PIC TANFOLYAM

Dr. Kónya László:

http://alpha1.obuda.kando.hu/~konya

konya@novserv.obuda.kando.hu

PIC MIKROKONTROLLEREK ALKALMAZÁSTECHNIKÁJA

Tanfolyami tananyag

2000

2000.09.02

1. BEVEZETÉS

Jelenleg az informatika forradalma zajlik.

Két sokkoló dátum:

• 1971 - az első 4 bites mikroprocesszor INTEL 4004:

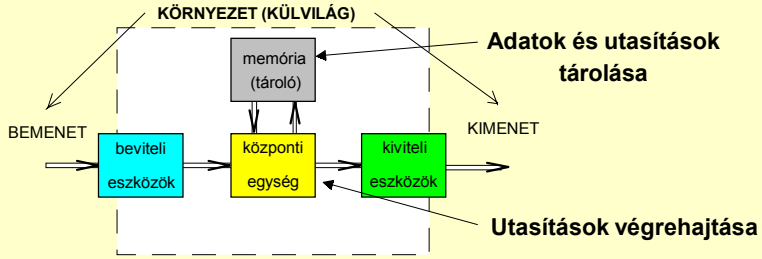
csak 28 év !!!

• 1981 - az IBM PC megjelenése:

csak 18 év !!!

• A számítógépek részt vesznek az újabb számítógép-generációk előállításában - evolúció ?!

1.1 AZ INFORMÁCIÓFELDOLGOZÁS ÁLTALÁNOS MODELLJE.



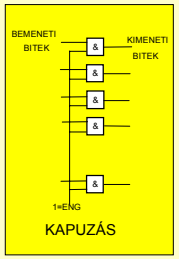
Algoritmus: hogyan hozzuk létre a kimeneti információit a bemeneti információból.

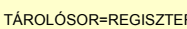
Program: az algoritmus számítógépen történő megvalósítása, utasítások sorozata

A program határozza meg hogy mit végezzen el, és ez cserélhető!!!

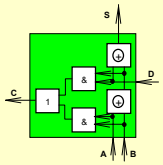
Megvalósítás: digitális áramkörökkel: MIÉRT két állapottal?

1.2 A MIKROSZÁMÍTÓGÉPEK ALAPÁRAMKÖREI



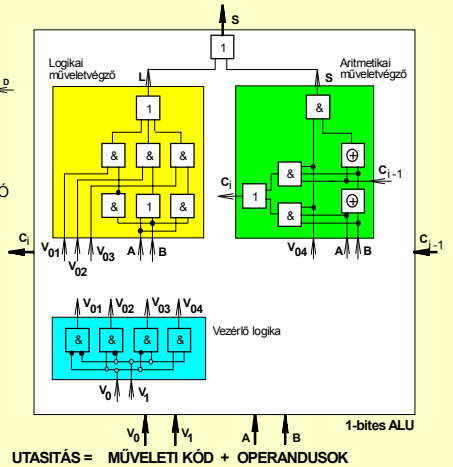


TÁROLÓSOR=REGISZTER



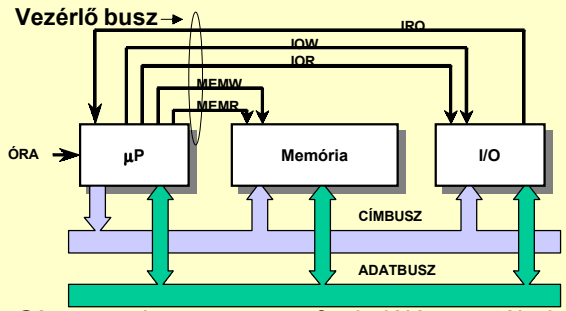
TELJES ÖSSZEADÓ
S=A+B+D

A	B	D	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



UTASITÁS = MŰVELETI KÓD + OPERANDUSOK

1.3 A MIKROSZÁMÍTÓGÉPEK ÁLTALÁNOS FELÉPÍTÉSE



Mikroszámítógép:
IC-k-ből áll
fejlődés:
- kártyamodulok
- egy kártyán:(SBC)
- egy tokban:
mikrokontroller
= (mikrovezérlő)

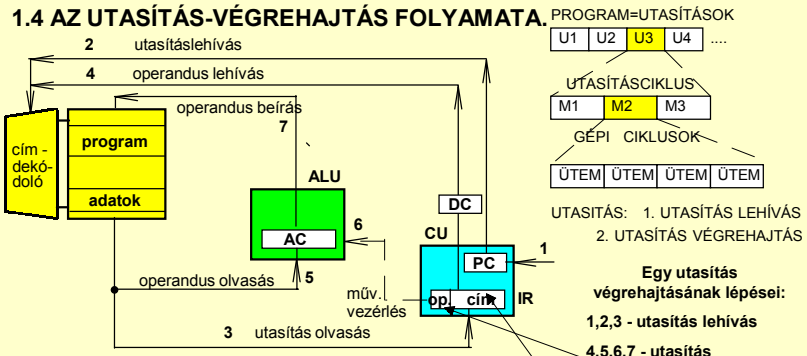
Sín, vagy busz: azonos funkciójú vezetékek összessége (cím, adat, és vezérlő busz

a processzor szemszögéből a memória és I/O megkülönböztetés nem indokolt

az I/O regiszter olyan memória amelyeknek másik oldala a külvilágra „nyílik"

A mikrogépek teljesítményének meghatározására elterjedt egy nem teljesen objektív, de elfogadott jellemző, a MIPS. Ez azt mutatja meg, hogy hány utasítást képes az adott eszköz egy másodperc alatt végrehajtani.

1.4 AZ UTASÍTÁS-VÉGREHAJTÁS FOLYAMATA.



Egy utasítás végrehajtásának lépései:

1,2,3 - utasítás lehívás

4,5,6,7 - utasítás végrehajtás

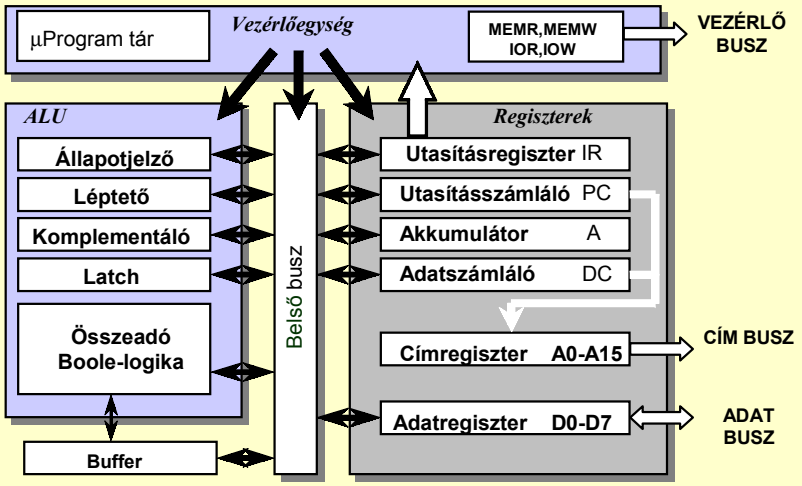
utasítás = művelet (mit?) + operandus(ok) (mivel?)

ALU - Aritmetikai- logikai egység
CU - Vezérlő egység

Funkcionális egységek: PC-program counter, DC - data counter, IR-utasításregiszter, AC - akkumulátor

Neumann elv: közös program és adatmemória

1.5 CISC PROCESSZOROK - MIKROPROGRAMOZÁS



PC-program counter, DC - data counter, IR-utasításregiszter, AC - akkumulátor

1.6 RISC PROCESSZOROK.

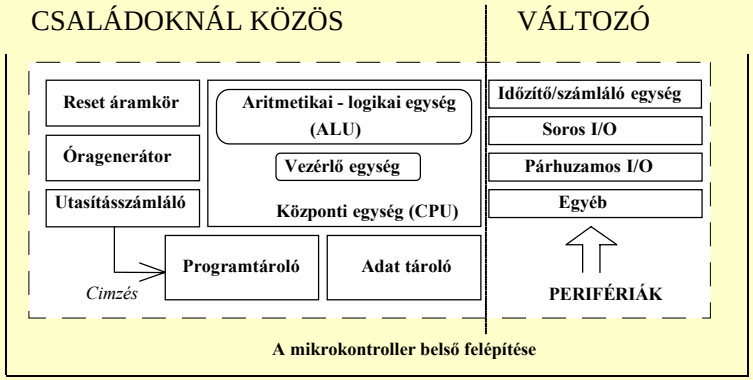
Ha egyszerűbb a felépítés, egyszerűbb, de gyorsabb a hardver is.

RISC processzorok főbb jellemzői:

- **Kevésbé bonyolult utasítások; a felhasználói terület elemzése alapján a bonyolult utasítások elhagyása. Az utasítások végrehajtásához egy gépi ciklust használnak fel.**
- **Kevés utasítás (<128) és címzési mód(2-4) használata.**
- **Az utasítások végrehajtásához nincs mikroprogram; az igen bonyolult fordítóprogram állítja el a végs formát.**
- **Az utasítások rögzített hosszúságúak, hosszuk nem változik**

1.7 MIKROKONTROLLEREK FELÉPÍTÉSE

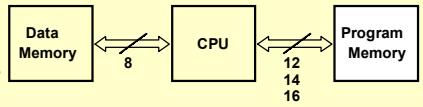
Mikrokontroller vagy mikrovezérlő: mikroszámítógép egy tokban



A PERIFÉRIÁK HASZNÁLATA CSÖKKENTI A CPU TERHELÉSÉT (a perifériakezelés hardveres megoldása miatt)

2.0 PIC ARCHITEKTÚRA: RISC-TULAJDONSÁGÚ

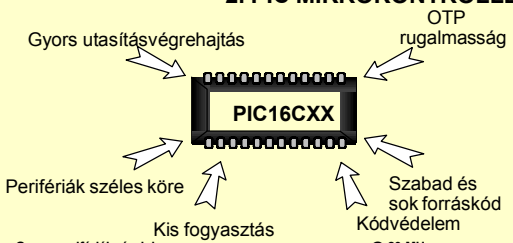
A nagy teljesítmény okai:



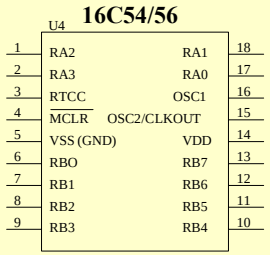
- Harvard architektúra
- Eltérő szélességű adat és programbusz
- Minden utasítás egyszavas
- Regiszter fájl szervezés
- LWI (Long Word Instruction)
- Utasítás csővonal (pipelining)
- Egyciklusos utasítások
- Csökkentett utasításkészlet
- Ortogonális utasításkészlet
- Nagyobb áteresztő-képesség

Harvard felépítés két külön tárolót: egy adat és egy programtárolót használ, megszűnik a sorban állás a memóriáért

2. PIC MIKROKONTROLLEREK



Gyors perifériák, órajel		@ 20 MHz
Utasítás-ciklusidő		200 ns
I/O váltás üteme		200 ns
Komparálási idő		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
	Írási szám	1 M E/W (Typical)

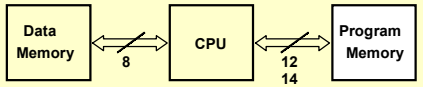


2.1 UTASÍTÁS = MŰVELETI KÓD + OPERANDUS

AZ UTASITÁSHOSSZ A DÖNTŐ !!!

INTEL MEGOLDÁS: A MÜVELETI KÓD + OPERANDUS MINDIG EGÉSZ BÁJT HOSSZÚSÁGÚ, MERT A PROGRAM ÉS ADATMEMÓRIA KÖZÖS

PIC: KÜLÖN PROGRAM- ÉS KÜLÖN ADATMEMÓRIA



A NAGYOBB BITSZÁMÚ UTASÍTÁSHOSSZ MIATT EGY SZÓBAN ELFÉR MIND A MŰVELETI KÓD, MIND AZ OPERANDUS (CÍME) !!!

FONTOS !!!

FELOSZTÁS A CSALÁDOK SZERINT

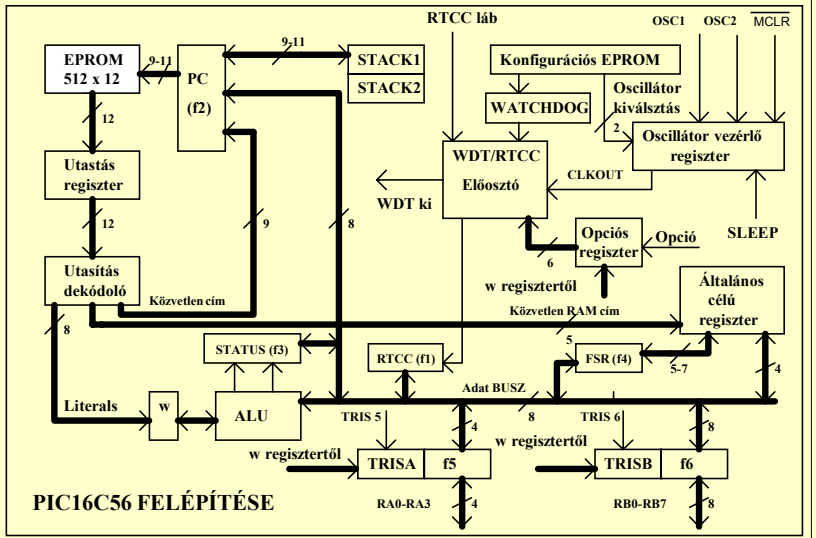
12 BITES: 7 BIT MŰVELETI KÓD 5 BIT OPERANDUS CÍM 16C5XX

14 BITES: 7 BIT MŰVELETI KÓD 7 BIT OPERANDUS CÍM 16CXX

16 BITES: 8 BIT MŰVELETI KÓD 8 BIT OPERANDUS CÍM 17CXX, 18CXX

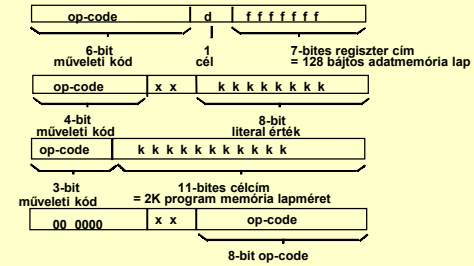
MIND AZ UTASÍTÁSBAN SZEREPLŐ MEMÓRIACÍM, MIND AZ OPERANDUSCÍM TARTOMÁNYA ERŐSEN BEKORLÁTOZOTT !!! -> EZÉRT LAPOZÁS (PROGRAMMEMÓRIA), ILLETVE BANKVÁLTÁS (ADATMEMÓRIA) ALKALMAZÁSA

2.2 PIC MIKROKONTROLLEREK FELÉPÍTÉSE



PIC16C56 FELÉPÍTÉSE

2.4 14 BITES PIC UTASÍTÁSOK FELÉPÍTÉSE



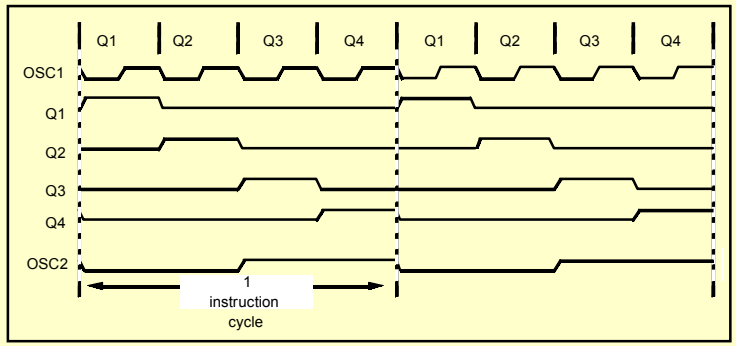
Hogyan tárolhatók a program memóriában adatok (állandók)?

- 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
```


2.3 UTASÍTÁSCIKLUS, PIPELINE

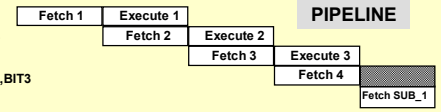
- Az utasításciklus az órafrekvencia 1/4-e
- A cikluidő 200 ns 20 MHz-es óránál, 1 mikrosec 4 MHz-nél



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

Operandussorrend:
Honnan -> hova
MOV=mozgasd
L = literal (konstans)
F = fájlregiszter
B = bit
S = set (1-be ír)

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



PIPELINE

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

2.4.1 UTASÍTÁSTÍPUSOK

LOGIKAI: AND, OR, XOR, NOT

ARITMETIKAI + SHIFT: ADD, SUB, CLR, DEC, DECSZ, INC, INCSZ,RL,RR, SWAP, TEST

BIT: CLRB, SETB, SB, SNB

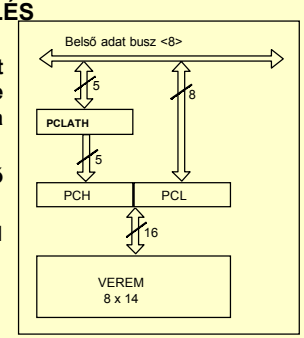
ADATMOZGATÓ: MOV

PROGRAMVEZÉRLŐ: CALL, JMP, NOP, RET, RETI, RETLW

RENDSZERVEZÉRLŐ: SLEEP

2.4.2 VEREMKEZELÉS

- Mikor egy CALL utasítást végrehajtunk, az azt követő utasítás címe a verem tetejére másolódik, és az alatta lévő tartalmak a veremben lefelé mozognak.
- RETURN végrehajtásakor verem tetején lévő cím kerül az utasításszámlálóba
- Helytelen veremkezelés (pl CALL RETURN nélkül hibás program működést eredményez



PIC16C5X-nél csak külön 2 x 11 bites hardver verem

PIC16CXX-nél külön 8 x 14 bites hardver verem

Verembe rakunk: CALL vagy megszakítás

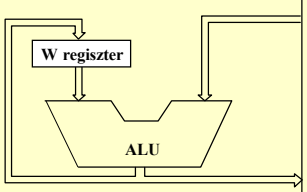
Veremből veszünk: RETURN, RETLW, RETFIE

A verem: (last in first out) – LIFO

TÚLÍRÁSKOR AZ UTOLSÓ KIPOTTYAN

2.5.1 PIC UTASÍTÁSOK VÉGREHAJTÁSA, STATUS REGISZTER

A műveletek eredménye állítja a STATUS regiszter bitjeit



Indirect Addr 00h	Indirekt címre hivatkozásnál használt
RTCC	Időzítő/számláló regiszter
PC	Utasításszámláló
Status	Program státusz regiszter
FSR	Indirekt címet tartalmaz
PortA	A port
PortB	B port
PortC	C port (ha van)
Általánosan felhasználható regiszterek (File registers) (RAM) (24-72)	

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: adatmemória bankválasztás
TO: time out (watchdog)
PD: power down (sleep)
Z: zéró bit
DC: half carry (4. biten túlcsord.)
C: carry

OPERANDUSOK: W, REGISZTER(fr), #literal (KONSTANS), bit

2.5 PIC 16CXX UTASÍTÁSKÉSZLET

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 W from Literal	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

2.5.2 ILLUSZTRÁCIÓ: PIC PROGRAMRÉSZLETEK

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 fel-használásával:

ADDLW 256-3 vagy ADDLW 253 esetleg ADDLW -3

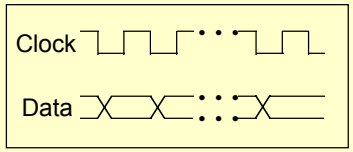
Emiatt a W regiszter tartalma kettes komplemensének képzése: SUBLW 0
utasítással történhet a szokásos XORLW 0ffh
ADDLW 1 utasítások helyett.

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
; hh  a legmagasabb helyiértékű bájt

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                    HÁZI FELADAT !!!
      ANDWF  hh,W
      XORWF  hh,F
      IORWF  mh,F
      RETURN
```

2.5.3 PÉLDA: SOROS ADATÁTVITEL

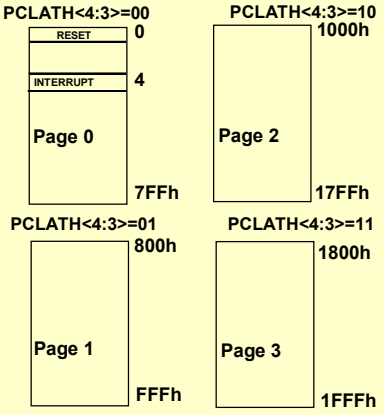


Csak 11 szó a program !!!
74 utasítás ciklus = 14.6µs @ 20 MHz
Ez megfelel 540 Kbit/sec-nek

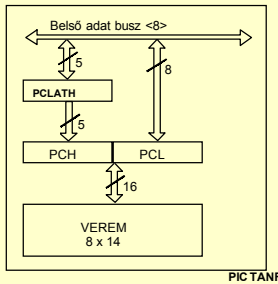
Egy egyszerű szinkron (SPI) soros adatátviteli program:

```
XMIT     MOVLW   08h            ;  8 bites átvitel
         MOVWF   bit_count      ;

XM_LOOP  BCF     PORTB, bit0    ;  0 ® Adata láb
         BCF     PORTB, bit1    ;  0 ® Clock láb
         RRF     XDATA          ;  Forgatás carry-n keresztül
                                ;  XDATA = küldendő adat
         BTFSC   STATUS, CARRY  ;  teszt carry bit
         BSF     PORTB, bit0    ;  1 ®  Adat láb
         BSF     PORTB, bit1    ;  1 ® Clock láb
         DECFSZ  bit_count      ;
         GOTO    XM_LOOP        ;  vissza
         BCF     PORTB, bit1    ;  clock láb = L
```


2.6.2 PIC16CXXX PROGRAM-MEMÓRIA TÉRKÉP

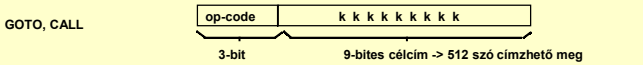


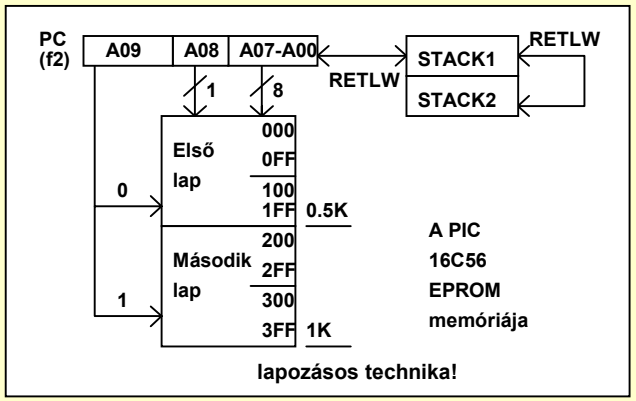
- **Maximum 8K szó (13 bit) program memória. Jelenlegi típusok kapacitása: 4K**
- **4 darab 2K szavas lap (11 bit)**
- **Lap váltás: PCLATH<4:3>**
- **Reset vektor: 0000h**
- **Megszakítás vektor: 0004h**



Példaprogram: ADVPAGE.ASM

2.6.1 PROGRAM-MEMÓRIA 12 BITES ESZKÖZÖKNÉL





14 bites eszköznél a célcím 11 bites, ezért a megcímezhető memória 2048 szó

2.6.3 PROGRAM MEMÓRIA LAPOZÁS (PAGING)

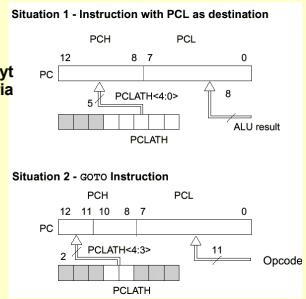
Lap / Bank határok

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

- **A PCLATH lapozó bitjeinek 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 módosítani, 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)**

2.6.4 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



Összefoglalás:

- 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

2.7.1 ADATMEMÓRIA: FÁJL REGISZTEREK II.

Addr	Name	Function
$00	IND	Used for indirect addressing
$01	RTCC	Real Time Clock Counter
$02	PC	Program Counter (low byte)
$03	STATUS	Holds Status bits of ALU
$04	FSR	File Select Register
$05	RA	Port A Control register
$06	RB	Port B Control register
$07	RC*	Port C Control register

A tok működésében alapvető szerepet játszanak

W - akkumulátor - műveletek operandusa

INDF - INDirect File register Álcíme: 00H ??? A 0 című regiszterre történő hivatkozás ténylegesen az FSR regiszterben lévő tartalom által meghatározott című regiszterre utal

RTCC (01H) - 8 bites számláló túlcsordulása IT-t okoz külső impulzus, vagy belső órajel növelheti

PC (02H) - az utasításmutató alsó 8bites része

STATUS (03H) - ld. később

FSR (04H) - File Select Register (04H) - tartalma határozza meg az indirekt utasításban szereplő regiszter címét, felső három bitje az aktuális regiszter bankot

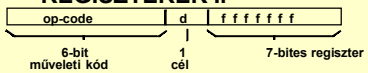
RA,RB,RC (05,06,07) portok !RA,!RB,!RC - portvezérlő regiszterek

MODE - portkonfigurálás mutató regisztere,

OPTION - ld. később

2.7.1 ADATMEMÓRIA: FÁJL REGISZTEREK I.

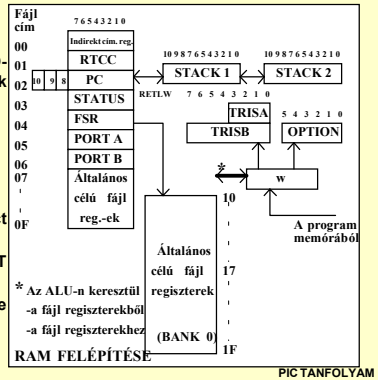
- Utasítások közvetlen címzéssel



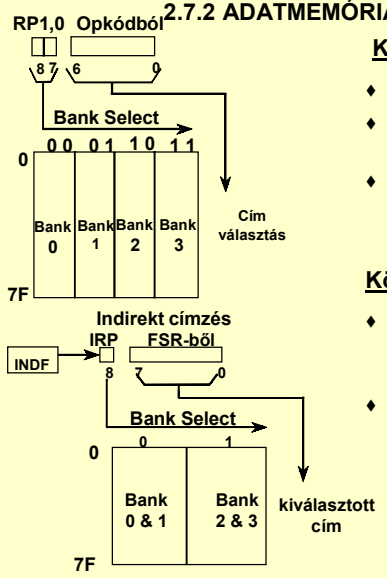
A 7 bites címrész miatti korlátozás feloldható a adatmemória bankok alkalmazásával, ezek összessége alkotja a fájlregiszter tömböt. Azaz egy cím egy adott bankbeli címet jelent!

A regisztermező két részre tagozódik.

1. működtető fájlregiszterek (belső működéshez és I/O-hoz) más néven Speciális Funkcióju Regiszterek (SFR)
   - A valósidejű óra / számláló regisztere (RTCC)
   - programszámláló (PC, Program Counter)
   - állapotregiszter (Status Register)
   - I/O regiszter (I/O Registers, PORTs)
   - fájlregisztert választó regiszter (FSR, File Select Register)
   - További speciális regiszterek szolgálnak az I/O PORT konfigurálására és az előosztó kezelésére.
2. általános célú regiszterek (General Purpose Registers).



2.7.2 ADATMEMÓRIA LAPJAI: A BANKOK



Közvetlen memória bank kezelés

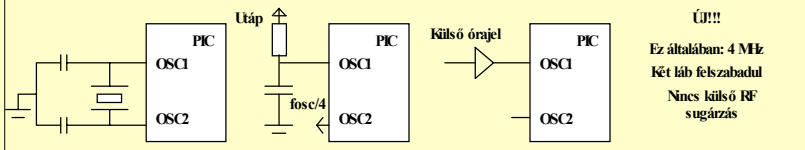
- 4 darab 128 bájtos adatmemória bank
- Speciális Funkcióju Regiszterek (SFR) ebben a RAM-ban vannak
- Bankok kiválasztása a Státusz Regiszter RP0 és RP1 bitjeivel történik

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

2.8 ÓRAJEL GENERÁLÁS



Órajel generálás PIC kontrollereknél

A belső oszcillátor -- kisebb pontossága és frekvenciastabilitása miatt olyan alkalmazásokban használható – ahol ezek nem okoznak problémát. A frekvenciát a tápfeszültség változása és a hőmérséklet változása befolyásolja.

FONTOS!!!

A tokban gyárilag beállítanak egy kiolvasható értéket, amely mellett a belső oszcillátor a legpontosabb (gyári jusztírozás). Ezt az értéket a tok égetésekor is be kell írni. (EPROM-os változat).

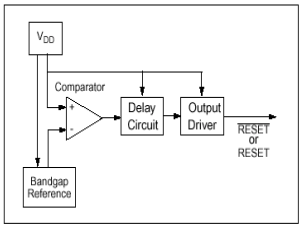
2.9.1 MICROCHIP RESET ÁRAMKÖRÖK

The Microchip Supervisor Device Family

Push-Pull (Active Low) MCP100/MCP809	Push-Pull (Active High) MCP101/MCP810	Open-Drain (no pull-up) MCP120	Open-Drain (internal pull-up) MCP130

- **CMOS Push-Pull kimenet**
- **7 feszültség szintű reset pont**
- **(MCP100, MCP809)**
- **Aktív magas reset (MCP101, MCP810) (pl.8051-hez!!!)**
- **T0-92 és SOT23 3-pin tokozás**
- **Ipari hőmérséklet tűrésű minden eszköz**
- **Lábkiosztás és a specifikáció kompatibilis a Maxim és Dallas hasonló eszközeivel**

BLOCK DIAGRAM

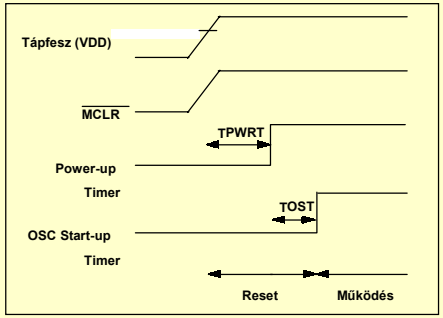


2.9 RESET KEZELÉS

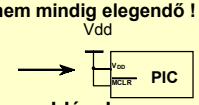
RESET FELADATA A BELSŐ TÁROLÓK ALAPÁLLAPOTBA HOZÁSA. EHHEZ A BELSŐ ÓRAJEL STABIL MŰKÖDÉSE SZÜKSÉGES!

BEKAPCSOLÁSKOR ELINDUL EGY FÜGGETLEN BELSŐ RC OSZCILLÁTORÓL MŰKÖDŐ SZAMLÁLÓ (POWER UP TIMER). ENNEK SZEREPE AZ ESETLEGES LASSÚ TÁPFESZÜLTSÉG NÖVEKEDÉSÉNEK A KOMPENZÁLÁSA.

AMIKOR EZ TÚLCSORDUL, ELINDUL EGY, AZ OSZCILLÁTOR ESETLEGES LASSÚ BEREZGÉSE MIATTI HIBÁS MŰKÖDÉS KIVÉDÉSÉT CÉLZÓ MÁSODIK KÉSLETETÉST BIZTOSÍTÓ SZÁMLÁLÓ (OSC START-UP TIMER)

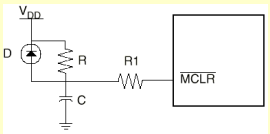


Ez majdnem mindig elegendő !



További megoldások:

•Tranzisztoros áramkör használata



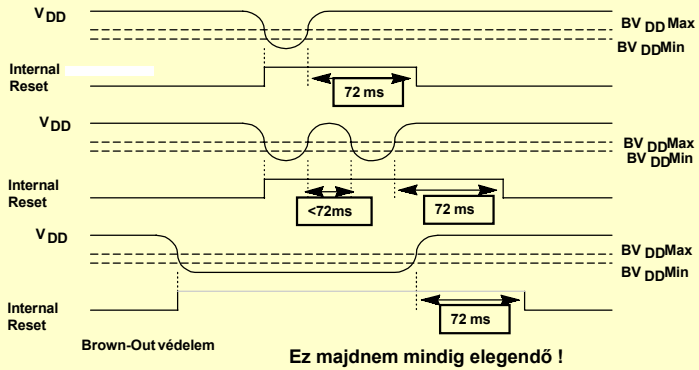
•Külső reset áramkör használata

2.9.2 BROWN-OUT

HA A TÁPFESZÜLTSÉG RÖVID IDŐRE LEESIK, AZ RESET-ET OKOZ.

MI VAN, HA A RESET ALATT ISMÉT LEESIK A FESZÜLTSÉG ? HIBÁS RESET !!!

A MEGOLDÁS: MINDIG AZ UTOLSÓ TÁPCSÖKKENÉSTŐL INDUL A RESET FOLYAMAT!



Ez majdnem mindig elegendő !

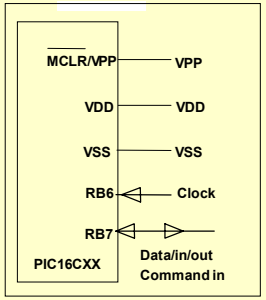
2.10 A TOKOK FELPROGRAMOZÁSA

FOGALMAK: PÁRHUZAMOS PROGRAMOZÁS, SOROS PROGRAMOZÁS FEJLESZTŐI PROGRAMOZÓ, IPARI PROGRAMOZÓ

PROGRAMOZÓI ÁLLAPOTBA KERÜLÜNK: HA AZ MCLR LÁBAT VPP FESZÜLTSÉGRE (12V) EMELJÜK. EZUTÁN A PROGRAMOZÓ BERENDEZÉS A CLOCK LÁBON ÓRAJELEKKEL ÉRVÉNYESÍTVE ELKÜLD EGY PARANCSOT (PL. 6 BITES A PARANCS), MAJD A TOK AZ ÓRAJELLEL ÜTEMEZVE AZ RB7 LÁBÁN KERESZTÜL VÁLASZOL.

HÁROM TOKTÍPUS: OTP (=One Time Programming), EPROM, (ABLAKOS), FLASH

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)**

2.11 PICSTART PLUS PROGRAMOZÓ

Picstart Plus Options Too	
Enable Programmer	A programozó engedélyezése
Program/Verify	Programozás, a beírt program ellenőrzése
Read Device	A programozóba helyezett PIC tok olvasása
Blank Check All	A tok üresség ének (memória + konfigurációs bitek) vizsgálata
Blank Check OTP	OTP tok üresség ének vizsgálata
Display Error Log	A programozás vagy ellenőrzés során kiírt hibák megnézése
Erase Program Memory	A program memória puffer törlése
Erase Configuration Bits	Konfigurációs bitek törlése
Reset Programmer	Programozó alaphelyzetbe állítása

EPROMOS ÉS FLASH EPROMOS TOKOK PROGRAMOZÁSÁRA IS ALKALMAS !

FEJLESZTŐI ÉS IPARI PROGRAMOZÓK VANNAK, PICSTART AZ ELŐBBI KATEGÓRIÁBA TARTOZIK

Hogyan lehet a programozót az új tokok programozására is képessé tenni?

3. PIC PERIFÉRIÁK

VANNAK KÖZÖS ELEMEK („SZÉRIATARTOZÉKOK” ;-)))

I/O LÁBAK

8 BITES SZÁMLÁLÓ/IDŐZÍTŐ (RTCC vagy TMR0)

WATCHDOG

CSALÁDELVET MEGVALÓSÍTÓ PERIFÉRIÁK:

- **TMR1, TMR2 időzítők**
- **A/D**
- **SOROS I/O**
- **PWM**
- **CAPTURE/COMPARE**
- **PÁRHUZAMOS (SLAVE) PORT**
- **I2C/SPI**

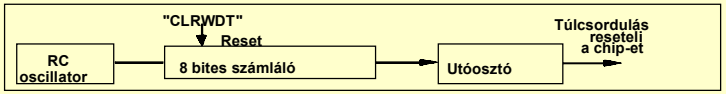
CSALÁDELV:
CPU AZONOS
ADAT ÉS PROGRAM MEMÓRIA MÉRETE
PERIFÉRIÁK SZÁMA
FAJTÁJA
KÜLÖNBÖZIK

A PERIFÉRIÁK MŰKÖDTETÉSE ÁLTALÁBAN NÉGY REGISZTER SEGÍTSÉGÉVEL VALÓSÍTHATÓ MEG, EZEK:

- **BEMENETI**
- **KIMENETI**
- **ÁLLAPOT (STÁTUSZ)**
- **VEZÉRLŐ (CONTROL) REGISZEREK**

EZEK A REGISZTERFÁJL ADOTT CÍMEIN HELYEZKEDNEK EL

3.0.1 WATCHDOG



- **Segít, ha a program „elkószál”**
- **Saját szabadonfutó RC oszcillátors van**
- **WDT programból nem kezelhető, 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**

3.0.2 SLEEP (POWER-DOWN) MÓD

SLEEP utasítással kerül a PIC16CXX SLEEP (power-down) állapotba.

Ilyenkor:

- 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.

Ébredés (Wake up):

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

A PROGRAM A SLEEP UTASÍTÁST KÖVETŐ UTASÍTÁSSAL FOLYTATÓDIK

Fogyasztás: 1.5 µA typical @ 4V

3.0.3 MELEG RESET

Meleg reset: tápfesz. megvan, és úgy reset !!!

A STÁTUSZ regiszter T0 (watchdog állítja) és PD (sleep állítja) bitje alapján dönthető el:

TO	PD	RESET oka
0	0	WDT időtúlfutás energiatakarékos (SLEEP) üzemben
0	1	WDT időtúlfutás normál üzemben
1	0	külső RESET (MCLR) energiatakarékos SLEEP) üzemben
1	1	Power-On, Bekapcsolás
X	X	külső RESET (MCLR) normál üzemben

3.0.4 PIC16CXX MEGSZAKÍTÁSOK

- MEGSZAKÍTÁS: SPECIÁLIS SZUBRUTINHÍVÁS Esemény hatására PC-be egy cím (vektor) kerül)
- Egyszintű IT (egyszerre csak egyet)
- egyidőben bekövetkező megszakítás esetén a prioritást a IT-ben való lekérdezési sorrend határozza meg. 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.

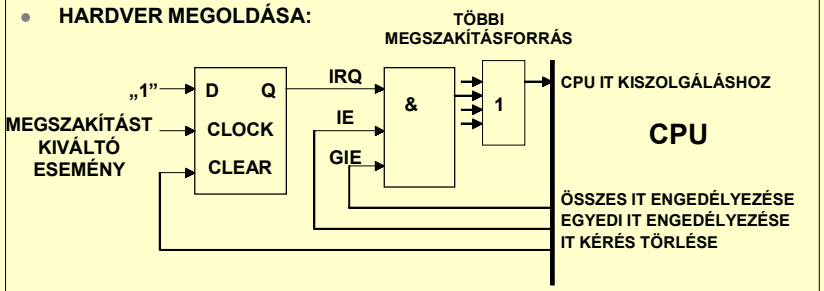
Prioritás: a kiszolgálás sorrendje

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
```


3.0.4 MEGSZAKÍTÁSOK (FOLYT)

- 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
- HARDVER MEGOLDÁSA:

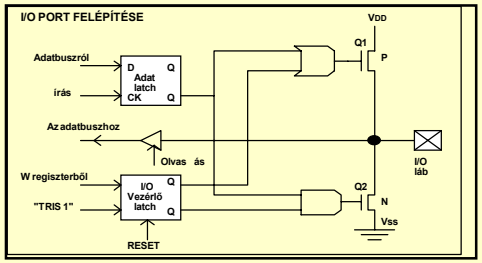


3.1 I/O PORTOK

Portok: kapuk a külvilág felé - bemenetek vagy kimenetek

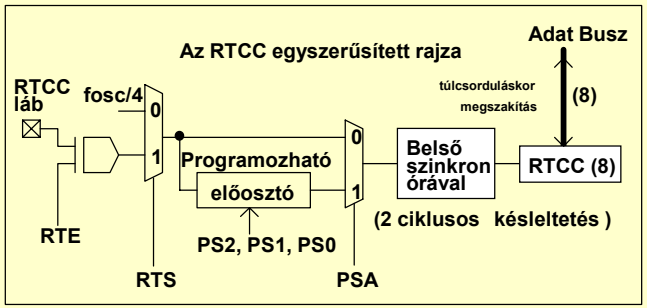
BÁRMELYIK UTASÍTÁS, AMELYIK A RÉGI ÉRTÉKÉT HASZNÁLVA ÁLLÍTJA ELŐ A REGISZTER ÚJ ÉRTÉKÉT READ-MODIFY-WRITE, AZAZ BEOLVAS - MÓDOSÍT - VISSZAÍR UTASÍTÁSNAK HÍVJUK.

AZAZ AZ ILYEN UTASÍTÁSOK - MÉG HA CSUPÁN A REGISZTER EGYETLEN BITJÉRŐL IS VAN SZÓ - ELŐSZÖR BOLVASSÁK A REGISZTER TARTALMÁT, ELVÉGZIK A MŰVELETET ÉS AZ EREDMÉNYT VISSZAÍRJÁK A REGISZTERBE.
PL.: CLR FR.BIT, ADD FR,W, DEC FR, STB.



Kimeneti tárolóba írunk, de lábat olvasunk!!!

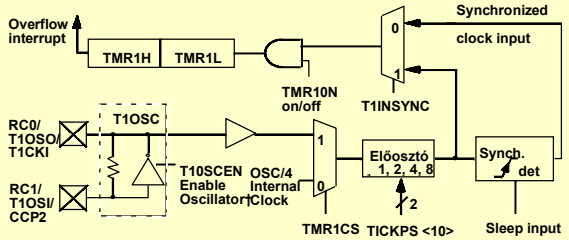
3.2.1 TMR0 = REAL TIME CLOCK COUNTER (RTCC)



- **Olvasható és írható**
- **Túlcsordulásakor megszakítást generál a 14 és 16 bites családnál, a 12 biteseknél nincs megszakítás**
- **Az előosztója programozható (de a watchdog is használja!)**
- **Időzítőként: Az órajel frekvencia: OSC/4 (5 MHz @ 20 MHz-es kontroller órajelnél inkrementálódik ,**
- **Előosztóval a külső órajel 50 Mhz-ig mehet**
- **Számlálóként:programozható az élváltás, amire lép (fel- vagy lefutó él)**

3.2.2 TMR1

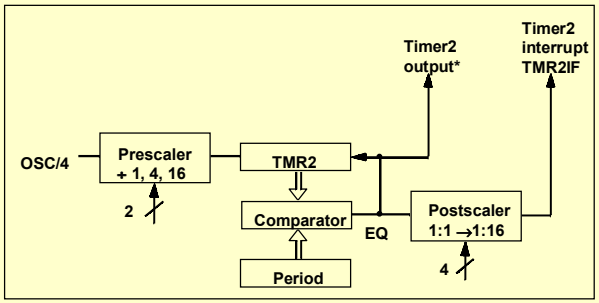
- **Aszinkron számláló mód**
- **Sleep ideje alatt is működik**
- **Ha TMR1 túlcsordul, ébreszti a processzort (külső órajel lépteti)**
- **16-bites szaámláló/időzítő, 3 bites előosztó (÷ 1, 2,4,8)**
- **SLEEP módban is fut az idő**
- **LP oszcillátor**
- **A CCP capture és comparátor (CCP) modul(ok) időalapja**
- **Aszinkron számláló mód**
- **Direkt 32 KHz - 200 KHz kvarc működés RC0 és RC1-en keresztül**



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

3.2.3 TMR2

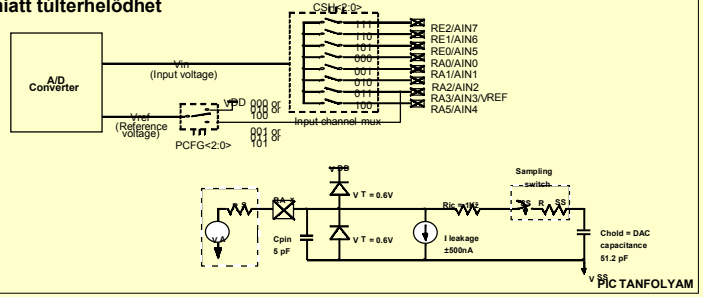
- **8 bites időzítő**
- **4 bites előosztó (÷ 1, 4, 16)**
- **4 bites utóosztó (÷ 1 to ÷16)**
- **Az utóosztó túlcsordulása megszakítást okoz**
- **A Szinkron Soros Port (SSP) modul baud-rate generátora**



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

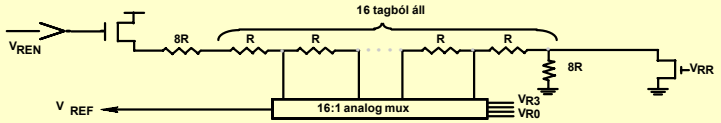
3.3 A/D

- A/D konverter modul:
- Maximum 8 analóg bemenet multiplexelődik egy A/D konverterre
- Mintavevő-tartót tartalmaz
- 11 µs-os mintavételi idő (10K forrás impedanciánál)
- 16 µsec konverziós idő csatornánként (20 µsec PIC16C71-nél)
- 8 bites felbontás (±1 LSB accuracy (±2 LSB @ VDD=3VA/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



3.4 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

3.3.1 MCP320X 12 BITES AD KONVERTER

- 12-bit +/- 1 LSB pontosság
- 1,2,4 és 8 analog bemeneti csatorna
- 2.7 - 5.5V működési tartomány
- 2.0 MHz-s SPI™ illesztés
- 100 000 minta másodpercenként
- -86 dB typical THD
- 72 dB jel-zaj és torzítás (Signal to Noise and Distortion)
- Olcsó 8-, 14- and 16-lábú SOIC and DIP tokozás

ÖNÁLLÓ TOKBAN 12 BITES AD ÁTALAKÍTÓ !!!

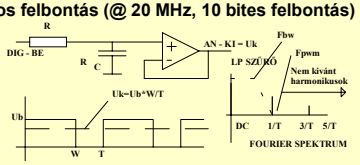
3.5 CAPTURE/COMPARE/PWM (CCP) MODUL I.

<u>Capture: adott feltétel teljesülése esetén egy számláló értékének beírása egy regiszterbe</u>

- 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

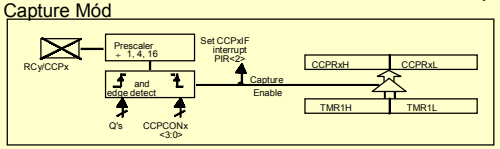
8 vagy 10-bit felbontás: 80 KHz frekvencia 8-bites felbontásnál
20 KHz frekvencia 10-bites felbontásnál

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



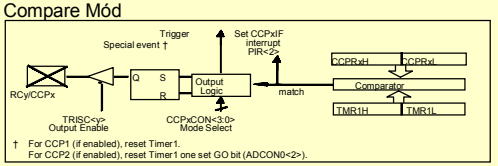
3.5 CAPTURE/COMPARE/PWM (CCP) MODUL II.

Capture Mód



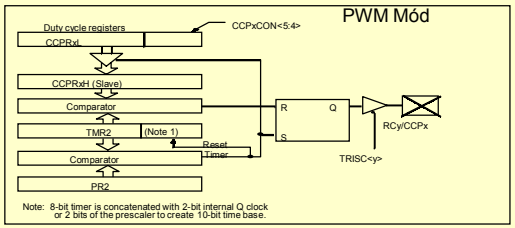
CAPTURE (BEÍRÁS): ***Egy lábon fellépő szintváltás hatására a TMR1 számláló 16 bites tartalmát a <CCPRxH:CCPRxL> 16 bites regiszterbe írjuk.***

Compare Mód



COMPARE (ÖSSZEHASONLÍTÁS): ***A TMR1 számláló 16 bites tartalmát a <CCPRxH:CCPRxL> 16 bites regiszterben lévő tartalommal hasonlítjuk össze. Egyezés esetén egy lábon kiadunk egy jelet, illetve megszakítást generálunk.***

PWM Mód



A KITÜLTÉSI TÉNYEZŐ A FREKVENCIÁTÓL FÜGGETLEN, DE A FREKVENCIA A FELHARMONIKUS TARTALMAT BEFOLYÁSOLJA (SZŰRÉS)

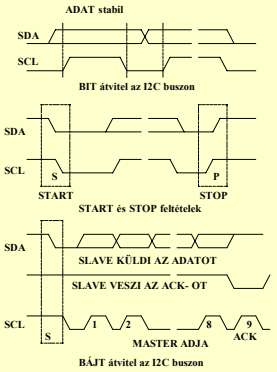
3.7.1 I2C BUSZ (FAKULTATIV)

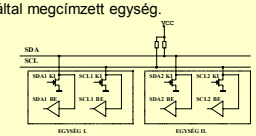
Az adatforgalom két vezetéken történik. Az SCL vezeték az órajelet szolgáltatja , az SDA jelű végzi az adatforgalmazást. A közös potenciált a GND összekötés biztosítja. Az SCL és SDA vonalak kimenetei nyitott kollektoros megoldásúak, így a vonalak alaphelyzetben magas állapotban vannak. Ennek előnye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

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

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

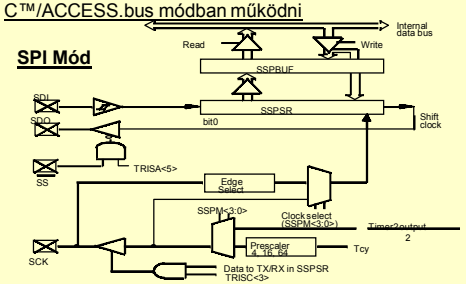
A **START** és **STOP** állapotokat csak a mester generálhatja. A busz aktív a **START** és **STOP** állapot között.

3.6 SSP MODUL

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

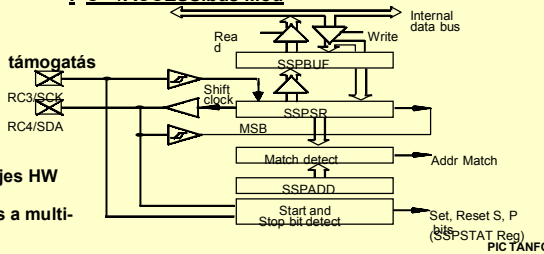
Max baud ütem (20 MHz)-nél
Mester 5 MHz
Szolga 2.27 MHz
Programozható baud ütem. (OSC ÷ 4, 16, 64, vagy TMR2 output ÷ 2)
Programozható órajel polaritás adás/vételnél

SPI Mód



I²C™/ACCESS.bus Mód

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

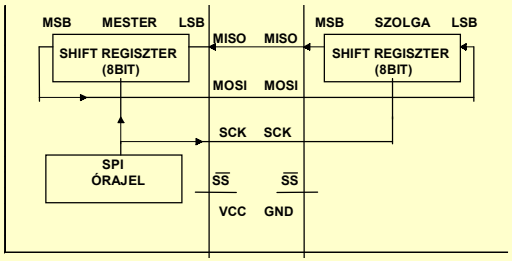


- **A szolga funkciók teljes HW kialakítása**
- **HW segíti a mester és a multi-mester funkciókat**

3.7.2 SPI BUSZ (FAKULTATIV)

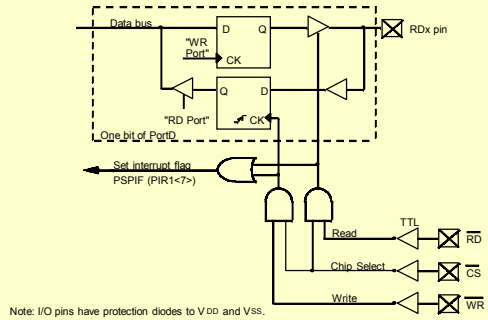
MOTOROLA mikroprocesszoroknál vezették be

- **Az SPI lényegében egy két vonalon összekötött shift regiszter,**
- **Az adatátvitelnél egy master és több szolga lehet. A mester vezérli a folyamatot, adja az órajelet.**
- **Az SPI két adat és két vezérlő vonalból áll.**
  - **Master Out Slave In – MOSI**
  - **Master In Slave Out – MISO**
  - **Serial Clock – SCLK a mester által küldött órajel**
  - **Slave Select - SS a szolgát jelöli ki**
- **Két javasolt protokoll van: egy mesteres - többmesteres**
  - **Egy mesteres (kontroller – perifériák) A mester kiküldi a szolga címét a keretben. A megcímzett eszköz a MISO vonalon küldi be a csomag elfogadását, míg a többi eszköz a saját MISO vonalait kikapcsolja.**
  - **Több-mesteres protokoll: mesterek ütközésének elkerülése - arbitráció**



3.8 PÁRHUZAMOS SZOLGA PORT

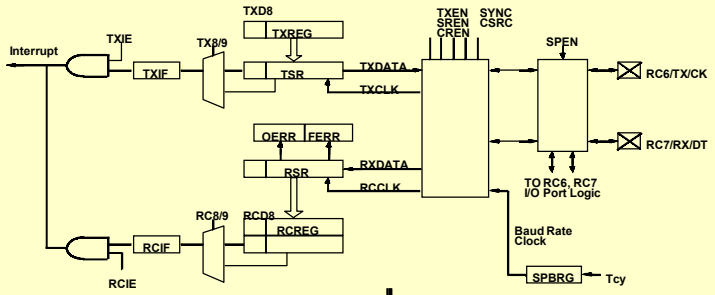
Parallel Slave Port Blokkvázlata



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)

3.9 SCI MODUL

SCI blokkvázlata



SCI Aszinkron Mód

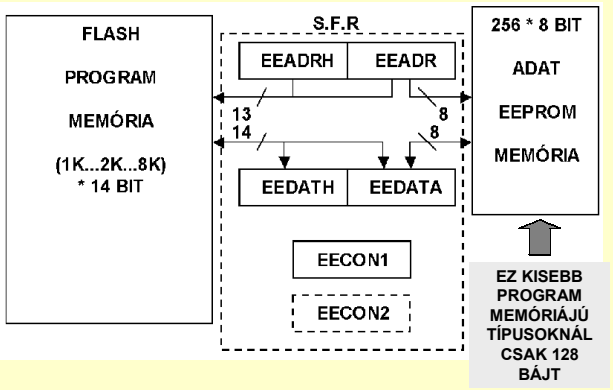
Adás
Adó regiszter kettős pufferelésű
Ha az adó kész az adatátvitelre, megszakítást generál
8/9 bites adás
Vétel
Vevő regiszter kettős pufferelésű
Felülírási hiba jelzőbit
Keretezési hiba detektálása
Minden adatbit többségi detektálása (zajvédelem)
Ha bejött egy teljes adat, megszakítást generál

SCI Szinkron Mód

Adás
Adó regiszter kettős pufferelésű
Egymást követő 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 követő bájtokat úgy veszi, hogy lehetséges 16 bites szavak vétele, vagy akár hosszabbaké is

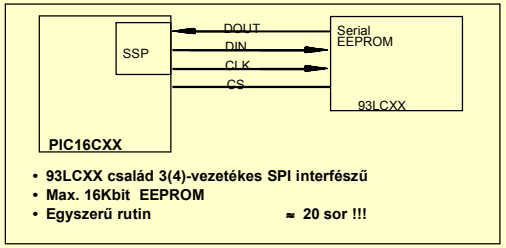
3.10 BELSŐ FLASH EEPROM PROGRAM ÉS ADAT MEMÓRIA

MEMÓRIÁK A PIC16F87X TÍPUSOK ESETÉN

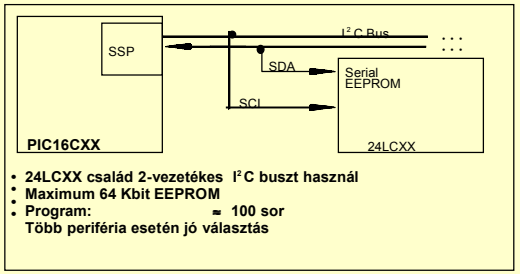


A PROGRAM-MEMÓRIA PROGRAMBÓL ÍRHATÓ, OLVASHATÓ !!!

3.11 KÜLSŐ ESZKÖZÖK: EEPROM ILLESZTÉS



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



- **24LCXX család 2-vezetékes I²C buszt használ**
- **Maximum 64 Kbit EEPROM**
- **Program: ≈ 100 sor**

Több periféria esetén jó választás

4. PIC CSALÁD (RÉGEN)

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/COMPARE/PWM										1		2		2
SOROS PORT										SPI/I2C				SCI
PÁRHUZAMOS SLAVE PORT										IGEN		IGEN		
8 bites AD											4	8		
MEGSZAKÍTÁSOK									3	8	4	12	4	11
I/Ovonalak	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

4.1 ÚJ ESZKÖZÖK A MICROCHIP-TŐL

- Flash EEPROM program-memóriájú eszközök megjelenése
  - Gyors és egyszerű újra programozhatóság
  - In-Circuit Debugging (ICD) - azaz lépésenkénti ill. töréspontos programfuttatás
- Választható belső oszcillátor
- Nagyobb órajelsebesség - nagyobb teljesítmény (MIPS)
- Nagyobb kapacítású belső EEPROM-os adatmemória
- Programozható, feszültségcsökkenést figyelő áramkör
- Új eszközök:
  - RESET áramkörök
  - Műveleti erősítők
  - 10 és 12 bites AD átalakítók
  - CAN interfész (MCP2510 CAN kontroller SPI illesztéssel)
  - RFID : rádiófrekvenciás azonosító eszközök

4.2 PIC16F87X FLASH EEPROM-OS KONTROLLER TULAJDONSÁGAI

PIC16F87X Features at a Glance

- **8K x 14 FLASH Program Memory**
  - Typ. 1000 E/W
  - Byte/Word Read/Write at VDD
- **256 x 8 EEPROM Data Memory**
  - Min. 100K E/W
- 368 x 8 Data Memory (RAM)
- 33 I/O ports
  - 25mA sink/source
- 3 Timers
  - 1 - 16-bit
  - 2 - 8-bit
- **10-bit A/D**
- Two **C**apture/**C**ompare/**P**WMs
- USART
  - 9-bit addressable
  - High Speed
- **Enhanced SPI™**
  - All 4 SPI modes supported
  - Microwire® Support
- **Master I²C™**
  - **Hardware Write to I²C devices**
- **I**n-**C**ircuit-**S**erial **P**rogra**m**ming™
- **In-Circuit-Debugger**
- **P**arallel **S**lave **P**ort

4.3 PIC16F87X CSALÁD

Device	PIC16F873	PIC16F876	PIC16F874	PIC16F877
Operating Freq.	DC-20MHz	DC-20MHz	DC-20MHz	DC-20MHz
Resets/Delays	POR,BOR	POR,BOR	POR,BOR	POR,BOR
	PWRT,OST	PWRT,OST	PWRT,OST	PWRT,OST
Prog. Mem	4K FLASH	8K FLASH	4K FLASH	8K FLASH
Data Mem	192	368	192	368
EEPROM	128	256	128	256
Interrupts	13	13	14	14
I/O Ports	22	22	33	33
Timers	3	3	3	3
CCP	2	2	2	2
SPI	Y, Enh.	Y, Enh.	Y, Enh.	Y, Enh.
I2C	Mstr/Slave	Mstr/Slave	Mstr/Slave	Mstr/Slave
USART	Y, 9-bit	Y, 9-bit	Y, 9-bit	Y, 9-bit
Parallel Port	N	N	Y	Y
10-bit A/D	5 ch	5 ch	8 ch.	8 ch.
ICSP/ICD	Y/Y	Y/Y	Y/Y	Y/Y

4.4 PIC16C7XX CSALÁD TULAJDONSÁGAI

Device	PIC16C712/716	PIC16C717	PIC16C770/771
Operating Freq.	DC - 20MHz	DC - 20MHz	DC - 20MHz
Resets	BOR	P.BOR, PLVD	P.BOR, PLVD
Channels A/D	4 x 8-bit	6 x 10-bit	6 x 12-bit
Serial Communications	None	SSP	SSP
I/O Ports	13	16	16
Program Memory	1K/2K X 14	2K X 14	2K/4K X 14
Data Memory (Bytes)	128	128	256
CCP Module	1	1 w/Enh. PWM	1 w/Enh. PWM
Timers	1 x 16-bit	1 x 16-bit	1 x 16-bit
	2 x 8bit, WDT	2 x 8bit, WDT	2 x 8bit, WDT
Packages	18P,18SO,	18P,18SO,	20P, 20SO,
	18JW, 20SS	18JW, 20SS	20JW, 20SS
Internal RC System Clk	No	4MHz	4MHz

PLVD: Programmmable Low Voltage Detect (IT-t okozhat!)

4.5 ÚJ ÉS RÉGI TÍPUSOK ÖSSZEHASONLÍTÁSA

Device	PIC16C74A	PIC16C74B	PIC16F874	PIC16C77	PIC16F877
Operating Freq.	DC-20MHz	DC-20MHz	DC-20MHz	DC-20MHz	DC-20MHz
Resets/Delays	POR,BOR PWRT,OST	POR,BOR PWRT,OST	POR,BOR PWRT,OST	POR,BOR PWRT,OST	POR,BOR PWRT,OST
Prog. Mem	4K EPROM	4K EPROM	4K FLASH	8K EPROM	8K FLASH
Data Mem	192	192	192	368	368
EEPROM	0	0	128	0	256
Interrupts	12	12	14	12	14
I/O Ports	33	33	33	33	33
Timers	3	3	3	3	3
CCP	2	2	2	2	2
SPI	Y	Y	Y, Enh.	Y, Enh.	Y, Enh.
I2C	Slave	Slave	Mstr/Slave	Slave	Mstr/Slave
USART	Y	Y	Y, 9-bit	Y	Y, 9-bit
Parallel Port	Y	Y	Y	Y	Y
8-bit A/D	8 ch.	8 ch.		8 ch.	
10-bit A/D			8 ch.		8 ch.
ICSP/ICD	Y/N	Y/N	Y/Y	Y/N	Y/Y

4.6 PIC16F62X CSALÁD TULAJDONSÁGAI

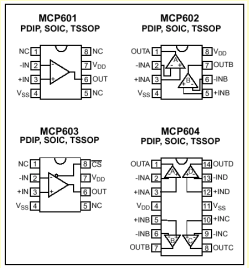
SZUPER!!!

Device	PIC16F627	PIC16F628
Operating Freq.	DC - 20MHz	DC - 20MHz
Resets	BOR	BOR, RLVD
Comparators	2 + VREF	2 + VREF
Serial communications	USART	USART
I/O Ports	16	16
Program Memory	1024 x 14	2048 x 14
Data Memory (Bytes)	224	224
EEPROM Data Memory (Bytes)	128	128
CCP module	1	1
Timers	1 x 16-bit	1 x 16-bit
	2 x 8-bit, WDT	2 x 8-bit, WDT
Packages	18P,18SO,	18P,18SO,
	20SS	20SS
Internal RC System Clk	4MHz	4MHz

4.7 PIC18CXXX CSALÁD

Device	PIC18C242	PIC18C442	PIC18C252	PIC18C452
Max Clk Speed	40MHz	40MHz	40MHz	40MHz
Resets	POR,PBOR	POR,PBOR	POR,PBOR	POR,PBOR
LVD & Vref	Y	Y	Y	Y
Prog. Mem	8K X 16	8K X 16	16K X 16	16K X 16
Data Mem	512	512	1536	1536
Hardware MPY	8 X 8	8 X 8	8 X 8	8 X 8
Interrupts	13	13	14	14
I/O Ports	23	23	34	34
Timers	3	3	3	3
CCP	2	2	2	2
SPI	Y, Enh.	Y, Enh.	Y, Enh.	Y, Enh.
I2C	Mstr/Slave	Mstr/Slave	Mstr/Slave	Mstr/Slave
USART	Y, 9-bit	Y, 9-bit	Y, 9-bit	Y, 9-bit
Parallel Port	N	N	Y	Y
10-bit A/D	5 ch	5 ch	8 ch.	8 ch.
ICSP	Y	Y	Y	Y

4.8 MŰVELETI ERŐSÍTŐK



- 1, 2 ill. 4 erősítő/tok, erősítő egedélyező bemenettel
- 2.7V - 5V működési tartomány!!!
- 2mV ofszet
- teljes kimeneti vezérelhetőség (Rail-to-Rail)
- Tokozás: PDIP, 150mil SOIC és SOT-23

DC Specifications(1)	LMC6492A	OPA2337A	TLC2262	MCP602
Input Offset Voltage(mV)	3.8	3.0	2.5	2.0
I_B @ 85 deg C (pA)	200	10	100	80
Input Voltage Swing (V)	0 to 5	-0.2 to 4.8	0 to 4.2	-0.2 to 3.8
Aol (dB)		100	98	100
PSR (uV/V)	562	125	100	178
CMRR (dB)	65	74	70	80
Output Voltage Swing (V)	0.18 to 4.8	0.125 to 4.875	0.15 to 4.85	0.1 to 4.9
Power Supply Current (uA)	2100	750	500	325
Power Supply Voltage (V)	5 to 15	2.7 to 5.5	4.4 to 16	2.7 to 5.5
AC Specifications				
Gain Bandwidth Product (MHz)	1.5	3.0	0.71	3.0
Slew Rate (SR, V/us)	1.3	1.2	0.55	1.8
Settling Time (us)		2.5	6.4	8.0
Input Voltage Noise (nV/rt Hz)	37	26	12	27

4.9 RFID - TAG

RFID =

RÁDIÓFREK-VENCIÁS AZONOSÍTÓ ELEM

Single, MCRF200/202/250

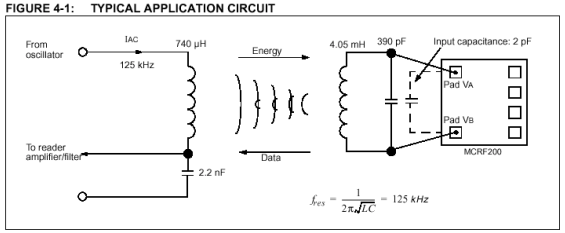
- 125kHz, passzív (nem kell elem), 128 vagy 96 bit
- MCRF200: EM, Temic, Philips kompatibilis
- MCRF202: Binaris érzékelő bemenet
- MCRF250: Többszörös olvasás, ütközést kivédő

MCRF355/360

All: 13.56MHz, paszív, anti-collision, cloaking

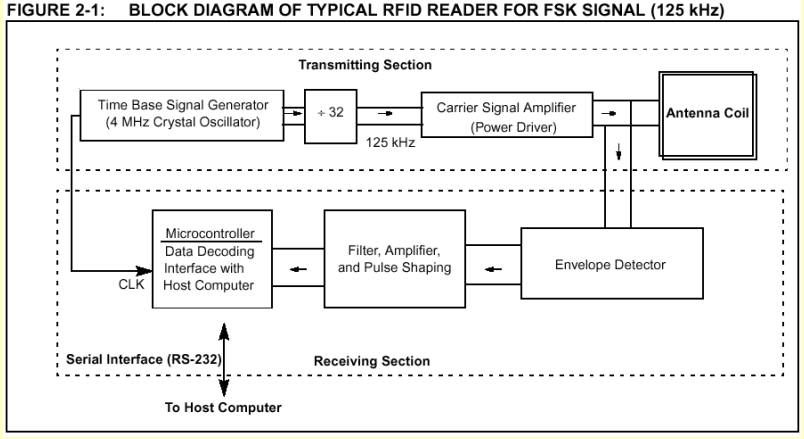
- MCRF355: Ipari vezető csak olvasható típus
- MCRF360: 100pF tokon belüli rezonancia kapacitás

FIGURE 4-1: TYPICAL APPLICATION CIRCUIT

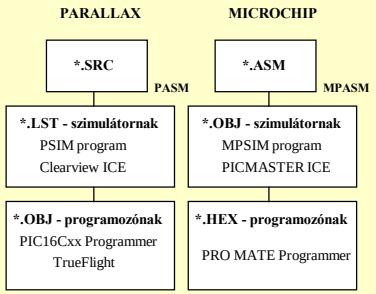


4.10 RFID - READER

FIGURE 2-1: BLOCK DIAGRAM OF TYPICAL RFID READER FOR FSK SIGNAL (125 kHz)



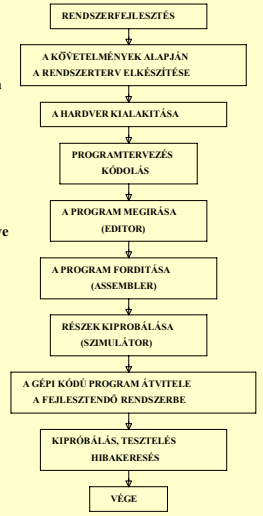
5. RENDSZERFEJLESZTÉS



Szövegszerkesztővel forráskód megírása

Assemblálás

A fordítás eredménye



A RENDSZERFEJLESZTÉS BLOKKVÁZLATA

HOGYAN LEHET KONTROLLERREL EGY ADOTT FELADATOT MEGOLDANI?

- *Mit akarunk megvalósítani?*
- *Megvalósítható-e mikroszámítógéppel?*
- *Ember-gép kapcsolat megtervezése !!!!!*
- *Be- és kimenetek száma, típusa (analóg, digitális)*
- *Programtár nagysága*
- *A megfelelő mikroszámítógép kiválasztása*

5.1 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)

- 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 / I²C busz illesztés
- Párhuzamos slave port
- 64 bájtos EEPROM adatmemória

@ Insert one line of assembly language code.
ASM..ENDASM Insert assembly language code section.
BRANCH Computed **GOTO** (equiv. to ON..GOTO).
BRANCHL BRANCH out of page (long **BRANC**H).
BUTTON Debounce and auto-repeat input on specified pin.
CALL Call assembly language subroutine.
CLEAR Zero all variables.
COUNT Count number of pulses on a pin.
DATA Define initial contents of on-chip EEPROM.
DEBUG Asynchronous serial output to fixed pin and baud.
DISABLE Disable **ON INTERRUPT** processing.
DTMFOUT Produce touch-tones on a pin.
EEPROM Define initial contents of on-chip EEPROM.
ENABLE Enable **ON INTERRUPT** processing.
END Stop execution and enter low power mode.
FOR..**NEXT** Repeatedly execute statements.
FREQOUT Produce up to 2 frequencies on a pin.
GOSUB Call BASIC subroutine at specified label.
GOTO Continue execution at specified label.
HIGH Make pin output high.
HSERIN Hardware asynchronous serial input.
HSEROUT Hardware asynchronous serial output.
I2CREAD Read bytes from I2C device.
I2CWRITE Write bytes to I2C device.
IF..THEN..ELSE..ENDIF Conditionally execute statements.
INPUT Make pin an input.
{LET} Assign result of an expression to a variable.
LCDOUT Display characters on LCD.
LOOKDOWN Search constant table for value.
LOOKDOWN2 Search constant / variable table for value.
LOOKUP Fetch constant value from table.
LOOKUP2 Fetch constant / variable value from table.
LOW Make pin output low.
NAP Power down processor for short period of time.
ON INTERRUPT Execute BASIC subroutine on an interrupt.
OUTPUT Make pin an output.
PAUSE Delay (1mSec resolution).
PAUSEUS Delay (1uSec resolution).
PEEK Read byte from register.
POKE Write byte to register.
POT Read potentiometer on specified pin.
PULSIN Measure pulse width on a pin.
PULSOUT Generate pulse to a pin.
PWM Output pulse width modulated pulse train to pin.
RANDOM Generate pseudo-random number.
RCTIME Measure pulse width on a pin.
READ Read byte from on-chip EEPROM.
RESUME Continue execution after interrupt handling.
RETURN Continue at statement following last **GOSU**B.
REVERSE Make output pin an input or an input pin an output.
SERIN Asynchronous serial input (BS1 style).
SERIN2 Asynchronous serial input (BS2 style).
SEROUT Asynchronous serial output (BS1 style).
SEROUT2 Asynchronous serial output (BS2 style).
SHIFTIN Synchronous serial input.
SHIFTOUT Synchronous serial output.
SLEEP Power down processor for a period of time.
SOUND Generate tone or white-noise on specified pin.
STOP Stop program execution.
SWAP Exchange the values of two variables.
TOGGLE Make pin output and toggle state.
WHILE..WEND Execute statements while condition is true.
WRITE Write byte to on-chip EEPROM.
XIN X-10 input.
XOUT X-10 output.

5.3 PICBASIC- PRO COMPILER UTASÍTÁSOK

PARALLAX CÉG UTASÍTÁSKÉSZLETE

A MNEMONIKOK NAGYON HASONLÓAK AZ MCS-51-ES UTASÍTÁSKÉSZLETÉHEZ

ÚJJÁSZÜLETÉS:

A SCENIX CÉG SX KONTROLLERÉBEN

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.)

5.4 SX – A VILÁG LEGGYORSABB MIKROVEZÉRLŐJE

- 50 MIPS (millió utasítás mp.-ként) sebesség
- DC-50MHz működési tartomány
- 1 utasítás órajelenként (ugrás 3 órajel)
- 20ns utasításciklusidő, 60ns megszakítási válaszidő
- E² Flash Eprom Technológia
- Helyben programozható az OSC lábakon
- 10,000 írás olvasási ciklus

Gyors megszakításkezelés
- Megszakításkor a PC, W, STATUS, és FSR regiszterek automatikus mentése
- A belső időzítő/számláló (RTCC) túlcsordulás 3 órajel hosszúságú megszakítást generál
- RB port lábain a jelváltás megszakítást generálhat, vagy felébredést a kisfogyasztású (szundi) állapotból

Számos be/kimeneti (B/K) lehetőség
- Az összes portláb egyenként állítható be- vagy kimenetnek
- A bemenetek állíthatóan TTL vagy CMOS szint kompatibilisek
- Minden lábon aktivizálható egy belső felhúzó ellenállás (~20kohm a tápfeszre (VDD))
- RB és RC bemeneteknél választható Schmitt triggeres működés
- Az összes kimenet 30mA-es áramot képes elnyelni/kiadni
- RA kimenetek szimmetrikus meghajtóként viselkednek (azonos Vdrop +/-)
- Analóg komparátor az RB porton (RB0 a komparátor kimenete, RB1: in-, RB2: in+)

Új megoldás: virtuális perifériák

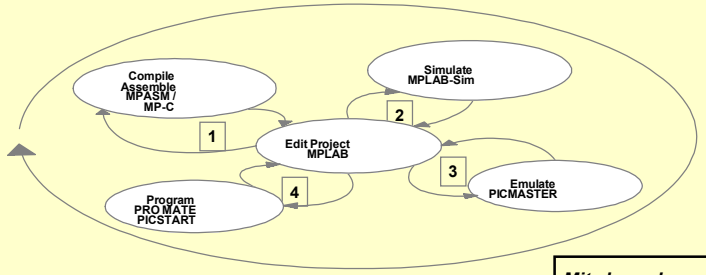
Járulékos komponensek csökkentése
- Belső oszcillátor (off, 4MHz...32768Hz 8 lépésben) elektromágneses sugárzást (EMI) csökkenti
- Beépített rövid idejű tápfeszcsökkenés zavar (brown-out) érzékelés (off, 4.0V)
- Bekapcsoláskor belső reset áramkör

PIC16C5x kompatibilitás
- A hatékonyabb kódolás érdekében tíz új utasítás
- Könnyen kezelhető nagysebességű megszakítás
- 8 szintű veremkezelés is választható
- A Carry jelzőbitet az összeadásnál és kivonásnál figyelembe vehetjük (megfelel az ADDC ,SUBB utasításoknak, az eredeti ADD és SUB a műveletvégzéskor a Cy - t nem veszi figyelembe!)
- W akkumulátor az RTCC regiszter helyébe kerülhet, egyszerűsítve a számláló kezelését
- A kód memória programból olvasható, lehetővé téve a program sérülésének a detektálását ('98 UL compliance)
- E²Flash programmemória és a fájl-regiszter RAM biztosítóval a '54 *- '58 elemknél található méretrekorlátozható
- A Turbo biztosítóval állítható 1utasítás/1órajel vagy 1utasítás/4órajel utasítás végrehajtás

www.scenix.com
www.parallaxinc.com
alpha1.obuda.kando.hu/~konya

Általános jellemzők
- E²Flash kódmemória: 2048 x 12 bit
- RAM regiszterek száma: 136 bájt
- Működési feszültség: 3.7V - 6.25V
- Fogyasztás: csak 12mA @ 50MHz, 5V !!!
- A Parallax Inc. - től komplett fejlesztőrendszer vásárolható (SX-Key)

5.5 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*

Mit akarunk megvalósítani?
Megvalósítható-e mikroszámítógéppel?
Ember-gép kapcsolat megtervezése !!!!!
Be- és kimenetek száma, típusa (analóg, digitális)
Programtár nagysága
A megfelelő mikroszámítógép kiválasztása

5.5.0.1 ASSEMBLER

Az gépi kódú bináris utasításokat és paramétereit könnyen megjegyezhető szavakkal helyettesíti. Ezekkel és az adatokhoz rendelt szimbólumokkal írjuk meg a programot az ún. assembly nyelven. Az így megírt program forrásszövegét assembler-rel fordítjuk le.

Egy assembler sor négy mezőből áll: (lényegében egy 4 oszlopos táblázat)

```
CIMKE       UTASÍTÁS    OPERANDUS       MEGJEGYZÉS
Például:
UJRA:       MOV         A,45H           ;A ÉRTÉKE
```

Kereszfordító: (cross assembler): a gépi kódot előállító program egy másik gépen fut.
Egy assembler program utasításokat és direktívákat tartalmaz

Az assembler direktívák tulajdonképpen a fordítóprogramnak szóló, a fordítást vezérlő parancsok. Ezek segítségével:

• definiálhatók szimbólumok (pl. EQU, DATA)

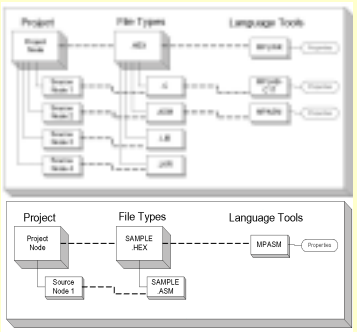
```
HATAR       EQU     120     ; pl. MOV A,#HATAR   A tartalma 120 lesz
BEM1        DATA    20H     ;20H mem.címre BEM1 néven hivatkozunk
```

• foglalhatók le és inicializálhatók mem. területek (pl. DS, DB, DW)

```
            DS      20      ;20 bájt lefoglalása a memóriában
SZOV:       DB 'EZT ÍROM KI' ;szöveg definiálása
```

• állíthatók be címek (ORG, END)

```
START       ORG     1000H   ;1000H címen kezdődik a program
            END             ;ez a program vége
```

5.5.0 MPLAB FEJLESZTÉS

Az MPLAB-ban a projekt egy projekt objektum (ez a node, azaz csomópont) egy vagy több forrás objektumból (node-ból) épül fel. Ezek a források általában MPASM assembler, illetve magas szintű C vagy BASIC forrásfájlok, előre lefordított könyvár fájlok, más tárgykódok (object fájlok), illetve az ezek összekapcsolását végző linker script vezérlő szövegfájlok (hasonlók a DOS batch fájljaihoz).

Linkeléskor az összelinkelni kívánt ún. object fájlokból hozzuk létre a végleges lefordított tárgykódot, vagyis egy .HEX fájlt ilyen node-ok építik fel. Egy node egy forrásprogramot és egy fordítóprogramot (BASIC, C) vagy assembler programot is meghatároz ezeket hívjuk nyelvi eszközöknek (Language Tool = Nyelvi eszköz).

1. Új projekt létrehozásakor először a program fejlesztési módját, és a használt PIC processzor típusát kell megadni az *Options>Development Mode* menüpontban:
   - Editor Only - csak program írás és fordítás
   - MPLAB-SIM – szimulator
   - MPLAB-ICE Emulátor
   - PICMASTER Emulátor
   - SIMICE Simulator
   - ICEPIC
   - MPLAB-ICD Debugger - ez az új lehetőség !
   - *A megadás után a RESET-re kattintva történik meg a kiválasztás.*
2. Utána a *Project>New Project* menüpontnál kell megadni a projekt fájl nevét (pl.: minta.pjt), és a könyvtárát, ahol el fogjuk helyezni.
3. Ezek után az *Edit Project* párbeszédes doboz jeleneik meg, ahol a benne lévő Project Files ablakban már látjuk a minta[.hex] bejegyzést. (Itt adhatjuk meg az Include, Library és Linker Script fájlok útvonalát, változathatjuk meg a már beállított fejlesztési módot, kontroller típust, valamint a használt nyelvi eszközt (fordító, vagy assembler programot)).
4. Ha rákattintunk a minta[.hex] bejegyzésre, megjelenik a NODE Properties párbeszédes ablak, ahol megnézhetjük illetve átállíthatjuk a nyelvi eszköz beállított alapértelmezéseit:

5.5.1 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**

5.5.1.1 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

5.5.1.2 MPASM UNIVERZÁLIS ASSEMBLER : INCLUDE DIREKTÍVA

- Speciális funkcójú 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
```


5.5.1.3 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 "JOE",0 ; Hello szöveg tárolása

Ami azonos a következővel:

```
RETLW 'J'
RETLW 'O'
RETLW 'E'
RETLW 0
```


5.5.1.4 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 programirást tesz lehetővé
Szimulátoros hibakeresés

IF - Ezzel kezdődika feltételes blokk
Szintakszis: IF <expr>
ENDIF - a feltételes blokk vége ENDIF

5.5.2.1 MPLAB-SIM I.

- 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
```


5.5.2.2 MPLAB-SIM II.

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

- Perifériamodulok szimulálásakor hasznos
- Register Stimulus új regiszterértéket tölt a regiszterekbe, mikor egy adott címet elértünk
- 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.
```

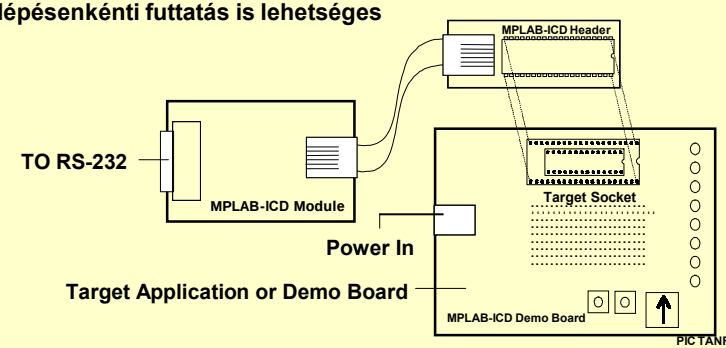

5.5.2.3 MPLAB-SIM III.

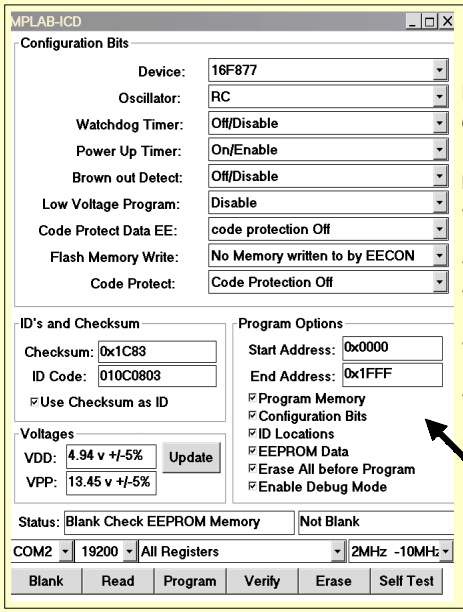
- A nyomkövetés során fájlban tárolhatjuk el az utasításokat és a regiszterek állapotát
- A fájlt a szimulátor hozza létre és többek között tartalmazza:
  - Lépésszám
  - A stopper által mutatott időpontot
  - Megváltozott regiszter értékeket
- A nyomkövetés adatai a képernyőn fognak megjelenni és egy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

Példaprogram: ADVSTIM.ASM

5.6.1 ICD=IN-CIRCUIT DEBUGGER

- Az ICD az MPLAB-ba van illesztve
- a program letölthető, és teljes sebességgel végrehajtható
- egy töréspont helyezhető el
- választható a töréspont alatt a perifériák felfüggesztése
- töréspont esetén:
  - a 0x1F00 címen kezdődő debug program végrehajtódik
  - a RAM vagy SFR-ek tartalma letölthető az MPLAB-ba
- lépésenkénti futtatás is lehetséges





5.6.2 MPLAB-ICD KEZELÉSE

Kell: MPLAB V4.11 vagy nagyobb

Options-Development Mode:

MPLAB-ICD Debugger

Programírás: Open vagy New Project

- **Reset vector címen 'NOP'-nak kell lenni**
- **1 verem szintet felhasznál**
- **Felhasználja a program memória utolsó 256 szavát, (1F00h-1FFFh)**
- **6 bájt adatmemóriahelyet (70h, 1EBh - 1EFh) használ**
- **továbbá két SFR-ben:**
  - **RB6 és RB7 lábakat, ezért**
  - **csak a PORTB (006h) & TRISB (086h) 0-5 bitje használható**

Magyarázat szóban !!!

6. PROGRAMOZÁS: ARCHITEKTÚRA (ISMÉTLÉS)

- **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>**

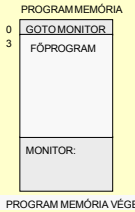
5.6.3 LETÖLTŐ (DOWNLOADER)

A PIC16F87x családnál nem csak az adatmemória, hanem a 14 bites programmemória is programmal írható, és így az eredeti program módosítható. Ehhez a memória nem használt területén el kell helyezni egy valamilyen perifériális eszközt (soros, I2C busz, SPI) kezelő hexa letöltő programot, amely képes a periférián érkező INTEL hexa programfájlt fogadni, és a benne lévő programszavakat a flash program memóriába írni. A monitorprogram:

Parancs	Bemenet	Kimenet	Magyarázat
I	XX	YY	XX című fájlregiszter tartalma (YY) a képernyőre
O	XX YY	----	XX című fájlregiszterbe YY hex tartalom bevitele
D	----	----	A PC-ről töltött hexa fájl memóriába írása és futtatása

Itt XX és YY hexa formátumú billentyűzetről beírt illetve kijelzett számok.

Első lépésként, a már kész monitorprogramot egy PIC16F87X memóriájába írjuk egy programozó (vagy ICD) segítségével. A program a memóriában a következő módon helyezkedik el:




```
ORG     0X0000   ; PROCESSZOR
                 ; RESET VEKTOR
MOVLW   0X018    ;HOSSZÚ
MOVWF   PCLATH   ;UGRÁS A MONITOR
GOTO    MONITOR  ; PROGRAMRA
START_PRG        ;3-MAS CÍM,
                 ; PROGRAMKEZDET
```


6.1PROGRAMOZÁS: 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          ; Bank 1 Indirekt regiszterek
CBLOCK 0x20                        SWSTACKSTART  EQU  0xA0
       COUNTLO
       COUNTHI
       SWSTACKPTR
       ...
ENDC
```

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
      DECFFSR,F                 ; SW verem mutató csökkent.
      MOVF      INDF,W          ; Adat W-be
      RETURN                    ; POP vége
```

6.2 PROGRAMOZÁS: TÁBLAKEZELÉS

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
IF ((Table&&0xFF00)!=(TableEnd-1 && 0xFF00))
ERROR  „A tábla átlépte a 256 szavas laphatárt”
ENDIF
```

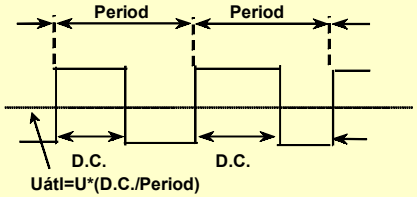
Ugrótáblák

- 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
```

Példa: ADVJUMP.ASM

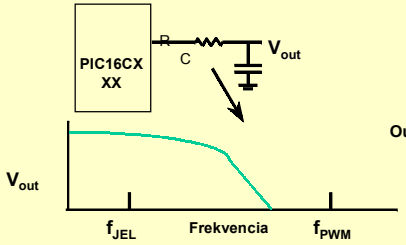
6.3 PROGRAMOZÁS: PWM



CCP PWM mód

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

Példaprogram: 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
```


6.4 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
```


6.5 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

6.6 PROGRAMOZÁS: TASZK KEZELÉS MEGSZAKÍTÁSSAL

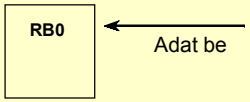
- TMR0 felhasználása periodikus megszakításra
- A következő taszk számontartása
- Az időszeletelés felhasználható a soros kommunikációra, stb.
- Két megvalósítási lehetőség:
  - Taszk mutató növelése a megszakítási programban (ISR-Interrupt Service Routine)
    - A taszk végrehajtása az ISR-ben

  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

6.7 PROGRAMOZÁS: ADATBEVITEL AZ I/O FORGATÁSSAL

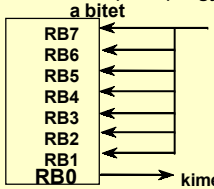
- Az adatbemenet az I/O portok LSB (bit 0) vagy az MSB (bit 7) kivezetése
- RRF utasítás (LSB) vagy RLF utasítás (MSB) a bemenet értékét a carry-be tolja
- Ugyanezen RRF vagy RLF utasítások a biteket egy fájl regiszterbe tolják




```
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
```


6.8 PROGRAMOZÁS: SOROS VONAL, SPI

- 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 vezeté-kes 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

6.9 PROGRAMOZÁS: 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
- 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: először a RAM minta feltöltése, utána a WDT reset végrehajtása

Szubrutin számlálás

- Két számláló: szubrutin hívó és végrehajtó számláló
- Hívó számláló inkrementálása minden rutinhívásknor
- 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

6.10 PROGRAMOZÁS:UTASÍTÁSSZÁM CSÖKKENTÉSE

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

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

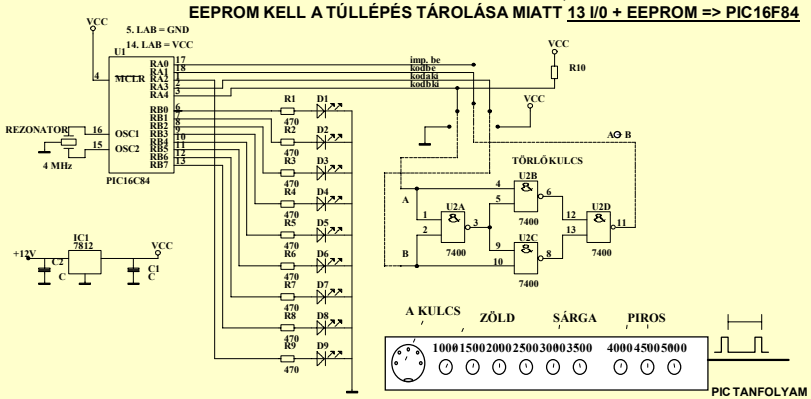
- Egy bit átvitel két regiszter között (REGA-ból REGB-be)
- REGB bit előzetes beállítása
- REGA bit-je tesztelése alapján a bitet esetleg megváltoztatjuk
- Portok esetén, két ciklus hosszúságú váltás (glitch) lehetséges!!!

```
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
```


7.1.1 EGY MEGOLDOTT FELADAT I.

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 => PIC16F84



6.11 PROGRAMOZÁS: UTASÍTÁSSZÁM CSÖKKENTÉSE

- 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 idő szink. miatt
kimkesz
```

- Regiszterek hasonlítása
- A két folytatás egy GOTO-val

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

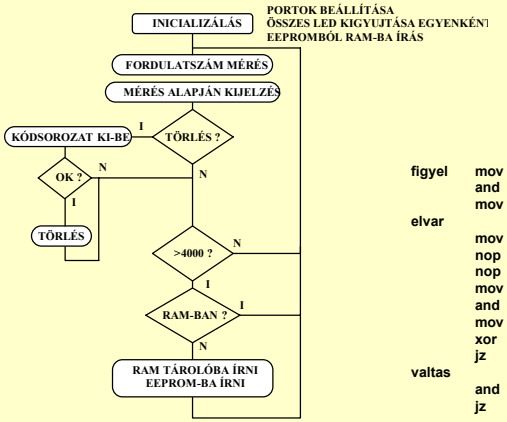
További ötletek a programok tanulmányozása alapján!

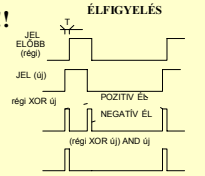
7.1.2 EGY MEGOLDOTT FELADAT II.

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 !!!






```
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
```

7.1.3 EGY MEGOLDOTT FELADAT III.

FORDULATONKÉNT EGY IMPULZUS

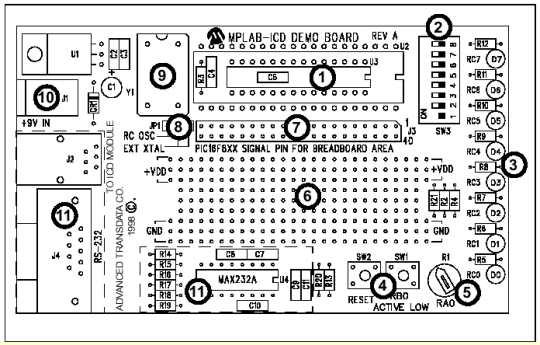
fordulatszám (ford/perc)	periódusidő *4 (msec)		fszam sec-ként	3sec alatt
500	120	480	8.3	25
1000	60	240	16.6	50
1500	40	160	24.9	75
2000	30	120	33.3	100
2500	24	92	41.7	125
3000	20	80	50	150
3500	17.1	69	58.3	175
4000	15	60	66.7	200
4500	13.3	53	75	225
5000	12	48	83.3	250

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 REGISTRS
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        ; 1 SEC. ELTELT?
        GOTO    EX67            ;IF NOT, EXIT
        CLRF    INT_COUNT       ;IF YES, CLEAR
        dec     sec
EX67
        CALL SIRWS      ;"POP" W AND STATUS REGISTERS
        RETFIE          ;ENAB. GLOBAL INT AND RETURN
```


7.2.2 MPLAB-ICD DEMO BOARD

1. EMULÁTOR FÉSZEK
2. LEDEKET BEKAPCSOLÓ DIP SOR
3. LEDEK A C PORTON
4. NYOMÓGOMBOK: RESET+PB0-RA
5. POTMÉTER RA0 BEMENETEN (ANALÓG)
6. SZABAD TERÜLET
7. PIC16F87X KIVEZETÉSEI EGY CSATLAKOZÓN
8. RC OSZCILLÁTOR (KB. 2 MHz) KIVÁLASZTÓ ÁTKÖTÉS
9. KÜLSŐ KRISTÁLY FOGLALATA
10. 9 V 0.75A TÁPEGYSÉG CSATLAKOZÓJA, A KÖZEPE A POZITÍV
11. RS232 SOROS VONAL KIALKÍTÁSÁHOZ

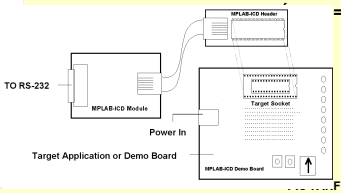


Az 1. jelű helyre természetesen egy, az ICD-vel beprogramozott példányt is be tudunk dugni, és így azt a tokot tudjuk élesben kipróbálni

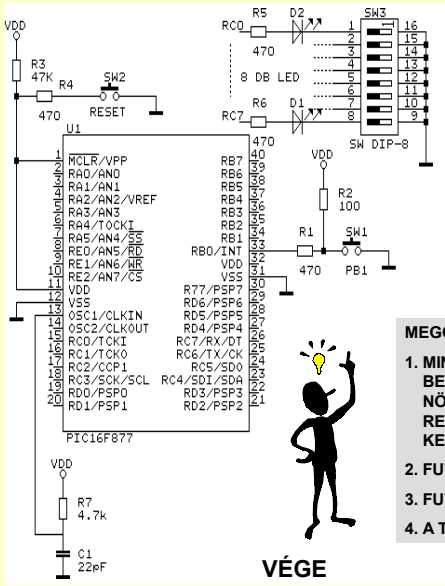
7.2.1 MPLAB-ICD HASZNÁLATA: A PROGRAM

```
;*****************************************************************
;*  DPRIM877.ASM
;*****************************************************************
;*  DRK
;*  16 AUGUST 1999
;*  Assembled with MPASM V2.20
;*****************************************************************
;*  AZ RB0-RA KOTOTT GOMB NYOMOGATASAVAL LEHET A PORTC-RE
;*  KOTOTT LEDEK KIJELZESET BINARISAN LEPTETNI
;*  A  panelen lévő SW3 DIP kapcsoló mindegyikének
;*  ON állásban kell lennie.
;*****************************************************************

        list p=16f877

        ; Include fájl, ha kell a könyvtár beállítása
        include "p16f877.inc"


        ; Start at the reset vector
        org     0x000
        nop                     ;KELL A DEBUG MIATT!!!
Start
        banksel TRISC
        clrf    TRISC           ;PORTC mind kimenet
        banksel PORTC
        clrf    PORTC           ;Clear PORTC
uj      incf    PORTC

WaitPus
        btfsc   PORTB,0         ;megnyomtuk?
        goto    WaitPus
w2
        btfss   PORTB,0         ;elengedtuk?
        goto    w2

        goto    uj
        end
```

- Reset vector címen 'NOP'-nak kell lenni
- 1 verem szintet felhasznál
- Felhasználja a program memória utolsó 256 szavát, (1F00h-1FFFh)
- 6 bájt adatmemóriahelyet (70h, 1EBh - 1EFh) használ
- továbbá két SFR-ben:
  - RB6 és RB7 lábakat, ezért
  - csak a PORTB (006h) & TRISB (086h) 0-5 bitje használható



7.2.3 MPLAB-ICD HASZNÁLATA: AZ ÁRAMKÖR



VÉGE

MEGOLDANDÓ FELADATOK:

1. MINDEN GOMBNYOMÁS HATÁSÁRA EGY BELSŐ REGISZTER TARTALMA EGYENKÉNT NÖVEKSZIK (BINÁRIS SZÁMLÁLÁS) ÉS A REGISZTER TARTALMA A 8 LED KIJELZŐRE KERÜL.
2. FUTÓFÉNY SZOFTVER IDŐZÍTÉSSEL.
3. FUTÓFÉNY MEGSZAKÍTÁSOS IDŐZÍTÉSSEL
4. A TANFOLYAMVEZETŐ ÖTLETEI….

7.2.4 ASM PROGRAM FELÉPÍTÉSE

```
AZONOSITÓ RÉSZ
        HOZZÁNYÚLÁST ÉS MÓDOSÍTÁST MUTATÓ SORSZÁM
        VERZIÓSZÁM
        UTOLSÓ MODOSÍTÁS DÁTUMA
        FÁJLNÉV
        SZERZŐ
        CÉG
        A PROGRAM RÖVID LEIRÁSA
        A PROGRAM TÖRTÉNETE DÁTUMOZVA
        A PROGRAM RÉSZLETESEBB LEÍRÁSA ÉS SPECIFIKÁCIÓJA
DEFINICIÓK, DEKLARÁCIÓK
        PROCESSZOR TÍPUS, KONFIGURÁCIÓ, INCLUDE FÁJLOK
        I/O PORTOK
                BEMENETEK
                KIMENETEK
        ALLANDOK
        VALTOZOK
        BITES VALTOZOK
        MAKROK
PROGRAM KEZDETE
MEGSZAKITASVEKTOR
        MEGSZAKITAS AZONOSITAS, FELDOLGOZAS
        REGISZTEREK VISSZAALLITASA
FOPROGRAM KEZDETE
        INICIALIZALASOK
                PORTOK
                REGISZTEREK
                BITEK
                EGYEB
        TENYLEGES FOPROGRAM KOD
SZUBRUTINOK
        MI A FELADATA
        BEMENET
        KIMENET
        HASZNALT REGISZTEREK
        KOD
PROGRAM VÉGE
```


7.2.5 FUTÓFÉNY SW IDŐZÍTÉSSEL I.

```
;#1        MINDEN FAJLHOZ-NYULASKOR 1-EL NOVELNI!!!
;VER: 1.0 DATE: 00.01.18 UTOLSO MODOSITAS
;FAJLNEV: FFENYSW.ASM NOTE:
;IRTA:    DR. KONYA LASZLO KONYA@NOVSERV.OBUDA.KANDO.HU
;CEG:     KKMF
;------------------[ MI EZ? ]-----------------------------------
;FUTOFENY SW IDOZITESSEL
;A  panelen lévő SW3 DIP kapcsoló mindegyikének
;ON állásban kell lennie.
;------------------[ TORTENET IDE KERUL MINDEN INFORMACIO ]---------
;00.01.12 ELSO VALTOZAT
;
;------------------[ SPECIFIKACIOK, KIEGESZITESEK ]-----------------
;
;==================[ D E F I N I C I O K ] ==========================
;
;------------------[ PROCESSZOR + KONFIGURACIO + INCLUDE ]----------
;
          LIST      P=16F877 ;PROCESSZOR DEFINICIO
          #INCLUDE  <P16F877.INC> ;A SZUKSEGES INCLUDE FAJL

            CONFIG  _CP_OFF &  _WDT_OFF &  _BODEN_ON &  _PWRTE_ON &
        _RC_OSC & _WRT_ENABLE_ON &  _LVP_ON &  _DEBUG_ON &  _CPD_OFF
;
;------------------[ I/O PORTOK ]-----------------------------------
;PORTC A LED PORT
;
;------------------[ ALLANDOK ]-------------------------------------
;
;------------------[ VALTOZOK ]-------------------------------------
TIMER1    EQU       20H
TIMER2    EQU       21H
SEG       EQU       22H
;------------------[ BITES VÁLTOZOK ]-------------------------------
;
;------------------[ MAKRÓK ]---------------------------------------
;
;==================[ P R O G R A M  K E Z D E T E ]=================
;
          ORG       0X000     ;PROCESSZOR RESET VEKTOR
          NOP                 ;DEBUG MIATT
          CLRF      PCLATH    ;PAGE BITEK BIZTOS TORLESE
          GOTO      INI_MAIN  ;UGRAS A PROGRAM KEZDETERE
;
;==================[ M E G S Z A K I T A S V E K T O R ]============
;NEM HASZNÁLJUK!
;         ORG       0X004     ;EZ A 4H CIM
```


7.2.5 FUTÓFÉNY SW IDŐZÍTÉSSEL II.

```
;==================[ F Ő P R O G R A M  K E Z D E T E ]============
;
;------------------[ INICIALIZALASOK ]------------------------------
INI_MAIN

INI_PORT
          BANKSEL TRISC
          CLRF      TRISC     ;PORTC mind kimenet
          BANKSEL PORTC
INI_REG
          CLRF      SEG       ;SEG=0
INI_BITS
          BSF       SEG,0     ;SEG='00000001'B
INI_MISC
;------------------[ PROGRAM TENYLEGES KEZDETE ]--------------------
MAIN
                              ;ADOTT REGISZTER ROTÁLÁSA CARRY KIHAGYÁSÁVAL
          RRF       SEG,W     ;ELŐTTE:SEG=76543210 UTÁNA:W=C7654321 CY=.0
          RRF       SEG,F     ;SEG=07654321 CY=CY
          MOVF      SEG,W     ;W=SEG

          MOVWF     PORTC     ;PORTC-N VANNAK A LEDEK
          CALL      WAIT      ;SW KÉSLELTETÉS
          GOTO      MAIN      ;UJRA
;==================[ S Z U B R U T I N O K ]=======================
WAIT
          CLRF      TIMER1
          CLRF      TIMER2
DLY1
          NOP
          DECFSZ    TIMER1,F
          GOTO      DLY1
          DECFSZ    TIMER2,F
          GOTO      DLY1
          RETURN
          END                 ;PROGRAM VÉGE
```

FELADAT: NYOMOGOMBRA LÉPKEDJEN!!!

7.2.6 FUTÓFÉNY MEGSZAKITÁSSAL I.

```
;         PORTC     =>        LEDEK
;------------------[ ALLANDOK ]-------------------------------------
;
RTCBE     EQU       D'105'              ;EZT KELL A SZAMLALOBA TENNI
;
;------------------[ VALTOZOK ]-------------------------------------
W_TMP               EQU     0X20        ; VARIABLE USED FOR CONTEXT SAVING
STA_TMP             EQU     0X21        ; VARIABLE USED FOR CONTEXT SAVING
SEG                 EQU       0X22
          CBLOCK    0X23
;IT-BEN HASZNALTAK
MS1                           ;1MS SZAMLALO
MS10                          ;10MSEC SZAMLALO
SEC                           ;1SEC SZAMLALO
SEGEDRI                       ;SEGÉDREGISZTER AZ IT-BEN
          ENDC

;------------------[ BITES VÁLTOZÓK ]-------------------------------
;
#DEFINE   CY        STATUS,C
;
;------------------[ MAKRÓK ]---------------------------------------
;
JNZ       MACRO ADDR                  ;JUMP IF ZERO BIT IS NOT SET
          BTFSS 3,2
          GOTO ADDR
          ENDM

MOV       MACRO     REG1,REG2 ;R2->R1
          MOVF      REG2,W
          MOVWF     REG1
          ENDM

SUBRK     MACRO     REG,KONST
          MOVLW     KONST
          SUBWF     REG,F
          ENDM

;==================[ P R O G R A M  K E Z D E T E ]=================
;
          ORG       0X000               ;PROCESSZOR RESET VEKTOR
          NOP                           ;A DEBUG MIATT
          CLRF      PCLATH              ;PAGE BITEK BIZTOS TÖRLÉSE
          GOTO      INI_MAIN  ;UGRÁS A PROGRAM KEZDETÉRE
```

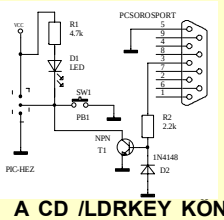
A PROGRAM ELEJE A HELYHIÁNY MIATT LEMARADT, LÁSD ELŐZŐ

7.2.6 FUTÓFÉNY MEGSZAKITÁSSAL II.

```
;==================[ M E G S Z A K I T A S V E K T O R ]============
;
          ORG       0X004               ;EZ A 4H CIM
          MOVWF     W_TMP               ;W MENTÉSE
          MOVF      STATUS,W  ;
          BCF     STATUS,RP0  ;BIZTOS REGISTER BANK SET TO 0
          MOVWF     STA_TMP             ;STATUS MENT,SE
;
;------------------[ MEGSZAKITAS AZONOSITAS, FELDOLGOZAS ]----------
;
          CALL      RTCINT
;
;------------------[ REGISZTEREK VISSZAÁLLÍTÁSA ]--------------------
          BCF     STATUS,RP0  ; ENSURE FILE REGISTER BANK SET TO 0
          MOVF      STA_TMP,W ;STATuS VISSZAALLITAS
          MOVWF     STATUS
          SWAPF     W_TMP,F
          SWAPF     W_TMP,W             ;W VISSZAÁLLÍTÁSA
          RETFIE                        ;VISSZATÉRÉS IT-BŐL
;
;==================[ F Ő P R O G R A M  K E Z D E T E ]============
;
;------------------[ INICIALIZALASOK ]-----------------------------
INI_MAIN

INI_PORT
          BANKSEL   TRISC
          CLRF      TRISC               ;PORTC mind kimenet
          BANKSEL   PORTC
          CLRF      PORTC               ;Clear PORTC
INI_REG
          CLRF      SEG
INI_BITS
          BSF       SEG,0               ;SEG=00000001
INI_MISC

INI_RTC
          BSF       STATUS,RP0          ;BANK1
          MOVLW     B'11010001'         ;ELEMEZZÜK!
          MOVWF     OPTION_REG
          BSF       INTCON,5            ;TMR0 IT ENG
          BSF       INTCON,GIE          ;GENERAL IT ENG
          BCF       STATUS,RP0          ;BANK0
```


7.2.6 FUTÓFÉNY MEGSZAKITÁSSAL III.

```
MAIN

UJ        GOTO      UJ                  ;JÓ KIS FŐPROGRAM, MONDHATOM
;
;==================[ S Z U B R U T I N O K ]=======================
;TMR0 MEGSZAKÍTÁS
;4 MHZ ÓRA 1 MIKRO 1 LÉPÉS 4-ES ELŐOSZTÓ
RTCINT
          MOVLW     RTCBE               ;INNEN INDUL A SZÁMLÁLÓ
          MOVWF     TMR0
          BCF       INTCON,2            ;CLEAR RTCC INTERRUPT FLAG
          INCF      MS1,F               ;1MSEC NÖVEL

          MOV       SEGEDRI,MS1         ;
          SUBRK     SEGEDRI,D'10'       ;
          JNZ       ITVEG               ;
MSEC10
          CLRF      MS1                 ;
          INCF      MS10,F              ;10 MSEC NOVEL
          MOV       SEGEDRI,MS10
          SUBRK     SEGEDRI,D'100'
          JNZ       ITVEG
SEC1
                                        ;ADOTT REGISZTER ROTÁLÁSA CARRY KIHAGYÁSÁVAL
          RRF       SEG,W               ;ELÖTTE:SEG=76543210 UTÁNA:W=C7654321 CY=.0
          RRF       SEG,F               ;SEG=07654321 CY=CY
          MOVF      SEG,W               ;W=SEG
          MOVWF     PORTC               ;PORTC-N VANNAK A LEDEK

          CLRF      MS10
          INCF      SEC,F               ;1 SEC NOVEL
ITVEG
          RETURN                        ;ENABLE GLOBAL INT AND RETURN
          END ;PROGRAM VÉGE
```

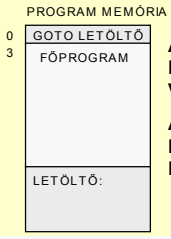



7.3.1 LDR-KEY HASZNÁLATA I.

A PROGRAMLETÖLTŐ KULCS A SAJÁT PROGRAM-MEMÓRIÁJUK ÍRÁSÁRA ÉS OLVASÁSÁRA KÉPES TÍPUSOK PROGRAMFEJLESZTÉSÉRE ILLETVE FRISSÍTÉSÉRE HASZNÁLHATÓ: PIC16F870/871, PIC16F872, 16F873/874/876/877.

A CD /LDRKEY KONYVTÁRÁBAN TALÁLHATÓ A LDRKEY.ASM FÁJLT - A LETÖLTŐ-LÁBAT MÓDOSÍTVA – KELL AZ MPLAB SEGÍTSÉGÉVEL LEFORDÍTANI, ÉS A PIC TOKBA ÍRNI. EZEK UTÁN MÁR A LEÍRT MÓDON KÉPES MŰKÖDNI. AZ ALKALMAZÁS HEXA FÁJLJÁT A SZINTÉN AZ EBBEN A KÖNYVTÁRBAN LÉVŐ DOWNLDR.EXE (WINDOWS ALATT FUTÓ) PROGRAM SEGÍTSÉGÉVEL KELL LETÖLTENI. A LETÖLTENI KÍVÁNT PROGRAM MEGÍRÁSAKOR FIGYELEMBE VENNI:

A PROGRAMMEMÓRIA ELSŐ HÁROM ILL. UTOLSÓ 256 SZAVA A MONITORPROGRAM SZÁMÁRA VAN FENNTARTVA. (PL.:PIC16F877 ESETÉN EZ A 0X0000-0X0002 ILL. 0X1F00-0X1FFF CÍMEKET JELENTI) HA MÉGIS OLYAN HEX FÁJLT PRÓBÁLNÁNK LETÖLTENI, AMELY EZT A TERÜLETET HASZNÁLJA A MONITORPROGRAM HIBÁT JELEZ ÉS AZ ÉRINTETT TERÜLET FELÜLÍRÁSA NÉLKÜL BEFEJEZI A LETÖLTÉST. (EZZEL MEGAKADÁLYOZVA A MONITORPROGRAM MŰKÖDÉSKÉPTELENNÉ TÉTELÉT)



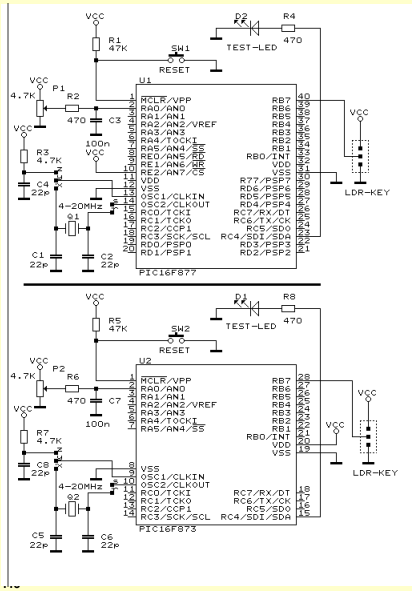
7.3.2 LDR-KEY HASZNÁLATA II.

A FELHASZNÁLÓ PROGRAMJÁNAK A 0X0003 CÍMEN KELL KEZDŐDNIE, MERT A MONITORPROGRAM IDE ADJA ÁT A VEZÉRLÉST INDULÁSKOR.

A MEMÓRIA 0 CÍMÉN LÉVŐ UTASÍTÁS ELUGRÁS A LETÖLTŐ-PROGRAMRA, RESET UTÁN AZONNAL EZ HAJTÓDIK VÉGRE. A KEZDŐ PROGRAMRÉSZLET:

```
          ORG       0X0000    ;PROCESSZOR RESET VEKTOR
                    MOVLW     0X018
                    MOVWF     PCLATH    ;HOSSZÚ
                    GOTO      MONITOR   ;UGRÁS A MONITOR PROGRAMRA
          START_PRG                     ;3-MAS CÍM, PROGRAMKEZDET
```

A MONITORPROGRAM NEM HASZNÁL MEGSZAKÍTÁSOKAT, AZOK MEGKÖTÉSEK NÉLKÜL ALKALMAZHATÓAK A SAJÁT PROGRAMBAN. AZ INTERRUPT VEKTOR IS VÁLTOZATLANUL A 0X0004 CÍMRE MUTAT.
AZ ELŐRE FELPROGRAMOZOTT ÁRAMKÖRÖK KONFIGURÁCIÓS SZAVA IS BE LETT ÉGETVE, AZOKAT A LETÖLTŐ KULCS SEGÍTSÉGÉVEL NEM LEHET MÓDOSÍTANI.
AZ ELŐRE FELPROGRAMOZOTT ÁRAMKÖRÖK AZ RB7 PORTOT HASZNÁLJÁK A LETÖLTÉSRE!
A RESET FOLYAMAT ALATT A LETÖLTŐ KULCS GOMBJÁNAK LENYOMVA TARTÁSÁVAL LEHET A LETÖLTŐ PROGRAMOT AKTIVIZÁLNI, MELYET A LED KÉTSZERI FELVILLANÁSA JELEZ.
A HIBÁS LETÖLTÉST A LED FOLYAMATOS VILÁGÍTÁSA JELZI, MÍG A SIKERES PROGRAMOZÁS UTÁN A LED HÁRMAT VILLAN, MAJD AZ ÁRAMKÖR RESET-ELÉSÉVEL INDÍTHATÓ A FELHASZNÁLÓI PROGRAM.



7.3.3 MINIÁRAMKÖR AZ LDRKEY-HEZ

KÉT 40 ILLETVE 28 LÁBÚ TOKOKHOZ MEGRAJZOLT ÁRAMKÖR A KEZDŐ KAPCSOLÁS AZ LDR-KEY HASZNÁLATÁHOZ.

HÁROM „PERIFÉRIÁJA” VAN:

EGY LED:

RC4 PONTRA KAPCSOLÓDIK

- **EGY NYOMÓGOMB:**

LDR-KEY NYOMÓGOMBJA

- **EGY ANALÓG BEMENET** (*A POTI*)

CÉLSZERŰ EGY KIS „CSUPALYUK” PANELRE PIC-NEK EGY 40(28) LÁBÚ IC FOGLALATOT FELFORRASZTANI, ÉS ENNEK A KIVEZETÉSEIHEZ FORRASZSZUK A TÖBBI ALKATRÉSZT. AZ ÁRAMKÖRI RAJZON A PIC BEKÖTÉSE A FELÜLNÉZETI TOKOZÁSI KIALAKÍTÁSÁVAL EGYEZIK MEG.

7.3.4 MINIPROGRAMOK

AZ ELKÉSZÍTETT ÁRAMKÖR KIPRÓBÁLÁSÁHOZ (ÉS A GYAKORLÁSHOZ) HÁROM PROGRAM KÉSZÜLT.

<u>VILLOG.ASM</u>: AZ RC PORT 4. LÁBÁRA KÖTÖTT LED-ET KAPCSOLJUK PERIODIKUSAN KI ÉS BE, SZOFTVER IDŐZÍTÉST HASZNÁLVA.

<u>LEPTET.ASM</u>: A LETÖLTŐ KULCS NYOMÓGOMBJÁT NYOMKODVA A AZ RC PORT 4. LÁBÁRA KÖTÖTT LED-ET KAPCSOLJUK KI ÉS BE.

<u>VALTVIL.ASM</u>: A PROGRAM AZ A/D EGYSÉG 0. CSATORNÁJÁN LÉVŐ FESZÜLTSÉGET ALAKÍTJA ÁT DIGITÁLIS ÉRTÉKKÉ, ÉS EZ A SW IDŐZÍTÉST ÁLLÍTJA. ÍGY A PORTC, 4. BITJÉNEK VILLOGÁSI SEBESSÉGE FOG VÁLTOZNI

EZT A HÁROM PROGRAMOT AZ MPLAB PROGRAMMAL LEFORDÍTVA, AZ LDR-KEY-EL LETÖLTHETJÜK. A KÉNYELMESEBBEKNEK MEGTALÁLHATÓK A KÖZVETLENÜL LETÖLTHETŐ VILLOG.HEX, LEPTET.HEX VALTVIL.HEX FÁJLOK IS.

LETÖLTÉSI HELY: http://alpha1.obuda.kando.hu/~konya

7.3.5 VALTVIL.ASM PROGRAM

```
;A PROGRAM AZ A/D EGYSÉG 0. CSATORNÁJÁN LÉVŐ FESZÜLTSÉGET ALAKÍTJA ÁT DIGITÁLIS ÉRTÉKKÉ, ÉS EZ A SW IDŐZÍTÉST ÁLLÍTJA. IGY A PORTC,4 BITJÉNEK VILLOGÁSI SEBESSÉGE  VÁLTOZIK
            LIST P=16F877
            INCLUDE "P16F877.INC"
TIMER1      EQU     20H
TIMER2      EQU     21H
SEG         EQU     22H
;START AT THE RESET VECTOR
            ORG     0X003
START
            BANKSEL TRISC
            CLRF    TRISC
            MOVLW   B'00001110'
;LEFT JUSTIFY,1 ANALOG CHANNEL
            MOVWF   ADCON1
            ;VDD AND VSS REFERENCES
            BANKSEL PORTC
            MOVLW   B'01000001'
            ;FOSC/8, A/D ENABLED
            MOVWF   ADCON0
MAIN
;---------------[ A/D KONVERZIÓ ]------------------------
            BSF     ADCON0,GO
            ;START A/D KONVERZIÓ
WAITAD      BTFSS   PIR1,ADIF
            ;VÁRAKOZÁS, MÍG KÉSZ NEM LESZ
            GOTO    WAITAD
```

```
;---------------[ VILLOGTATO RUTIN ]--------------------
UJRA
            BSF     PORTC,4
            ;PORTC,4 BITJE 1
            CALL    WAIT        ;VÁRAKOZÁS
            BCF     PORTC,4
            ;PORTC,4 BITJE 0
            CALL    WAIT        ;VÁRAKOZÁS
            GOTO    MAIN        ;CIKLUSBAN
;---------------[ SZOFTVER KÉSLELTETÉS ]----------
WAIT        MOVFW   ADRESH
            BTFSC   STATUS,Z
            INCF    ADRESH,F
            MOVWF   TIMER2
            CLRF    TIMER1
DLY1
            NOP
            NOP
            NOP
            NOP
            DECFSZ  TIMER1,F
            GOTO    DLY1
            DECFSZ  TIMER2,F
            GOTO    DLY1
            RETURN
            END
```