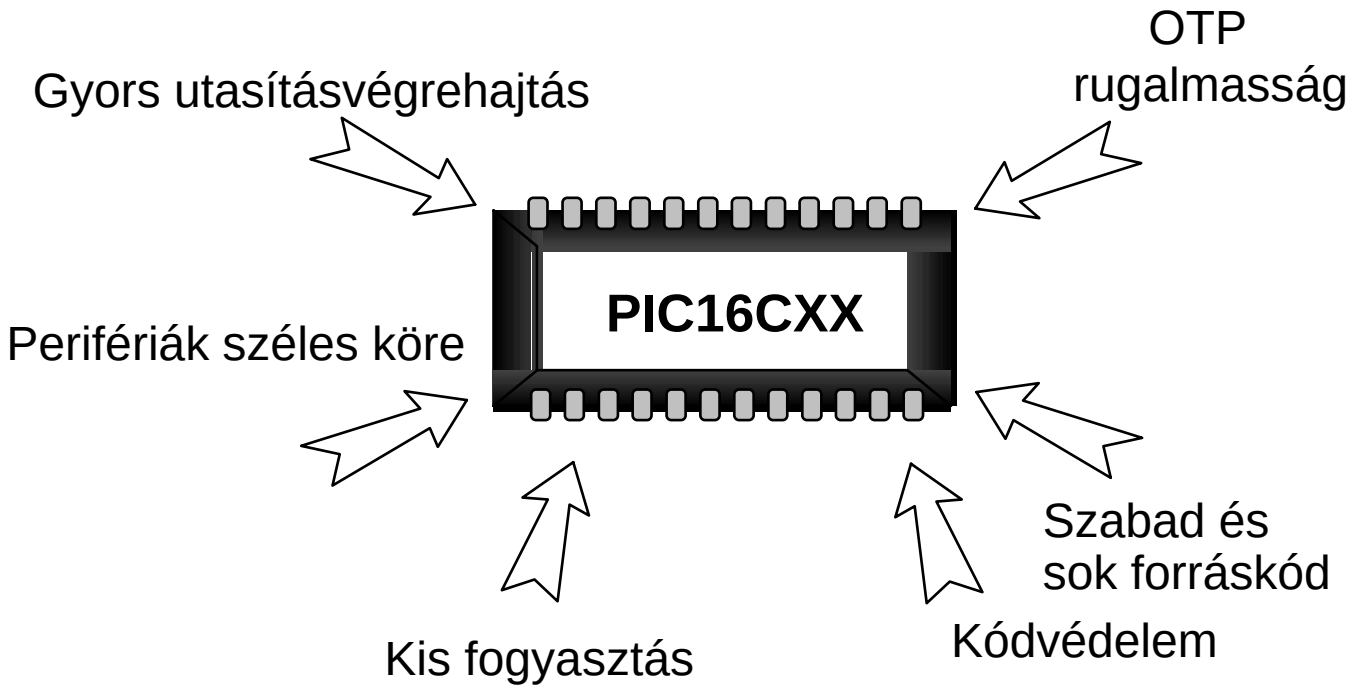
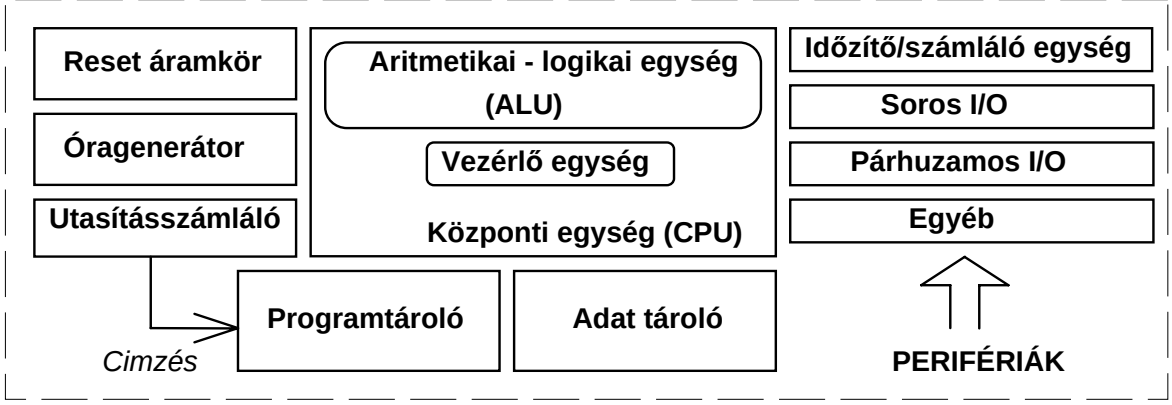
PIC mikrokontrollerek alkalmazástechnikája

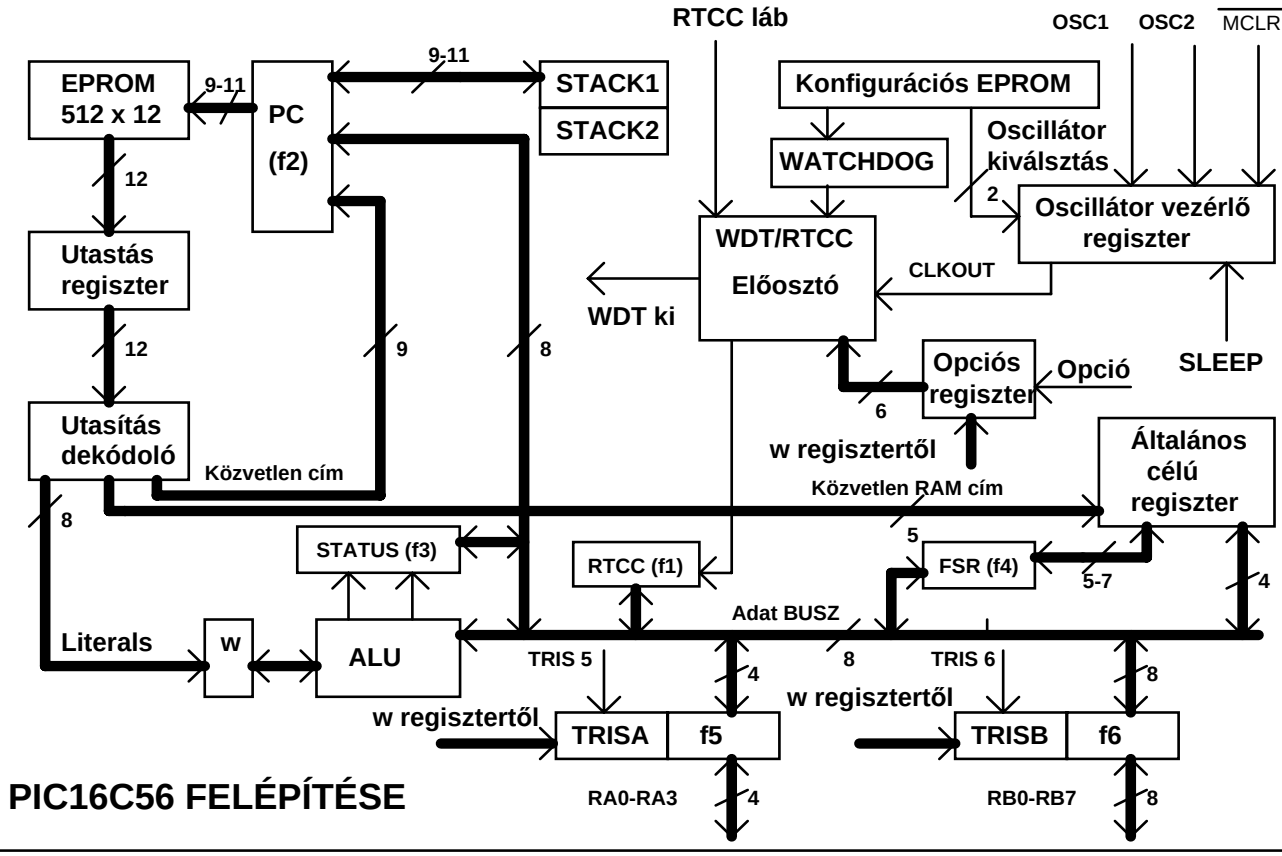


Gyors perifériák, órajel		**@ 20 MHz**
Utasítás-ciklusidő		**200 ns**
I/O váltás üteme		**200 ns**
Komparálási idõ		**200 ns**
Input Capture Resolution		**200 ns**
(with divide by 16 prescaler)		**50 ns**
PWM felbontás (8- és 10-bit)		**50 ns**
PWM frekvencia	**8-bit**	**80 KHz**
	10-bit	**20 KHz**
SCI (USART)	**Aszinkron sebesség**	**312.5-KBaud**
	Szinkron sebesség	**5000-KBaud**
SPI	**Master adatsebesség**	**5 MHz**
	Slave adatsebesség	**2.27 MHz**
A/D konverziós sebesség		**16 - 20 µsec**
Adat EEPROM	**Írási idő**	**10 ms**
	Irási szám	**1 M E/W (Typical)**

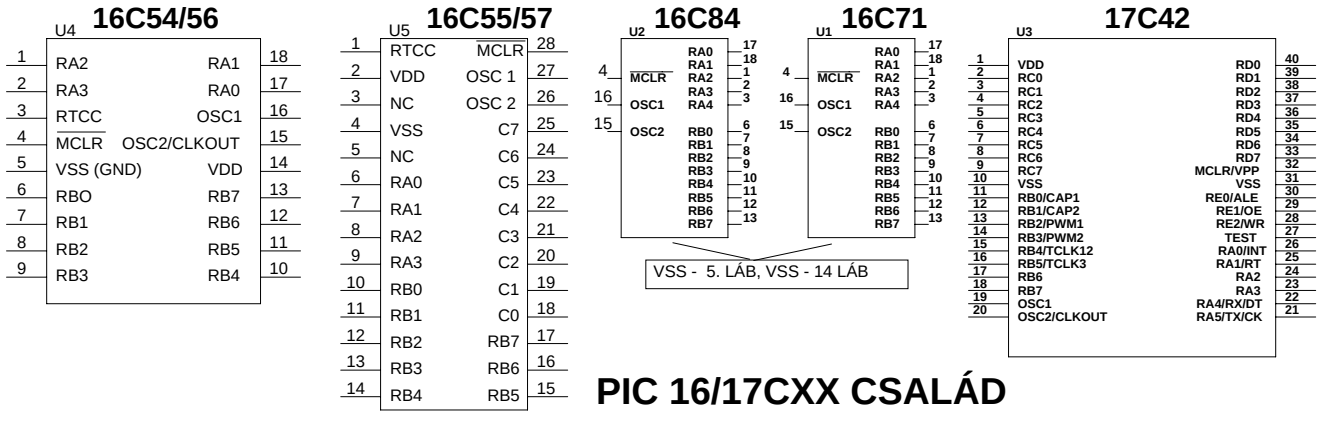
PIC -mikrokontrollerek



A mikrokontroller belső felépítése



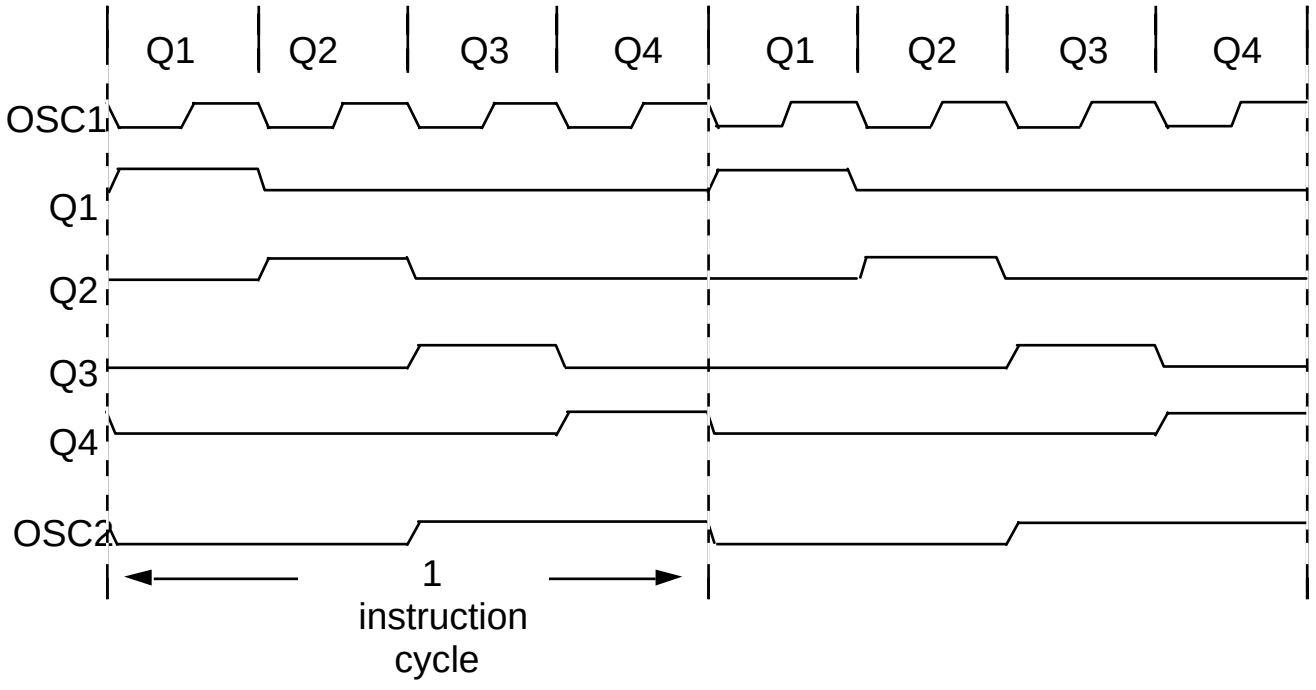
PIC16C56 FELÉPÍTÉSE



PIC 16/17CXX CSALÁD

UTASÍTÁSCIKLUS

- Az utasításciklus az órafrekvencia 1/4-e
- A cikluidõ 200 ns 20 MHz-es óránál, 1 mikrosec 4 MHz-nél

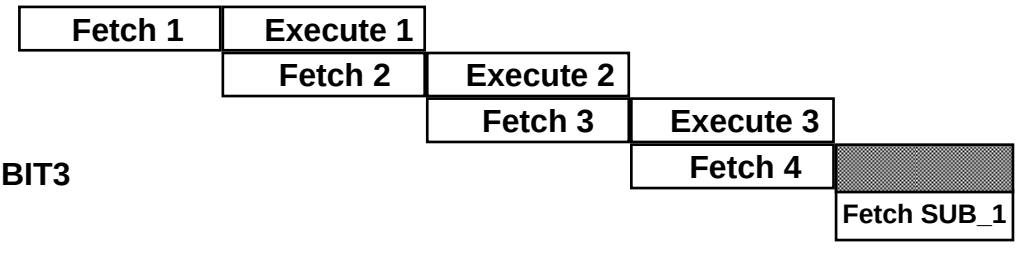


Megjegyzés:: általában: Q1 = Dekódolás Q2 = Olvas Q3 = Végrehajt Q4 = Írás

• Számos 8 bites mikrokontrollernél, az utasitások lehívása és végrehajtása sorban egymás után történik

• PIC16CXX pipeline felépítésű, a lehívás és végrehajtás átlapolja egymást és ezért egyciklusos utasítások vannak

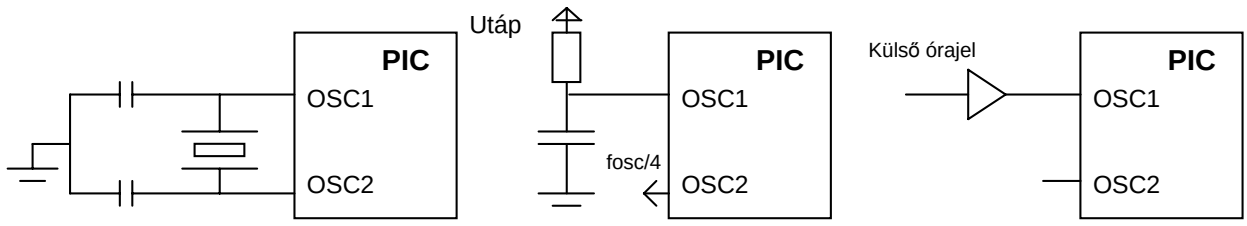
1. MOVLW 55h
2. MOVWF PORTB
3. CALL SUB_1
4. BSF PORTA,BIT3



• Bármelyik program elágazás (pl. GOTO, CALL vagy a PC-be írás) két ciklusos!

PIC működtetés

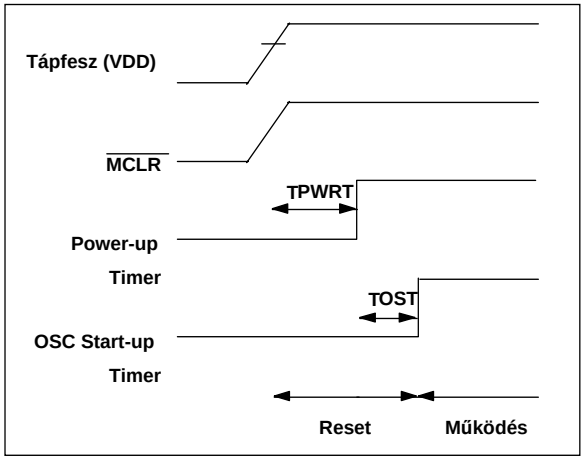
ÓRAJEL - RESET - PROGRAMOZÁS

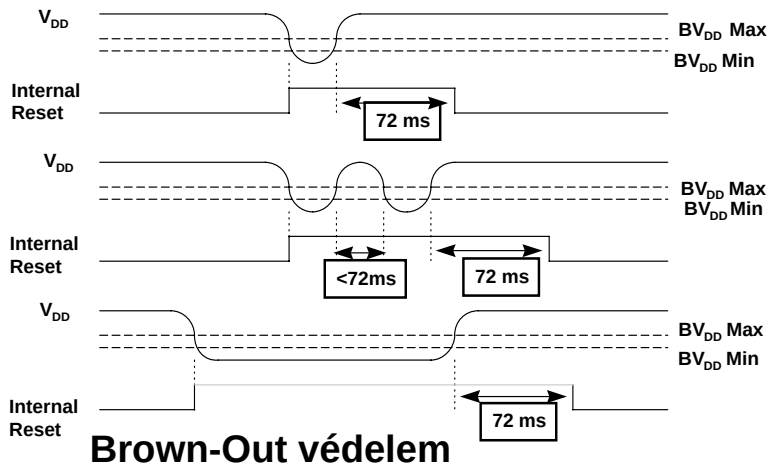


LP, XT, HS órajel kialakítás | RC órajel kialakítás | Külső órajel használata

Órajel generálás PIC kontrollereknél

RESET kezelés

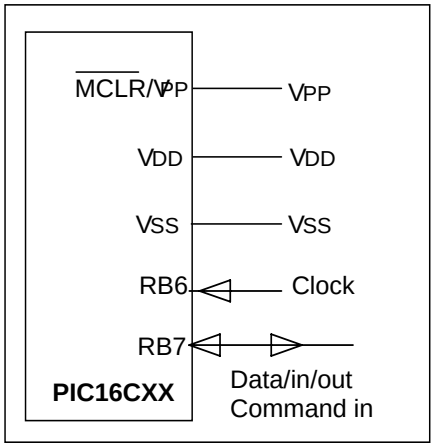




Brown-Out védelem

Ez majdnem mindig elegendő ! →

Soros programozás

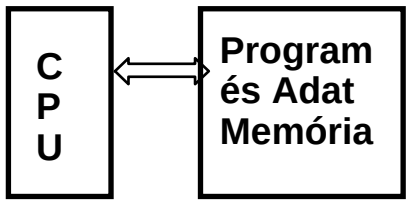


- Csak 2 láb kell a programozáshoz
- RB6 az órajel bemenet
- RB7 az adat be/ki- vagy a parancs bemenet
- Parancsok:
  - Load data (adatírás)
  - Read data (adatolvasás)
  - Begin programming (programozás indul)
  - End programming (programozás vége)
  - Increment address (memóriacím növelése)

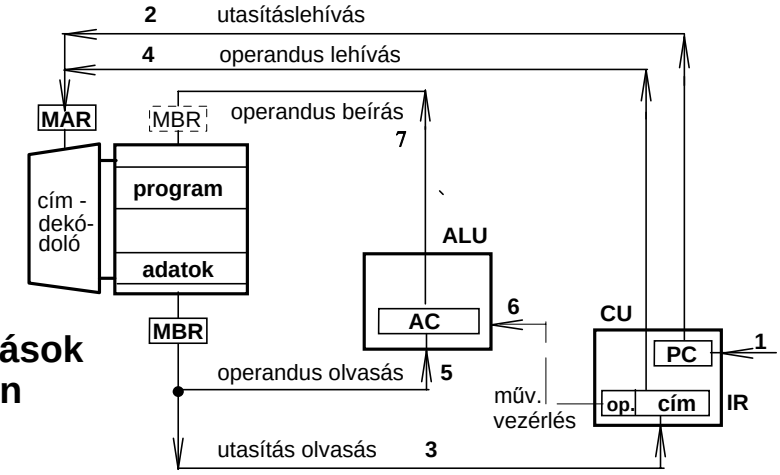
PIC16CXX Architektúra: RISC-tulajdonságú

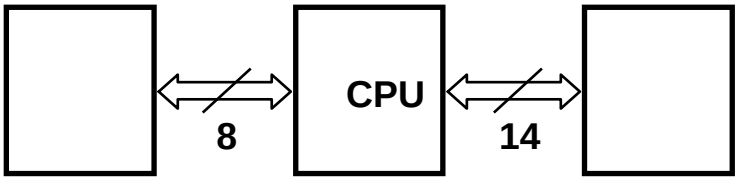
A nagy teljesítmény okai:

- **Harvard architektúra**
- **Regiszter fájl szervezés**
- **Minden utasítás egyszavas**
- **LWI (Long Word Instruction)**
- **Utasítás csõvonal (pipelining)**
- **Egyciklusos utasítások**
- **Csökkentett utasításkészlet**
- **Ortogonális utasításkészlet**



Neumann felépítésben az utasítások és az adatok közös memóriában helyezkednek el Az áteresztőképesség korlátozott



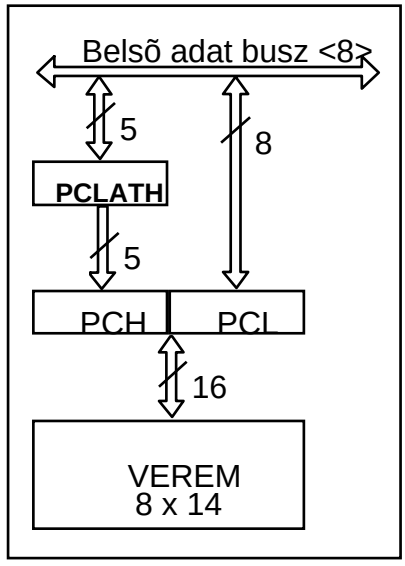


Harvard felépítés két külön tárolót: egy adat és egy programtárolót használ

- **Nagyobb ateresztőképesség**
- **Eltérő szélességű adat és programbusz lehetséges**

Veremkezelés

- **PIC16CXX-nél külön 8 x 14 bites hardver verem**
- **Verembe rakunk : CALL vagy megszakítás**
- **Veremből veszünk: RETURN, RETLW, RETFIE**
- **A verem körkörös (last in first out) - LIFO**

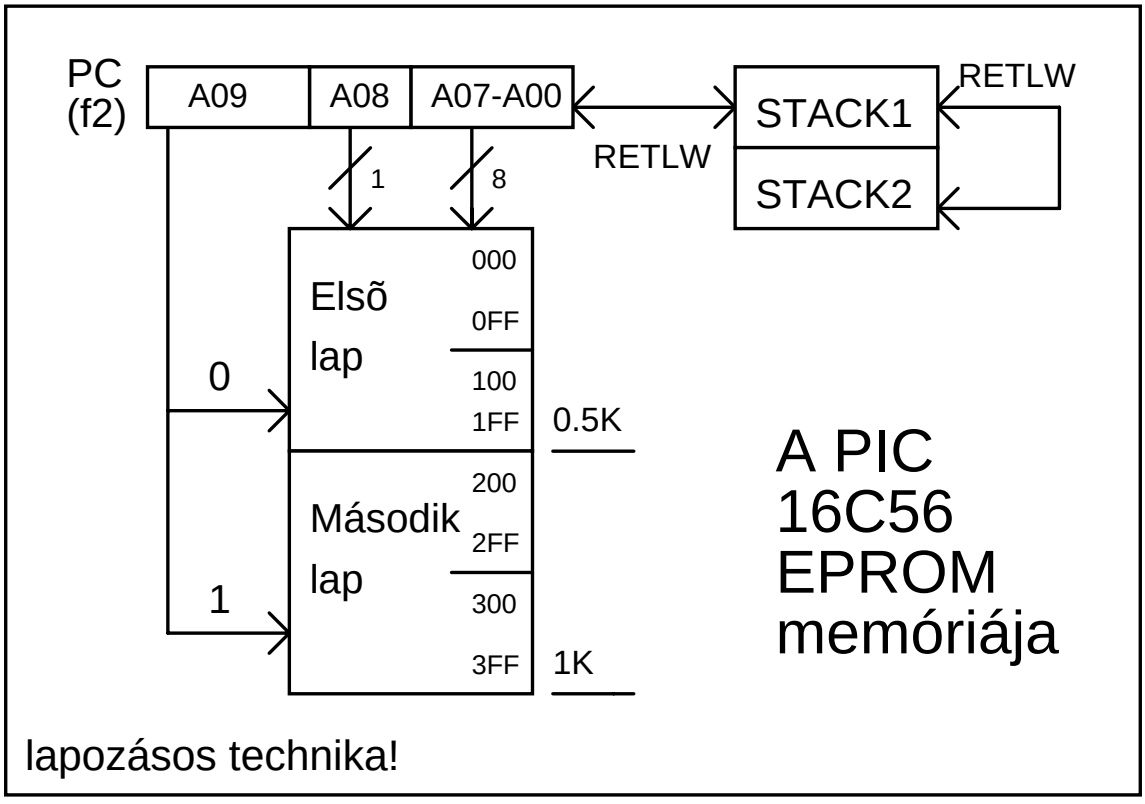


PIC PROGRAM és ADAT MEMÓRIA

HARVARD ARCHITEKTÚRA: különálló
adat memória (8 bit)
program memória (12-14-16 bit)

Miért jó?

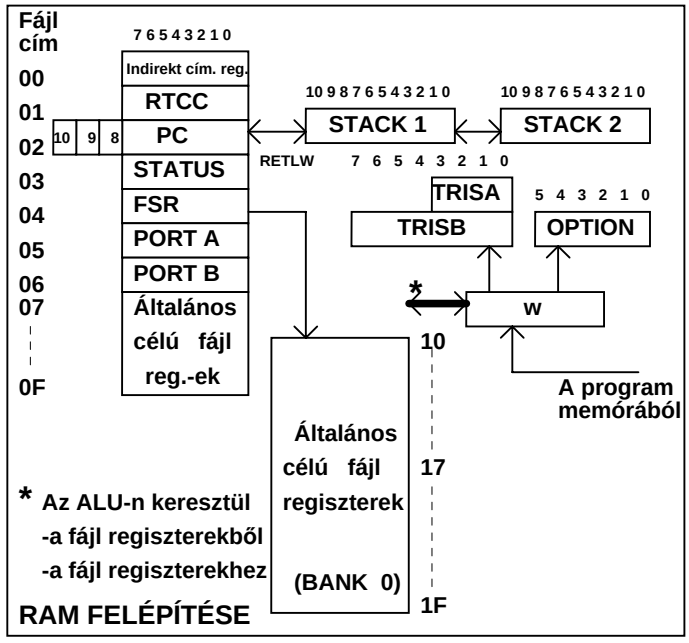
op code <7>	direct data adr <7>



A regisztermező két részre tagozódik.

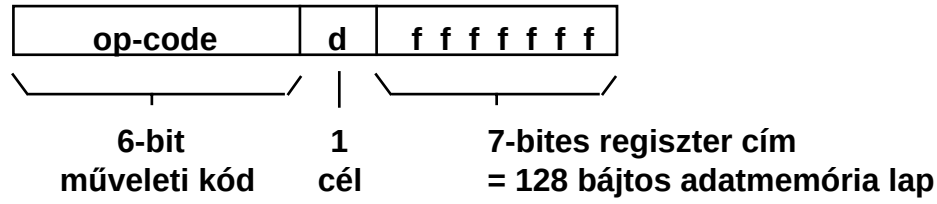
1.működtető file regiszterek (belső működéshez és I/O-hoz)
- **A valósidejű óra / számláló regisztere (RTCC)**
- **programszámláló (PC, Program Counter)**
- **állapotregiszter (Status Register)**
- **I/O regiszter (I/O Registers, PORTs)**
- **file regisztert választó regiszter (FSR, File Select Register)**
- **általános célú regiszterek (General Purpose Registers).**

2.További speciális regiszterek szolgálnak az I/O PORT konfigurálására és az előosztó kezelésére. A mikrovezérlő 512 szavas memóriát képes közvetlenül megcímezni, amit EPROM-ként alakítottak ki

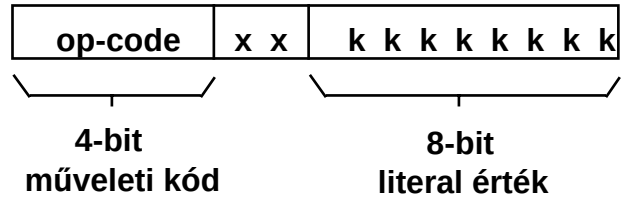


UTASÍTÁSKÉSZLET

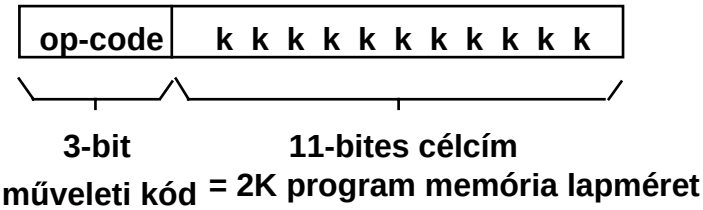
- **Utasítások közvetlen címzéssel**



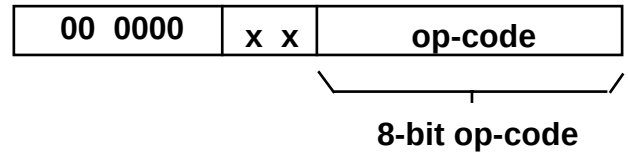
- **Utasítások állandóval literál = konstans**



- **GOTO, CALL**



- **Speciális utasítások NOP, SLEEP, CLRW OPTIONS, TRIS, CLRWDT**



Hogyan tárolhatók a program memóriában adatok (állandók)?

- **RETLW utasítás használható állandó értéknek W regiszterbe töltésére**
- **Nagyon hasznos táblázatoknál**

```
            CALL    TABLE   ;  W-ben az eltolás (table offset)
            MOVWF   PORTA   ;  W now has table value
            •
            •

TABLE       ADDWF   PC      ;  W = offset, PC=PC+offset
            RETLW   k0      ;  A tábla kezdete
            RETLW   k1      ;
            •
            •
            •
            RETLW   kn      ;  A tábla vége
```

Utasításkészlet - Összefoglaló

PIC16CXX Utasításkészlet - Összefoglaló

Byte-Oriented Operation - bájtos utasítások

No Operation	NOP	-
Move W to f	MOVWF	f
Clear W	CLRW	-
Clear f	CLRF	f
Subtract W from f	SUBWF	f,d
Decrement f	DECF	f,d
Inclusive OR W and f	IORWF	f,d
AND W and f	ANDWF	f,d
Exclusive OR W and f	XORWF	f,d
Add W and f	ADDWF	f,d
Move f	MOVF	f,d
Complement f	COMF	f,d
Increment f	INCF	f,d
Decrement f, skip if zero	DECFSZ	f,d
Rotate right f	RRF	f,d
Rotate left f	RLF	f,d
Swap halves f	SWAPF	f,d
Increment f, skip if zero	INCFSZ	f,d

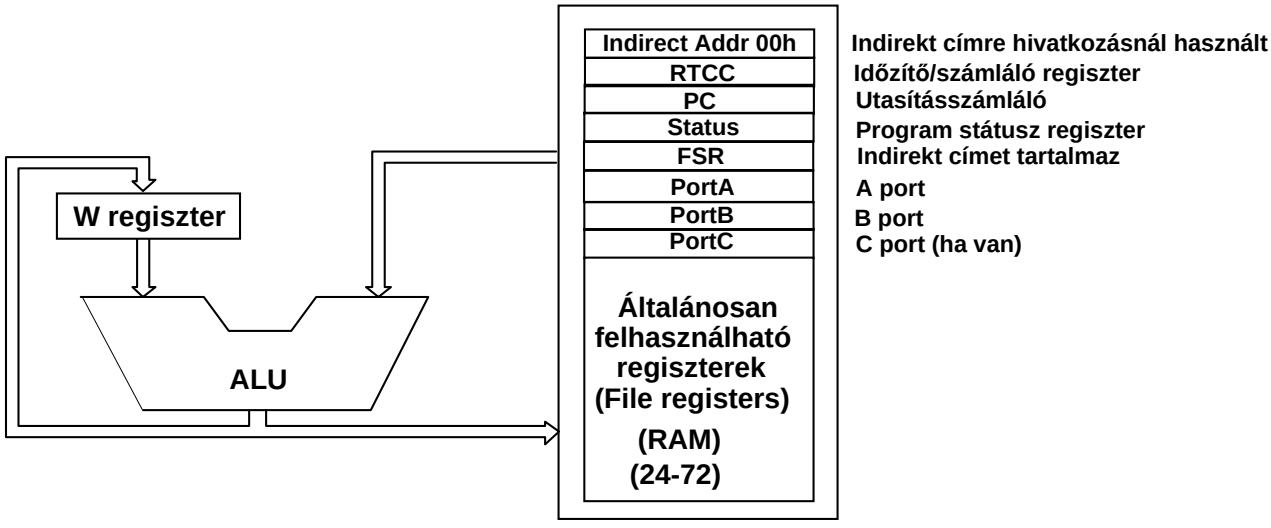
Bit-Oriented Operations - bit utasítások

Bit clear f	BCF	f,b
Bit set f	BSF	f,b
Bit test f, skip if clear	BTFSC	f,b
Bit test f, skip if set	BTFSS	f,b

Literal and Control Operations
Konstanskezelő és vezérlő utasítások

Go into standby mode	SLEEP	-
Clear Watchdog Timer	CLRWDT	-
Return, place Literal in W	RETLW	k
Return from interrupt	RETFIE	-
Return	RETURN	-
Call Subroutine	CALL	k
Go to address (k is 9 bit)	GOTO	k
Move Literal to W	MOVLW	k
Inclusive OR Literal and W	IORLW	k
Add Literal to W	ADDLW	k
Subtract Literal from W	SUBLW	k
AND Literal W	ANDLW	k
Exclusive OR Literal W	XORLW	k

Jelölések:: f = a RAM (file) regiszter címe
d = a művelet eredménye hova kerül; 0 = W regiszter, 1 = RAM (file) regiszter
k = egy 8 bites fix érték (konstans) vagy egy utasításra mutató cím (ez hoszabb mint 8 bit!)
Megjegyzés: A szürkével jelölt utasítások a 16CXX típusok "új" utasításai a PIC16C5X típushoz képes



STATUS WORD REGISZTER (f3)

(7)	(6)	(5)	(4)	(3)	(2)	(1)	(0)
PA2	PA1	PA0	T0	PD	Z	DC	C

RESET feltételek:

- = PA2, PA1, PA0 törlődnek, értékük :"0"
- = Jelzik a RESET okát.
- = Z, DC, C értéke nem változik meg.

PA0-PA2: Imemória lapválasztás
TO: time out (watchdog)
PD: power down (sleep)
Z: zéró bit
DC: half carry (4. biten túlcsord.)
C: carry

Illusztráció: PIC program részletek

A kivonások (SUBWF and SUBLW) a PIC-eknél egy kicsit másképp használhatók mint ahogy megszoktuk más 8 bites kontrollereknél. A számítás a PIC-eknél (memória - W) módon történik a (W - memória helyett).
Pl. Ha W-ből 3-at ki akarunk vonni, és azt írjuk:

```
SUBLW  3
```

valójában 3-ból vonjuk ki a W tartalmát
A jó megoldást az ADDLW utasítással érjük el a kettes komplemens felhasználásával:

```
ADDLW  256-3    vagy    ADDLW  253  esetleg    ADDLW  -3
```

Emiatt a W regiszter tartalma kettes komplemensének képzése:

```
SUBLW  0
```

utasítással történhet a szokásos

```
XORLW  0ffh
ADDLW  1
```

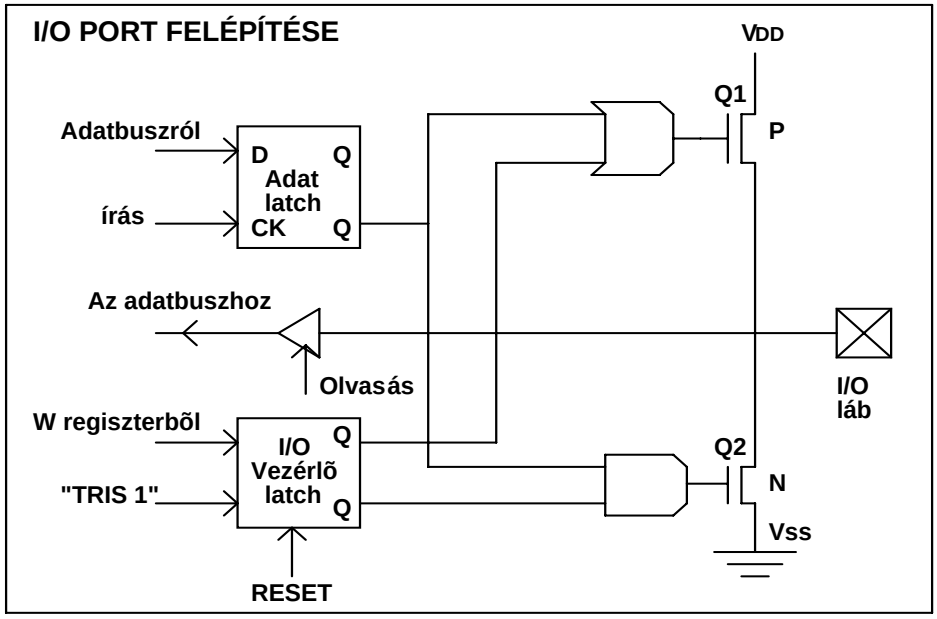
utasítások helyett.
A másik fontos dolog: kivonáskor a carry "nincs kölcsönvétel" bitként működik ("NOT borrow"). Azaz ha egy kivonási művelet kölcsönvétet okoz a carry törlődik!!! (egyébként 1)

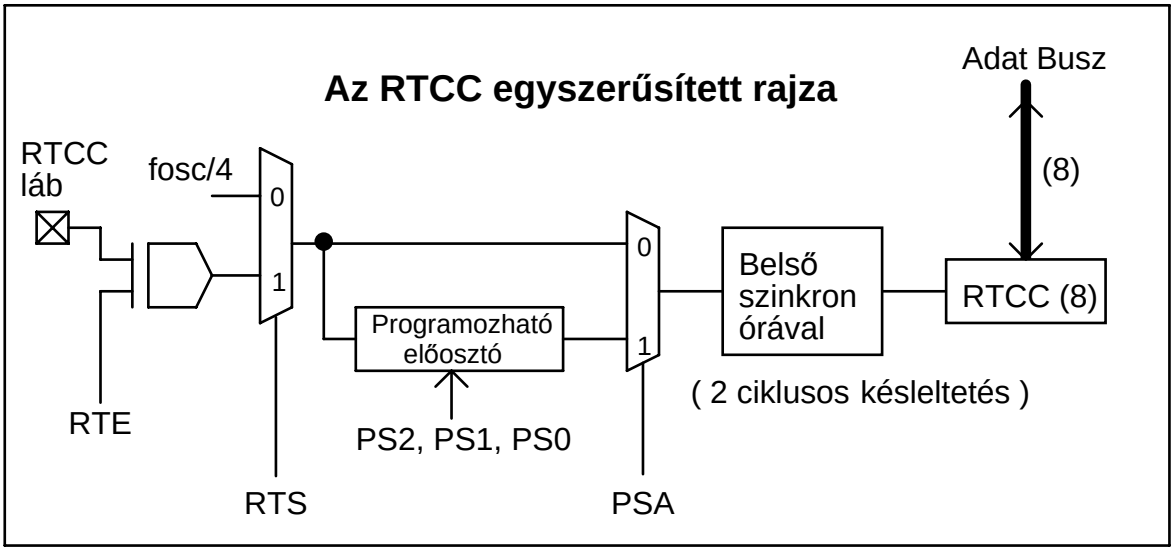
```
Feladat: egy bájt bitjeinek megfordítása
;pl.: 11000000 -ből 00000011 legyen

      MOVLW   0            ;
eredményregiszter (W) = 0
      BTFSC   NORMAL, 7      ; MSb set?
      IORLW   00000001B      ; Set LSb
      BTFSC   NORMAL, 6      ; bit set?
      IORLW   00000010B      ; Set bit
      BTFSC   NORMAL, 5      ; bit set?
      IORLW   00000100B      ; Set bit
      BTFSC   NORMAL, 4      ; bit set?
      IORLW   00001000B      ; Set bit
      BTFSC   NORMAL, 3      ; bit set?
      IORLW   00010000B      ; Set bit
      BTFSC   NORMAL, 2      ; bit set?
      IORLW   00100000B      ; Set bit
      BTFSC   NORMAL, 1      ; bit set?
      IORLW   01000000B      ; Set bit
      BTFSC   NORMAL, 0      ; bit set?
      IORLW   10000000B      ; Set bit

; az eredmény W-ben, NORMAL változatlan.
```

```
; 32 bites regiszter (hh:mh:ml:ll) eltolása
jobbra 4 bittel
cblock      0x20
      hh  ;legmagasabb helyiértékű bájt
      mh
      ml
      ll
endc
shr4
      MOVLW   0xf0
      ANDWF   ll,F
      SWAPF   ll,F
      SWAPF   ml,F    ;alsó-felső 4bit csere
      ANDWF   ml,W    ;felső (alsó volt)
      XORWF   ml,F    ;ml-ben törölni
      IORWF   ll,F     ;ll-hez hozzáadni
      MOVLW   0xf0
      SWAPF   mh,F
      ANDWF   mh,W
      XORWF   mh,F
      IORWF   ml,F
      MOVLW   0xf0
      SWAPF   hh,F
      ANDWF   hh,W
      XORWF   hh,F
      IORWF   mh,F
      RETURN
```

PIC PERIFÉRIÁK

- **I/O**
- **RTCC**
- **A/D**
- **soros I/O**
- **PWM**
- **capture/ compare**
- **párh.(slave) port**
- **időzítők**





SLEEP (Power-Down) Mód

SLEEP utasítással kerül a PIC16CXX SLEEP (power-down) állapotba. <u>Ilyenkor:</u>

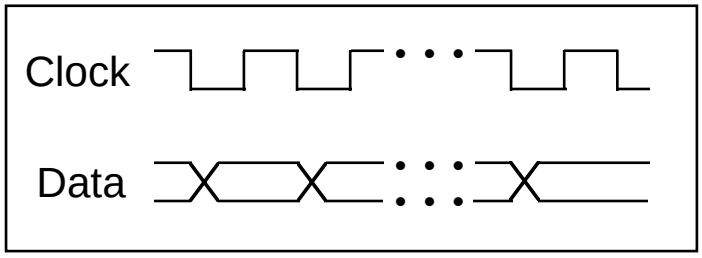
Minden belső óra és oszcillátor lekapcsol
Watchdog Timer, ha engedélyezett, tovább fut
a kimenetnek konfigurált I/O lábak tovább vezérelnek
A/D konvertál, ha A/D órajele RC oszc.

<u>Ébredés (Wake up)</u>:

Külső reset (MCLR lábat L-re, majd H-ra)
Watchdog Timer időtúllépése
Megszakítás

<u>Fogyasztás:</u> **1.5 µA typical @ 4V**

Példa: Soros adatátvitel



- Csak 11 szó a program !!!
- 74 utasítás ciklus = 14.6 μs @ 20 MHz
- Ez megfelel 540 Kbit/sec-nek

Egy egyszerű szinkron soros adatátviteli program:

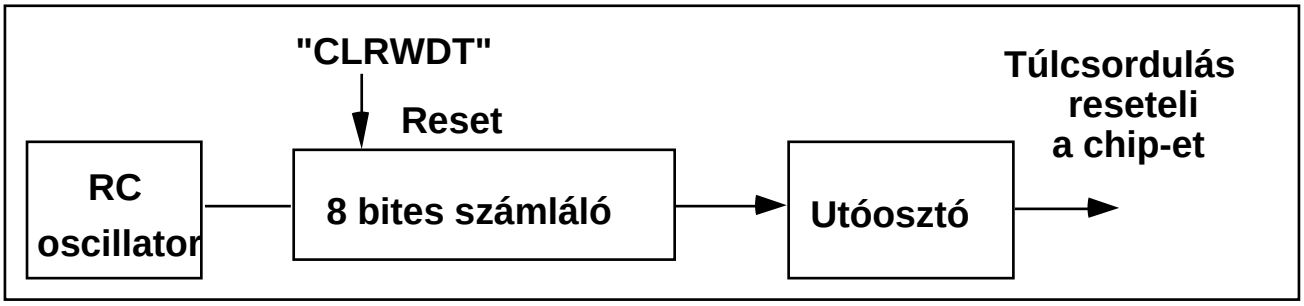
```
XMIT     MOVLW  08h            ;  Bit count
         MOVWF  bit_count      ;

XM_LOOP  BCF    PORTB, bit0    ;  0 ®  Data pin
         BCF    PORTB, bit1    ;  0 ®   Clock pin
         RRF    XDATA          ;  Rot right through carry
                               ;  XDATA = xmit data
         BTFSC  STATUS, CARRY  ;  test carry bit
         BSF    PORTB, bit0    ;  1 ®   Data pin
         BSF    PORTB, bit1    ;  1 ®  Clock pin
         DECFSZ bit_count      ;
         GOTO   XM_LOOP        ;  Not done yet
         BCF    PORTB, bit1    ;  Clear clock pin
```

Megszakítás - áttekintés

- **Akár 12 megszakítás forrás**
- **Csak egy megszakítási vektorcím (04H), prioritás programból**
- **Megszakítás engedélyezés: globális és egyedi**
- **Több megszakítás szundiból ébreszti a kontrollert**
- **Hardver megszakítás késleltetés - 3 utasítás ciklus**

Watchdog Timer (WDT) & TMR0

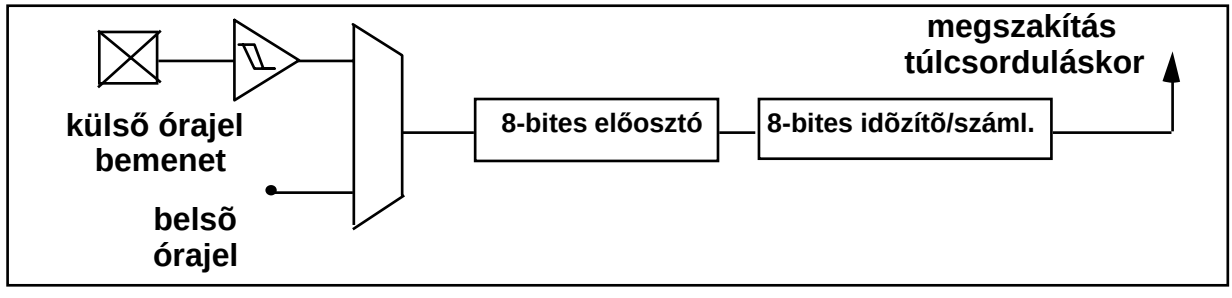
- Segít, ha a program „elkószál”
- Saját szabadonfutó RC oszcillátors van
- WDT programból nem kezelhetõ, csak a CLRWDT utasítással, ami törli
- WDT túlcsordulás reszeteli és újraindítja a kontrollert
- Programozható időtartam (time-out period): 18 ms-tól 2.5 sec-ig
- SLEEP-ben is működik. A túlcsordulás felébreszti a CPU-t



- WDT egedélyezés/tiltása a programozható biztosítékkal (WDT_ON/OFF)
- Egyszerű 8 bites számláló
- Mivel saját belső RC oszcillátor működteti, ezért kristályhibát is jelez

TMR0 (or RTCC)

- Olvasható és írható
- Túlcsordulásakor megszakítást generál
- Az előosztója programozható
- Időzítőként: Az órajel frekvencia: OSC/4 (5 MHz @ 20 MHz-es kontroller órajelnél
- Számlálóként: inkrementálódik , programozható az élváltás, amire lép (fel- vagy lefutó
  Elõosztóval a külsõ órajel 50 Mhz-ig mehet



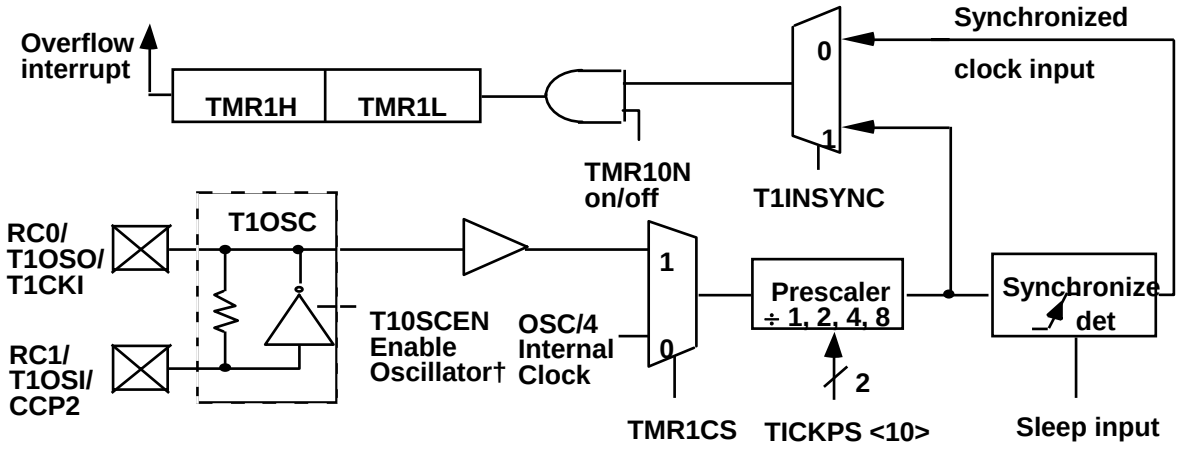
TMR1&2

- **16-bites szaámláló/időzítő, 3 bites előszto (÷ 1, 2,4,8)**
- **Aszinkron számláló mód**
  Sleep ideje alatt is működik
  Ha TMR1 túlcsordul, ébreszti a processzort
  (külső órajel lépteti)

Timer 1 (TMR1)

- **LP oszcillátor**

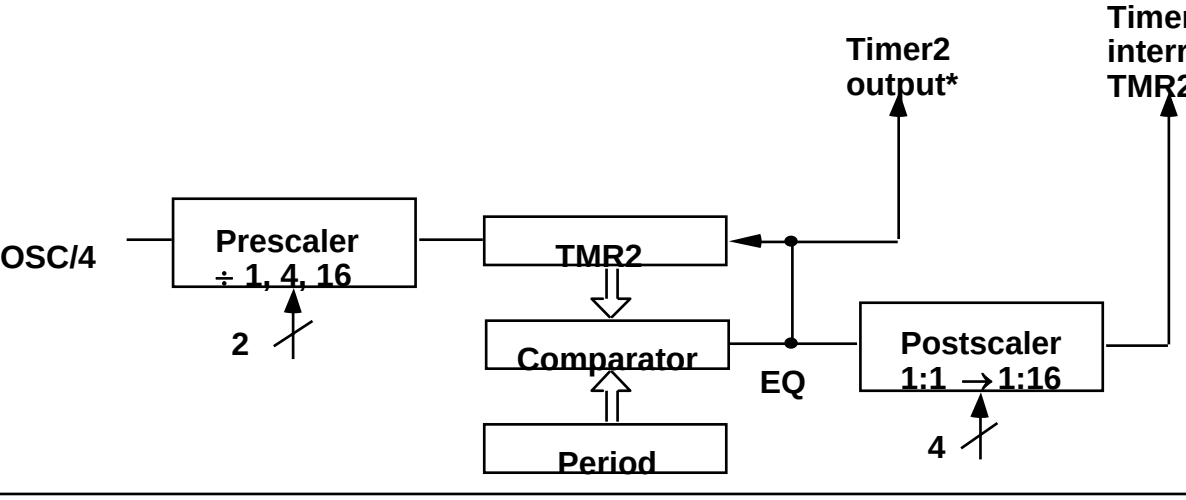
  Direkt 32 KHz - 200 KHz kvarc működés RC0 és RC1-en keresztül
  SLEEP módban is fut az idő
- **A CCP capture és comparátor (CCP) modul(ok) időalapja**



†T1OSCEN törölve van, akkor az inverter és az ellenállás kikapcsolódik. (Nincs fogyasztás)

Timer 2 (TMR2)

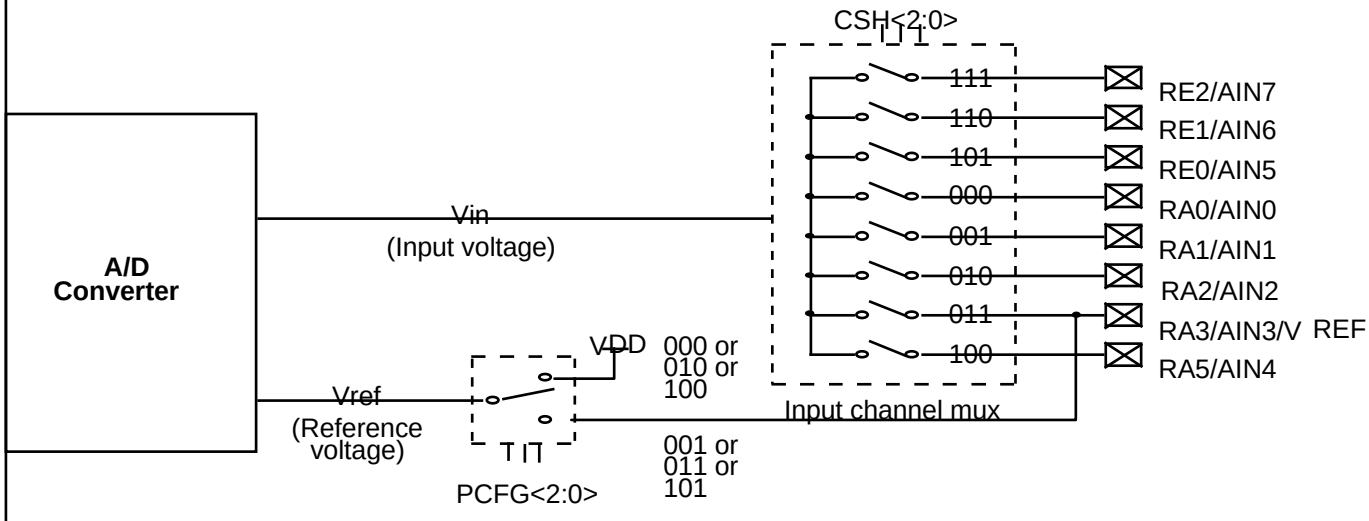
- **8 bites időzítő**
- **4 bites előosztó (÷ 1, 4, 16)**
- **4 bites utóosztó (÷ 1 to ÷16)**
  - **Az utóosztó túlcsordulása megszakítást okoz**
- **A Szinkron Soros Port (SSP) modul baud-rate generátora**

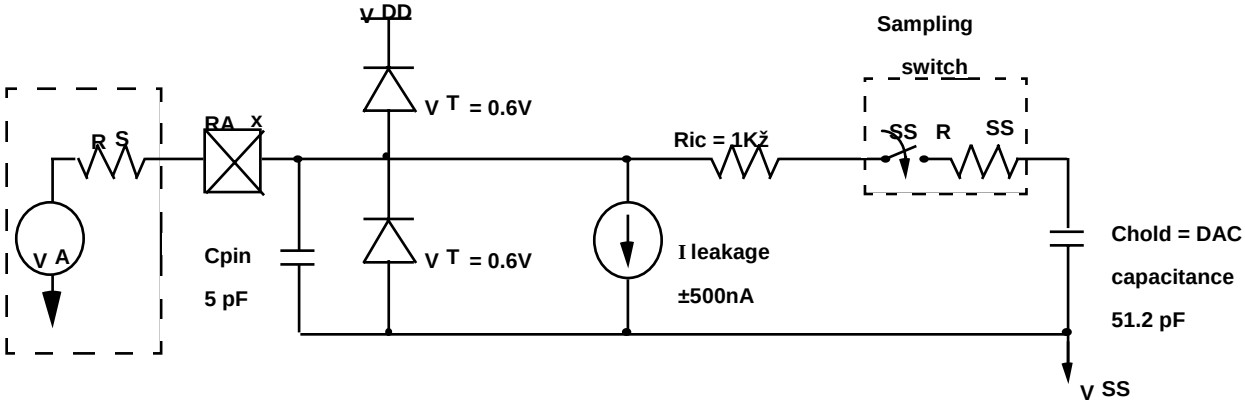


* TMR2 output can be software selected by the SSP module as baud clock.

A/D

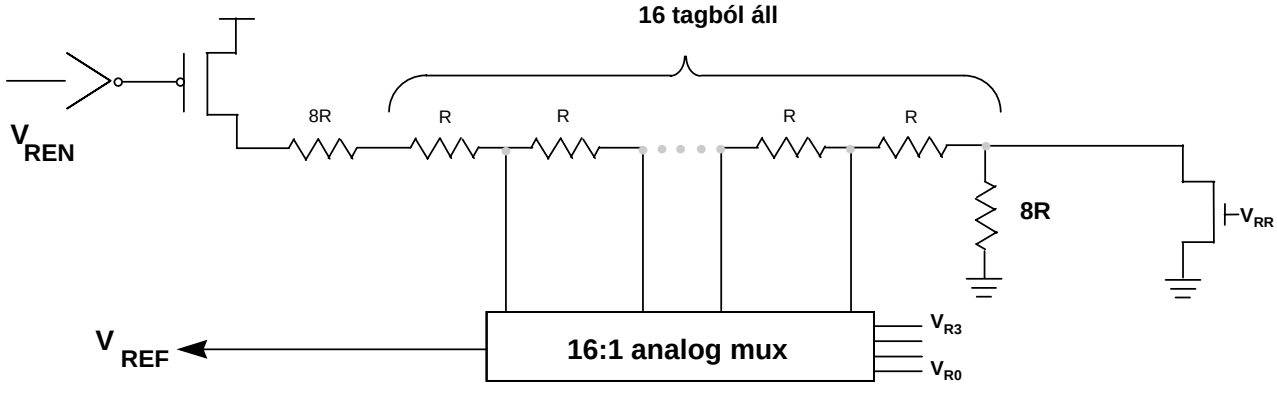
- **A/D konverter modul:**
  - **Max 8 analog bemenet multiplexelődik egy A/D konverterre**
  - **Mintavevő-tartót tartalmaz**
  - **11 µs-os mintavételi idő (10K forrás impedanciánál)**
  - **16 µsec konverziós idõ csatornánként (20 µsec PIC16C71-nél)**
  - **8 bites felbontás (±1 LSB accuracy (±2 LSB @ VDD=3V**
  - **A/D konverzió sleep alatt. A/D kész ébresztheti a processzort**
  - **Külső referencia bemenet, VREF**
  - **Bemenő feszültség tartomány: VSS-VREF**
- **A port lábai programból konfigurálhatók (analóg vagy digitális bemenet**
- **Az analóg bemeneteknek konfigurált lábak digitális kimenetként is működhetnek ha a TRIS bitjeiket töröljük**
- **Digitális bemenetként konfigurált lábra analóg feszültséget téve, a bemeneti puffer árama miatt túlterhelődhet**





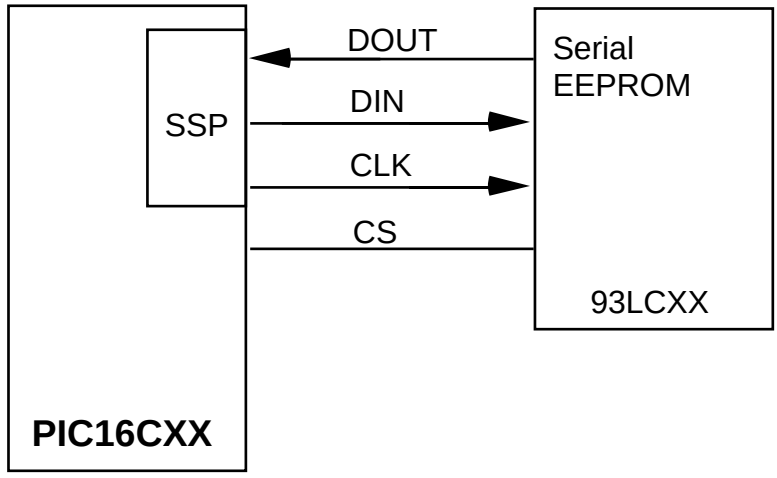
ANALÓG KOMPARÁTOR

- **2 analóg komparátor van egy tokban**
- **Programozható referencia feszültség**
- **8 programozható működési mód (reset komparátor, kikapcsolva, két független komparátor, közös referenciájú, stb.)**
- **Komparátor I/O multiplexelve van a digital I/O-val**
- **Komparátor kimenet megszakítást okozhat,**
- **és ez ébresztheti a tokot sleep-ből**

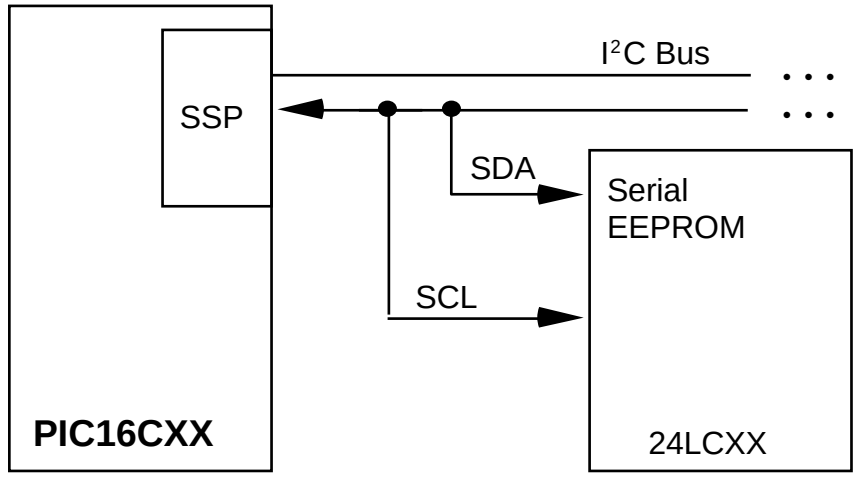


- **V_{REF} 16 lépésből álló feszültséget ad ki**
- **V_{REN} kapcsolja (ON/OFF) a feszültséget a referencia áramkörre**
- **D/A átalakítónak használható**

EEPROM



- **93LCXX család 3(4)-vezetékes SPI interfészű**
- **Max. 16Kbit EEPROM**
- **Egyszerű rutin ≈ 20 sor !!!**



- **24LCXX család 2-vezetékes ^{2}C buszt használ**
- **Maximum 64 Kbit EEPROM**
- **Program: ≈ 100 sor**
- **Több periféria esetén jó választás**

Capture/Compare/PWM (CCP) Modul I.

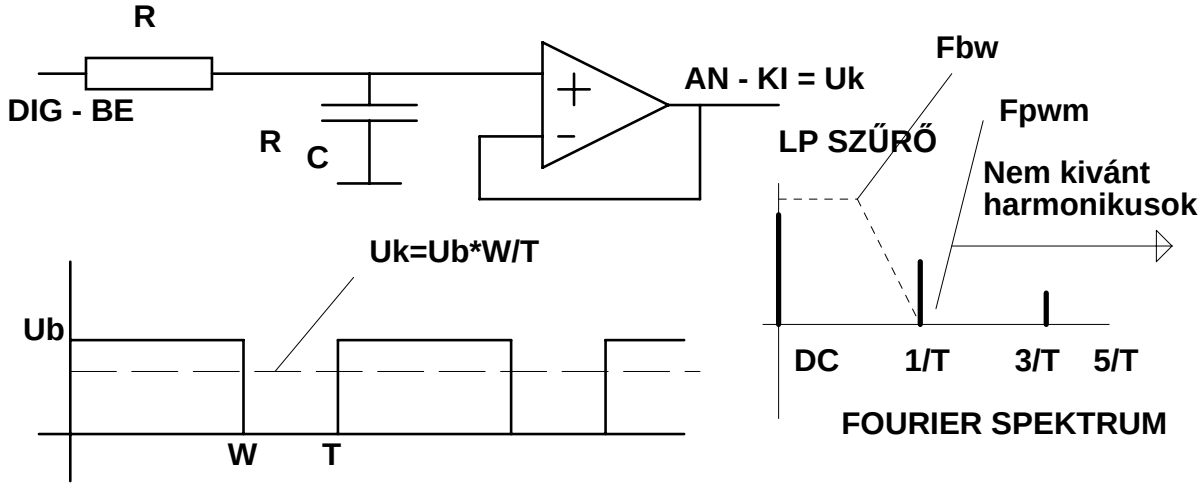
Capture: adott feltétel teljesülése esetén egy számláló értékének beírása egy regiszterbe

- **TMR1 és TMR2 számlálókat használják**
- **Capture mód: TMR1 16 bites értéke a capture regiszterbe íródhat:**
  - **Minden lefutó élnél**
  - **Minden felfutó élnél**
  - **Minden 4.-edik felfutó élnél**
  - **Minden 16.-adik felfutó élnél**
- **Komparátor mód: egy 16-bites értéket TMR1-hez hasonlít, és egyezéskor generálhat:**
  - **CCPX lábon magas szintet,**
  - **CCPX lábon alacsony szintet,**
  - **Szoftver megszakítást, vagy**
  - **kiválthat speciális eseményt (TMR1 törlését, vagy A/D GO bit-jét 1-be)**
- **PWM Mód**

8 vagy 10-bit felbontás: 80 KHz frekvencia 8-bites felbontásnál

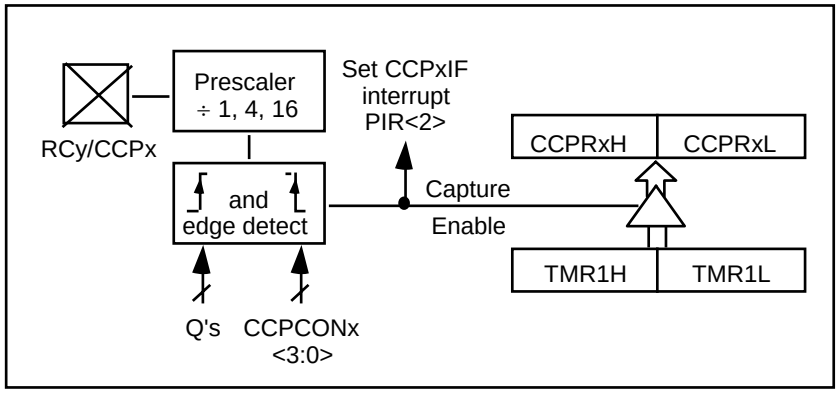
20 KHz frekvencia 10-bites felbontásnál

max. 50 nsec-os felbontás (@ 20 MHz, 10 bites felbontás)

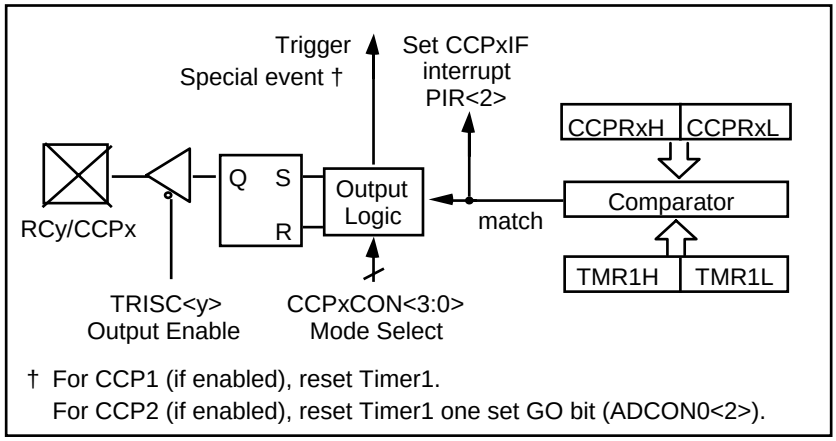


Capture/Compare/PWM (CCP) Modul II.

Capture Mód

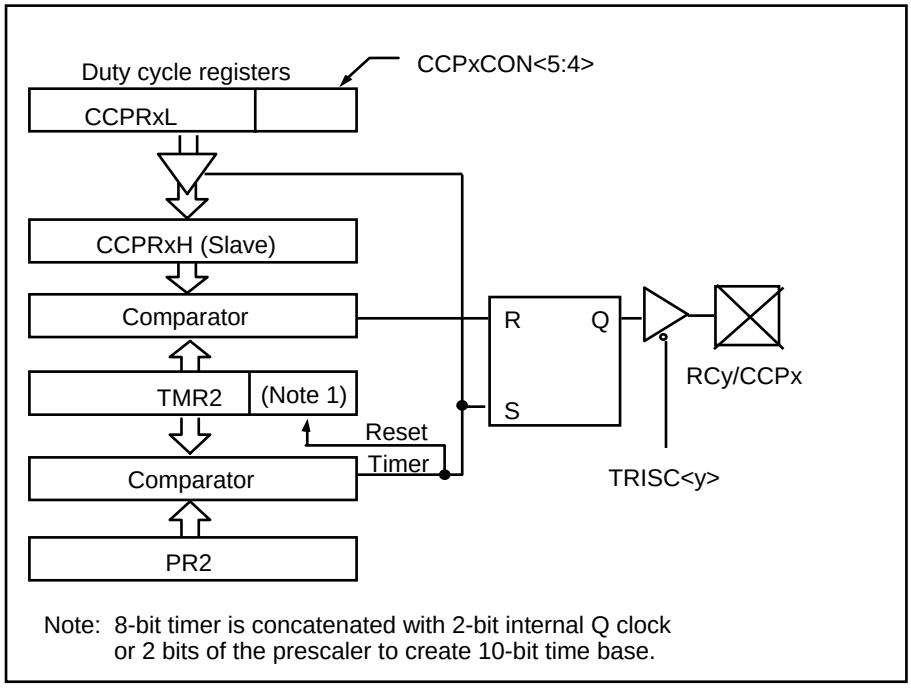


Compare Mód



† For CCP1 (if enabled), reset Timer1.
For CCP2 (if enabled), reset Timer1 one set GO bit (ADCON0<2>).

PWM Mód



Note: 8-bit timer is concatenated with 2-bit internal Q clock
or 2 bits of the prescaler to create 10-bit time base.

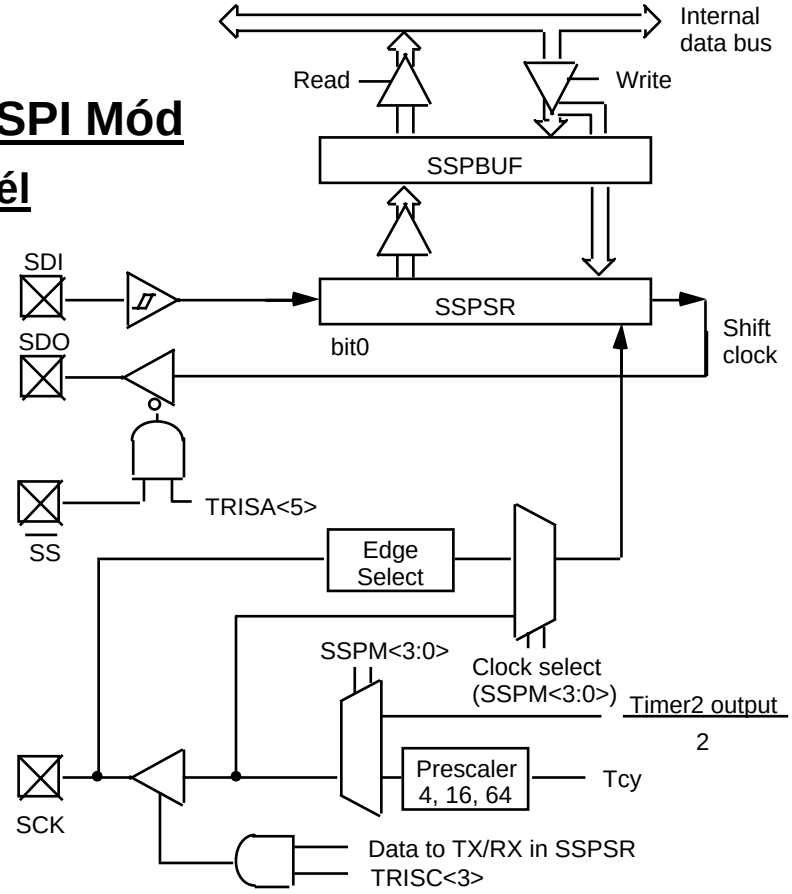
SSP modul

Képes vagy SPI vagy I²C™/ACCESS.bus módban működni

SPI Mód

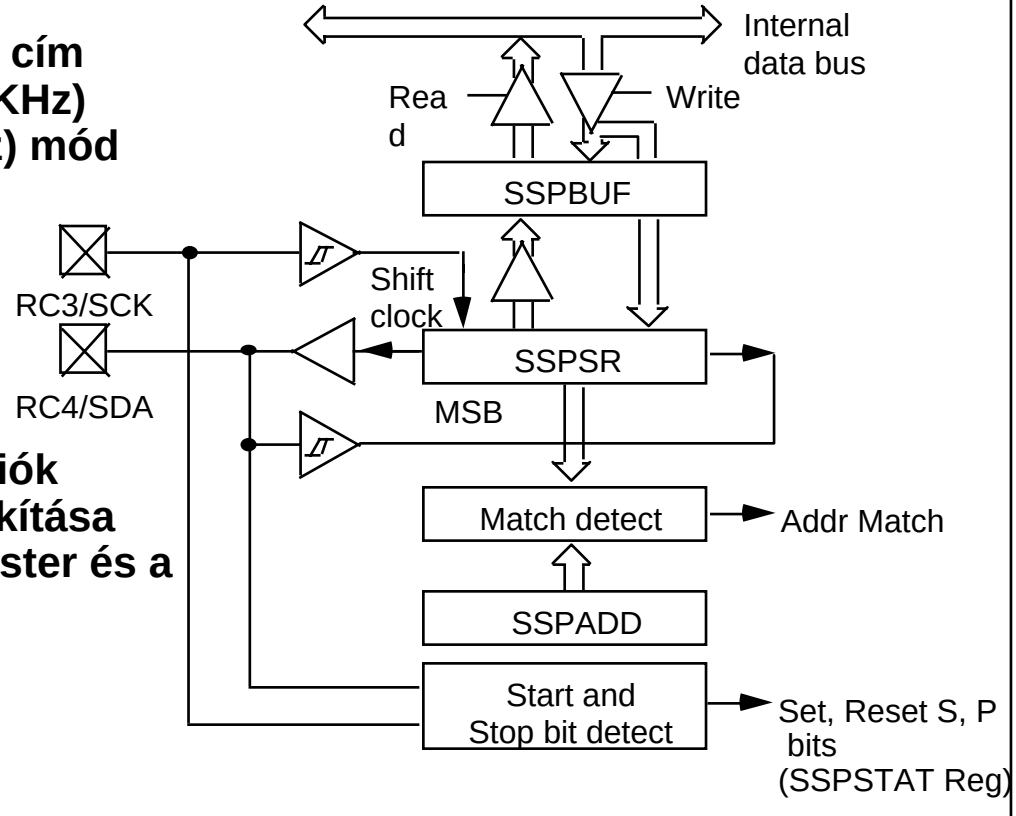
Max baud ütem (20 MHz)-nél
Mester 5 MHz
Szolga 2.27 MHz

Programozható baud ütem. (OSC ÷ 4, 16, 64, vagy TMR2 output ÷ 2)
Programozható órajel polaritás adás/vételnél



I²C™/ACCESS.bus Mód

- **7 vagy 10 bites cím**
- **Standard (100 KHz)**
- **gyors (400 KHz) mód támogatás**



- **A szolga funkciók teljes HW kialakítása**
- **HW segíti a mester és a multi-mester funkciókat**

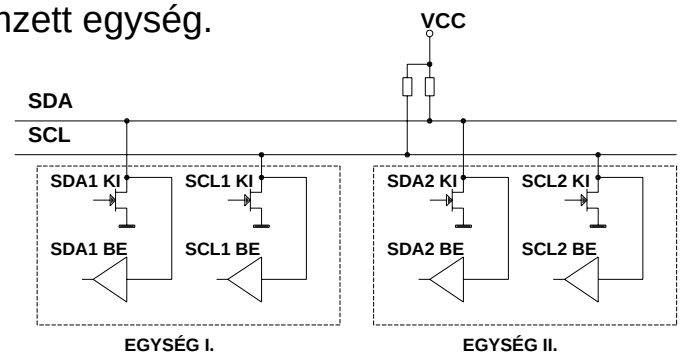
I2C busz

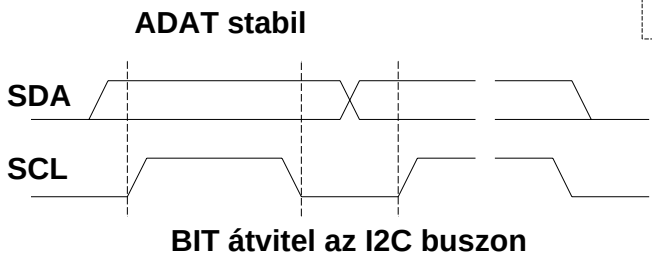
Az adatforgalom két vezetéken történik. Az SCL vezeték az órajelet szolgáltatja , az SDA jelű végzi az adatforgalmazást. A közös potenciált a GND összekötés biztosítja. Az SCL és SDA vonalak kimenetei nyitott kollektoros megoldásúak, így a vonalak alaphelyzetben magas állapotban vannak. Ennek elônye ebben rejlik, hogy nem csupán két , hanem számos eszközt köthetünk össze.

Minden egység lehet **Adó** ill **Vevô**. Ezen felül megkülönböztetünk **Master** és **Slave** eszközöket.:

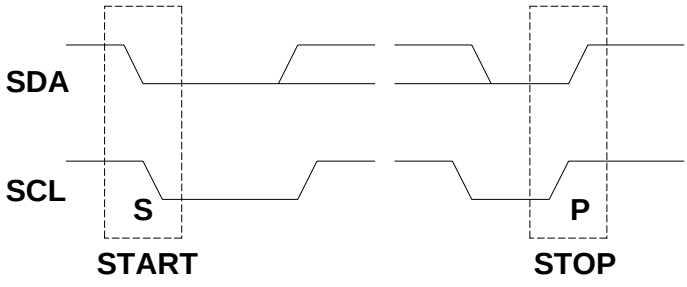
A funkciók: **TRX** = Transmitter (adó): Az az egység amelyik adatot küld a buszra. **RCV** = Recevier (vevô): Az az egység amelyik adatot fogad a buszról.

A szerepek: **MST** = Master (mester): Az az egység amelyik kezdeményezi az átvitelt, az átvitelhez az órajelet generálja, és be is fejezi az átvitelt. **SLV** = Slave (szolga): A mester által megcímzett egység.





BIT átvitel az I2C buszon



START és STOP feltételek

BÁJT átvitel az I2C buszon

Bit szintű átvitel: az eredetileg magas szinten lévő **SDA** vonalra kerül a 0 vagy 1 szint. Az **SCL** vonal magas szintje alatt érvényes az adat. Az adat csak az **SCL** vonal alacsony szintje alatt változhat.

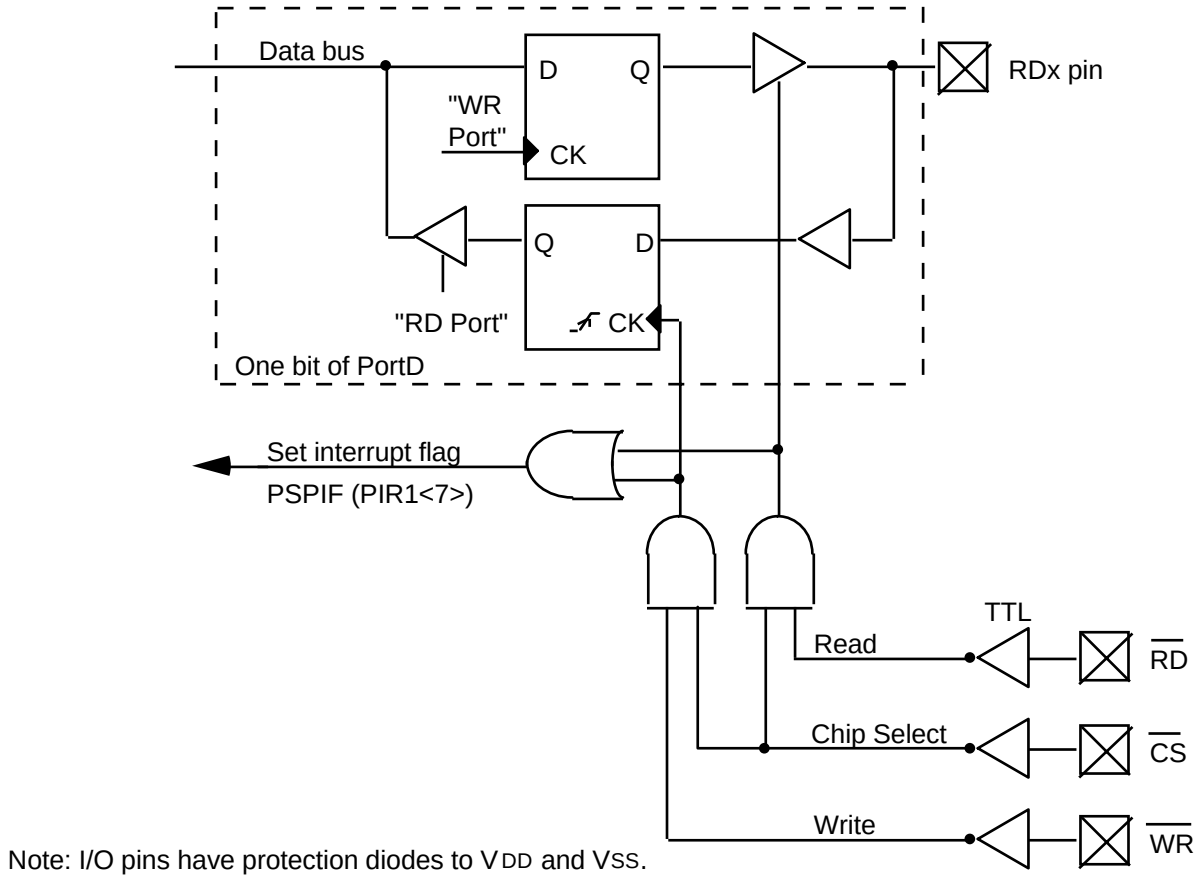
A busz aktív és inaktív állapotát a **START** és **STOP** feltételekkel tudjuk definiálni.

START feltétel akkor lép fel és a busz aktív lesz amikor **SCL** magas állapotában az **SDA** vonalon egy H-L átmenet van. **STOP** feltétel akkor lép fel, amikor **SCL** magas állapotában az **SDA** vonalon egy L-H átmenet van.

A **START** és **STOP** állapotokat csak a mester generálhatja. A busz aktív a **START** és **STOP** állapot között.

Párhuzamos szolga port

Parallel Slave Port Blokkvázlata

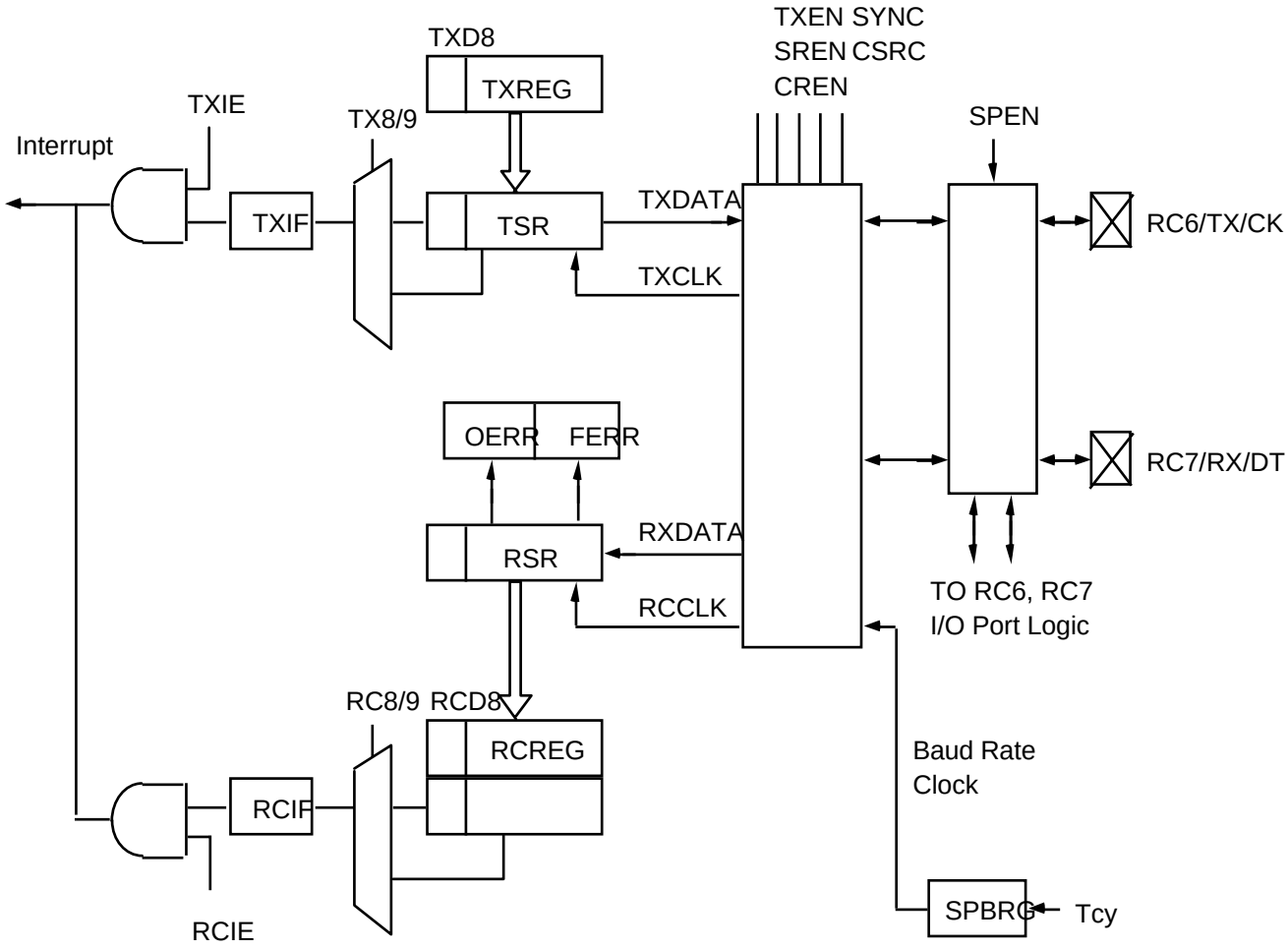


Note: I/O pins have protection diodes to V_{DD} and V_{SS}.

8 bites mikroprocesszor adatbuszához direkt köthető
Aszinkron működés (a külvilág felé)
A párhuzamos port külső írásakor vagy olvasásakor megszakítás generálódik
A Port D és Port E portokat használja
Port D Adatbusz
Port E vezérlő jelek (read, write, és chip select)

SCI modul

SCI blokkvázlata



SCI Aszinkron Mód

Adás

Adó regiszter kettős pufferelésű
Ha az adó kész az adatátvitelre, megszakítást generál
8/9 bites adás

Vétel

Vevő regiszter kettős pufferelésű
Felülírási hiba jelzőbit
Keretezési hiba detektálása
Minden adatbit többségi detektálása (zajvédelem)
Ha bejött egy teljes adat, megszakítást generál

SCI Szinkron Mód

Adás

Adó regiszter kettős pufferelésű
Egymást követõ bájtokat úgy viszi át, hogy lehetséges 16 bites szavak átvitele, vagy akár hosszabbakat is
8/9 bites adás

Vétel

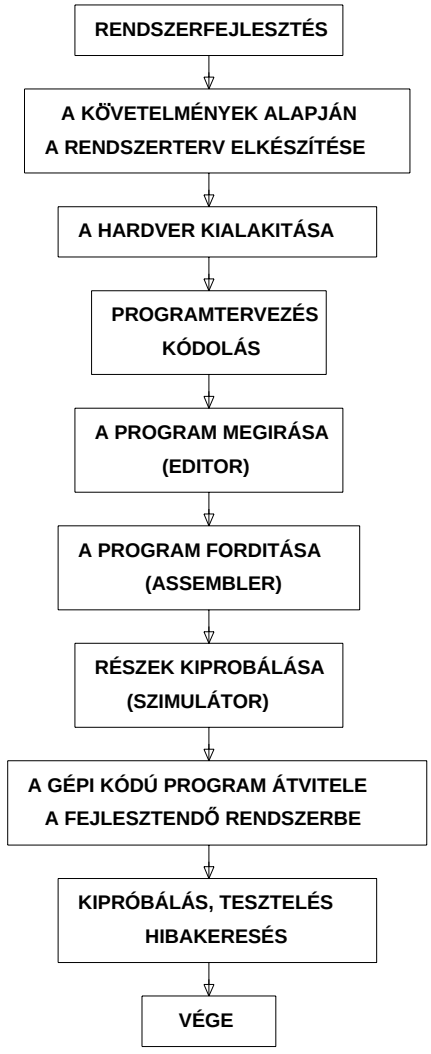
Vevő regiszter kettős pufferelésű
8/9 bites vétel
Egymást követõ bájtokat úgy veszi, hogy lehetséges 16 bites szavak vétele, vagy akár hosszabbaké is

PIC RENDSZERFEJLESZTÉS

Parallax-féle utasításkészlet

ADD	fr,#lit	ADD	fr1,fr2	ADD	fr,W
ADD	W,fr	ADDB*	fr,bit	AND	fr,#lit
AND	fr1,fr2	AND	fr,W	AND	W,#lit
AND	W,fr	CALL	addr8	CJA	fr,#lit,addr9
CJA	fr1,fr2,addr9	CJAE	fr,#lit,addr9	CJAE	fr1,fr2,addr9
CJB	fr,#lit,addr9	CJB	fr1,fr2,addr9	CJBE	fr,#lit,addr9
CJBE	fr1,fr2,addr9	\CJE	fr,#lit,addr9	CJE	fr1,fr2,addr9
CJNE	fr,#lit,addr9	CJNE	fr1,fr2,addr9	CLC	
CLR	fr	CLR	W	CLR	WDT
CLRB	bit	CLZ		CSA	fr,#lit
CSA	fr1,fr2	CSAE	fr,#lit	CSAE	fr1,fr2
CSB	fr,#lit	CSB	fr1,fr2	CSBE	fr,#lit
CSBE	fr1,fr2	CSE	fr,#lit	CSE	fr1,fr2
CSNE	fr,#lit	CSNE	fr1,fr2	DEC	fr
DECSZ	fr	DJNZ	fr,addr9	IJNZ	fr,addr9
INC	fr	INCSZ	fr	JB	bit,addr9
JC	addr9	JMP	addr9	JMP	PC+W
JMP	W	JNB	bit,addr9	JNC	addr9
JNZ	addr9	JZ	addr9	LCALL*	addrll
LJMP*	addrll	LSET*	addrll	MOV	fr,#lit
MOV	fr1,fr2	MOV	fr,W	MOV	OPTION,#lit
MOV	OPTION,fr	MOV	OPTION,W	MOV	!port fr,#lit
MOV	!port fr,fr	MOV	!port fr,W	MOV	W,#lit
MOV	W,fr	MOV	W,/fr	MOV	W,fr-W
MOV	W,++fr	MOV	W,--fr	MOV	W,<<fr
MOV	W,>>fr	MOV	W,<>fr	MOVB	bit1,bit2
MOVB	bitl,/bit2	MOVSZ	W,++fr	MOVSZ	W,--fr
NEG*	fr	NOP		NOT	fr
NOT	W	OR	fr,#lit	OR	fr1,fr2
OR	fr,W	OR	W,#lit	OR	W,fr
RET		RETW	lit,1it,...	RL	fr
RR	fr	SB	bit	SC	
SETB	bit	SKIP		SLEEP	
SNB	bit	SNC		SNZ	
STC		STZ		SUB	fr,#lit
SUB	fr1,fr2	SUB	fr,W	SUBB*	fr,bit
SWAP	fr	SZ		TEST	fr
XOR	fr,#lit	XOR	fr1,fr2	XOR	fr,W
XOR	W,#lit	XOR	W,fr		

Jelölések: **fr - file regiszter, #lit - konstans, W - w regiszter, addr8 - 8 bites cím,**
(*Aki ismeri a 51-es utasításkészletet, annak nagyon szembetűnő a hasonlóság.)*

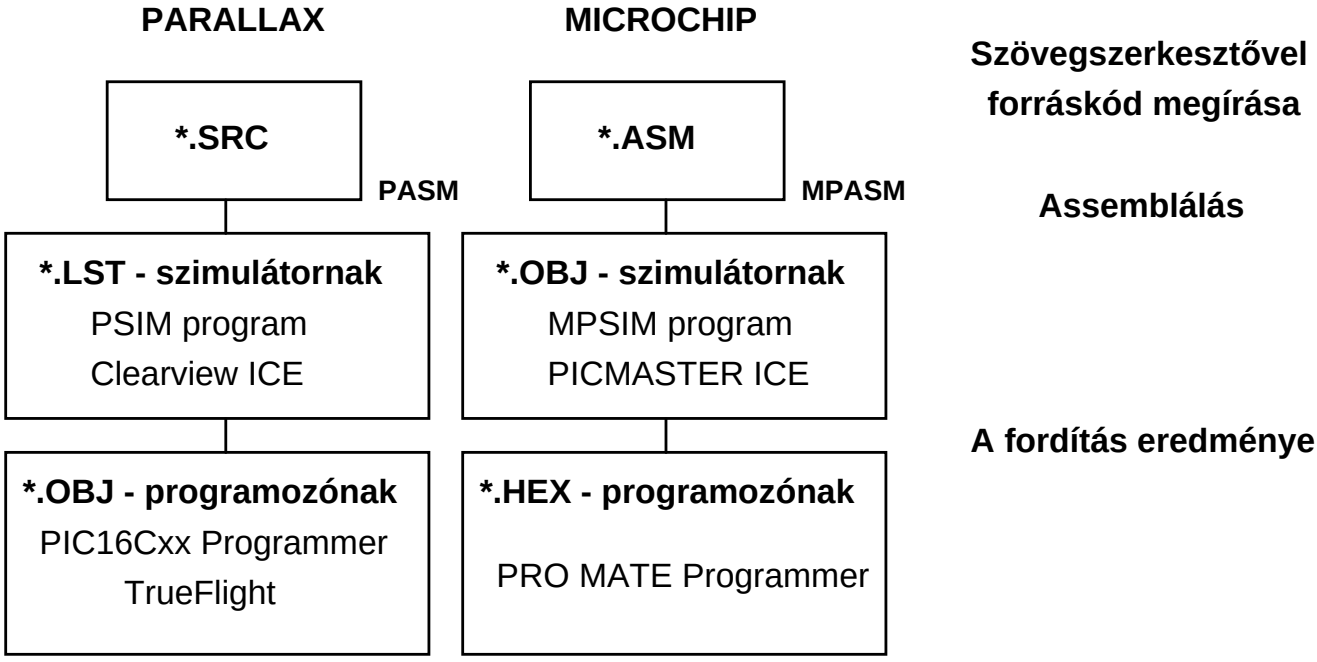


A RENDSZERFEJLESZTÉS BLOKKVÁZLATA

Parallax szimulátor

```
                        PIC 16C84 Simulator  v2.09
         HEX     BINARY         0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
STACK 1 0000 0000000000000   0 00 00 00 18 00 1F FF 00 00 00 00 00 00 00 00 00
STACK 2 0000 0000000000000   1 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
OPTION    FF      11111111   2 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
W         00      00000000   8 00 FF 00 18 00 1F FF 00 00 00 00 00 00 00 00 00
RTCC      00      00000000   9 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
PC      0000 0000000000000           LATCH           PIN            TRI-STATE
STATUS    18      00011000   PORT A  1F  00011111   00  00000000   1F  00011111
FSR       00      00000000   PORT B  FF  11111111   00  00000000   FF  11111111
INTCON    00      00000000
 MCLR RTCC IRP RP1 PR0 TO PD  Z DC  C   WD      CYCLES     TIME       XTAL
   1    0   0   0   0   1  1  0  0  0  0.0180  00000000  0.0000000   4mhz


000                 mov     !ra, #00000000b ;inicializalas
001
002                 mov     !rb, #0         ;kimenet
003
004                 inc     rb
005                 mov     szaml,#0ffh
F1-HELP F2-BRKPT F3-CLEAR F4-HERE F5-TIME F6-GO F7-STEP F8-NEXT F9-RUN F10-RST
```

PIC fejlesztés és a család

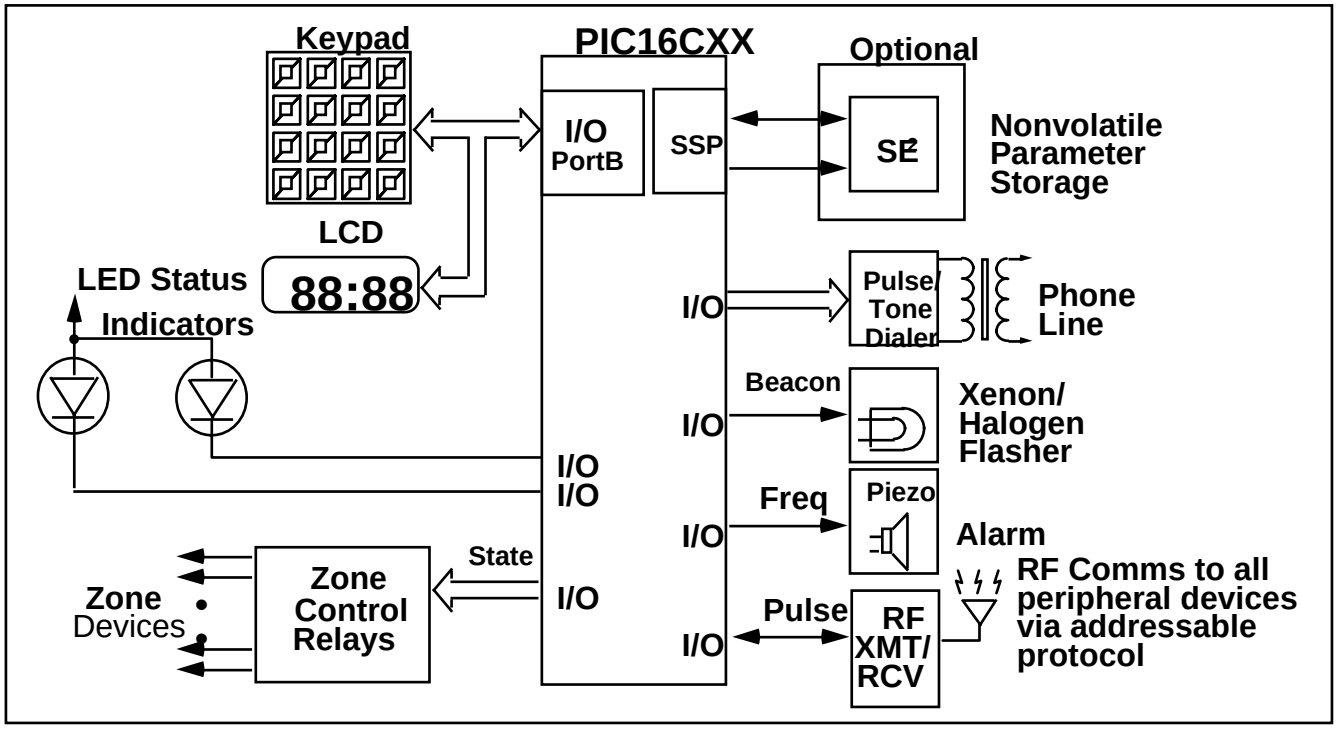


PIC CSALÁD - Összefoglaló

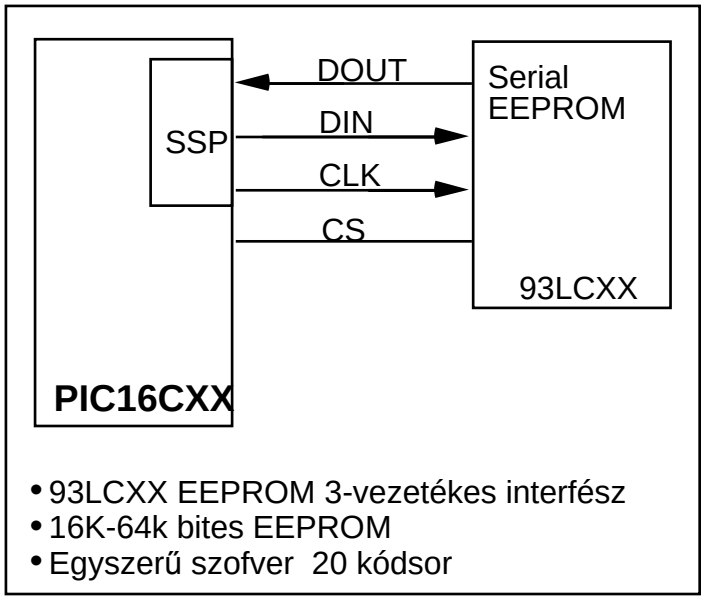
TIPUSOK	PIC16C5X								PIC16CXX					PIC17CXX
Utasítás-hossz (bit):	12								14					16
Jellemzők	54	54A	R54	55	56	57	57A	58A	61	64	71	74	84	42
SEBESSÉG	20	20	20	20	20	20	20	20	20	20	20	20	10	20
EPROM	512	512		512	1K	2K		2K	1K	2K	1K	4K		2K
ROM			512				2K							
EEPROM													1K	
RAM	25	25	25	25	25	72	72	72	36	128	36	192	36	232
EEPROM													64	
IDŐZÍTŐK	1+WDT	1+WDT	1+WDT	1+WDT	1+WDT	1+WDT	1+WDT	1+WDT	1+WDT	3+WDT	1+WDT	3+WDT	1+WDT	4+WDT
CAPTURE/COM-PARE/PWM										1		2		2
SOROS PORT										SPI/I2C				SCI
PÁRHUZAMOS SLAVE PORT										IGEN		IGEN		
8 bites A/D											4	8		
MEGSZAKÍTÁ-SOK									3	8	4	12	4	11
I/O vonalak	12	12	12	20	12	20	20	12	13	33	13	33	13	33
Tápfesz (V)	2.5-6.2	2.5-6.2	2.0-6.2	2.5-6.2	2.5-6.2	2.5-6.2	2.5-6.2	2.5-6.2	3.0-6.0	2.5-6.0	3.0-6.0	3.0-6.0	2.0-6.0	4.5-5.5
Utasításszám	33	33	33	33	33	33	33	33	35	35	35	35	35	55

PIC alkalmazások I.

Biztonsági rendszer

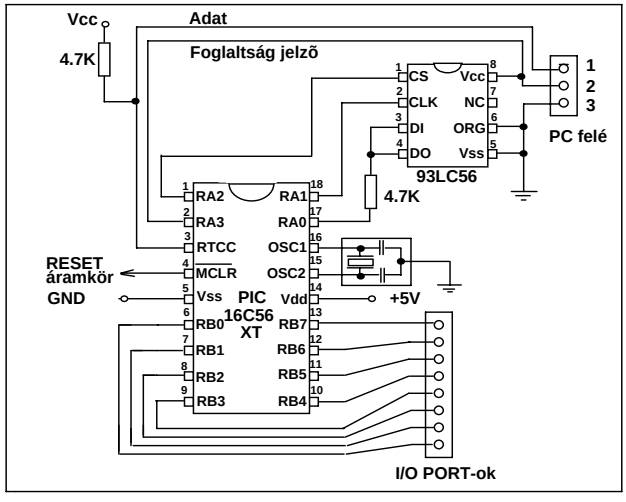


Soros EEPROM illesztése



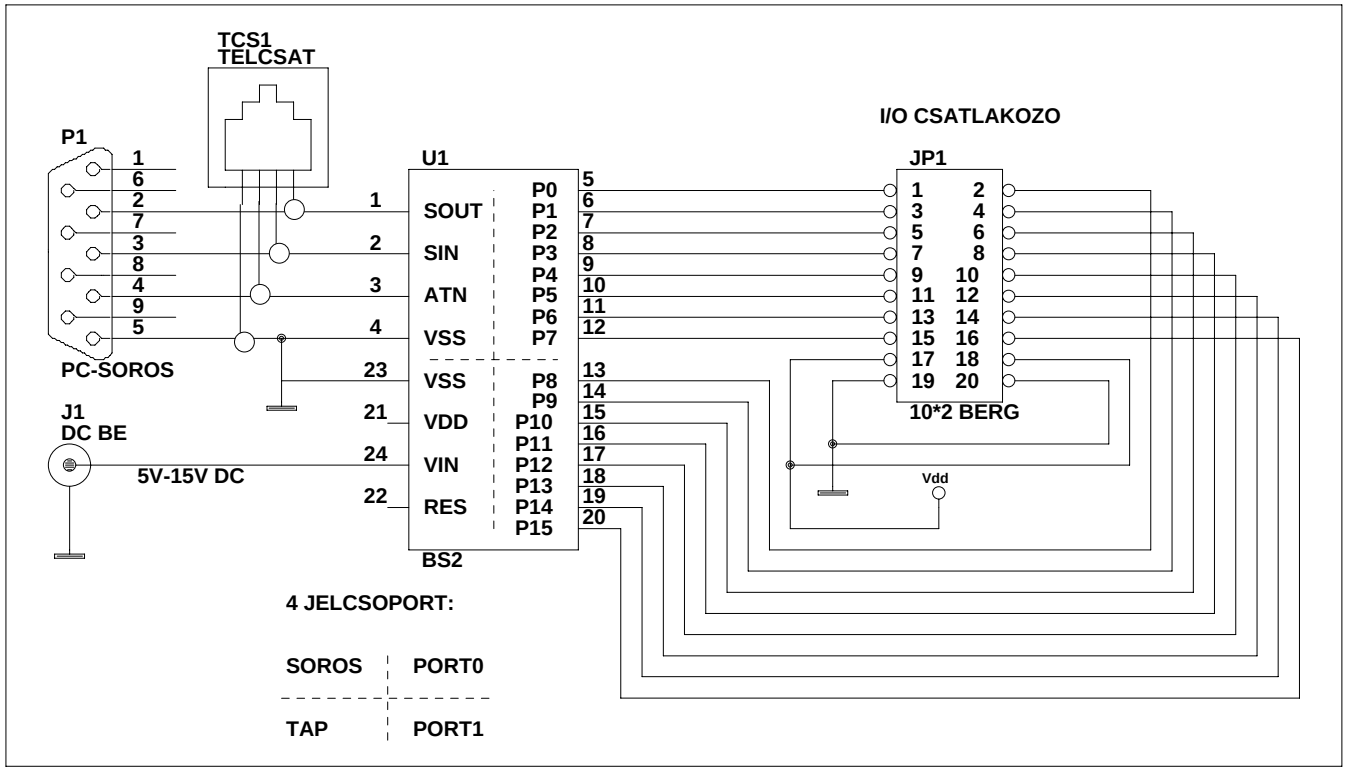
- 93LCXX EEPROM 3-vezetékes interfész
- 16K-64k bites EEPROM
- Egyszerű szofver 20 kódsor

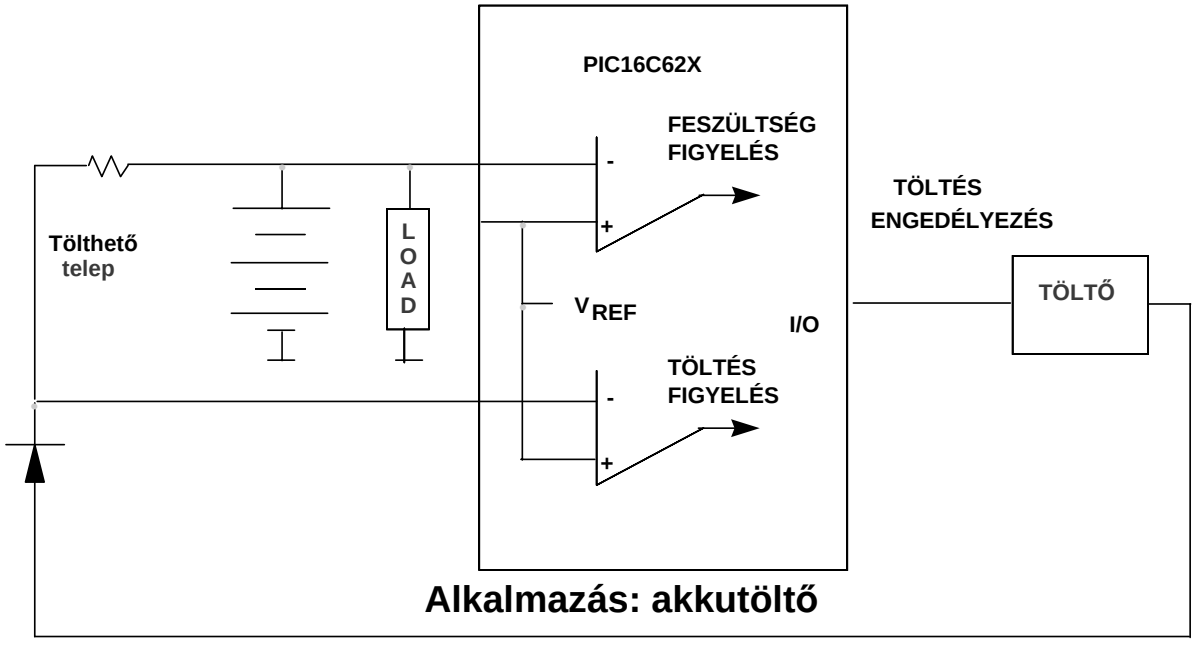
BASIC STAMP I.



PIC alkalmazások II.

BASIC STAMP II





Alkalmazás: akkutöltő

PIC16/17 mikrokontrollerek

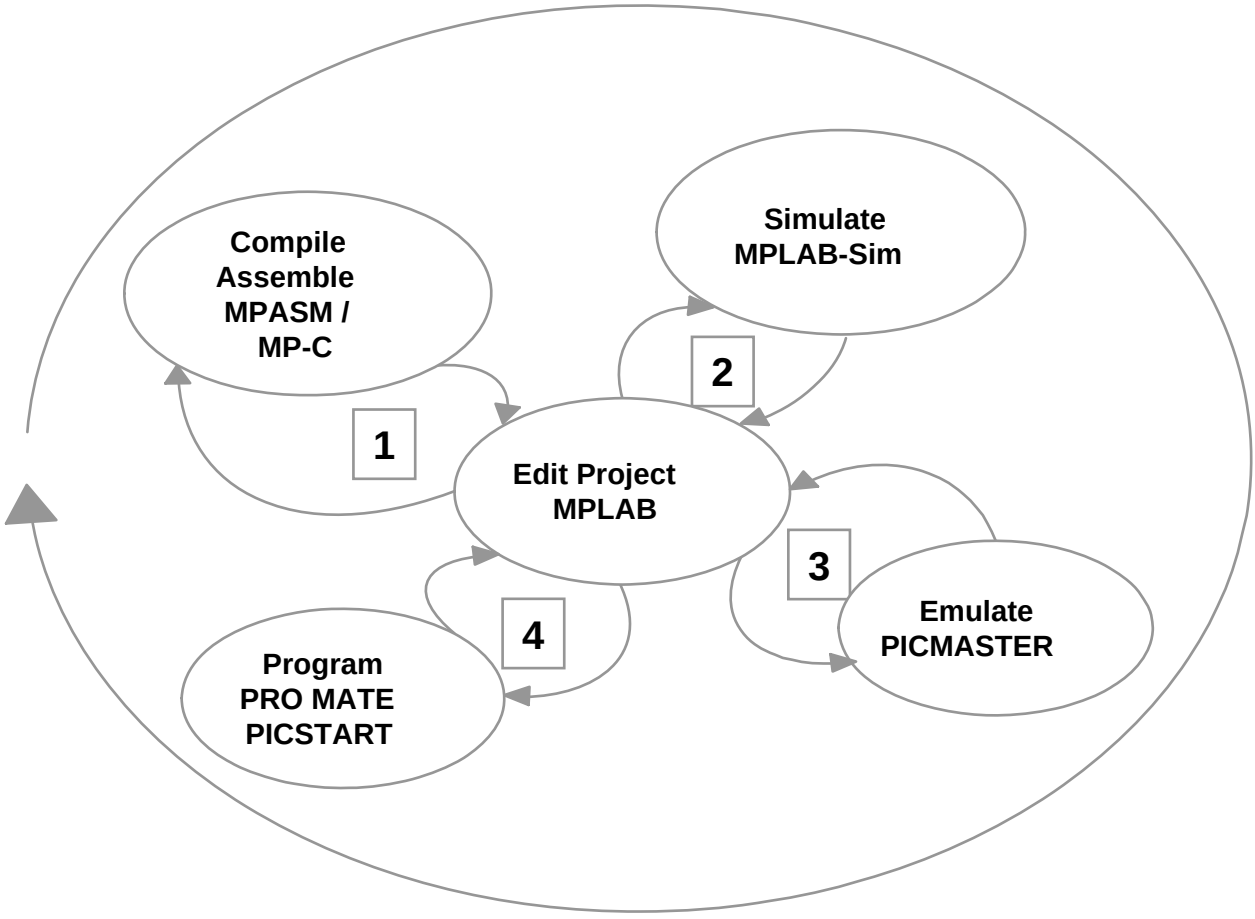
Hardver áttekintés

- **Harvard architektúra, külön program és adatmemória**
- **Kis áramfelvétel (15 mA tipikusan 3V, 32 KHz-nél)**
- **Teljes statikus kialakítás**
- **Kisfogyasztású SLEEP (szundi) üzemmód (< 1 mA 3V-nál)**
- **Minden utasítás egyszavas**
- **Minden utasítás (kivéve az ugrások) egy ciklusú**
- **Watchdog időzítő belső RC oszcillátorral**
- **Kódvédelem**
- **A digitális I/O vonalak nagy meghajtó/nyelő árama (25 mA)**
- **Beépített reszet, feszültségcsökkenés figyelő (brown-out)**

P E R I F É R I Á K

- **Duál 10-bit / 20 KHz, 8-bit / 80 KHz PWM**
- **8-bites max 8 csatornás, 26 msec konverziós idejű SAR A/D**
- **16-bites 16 csatornás integráló A/D**
- **Duál 5 mV-os ofszetű komparátor**
- **8-bites DAC**
- **Belső hőmérséklet érzékelő**
- **Belső feszültség referencia**
- **Belső oszcillátor**
- **116 pixeles 32 X 4 multiplexelt LCD meghajtó**
- **8/9-bites soros USART**
- **Telepes óra**
- **Duál 16-bites 200 nsec-os capture regiszter**
- **Duál 16-bites 200 nsec-os komparátor kimenet**
- **SPI / I^2C busz illesztés**
- **Párhuzamos slave port**
- **64 bájtos EEPROM adatmemória**

MPLAB fejlesztő rendszer



Fejlesztés lépései

- *Hardver és szoftver tervezés*
- *Programírás (editálás) (forráskód készítése)*
- *Gépi kód (tárgykód) előállítása assembler, compiler*
- *program jóságának ellenőrzése*
  - *szimulátor (program)*
  - *emulátor (hardver)*
  - *programozó*

MPASM univerzális assembler

Jellemzői

- Univerzális assembler minden PIC16/17 típushoz
- Windows alatt futtatható
- Makrózási lehetőség
  - Sztring helyettesítő makró (C stílusú #define)
  - Kód makró
- Feltételes assemblálás
- Include fájlok használata
- C stílusú kifejezés kezelés
- Több szabványos Intel hex formátum generálása

Fájlok létrehozása

- ***Automatikus*** assemblálás a MAKE projekt módszerrel (MPLAB)
- Gépi kód generálása a programozók és emulátorok számára
- A bemenet:
  *.ASM forrásfájl
- A kimenet:
  *.HEX fájl (programozó, emulátor, szimulátor számára)
  *.LST fájl listához
  *.ERR hibafájl
  *.COD fájl a szimulátor, emulátor szimbólumokhoz

MPASM univerzális assembler

List Direktívák

Minden parancssorban megadható direktíva az .ASM fájl LIST kulcsszava után is megadható

Szintakszis:

LIST <OPTION>=<VALUE>, ...

Opció	Default érték	Leírás
P	Nincs	Processzor típus-Megadni kötelező! 16C54, 17C42, etc..
E	ON	Eng./Tiltás Hiba fájl generálás
L	ON	Eng./Tiltás Lista fájl generálás
M	ON	Eng./Tiltás Makró kiterjesztés
C	ON	Eng./Tiltás Kis/nagybetű megkülönb.
Q	OFF	Eng./Tiltás hibaüzenetek kijelzése
R	HEX	Radix definiálása (HEX,DEC,OCT)
X	OFF	Eng./Tiltás keresztreferencia fájl generálás
F	INHX8M	HEX output fájl formátum INHX8M használt a MCHIP-nél

Példa:

```
LIST P=16C74
....
END
```

Megj.:
Processzor “P=<proc>“ *mindig* az első sor
END *mindig* az utolsó sor

MPASM univerzális assembler

Include direktíva

- Speciális funkcóju regisztereket, (STATUS, PORTok, stb.) a “Processor Definition” include fájlokban kell megadni, pl.:
  - P16C74A.INC
  - P17C44.INC
- A felhasználói könyvtárak, szubrutinok szintén include fájlba helyezhetők
- Assembler az include fájlokat úgy kezeli, mint forráskódot
- Szintaktika:
  - #include <path><FILENAME.INC>
  - #include C:\MPLAB\P16C74.INC
- A proceszort az #include előtt kell definiálni

ORG, CBLOCK, EQU direktívák

ORG: Beállítja a program kezdőcímét

```
ORG   0x04  ; a következő utasítás 4H címen lesz
```

EQU: Értékhez szimbólumot rendel, adatmemória címek helyett szimbólumok használhatók

```
COUNT      EQU   0x020  ; COUNT használható 20H című regiszter helyett
INPULSES   EQU   0x021  ; INPULSES használható 21H című reg. helyett
```

CBLOCK: Szimbólumtömböt definiál, jól használható regiszterek megadására

```
CBLOCK 0x20     ; A definíció a  20H című regiszternél kezdődik
COUNT           ; COUNT a 20H cimű fájlregisztert használja
INPULSES        ; INPULSES pedig a 21H-t
  ENDC
```

MPASM univerzális assembler

Konfigurációs bitek direktíva

Oszcillátor típus, kódvédelem, WDT beállítása

__CONFIG direktíva állítja be a biteket, mikor “Processzor definíció”-t pl. P16C74.INC használjuk

Csak az eszközben megtalálható bitek definiálása

A konfigurációs bitek között a & karaktert kell használni

Például: PIC16C54 - XT oszcillátor, WDT off, CP on, IDLOCS = 1234

```
#include “P16C5X.INC”
__CONFIG  _XT_OSC & _WDT_OFF & _CP_ON __IDLOCS  1234
```

Példaprogram: ADVGENIC.ASM

Szöveg tárolása

- **Az alkalmazásokban sokszor kell szövegeket megjeleníteni**
- **A szövegeket RETLW utasításokat tartalmazó táblázatokban tároljuk**
- **DT direktíva (define table) minden szövegkarakter elé teszi a RETLW utasítást:**

DT “Hello” ; Hello szöveg tárolása

Ami azonos a következővel:

```
RETLW‘H’
RETLW‘e’
RETLW‘l’
RETLW‘l’
RETLW‘o’
```

MPASM univerzális assembler

MACRO, ENDM direktívák

- Felhasználó által definiált utasítás sorozat forrásszövegbe illesztése
- Makrók előnyei:
  - Gyakran használt megoldások következetes alkalmazása
  - A forráskód egyszerűbb módosíthatósága
  - Könyebb tesztelhetőség
- Használat előtt kell definiálni !!!

Szintakszis:

- Definíció:

```
<macro_name>   MACRO        [<arg1>, ... ,<argn>]
                 . . .
               ENDM
```

- Használat:

```
<macro_name>          [<arg1>, ... , <argn>]
```

- Pl.:

```
LEDon   MACRO              ; ledon makró megadása
        BSF    PORTA,1     ; LED világít
        ENDM
        ....
        LEDon              ; LED be, makróhívás
```

Feltételes assemblálás

A forráskód feltételtől függően fordítható

Moduláris programírást tesz lehetővé

Szimulátoros hibakeresés

IF - Ezzel kezdődika feltételes blokk

Szintakszis:

```
IF        <expr>
```

ENDIF - a feltételes blokk vége

```
ENDIF
```

MPLAB-SIM

- Univerzális szimulátor PIC16/17 családhoz
- PC Windows és MPLAB-kompatibilis
- Diszkrét esemény, utasítás alapú szimuláció
- Teljes forrásszintű hibakeresés (debugging)
- Korlátok nélküli összetett töréspontok
- Megszakítás és perifériaműködés szimulációja (A/D és soros I/O nem!)
- Stimulus fájlok
- A DOS alapú MPSIM-et helyettesíti

Stimulus Fájl

- Stimulus fájlban adható meg bármelyik bemenet adott időpontba történő gerjesztése
- Utasítás szinten szimulálja a láb bemeneti feltételeit
- MPLAB-ban <FILENAME>.STI fájl szerkesztése
- Meg kell adni a portot, időt, állapotot
- Szintakszis:
  - STEP <PORT#> <PORT#>
  - <TIME> <VALUE> <VALUE>
- **Például:**

```
STEP    RA0     RA1     ! Portdefiniálás
3       0       1       ! 3. lépésnél, RA0=0, RA1=1
12      1       1       ! 12. lépésnél, RA0=1, RA1=1
```

MPLAB-SIM

Billentyű és időpont bemeneti gerjesztés

- Aszinkron, billentyű megnyomáshoz rendelt bemenet
  - Funkcióbillentyű I/O láb állapothoz rendelése
  - Mikor a Stimulus Dialog dobozban egy gombot megnyomunk, az I/O láb állapotot vált, az állapotok:
    - High - magas
    - Low - alacsony
    - Pulse - impulzus
    - Toggle - ellentétére vált
- Szinkron ismétlődő órajel esemény bemenet
  - I/O lábhoz óra rendelhető
  - Frekvenciája, periódusideje választható

Állapotváltás regiszter fájllal

- Register Stimulus új regiszterértéket tölt a regiszterekbe, mikor egy adott címet elértünk
- Perifériamodulok szimulálásakor hasznos
- Pl. az A/D átalakítás eredményét tartalmazó ADRES reigszter tartalmát egy fájlból töltjük, mikor kiolvassuk a tartalmát
- Példaprogram: ADVAL.INJ:

0x16	Első híváskor 16 HEX-et tölt
0x84	Másodikra 84 HEX
...	stb.

MPLAB-SIM

Trace (nyomkövetés) fájl

- A nyomkövetés során fájlban tárolhatjuk el az utasításokat és a regiszterek állapotát
- A fájlt a szimulátor hozza létre és többek között tartalmazza:
  - Lépésszám
  - A stopper által mutatott időpontot
  - Megváltozott regiszter értékeket
- A nyomkövetés adatai a képernyőn fognak megjelenni és egy generált ASCII szövegfájlba kerülnek

Példaprogram: ADVSTIM.ASM

Programozás

Architektúra

A nagy teljesítmény okai:

- Harvard architektúra Minden utasítás egyszavas (Long word)
- Egyciklusos utasítások
- Utasítás vonal (pipelining)
- Redukált utasításkészlet

Szoftver architektúra:

- Program memórai lapozása (*paging)* nagy programoknál
- Fájlregiszter bankváltás (*banking)* nagy SRAM esetén
- Ugró utasítások 2 ciklus hosszúak

Műveleti kód és operandus egy program-szóban

Programszó hossza típusfüggő:

16C5XX = 12 bit, 16CXXX = 14 bit,
17CXXX = 16 bit

Bennfoglalt utasításra példa:

MOVLW #imm<8>	1100XX	imm<8>

Közvetlen címzésre példa:

MOVWF #faddr<7>	0000001	faddr<7>

Programvezérlő utasításra példa:

GOTO #location<11>	101	location<11>

Programozás
Lap / Bank határok

PIC16/17 Family	Word Size (Bits)	Code Page Boundary (Words)	Current Max code Size (Words)	Register Bank Boundary (Bytes)	Current Max RAM Size (Bytes)
16C5XX	12	512	2K	32	73
16CXXX	14	2048	4K	128	192
17CXXX	16	8192	8K	256	454

Program memória lapozás (Paging)

- A lapozó bitek értéke határozza meg az aktív lapot
- A lapozásra CSAK akkor kell figyelni, ha CALL vagy GOTO utasítást hajtunk végre
- A lapozó biteket CSAK akkor kell módosítanai, ha más lapra ugrunk
- Lap bitek definiálják az aktuális lapot
- Utasítások, amelyeknél a lapozó bitekre tekintettel kell lenni:
  - GOTO <*address (0...8 bit)* >
  - CALL <*address (PC8=0)*>
  - <Utasítás> PCL,F ; e.g. ADDWF PCL,F (PC8=0)
- Verembe csak PC alsó 8 bitje kerül

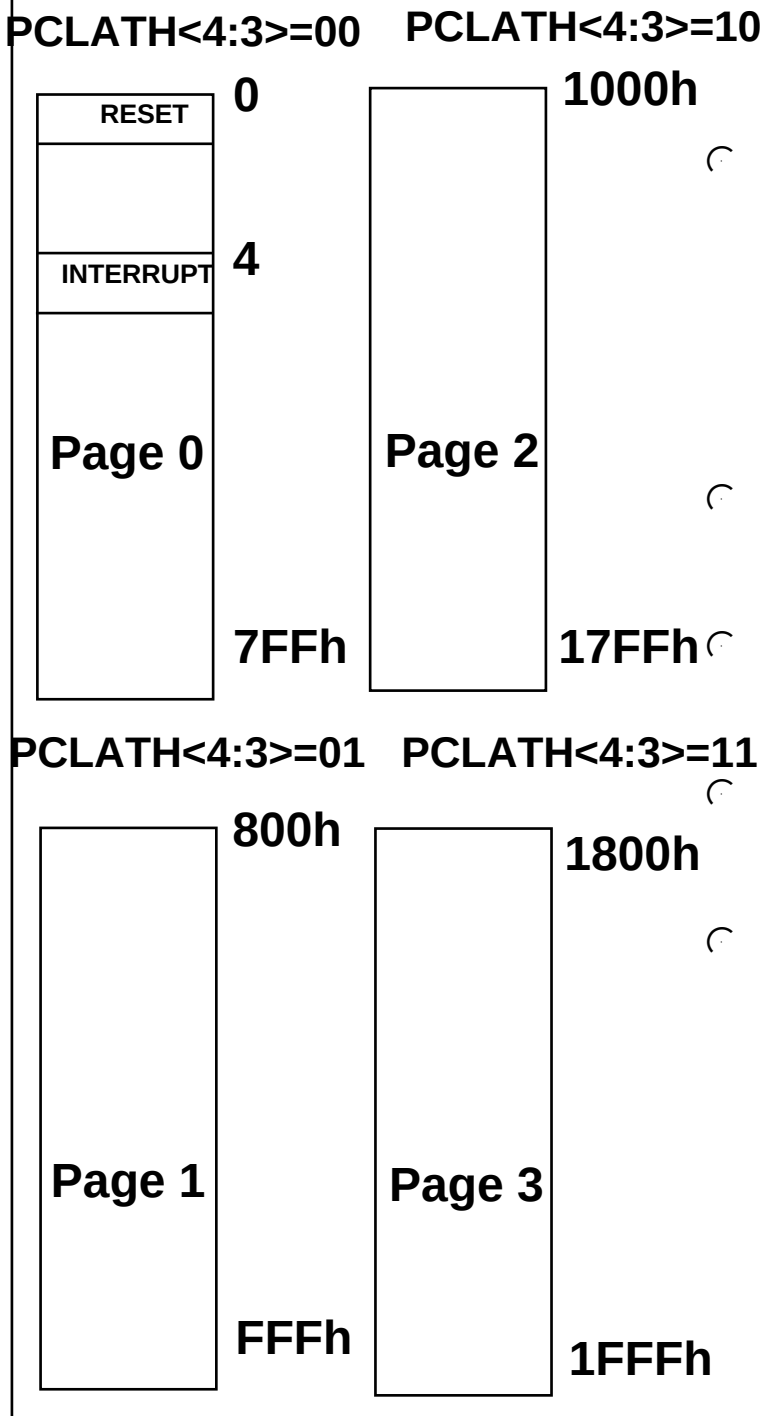
Programozás

ADDWF PCL címzés

- **ADDWF PCL F *nyolc* bites (256 szavas) címtartományt fog át, és mivel PC8=0, ezért csak a program memória lapok első 256 szavas részére hivatkozhatunk**
- **PIC16C5XX-nél PA0, PA1 határozza meg az aktuális lapot, de CALL és számított PC esetén:**
  - **PC 8- bitje törlődik**
  - **A célcím a lap első 256 címe lehet**
- **PIC16CXXX/17CXXX-nél, PCLATH használható a felső címekre**
  - **Célcím akárhol lehet**
  - **Olyan tábláknál amelyek átlépik a 256 szavas laphatárt PCLATH-ot állítani kell**

Programozás

PIC16CXXX Program-memória térkép

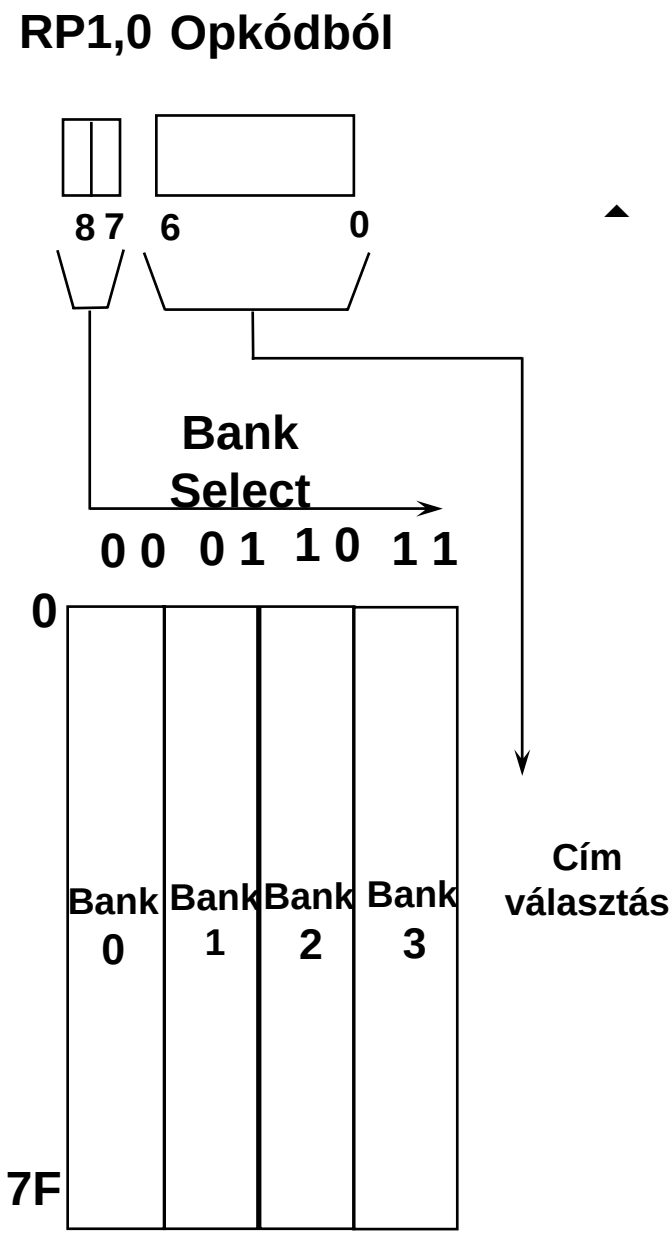


- Maximum 8K szó (13 bit) program memória. Jelenlegi típusok kapacítása: 4K
- 4 darab 2K szavas lap (11 bit)
- Lap váltás: PCLATH<4:3>
- Reset vektor: 0000h
- Megszakítás vektor: 0004h

Példaprogram: ADVPAGE.ASM

Programozás

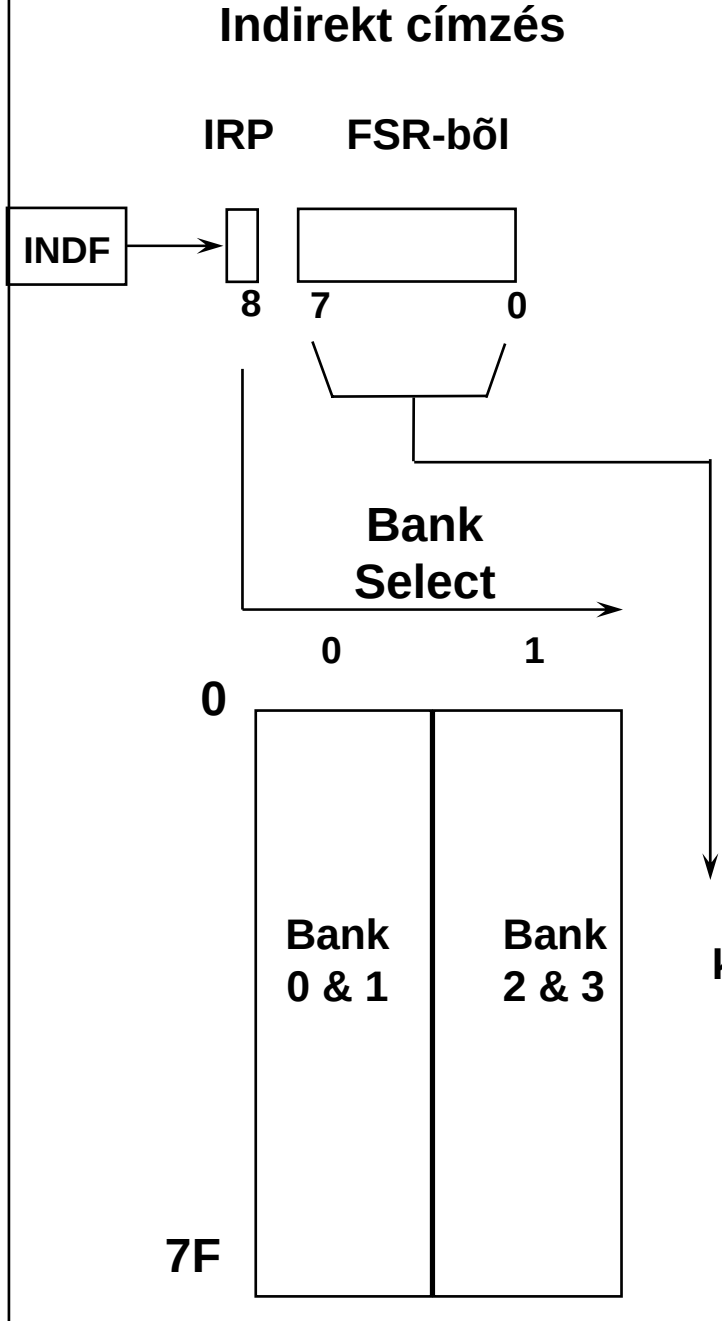
Adatmemória lapjai: a bankok



- 4 darab 128 bájtos adatmemória bank
- Speciális Funkcióju Regiszterek (SFR) ebben a RAM-ban vannak
- Bankok kiválasztása a Státusz Regiszter RP0 és RP1 bitjeivel történik

Programozás

Közvetett memória bank kezelés



- Az indirekt címzendő bankot a Status regiszter IRP bitje határozza meg
- Az INDF regiszter-rel végzett műveleteket az FSR regiszter által mutatott adatokkal hajtjuk végre

Példaprogram:ADVBANK.ASM

Programozás

RAM kiosztás (Allokáció)

- A közvetlenül címzett regisztereket a 0-ás Bankban célszerű tartani
- Adatstruktúrákat, szoftver vermet, és más közvetetten (FSR-en) keresztül címzett regisztereket az 1-es Bank-ba célszerű tárolni
- Más további bankváltás nem szükséges (kivéve a perifériák inicializálását)

```
; Bank 0 Direkt regiszterek
CBLOCK 0x20
        COUNTLO
        COUNTHI
        SWSTACKPTR
        ...
ENDC
```

```
; Bank 1 Indirekt regiszterek
SWSTACKSTART   EQU   0xA0
```

Példa szoftver veremre

- Bejövő, vagy kimenő adatok tárolására használható
- Létrehozható egy szoftver “stack” adat struktúra

```
initswstack
    MOVLW     SWSTACKSTART ; SW verem kezdőcím W-be
    MOVWF     FSR          ; FSR regiszter inicializálása
.......
pushswstack
    MOVWF     INDF         ; adat az SW verembe (Push)
    INCF      FSR ,F       ; FIFO mutató növelése
    RETURN                 ; PUSH vége
.......
popswstack
    DECF      FSR,F        ; SW verem mutató csökkent.
    MOVF      INDF,W       ; Adat W-be
    RETURN                 ; POP vége
```

Programozás

PIC16CXXX Tábla kezelés

Táblák az ADDWF PCL and RETLW utasításokkal kezelhetők

```
        ORG     0x10            ; Page 0
        MOVF    offset,W        ; w = ofszet (eltolás)
        CALL    Table
        . . .
        ORG     0x20            ; Page 0
Table
        ADDWF   PCL,F           ; hely=Tábla kcim + ofszet
        DT “ABCD”               ; RETLW sorozat def.
TableEnd                ; Page 0
```

Ugrótáblák

- Különféle bemenő feltételtől függő rutinra való ugrásra használható
- Hasonló az előbbi táblához GOTO utasítás van a CALL és RETLW helyett
- Használatával a veremszintek száma csökkenthető

```
        MOVF            NEXT_STATE,W
        GOTO            jump_table
        ...
jump_table              ; PC felső felét inicializáni kell
        ADDWF   PCL, F
        GOTO            state0_routine
        GOTO            state1_routine
        GOTO            state2_routine
```

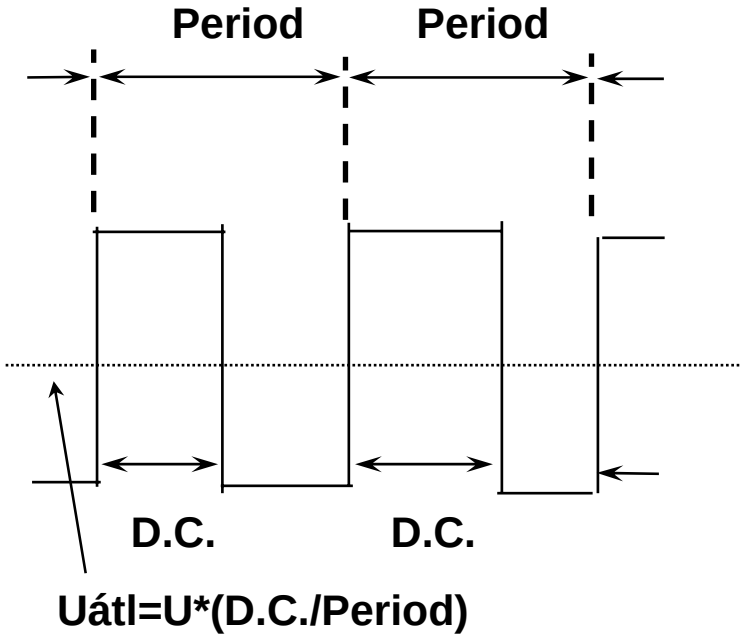
Szabályok:

- PCL-t az ADDWF PCL utasítással módosítjuk
  - ADDWF PCL eredménye nem lépheti át a 8-bites címhatárt (lapok miatt!)
  - PIC16C5XX - PA0 és PA1 inicializálandó
  - PIC16CXXX - PCLATH inicializálandó
  - PIC17CXXX - PCLATH inicializálható a MOVFP PCL,PCL utasítással

Példa: ADVJUMP.ASM

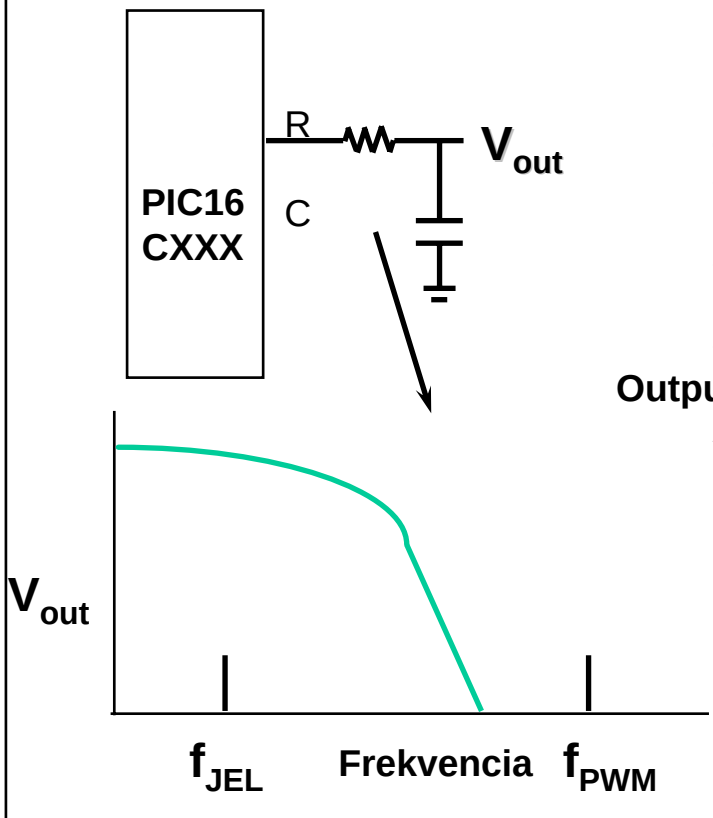
Programozás

CCP PWM mód



- A periódus végén a kimenet magas lesz
- A kitöltési ciklus (Duty Cycle (D.C.)) végén a kimenet alacsony szíintű lesz
- A periódus végén egy új kitöltési tényező tölthető

Pédaprogram: ADVPWM.ASM



PWM mint D/A átalakító

```
        org         0x10
        CLRF        PCLATH
        MOVF        Vout,W
        CALL        OutputVoltage
        MOVWF       CCPR1L
        GOTO        Continue
OutputVoltage
        ADDWF       PCL,F
        RETLW       0x00  ;  Vss
        RETLW       0x3F  ; (Vdd*255) / 64
        RETLW       0x7F  ; (Vdd*255) / 127
        RETLW       0xBF ; (Vdd*255) / 191
        RETLW       0xFF ;  Vdd
```

Programozás

Paraméter átadás

- Szubrutinoknál szükséges a változók értékeinek az átadása
  - Matematikai rutinoknál, pl. szorzás és osztás
  - Soros adó és vevő rutinoknál pl. PUTCHAR vagy GETCHAR
- Globális változók
  - Közvetlen címzésű regiszterek használata
  - A legjobb néhány paraméter esetén
- W regiszter
  - Egy paraméter átadásához a W-t használjuk
  - A paramétert W-be tesszük
  - Meghívjuk a szubrutint. PÉLDA:

```
MOVLW   'M'         ; ASCII M karakter W-be
CALL    putchar     ; Karaktert a kijelzőre
```

- FSR - összetett adatstruktúrákhoz
  - Nagyszámú paraméter esetén
  - Csak a kezdőcímet kell átadni
  - FSR használható az első adatelemre mutatónak. Példa:

```
CLK_MIN_LD      EQU  0x0C      ALM_MIN_LD      EQU  0x10
CLK_MIN_HD      EQU  0x0D      ALM_MIN_HD      EQU  0x11
CLK_HOUR_LD     EQU  0x0E      ALM_HOUR_LD     EQU  0x12
CLK_HOUR_HD     EQU  0x0F      ALM_HOUR_HD     EQU  0x13

MOVLW   CLK_MIN_LD
MOVWF   FSR
CALL    inc_time
MOVLW   ALM_MIN_LD
MOVWF   FSR
CALL    inc_time
```

Programozás

PIC16CXXX megszakítások

- Több belső és külső megszakítás forrás
- A megszakításoknak csak egy vektorcíme van (04h)
- A prioritás és azonosítás programból lehetséges
- Globális és egyedi megszakítás engedélyezés
- Több megszakítás a processzort a szundi (sleep módból ébreszti
- Megszakítás felismerés 3 utasításciklus, illetve 4 ciklus külső megszakításoknál
- A megszakítás kiszolgálásakor fontos megőrizni a STATUS a W regiszter (valamint PCLATH regiszter ha 4K-s vagy nagyobb memóriájú tokot használunk) tartalmát.

Rutinok a ‘push’ (mentés) és ‘pop’(visszatöltésre):

```
tempW       equ   0x20      ;W mentése bank 0 & 1-be
tempStatus  equ   0x21      ;STATUS mentése bank 0-ba
tempPclath  equ   0x22      ;PCLATH mentése bank 0-ba (16c74)
            ORG 0x04
push        movwf   tempW
            swapf   STATUS,W
            bcf     STATUS,RP0
            movwf   tempStatus
            movf    PCLATH,W
            movwf   tempPclath
            bcf     PCLATH,3
```

```
pop         movf    tempPclath,W
            movwf   PCLATH
            bcf     STATUS,RP0
            swapf   tempStatus,W
            movwf   STATUS
            swapf   tempW,F
            swapf   tempW,W
            retfie
```

Programozás
Megszakítások összhasonlítása

PIC16CXXX @ 12MHz

- 1 vagy 2 ciklusú utasítások
- ut.cikl = órajel / 4

```
                                ; szó / cikl
Int_Srv                         ; 0 / 3
    MOVWF     tempW             ; 1 / 1
    SWAPF     STATUS,W          ; 1 / 1
    BCF       STATUS,RP0        ; 1 / 1
    MOVWF     tempStatus        ; 1 / 1
              . . .
```

- 4 szó a programmemóriában
- Int késl = 3 cikl = 1.0 µs (latency is same for 1 or 2 cycle inst)
- Int adminisztr. = 4 cikl = 1.3 µs
- Teljes idő = 7 cikl = 2.3 µs

MCS-8X51 @ 12 MHz

1...4 ciklusú utasítások

- ut.cikl = órajel / 12

```
                                ; bájt / cikl
Int_Srv                         ; 0 / 3 to 9
      PUSH    PSW               ; 2 / 2
      PUSH    ACC               ; 2 / 2
      MOV     PSW,#08h          ; 3 / 2
      . . .
```

- 7 bájt a programmemóriában
- Int késl (min) = 3 cikl = 3 µs
- Int késl (max) = 9 cikl = 9 µs
- Int adminisztr = 6 cikl = 6 µs
- Teljes idõ = 9 - 15 cikl.=9 - 15µs

Többszörös megszakítások kezelése

- Az egyes megszakításforrások engedélyezése/tiltása az INTCON, PIE1 és PIE2 regiszterek segítségével lehetséges
- A megszakítások globális engedélyezése/tiltása GIE bittel lehetséges
- A megszakítás kiszolgálásának kezdetekor a hardver törli a GIE bitet és a visszatérési címet a verembe rakja
- A megszakítási esemény az INTCON, PIR1 és PIR2 regiszterek bitjeit állítja be
- Ezeket programból kell törölni!
- RETFIE utasítás GIE-t 1 be állítja és verem tetején lévő címet a PC-be tölti

Programozás
Taszk kezelés

- Szubrutinok adott időközönkénti futtatására használható
- Gyakran a TMR0 timert használják ütemadóként
- Vagy a lekérdezéses (polling) (PIC16C5XX) vagy megszakításos (PIC16CXXX / PIC17CXXX) technikát lehet használni

Taszk kezelés lekérdezéssel

- PIC16C5XX TMR0 regiszterének olvasása
- TMR0 aktuális és az előző értékének az összehasonlítása
- Annak meghatározása, hogy melyik bit változott
- A változott bit tesztelése alapján a megfelelő taszk (szubrutin) hívása

```
MOVF    TMR0,W          ; TMR0 aktuális értéke W-be
XORWF   PrevTMR0,W      ; Nézzük melyik bit változott...
MOVWF   ChaTMR0         ; Minden megváltozott bit ChaTMR0-
        ba
XORWF   PrevTMR0,F      ; Az aktuális lesz az előző érték...
```

Példaprogram: ADVPOLL.ASM

Programozás

Taszk kezelés megszakítással

- TMR0 felhasználása periodikus megszakításra
- A következő taszk számontartása
- Az időszeletelés felhasználható a soros kommunikációra, stb.
- Két megvalósítási lehetőség:
  - Taszk mutató növelése a megszakítási programban (ISR-Interrupt Service Routine)
    - A taszk végrehajtása az ISR-ben

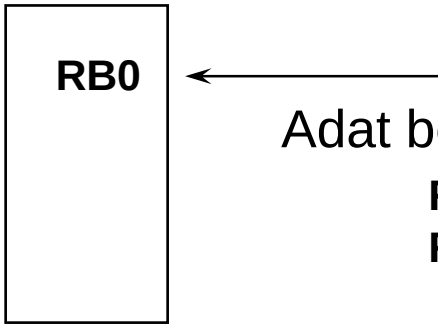
  VAGY:
  - Jelzőbitek állítása a taszkmutató alapján
  - Egy fő programbeli hurokban figyeljük a jelzőbiteket és futtatjuk a taszkot ha kell
- Ha a timer megszakít, a taszkok az ISR-ben futnak
- Pontos és precíz időzítést biztosít a valós idejű taszkok számára
- A megszakítási rutin hosszú lesz, ezért rövid taszkok esetén előnyös a használata
- Példa:
  - TMR0 megszakít 4 msec-ént
  - Billentyűzet figyelése 52 mse.-ként
  - Forgatási taszk 1 mp-ként

Példaprogram: ADVTASKI.ASM

Programozás

Adatbevitel az I/O forgatással

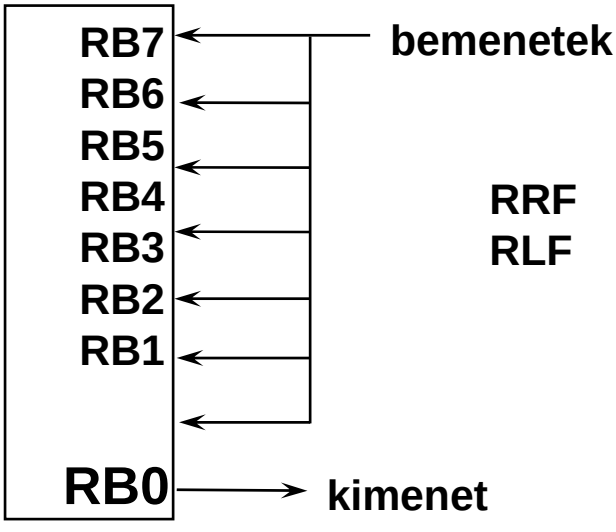
- Az adatbemenet az I/O portok LSB (bit 0) vagy az MSB (bit 7) kivezetése
- RRF utasítás (LSB) vagy RLF utasítás (MSB) a bemenet értékét a carry-be tolja
- Ugyanezen RRF vagy RLF utasítások a biteket egy fájl regiszterbe tolják




```
RRF     PORTB,W          ; RB0 bit Carrybe
RRF     DReg,F   ; Adatbit mentése DReg-be
```

Adatkivitel forgatással

- Adatkimenet az I/O port LSB (bit 0) vagy MSB (bit 7) bitje
- A port többi bitje csak bemenet lehet !!!
- RRF vagy RLF -el a kimeneti bitet carry-be
- RRF (MSB) vagy RLF (LSB) utasítással a carry-ből a kimenetre mozgatjuk a bitet




```
RRF     DReg,F   ; Kimeneti bit Carry-be
RLF     PORTB,F  ; Onnan a kimenetre
```

Programozás

Soros adatátvitel - UART-al

- A hatékony bitkezelő és forgató utasításokkal egyszerűn valósítható meg a szoftver UART
- Fél duplex rutinok a biteket időzítve tolják
- Az ADÓ és VEVŐ rutin összesen csak 50 szó hosszúságú!
- Az adattátviteli sebesség max. 115Kbit/sec lehet

Példaprogram: ADVUART.ASM

Soros adatátvitel - SPI-vel

- Számos perifériaáramkör használja a 4 vezetékes SPI szinkron soros protokollt
- A hatékony bitkezelő és forgató utasításokkal egyszerűn valósítható meg a szoftver SPI
- A teljes duplex rutinban a kimeneti bitek egyszerű kitolásával egyszerre történik a bemeneti bitek betolása
- Az ADÓ és VEVŐ rutin összesen csak 17 szó hosszúságú!
- Az adattátviteli sebesség max. 373Kbit/sec lehet

Példaprogram: ADVSPI.ASM

Programozás
Utasításszám csökkentése I.

- Egy utasítás megtakarítható két egymásutáni NOP esetén
- GOTO “következő utasítás”-t használjuk

NOP	GOTO $+1
NOP	
2 utasítás, 2 ciklus	1 utasítás, 2 ciklus

- A célbit meghatározza, hogy az eredmény a W vagy az F(ájl) regiszterbe kerüljön
- Célszerű az adatmozgatást átgondolni

Példa: A + B -> A

MOVF	A,W		MOVF	B,W
ADDWF	B,W		ADDWF	A,F
MOVWF	A			
3 utasítás			2 utasítás	

- Egy bit átvitel két regiszter között (REGA-ból REGB-be)
- REGB bit előzetes beállítása
- REGA bit-je tesztelése alapján a bitet esetleg megváltoztatjuk
- Portok esetén, két ciklus hosszúságú váltás (glitch) lehetséges!!!

BTFSS	REGA,2		BCF	REGB,5
BCF	REGB,5		BTFSC	REGA,2
BTFSC	REGA,2		BSF	REGB,5
BSF	REGB,5			
4 utasítás			3 utasítás	

Programozás

Utasításszám csökkentése II.

- A kimeneti bit állítása egyforma ideig tartson (szinkronizálás), ha ez fontos
- LSB vagy MSB esetén a ROTATE utasítás használható

```
        BTFSC   OutBit,0        ; Átlépés ha kimenet nulla
        GOTO    kim1            ; Ugrás 1-be állításra
        NOP                     ; Egy cikl. késl. a szink. miatt
        BCF     PORTA,1         ; Kimeneti bit törlése
        GOTO    kimkesz         ; végére ugrás
kim1
        BSF     PORTA,1         ; kimeneti bit 1
        GOTO    kimkesz         ; Végrehajtási idõ szink. miatt
kimkesz
```

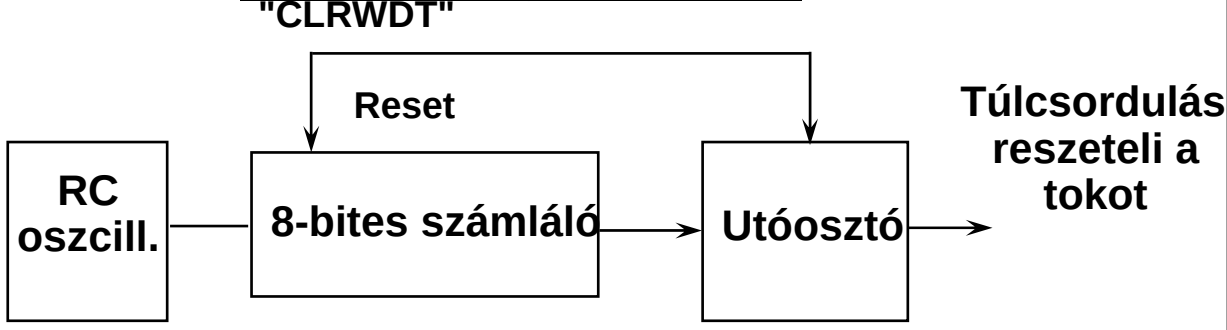
- Regiszterek hasonlítása
- A két folytatás egy GOTO-val

```
        MOVF    REGB,W    ;W tartalma REGB-be
        XORWF   REGA,W    ; REGA & REGB hasonlítása
        BTFSS   STATUS,Z  ; Átlép ha REGA=REGB
        GOTO    NotEqual  ; Ugrik ha nem egyenlõ
Equal
.
.
NotEqual
.
.
```

További ötletek a programok tanulmányozása alapján!

„Bombabiztos” tervezés

Watchdog hardver



- Programeltérülés esetén segíthet
- Saját belső RC oszcillátort használ
- WDT programból nem tíltható!
- WDT túlcsordulása reszeteli a tokot
- CLRWDT utasítás törli WDT-t
- Programozható time-out periódus: 18 ms-tól 2.5 másodpercig
- SLEEP üzemmódban is működik
- Túlcsorduláskor a CPU „felébred”

Watchdog szoftver

- A watchdog hatékonysága függ a felhasználói program jól megírtságától
- Az egész programban csak egy CLRWDT utasítást használjunk!
- A CLRWDT utasítást a főhurokba tegyük
- Ne tegyük a CLRWDT-t ISR(megszakítási) vagy egyéb szubrutinba
- Olyan minimális WDT túlcsordulási időt válasszunk, amit a fő hurok végrehajtási ideje megenged
- A nem használt memórát töltsük fel a GOTO wdtreset utasítás kódjával

„Bombabiztos” tervezés

Nem használt memória feltöltése

- A nem használt memórát töltsük fel a GOTO wdtreset utasítás kódjával, ami normál esetben soha nem kerül végrehajtásra
- Ha a PC értéke elromlik, akkor valószínűleg ez az utasítás végrehajtódik, és újra indítja a kontrollert
- FILL direktíva használható a feltöltésre:

```
FILL    (GOTO wdtreset), (400h-$)
ORG 3FFh
wdtreset
```

WDT reset törlése

- Bekapcsoláskor a WDT végrehajtatása:
  - a RAM memória minta ellenőrzése tápfesz. bekapcsoláskor
  - Ha a minta nincs a RAM-ban:
    - RAM minta feltöltése
    - WDT reset végrehajtása

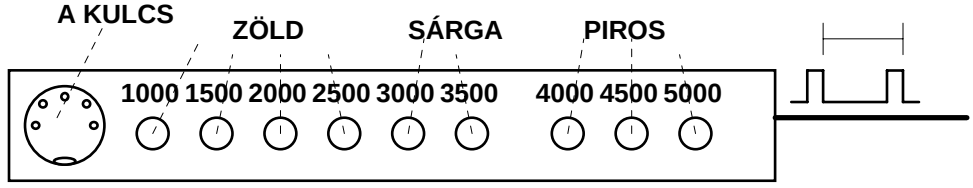
Szubrutin számlálás

- Két számláló: szubrutin hívó és végrehajtó számláló
- Hívó számláló inkrementálása minden rutinhíváskor
- Végrehajtó számláló inkrementálása minden szubrutin kezdetén
- A fő hurok elején a két számláló egyezőségének a vizsgálata, ha nem egyformák, akkor WDT reset

Példaprogram: ADVRELSW.ASM

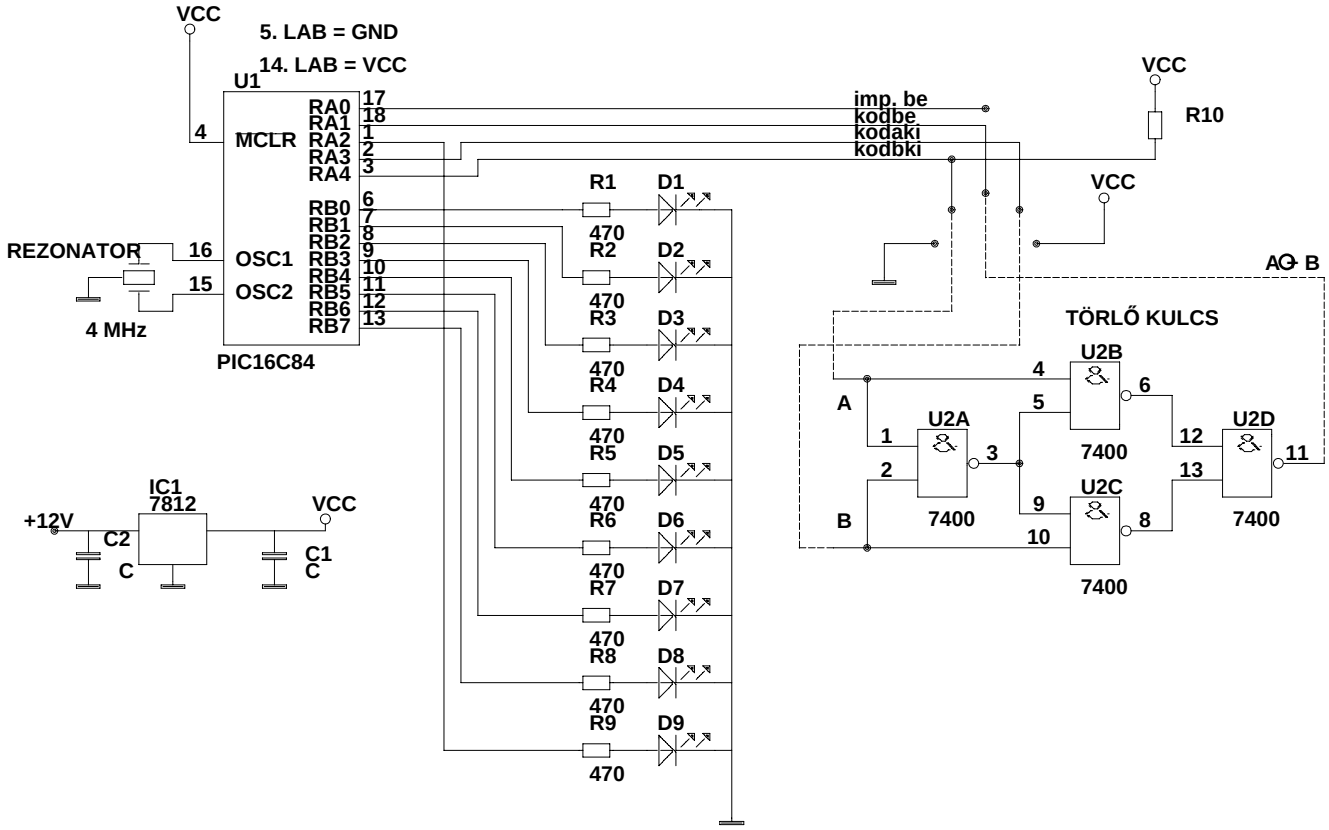
EGY „ÉLES” FELADAT - DEFINIÁLÁS

DIESEL AUTÓKNÁL A MOTOR TÚLPÖRGETÉSE KÁROS A MOTORRA. A FELADAT EGY OLYAN FORDULATSZÁM MÉRŐ KÉSZÍTÉSE, AMELY 1000-5000 FORD/PERC KÖZÖTT 500-AS LÉPÉSENKÉNT EGY LED KIGYUJTÁSÁVAL JELZI AZ AKTUÁLIS FORDULATSZÁMOT. 4000-ES FORDULAT TÚLLÉPÉSE ESETÉN A MEGFELELŐ LEDEKET BEKAPCSOLVA ÉGVE HAGYJA, JELEZVE A TÚLLÉPÉST. AZ ÍGY KIGYÚJTOTT LEDEK TÖRLÉSE CSAK EGY SPECIÁLIS ELEKTRONIKUS KULCCSAL LEHETSÉGES. (Tápfesz. kikapcsolása után is megőrzi az állapotát!)



KIMENETEK SZÁMA: 9 LED + 2 KULCS
BEMENETEK SZÁMA: 1 FORD.SZÁM IMPULZUS, 1 KULCS BEM.
EEPROM KELL A TÚLLÉPÉS TÁROLÁSA MIATT

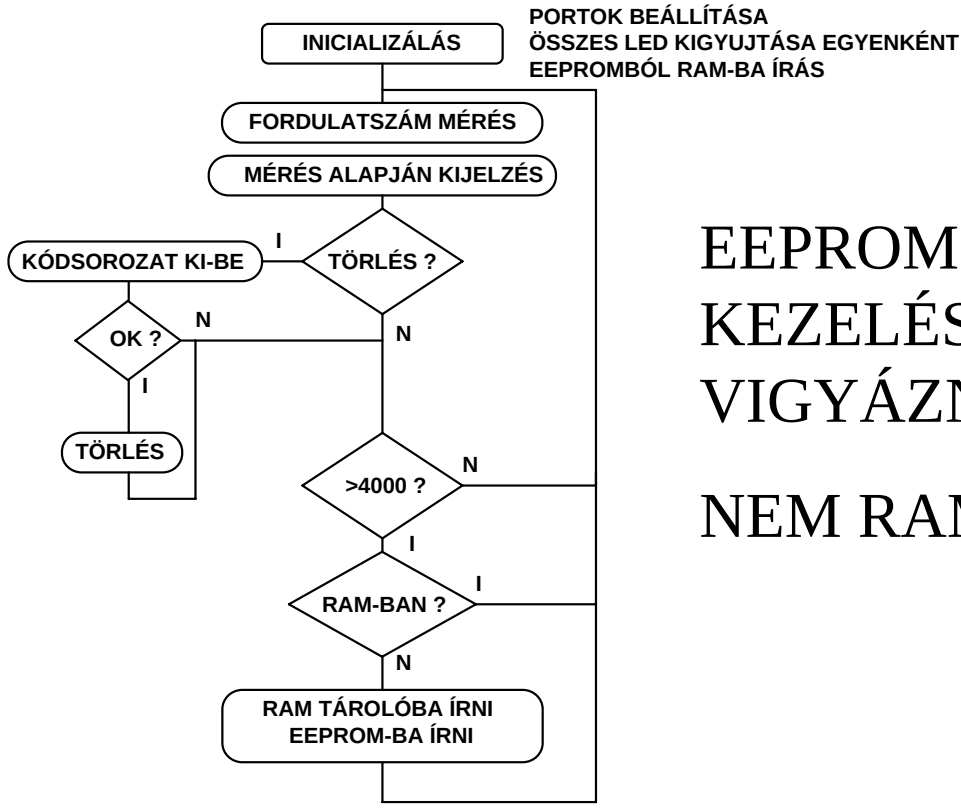
13 I/0 + EEPROM => PIC16C84



EGY „ÉLES” FELADAT - TERVEZÉS

MILYEN RÉSZFELADATOKAT KELL MEGOLDANI:

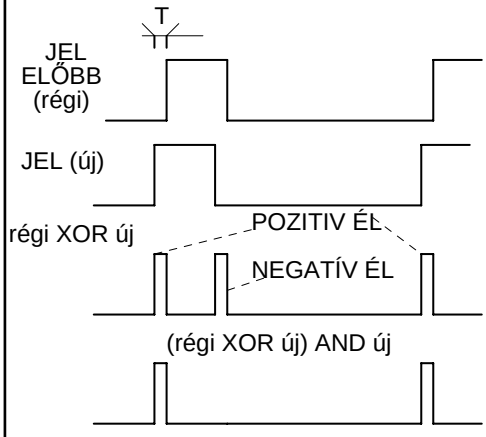
- *ÜZEMKÉPESSÉG ELLENŐRZÉSE*
- *FORDULATSZÁM MÉRÉS*
- *MÉRT ADATOK ALAPJÁN LED KIJELZÉS*
- *TÚLLÉPÉS TÁROLÁSA ÉS KIJELZÉSE*
- *TÚLLÉPÉS TÖRLÉSE KULCCSAL*



EEPROM KEZELÉSRE VIGYÁZNI !!!

NEM RAM !!!

ÉLFIGYELÉS




```
figyel   mov    w,ra
         and    w,#01     ;ra.0.bitje a JEL
         mov    uj,w      ;első minta
elvar
         mov    regi,uj   ;uj := regi
         nop
         nop
         mov    w,ra
         and    w,#01     ;0.bit a fordszám
         mov    uj,w
         xor    w,regi
         jz     elvar     ;regi=uj, nincs váltás
valtas
         and    w,uj      ;élváltás jó?
         jz     elvar     ;nem
                          ;ÉL FELDOLGOZÁSA
```

EGY „ÉLES” FELADAT

FORDULATONKÉNT EGY IMPULZUS

fordulatszám (ford/perc)	periódusidő *4 (msec)		fszam sec-ként	3sec alatt
500	120	480	8.3	25
1000	60	240	16.6	50
1500	40	160	24.9	75
2000	30	120	33.3	100
2500	24	92	41.7	125
3000	20	80	50	150
3500	17.1	69	58.3	175
4000	15	60	66.7	200
4500	13.3	53	75	225
5000	12	48	83.3	250

HA PERÓDUSIDŐT MÉRÜNK, AKKOR:
A mérést érdemes 0.25 msec-os megszakítással végezni (=4kHz)
Nem lineáris! (1/x függvény)
HA IDŐEGYSÉG ALATT BEJÖVŐ IMPULZUSSZÁMOT MÉRÜNK, AKKOR:
jobb a 3 mp alatti fordulatszámot mérni, vagy 3 impulzus/fordulat
ez lineáris !!!

```
INT_SERVICE     CALL SISWS          ;"PUSH" W AND STATUS REGISTERS
RTCCPOL         BTFSS INTCON,2      ;RTCC INTERRUPT?
                GOTO EX67           ;IF NOT, (FALSE INTERRUPT?) EXIT
                MOVLW  218          ;INNEN INDUL A SZÁMLÁLÓ
                MOVWF  RTCC
                BCF INTCON,2        ;CLEAR RTCC INTERRUPT FLAG
                setb    kodaki      ;hogy lássuk az it-t
                clrb    kodaki
                setb    kodaki
RTCCSER
                INCF    INT_COUNT,1     ;10 Hz INT RATE
                MOVLW 100
                SUBWF   INT_COUNT,0
                BTFSS   STATUS,2        ;ZERO_FLAG  1 SEC.
ELTELT?
                GOTO    EX67            ;IF NOT, EXIT
                CLRF    INT_COUNT       ;IF YES, CLEAR INT_COUNT
                dec     sec
EX67
                CALL SIRWS          ;"POP" W AND STATUS REGISTERS
                RETFIE              ;ENABLE GLOBAL INT AND RETURN
```