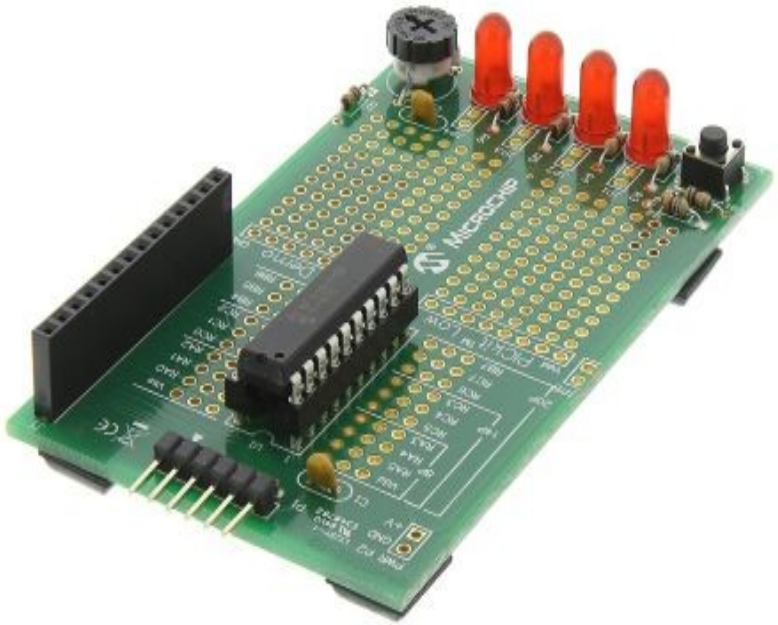
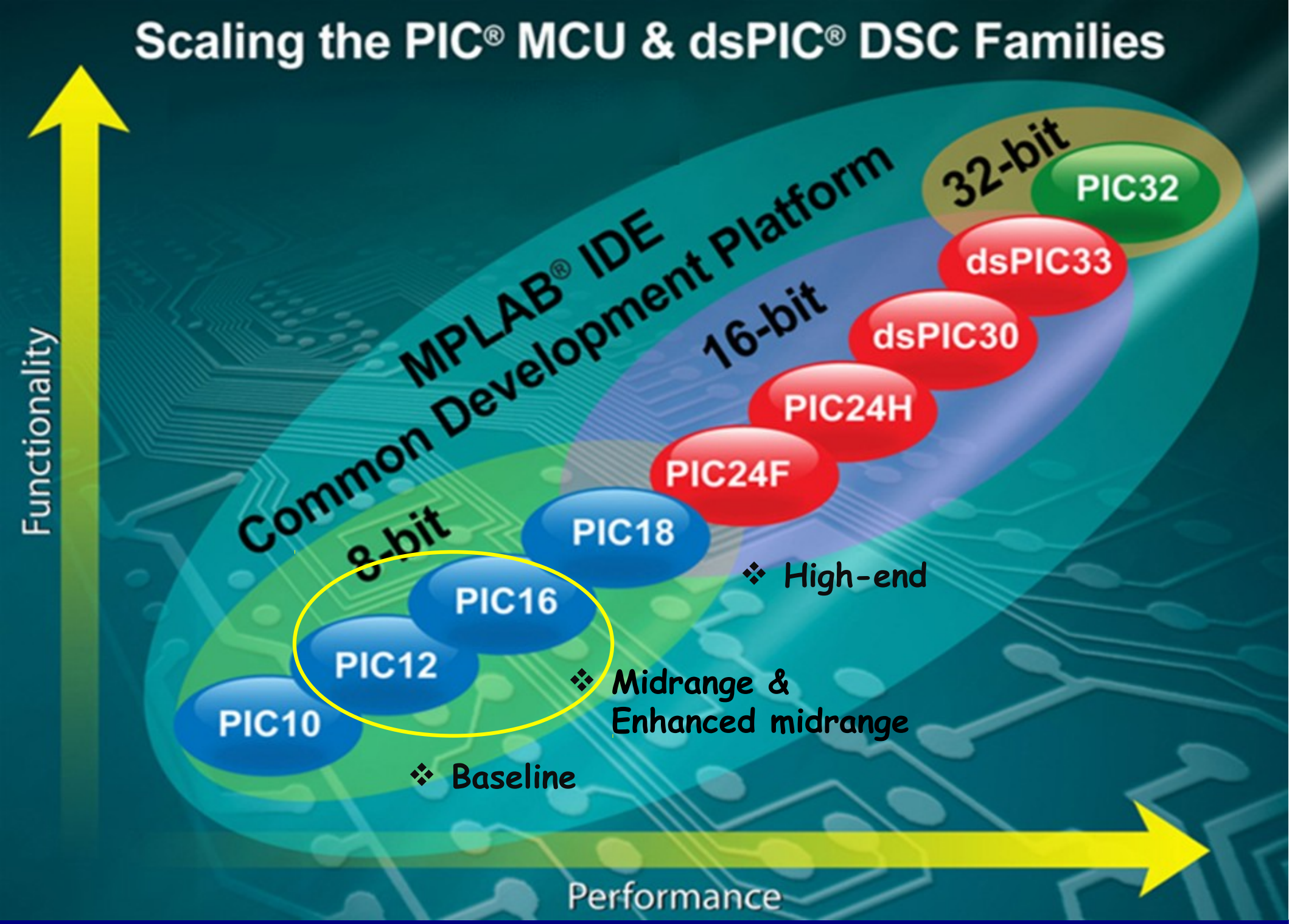
Vegyes témakörök



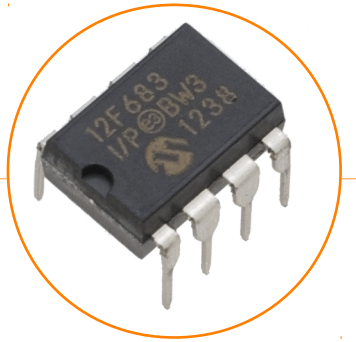
6. Microchip PIC mikrovezérlők programozása MPLAB X környezetben

Microchip PIC mikrovezérlők

- A Megtestesülés Plébánia Hobbielektronika foglalkozásain a 2016/2017-es évadban már tartottunk előadást a Microchip középkategóriás mikrovezérlőiről 2017. április 27-én, így az ott elhangzottakat nem ismételném meg
- Minden gyártó esetében célszerű megismerni azokat az a fejlesztéshez használható eszközöket, amelyeket a gyártó támogat
- Az újabb mikrovezérlő típusokat a régebbi eszközök nem kezelik
- Ha segítségre szorulunk, legnagyobb valószínűséggel ahhoz a fejlesztőeszközhöz kaphatunk segítséget, amit a legtöbben használnak („fősodor”)
- A Microchip esetében a „fősodor” jelenleg az MPLAB X integrált fejlesztői környezetet, az XC fordítókat és a PICkit3 (vállalati felhasználás esetén ICD3 vagy ICD4) programozó/hibavadász eszközöket jeleni.


Scaling the PIC® MCU & dsPIC® DSC Families
Functionality
Performance
MPLAB® IDE
Common Development Platform
8-bit
16-bit
32-bit
PIC10
PIC12
PIC16
PIC18
PIC24F
PIC24H
dsPIC30
dsPIC33
PIC32
❖ Baseline
❖ Midrange &
Enhanced midrange
❖ High-end

Egy konkrét típus



PIC12F683

Harvard felépítésű (14 bites utasítás, 8 bites adatút)
RISC processzor (35 utasítás)
VCC = 2.0 – 5.5 V

MPLAB X és a C fordítók

- A **Microchip MPLAB IDE** (integrált fejlesztői környezet) támogatja a kódszerkesztést, a nyomkövetést és a programletöltést a Microchip 8-, 16- és 32-bites PIC mikrovezérlőibe. Beépítetten csak assembler fordítót tartalmaz.
- Az **MPLAB X** az MPLAB IDE legutóbbi változata, ami a nyíltforrású NetBeans platformon alapul és keresztplatformos fejlesztői környezet (Windows, Linux és Mac OS X).
- Az **MPLAB X** ingyenesen használható.
- Külön kell telepíteni hozzá a C fordítókat, amelyek ingyenes változata nem, vagy csak korlátozottan optimalizál:
  - MPLAB XC8 — C fordító a 8-bites PIC mikrovezérlőkhöz
  - MPLAB XC16 — C fordító a 16-bites PIC mikrovezérlőkhöz
  - MPLAB XC32 — C/C++ fordító a 32-bites PIC mikrovezérlőkhöz

Az MPLAB X beszerzése

Szoftver

- Az **MPLAB X** fejlesztői környezet a Microchip honlapjáról tölthető le. Link: www.microchip.com/mplab/mplab-x-ide
- E sorok írásakor a legfrissebb változat a ver 4.05 kiadás. A korábbi kiadások **Microchip szoftver archívumban** találhatók: www.microchip.com/development-tools/downloads-archive

Dokumentáció

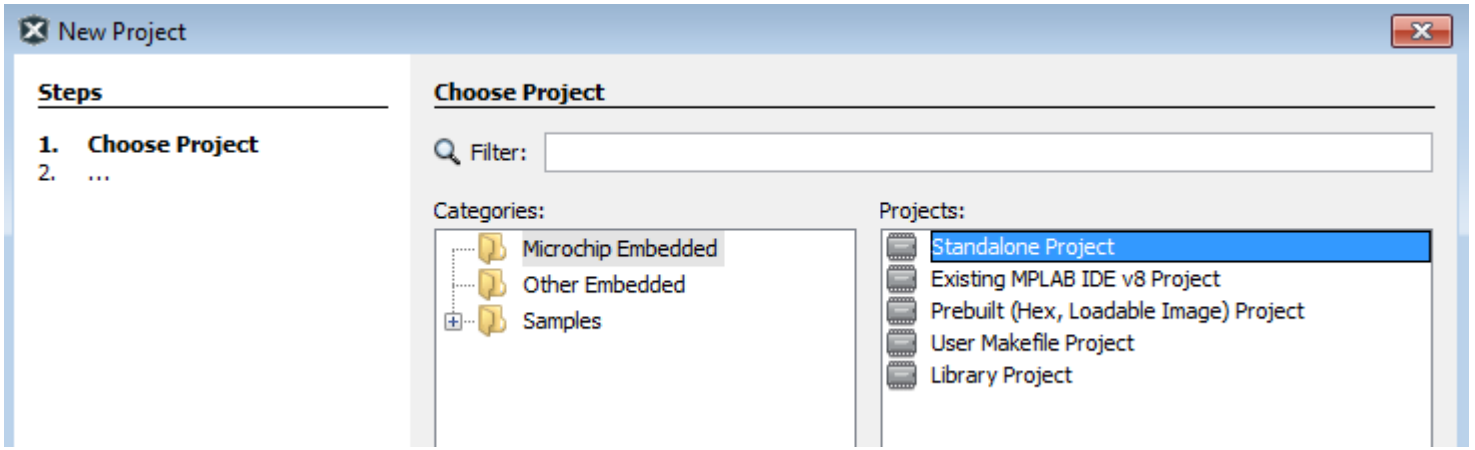
- MPLAB X IDE felhasználói kézikönyv (angol)
- A PICkit3 programozó használata MPLAB X környezetben
- PICkit3 programozó/hibakereső felhasználói kézikönyv (angol)
- Az MPLAB X IDE által támogatott eszközök listája

Az MPLAB X telepítése

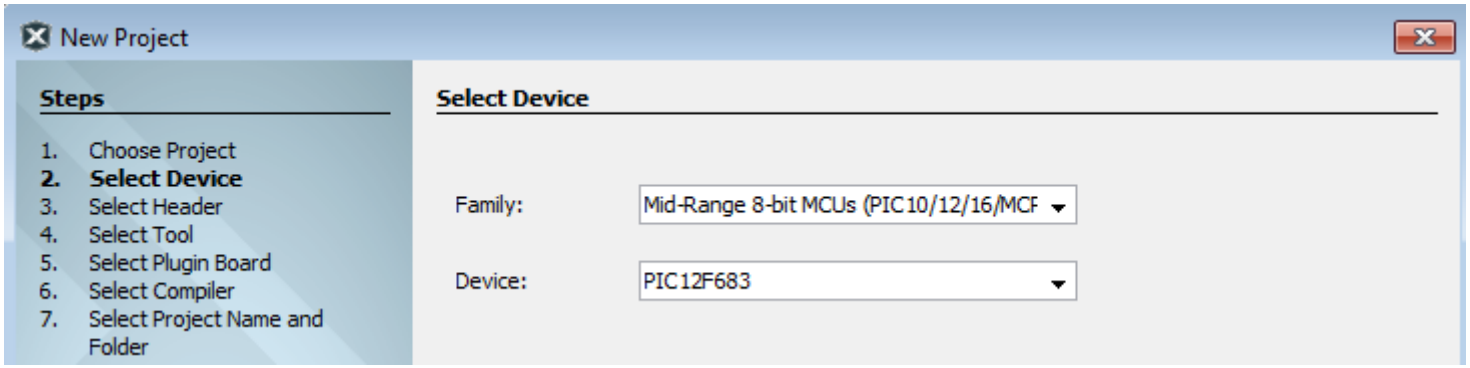
- A telepítési útmutató az MPLAB X felhasználói kézikönyvben található
- Amennyiben a gépünkön nincs megfelelő verziójú Java futtatói környezet (JRE) telepítve, úgy annak telepítésére is sor kerül az MPLAB X telepítésekor
- USB eszközmeghajtó telepítésére is szükség lehet az alábbi programletöltő/hibavadász eszközök esetében:
  - MPLAB REAL ICE in-circuit emulator
  - MPLAB ICD 3 in-circuit debugger
  - MPLAB PM3 device programmer
  - PIC32 Starter Kit
- A PICkit 2, PICkit 3 vagy az MPLAB Starter Kit-ek esetében nem kell eszközmeghajtót telepíteni (HID eszközként jelentkeznek be)

Új assembly projekt létrehozása

- **File → New project**
- **1. Choose projekt: Microchip Embedded → Standalone projekt** (önálló projekt) választása, kattintás a **Next** gombra

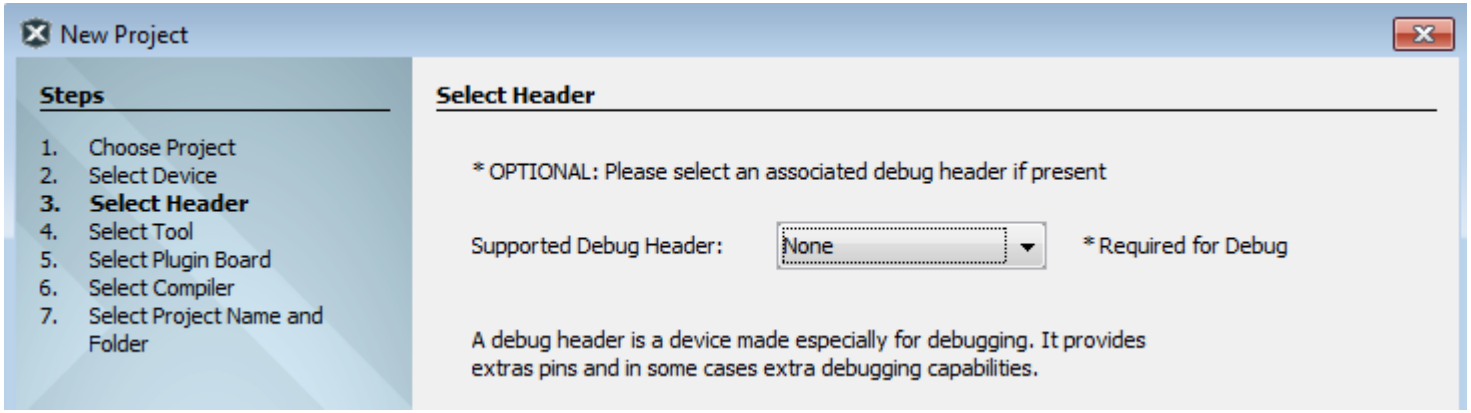


- **2. Select Device:** a **Mid-Range PIC12F683** típus kiválasztása, majd kattintás a **Next** gombra

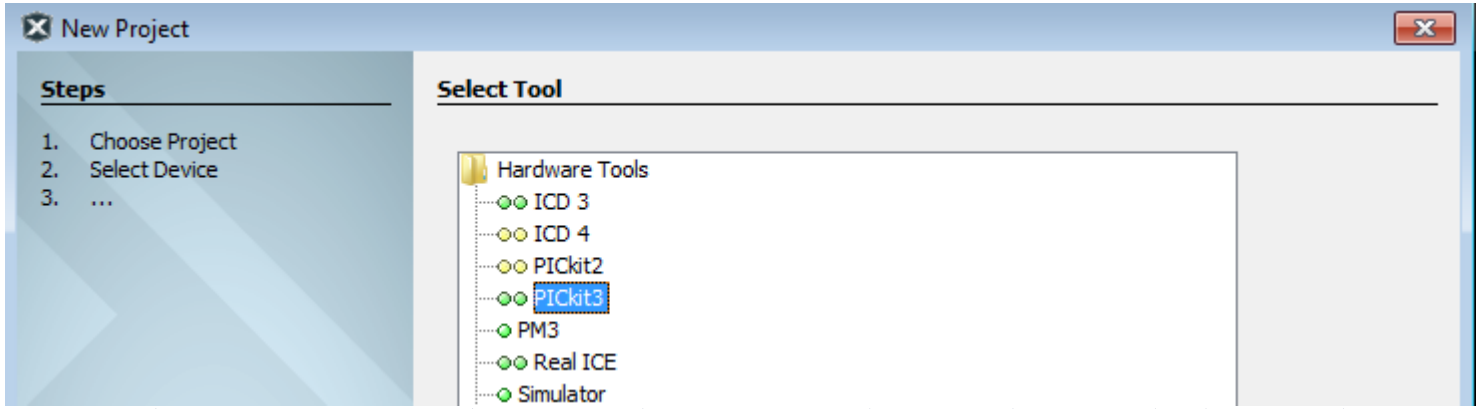


Új assembly projekt létrehozása

- **3. Select header:** Hardveres nyomkövetéshez kiegészítő kártya kellene. Nincs ilyenünk, tehát **None** választása, majd kattintás a **Next** gombra



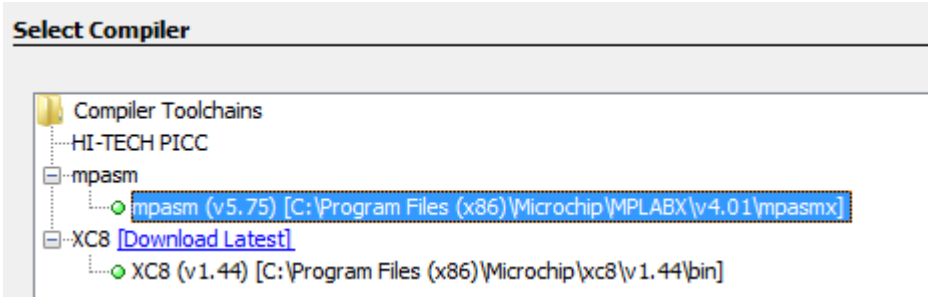
- **4. Select Tool:** a használni kívánt programletöltő (PICkit2 vagy PICkit3) kiválasztása, majd kattintás a Next gombra



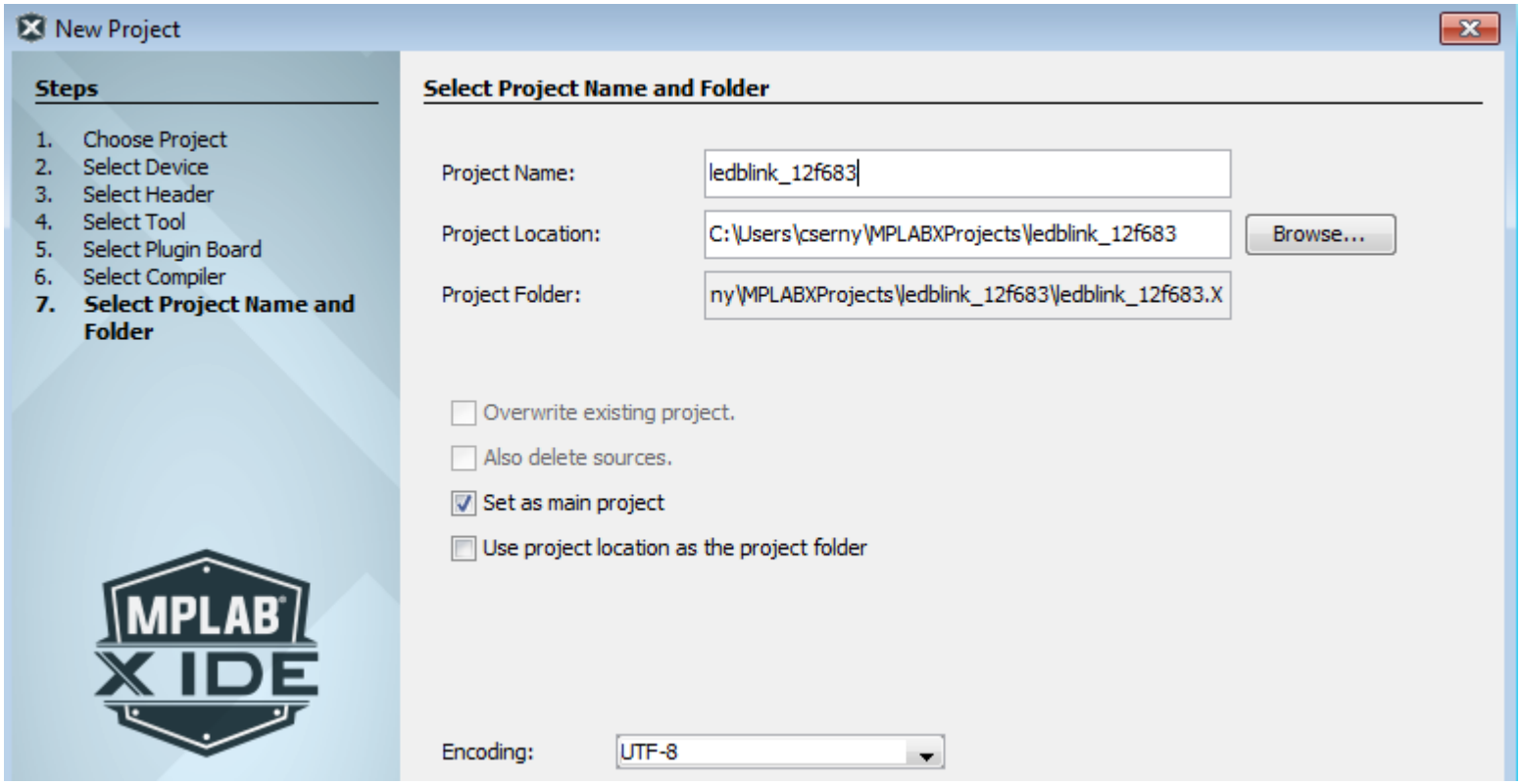
- **5. Select Plugin Board:** ez a lépés csak Real ICE hibavadász eszköz választása esetében jelenik meg!

Új assembly projekt létrehozása

- **6. Select Compiler:** a használni kívánt fordító (esetünkben **mpasm**) kiválasztása, majd kattintás a **Next** gombra

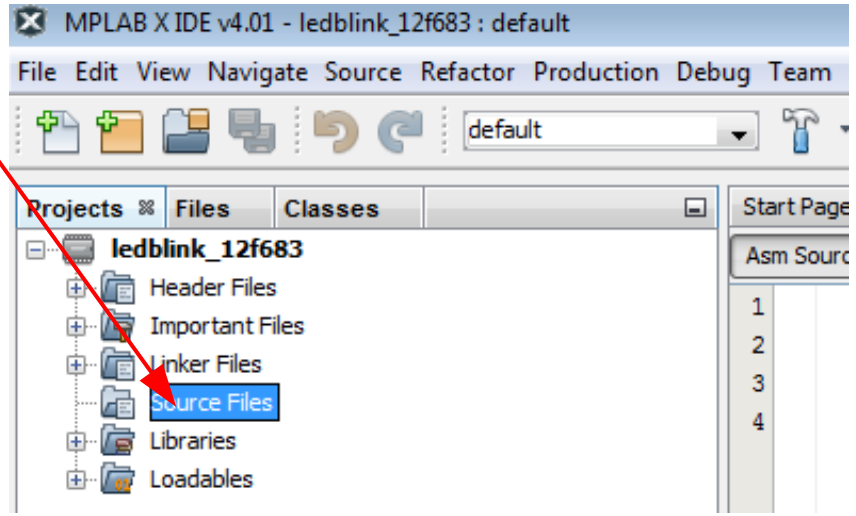


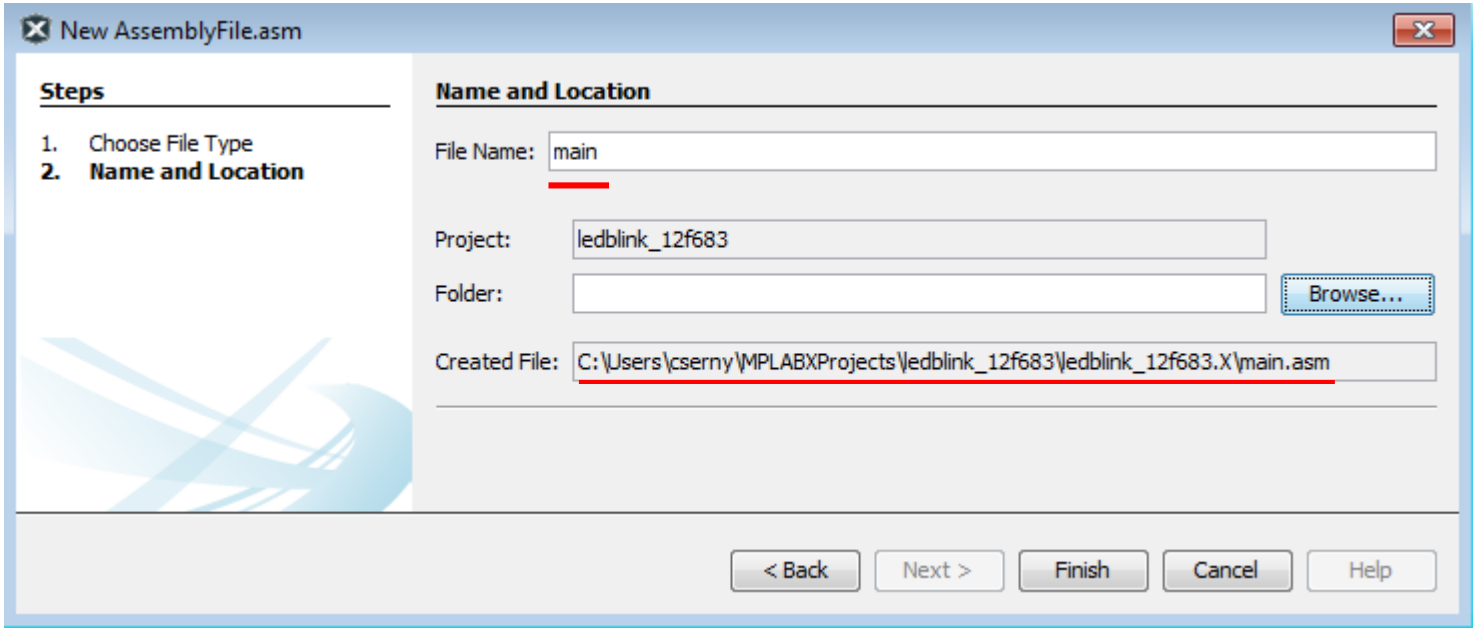
- **7. Select Project name and Folder:** adjuk meg a projekt nevét és mappáját, s állítsuk be a karakterek kódolását



Új forrásállomány hozzáadása

- A projekt ablakban jobb egérgombbal kattintsunk a Source files mappanévre!
- A felbukkanó menüben válasszuk a New ⟶ AssemblyFile.asm menüpontot!
- Adjuk meg az állomány nevét (pl. main) és az elhelyezési mappát (esetünkben megfelel az alapértelmezett mappa)



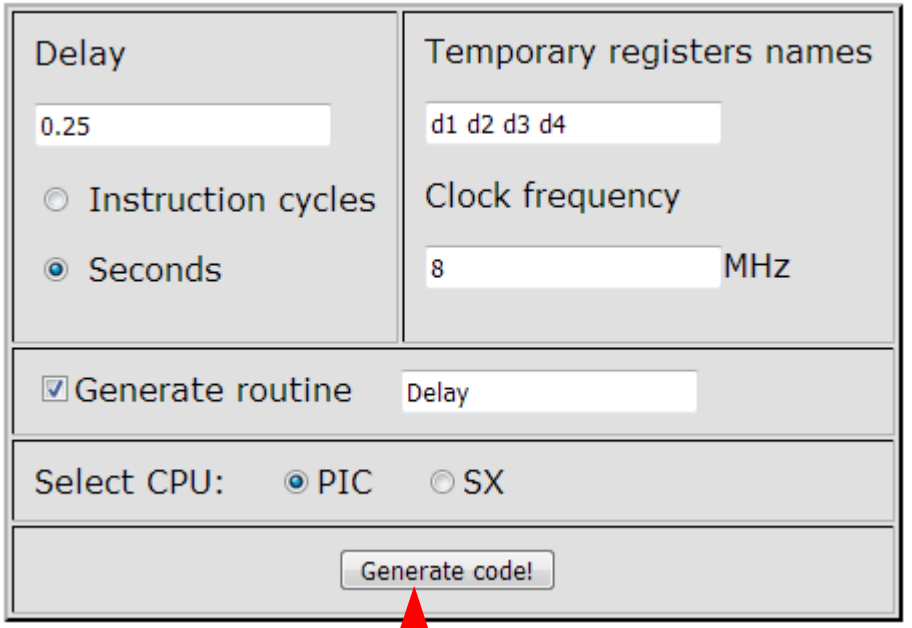


PIC12F683 LED villogtatás (main.asm)

```
    list P=12F683,ST=OFF              ; Symbol Table listázásának kikapcsolása
    errorlevel  -302                  ; Bankváltási figyelmeztetések tiltása

    #include "p12f683.inc"           ; processzor specifikus definíciók
        CBLOCK 0x20                   ; változók deklarálása
    d1
    d2
    d3
        ENDC

         ORG     0x000               ; processor reset vector
MAIN:
        BANKSEL CMCON0                ; Váltás Bank 0-ba
        MOVLW   b'00000111'           ; Analóg komparátor lekapcsolása
        MOVWF   CMCON0
        CLRF    GPIO
        BANKSEL TRISIO                ; Váltás Bank 1-be
        MOVLW   ~(1<<GP2)             ; (csak) GP2 legyen kimenet
        MOVWF   TRISIO
        CLRF ANSEL                    ; Digitális I/O engedélyezés
        MOVLW   b'01110001'           ; Fosc = 8MHz, IntOsc a rendszer óra
        MOVWF   OSCCON
        banksel GPIO                  ; Bankváltás 0-ba
LOOP:
        MOVLW   1<<GP2
        XORWF   GPIO,F                ; toggle LED
        CALL    Delay
        goto    LOOP
```

PIC12F683 LED villogtatás (main.asm)

```
;-----------------------------------------
;  Delay függvény: 0.25 s késleltetés
;  (8 MHz órajel frekvencia esetén)
;-----------------------------------------
Delay

    movlw   0x03
    movwf   d1
    movlw   0x18
    movwf   d2
    movlw   0x02
    movwf   d3
Delay_0
    decfsz  d1, f
    goto $+2
    decfsz  d2, f
    goto $+2
    decfsz  d3, f
    goto Delay_0
    goto $+1
    return
```



Kód generálás indítása

Az on-line használható, késleltető kódot generáló programot Nyikoláj Golovcsenkó írta PIC illetve SX mikrovezérlőkhöz.
Link: Microchip PIC Delay Code Generator

```
;-----------------------------------------------------------------------
    END                             ; 'program vége' direktíva
;-----------------------------------------------------------------------
```

PIC12F683 konfigurációs bitek

—	—	—	—	FCMEN	IESO	BOREN1	BOREN0
bit 15							bit 8

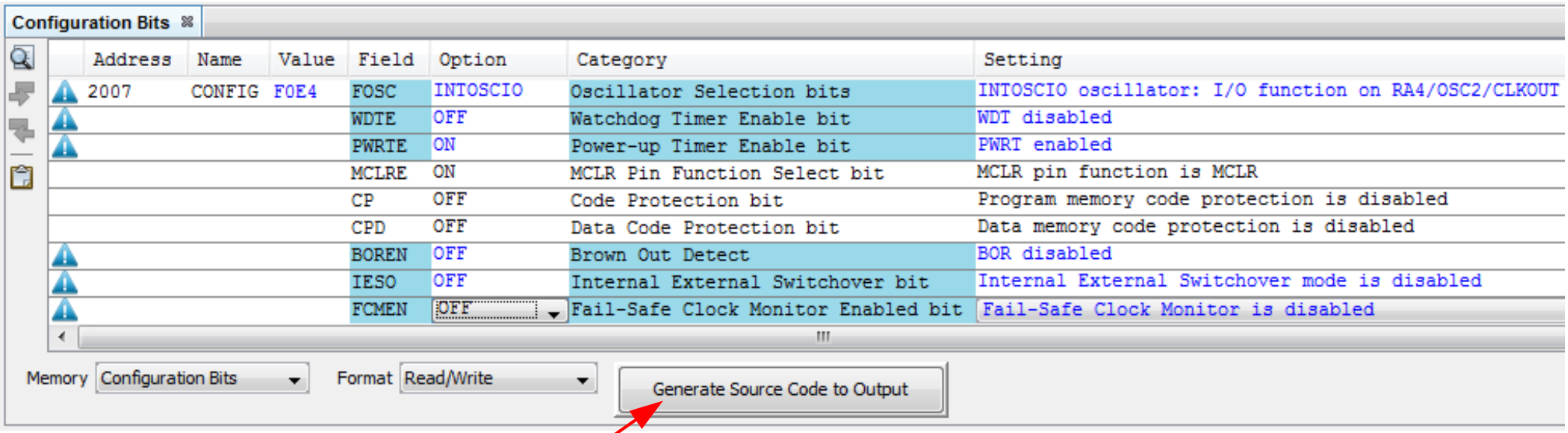
$\overline{\text{CPD}}$	$\overline{\text{CP}}$	MCLRE	$\overline{\text{PWRTE}}$	WDTE	FOSC2	FOSC1	FOSC0
bit 7							bit 0

- **FCMEN:** Fail-Safe Clock Monitor Enabled bit
- **IESO:** Internal External Switchover enable
- **BOREN<1:0>:** Brown-out Reset Selection
- **CPD:** Data Code Protection bit
- **CP:** Code Protection bit
- **MCLRE:** GP3/MCLR pin function select bit (MCLR láb RESET vagy I/O legyen)
- **PWRTE:** Power-up Timer Enable bit
- **WDTE:** Watchdog Timer Enable bit (Watchdog tiltás/engedélyezése)
- **FOSC<2:0>:** Oscillator Selection bits (órajelforrás választása)

Konfigurációs bitek:
A hardver „viselkedését” befolyásoló, menet közben nem változtatható beállítások

Konfigurációs bitek beállítása

- A **Production → Set Configuration Bits** menüpontot választva az **MPLAB X-ben** beállíthatjuk a konfigurációs biteket, majd forráskódot generálhatunk, s a main.asm állomány elejére másoljuk



- Kódgeneráláshoz kattintsunk a gombra!

```
;------------------------------------------------------------------
;  Konfigurációs bitek
;------------------------------------------------------------------
;   __config 0xF0E4
    __CONFIG _FOSC_INTOSCIO & _WDTE_OFF & _PWRTE_ON & _MCLRE_ON & _CP_OFF &
_CPD_OFF & _BOREN_OFF & _IESO_OFF & _FCMEN_OFF
```

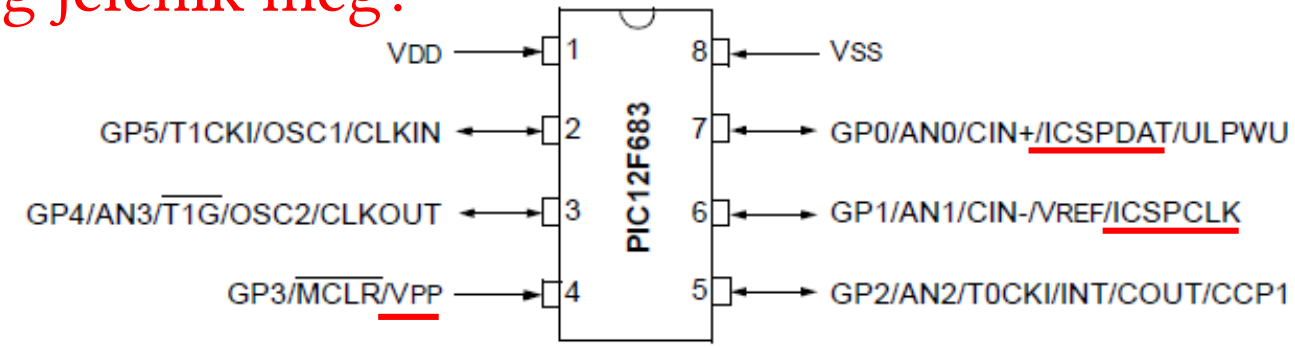
A fordítás menete

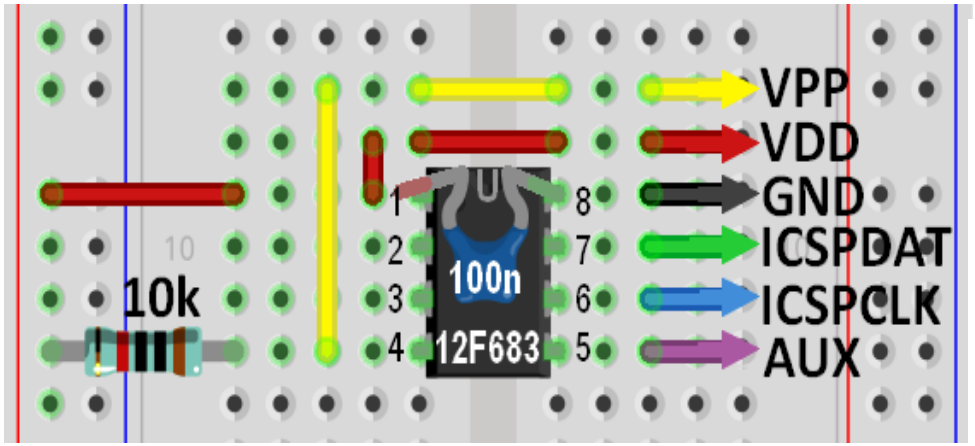
- A fordítás elindítására több lehetőségünk van:
  - **Production → Build Main Project** menüpont
  - Az **F11** gomb lenyomása
  - **Kalapács ikon**-ra kattintás
- Sikeres fordítás esetén ”Errors : 0” üzenetet kapunk

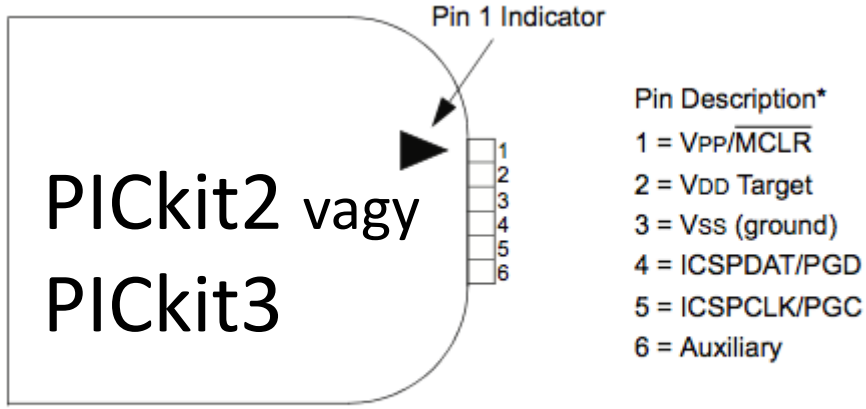
```
"C:\Program Files (x86)\Microchip\MPLABX\v4.01\mpasmx\mpasmx.exe" -q -p12f683
-l"build/default/production/main.lst" -e"build/default/production/main.err"
-o"build/default/production/main.o" "main.asm"

"C:\Program Files (x86)\Microchip\MPLABX\v4.01\mpasmx\mplink.exe"    -p12f683  -w
-m"dist/default/production/ledblink_12f683.X.production.map"   -z__MPLAB_BUILD=1
-odist/default/production/ledblink_12f683.X.production.cof  build/default/production/main.o


MPLINK 5.08, LINKER
Device Database Version 1.38
Copyright (c) 1998-2011 Microchip Technology Inc.
Errors    : 0

MP2HEX 5.08, COFF to HEX File Converter
Copyright (c) 1998-2011 Microchip Technology Inc.
Errors    : 0
```

Programletöltés

- A programot egy **PICkit2** vagy **PICkit3** programozóval töltjük le. A letöltés az **ICSPDAT** és **ICSPCLK** lábakon folyik.
- Az **AUX** kivezetést esetünkben nem használjuk.
- Figyelem! Programozás módban az **MCLR/VPP** lábon 10 – 13 V-os feszültség jelenik meg!

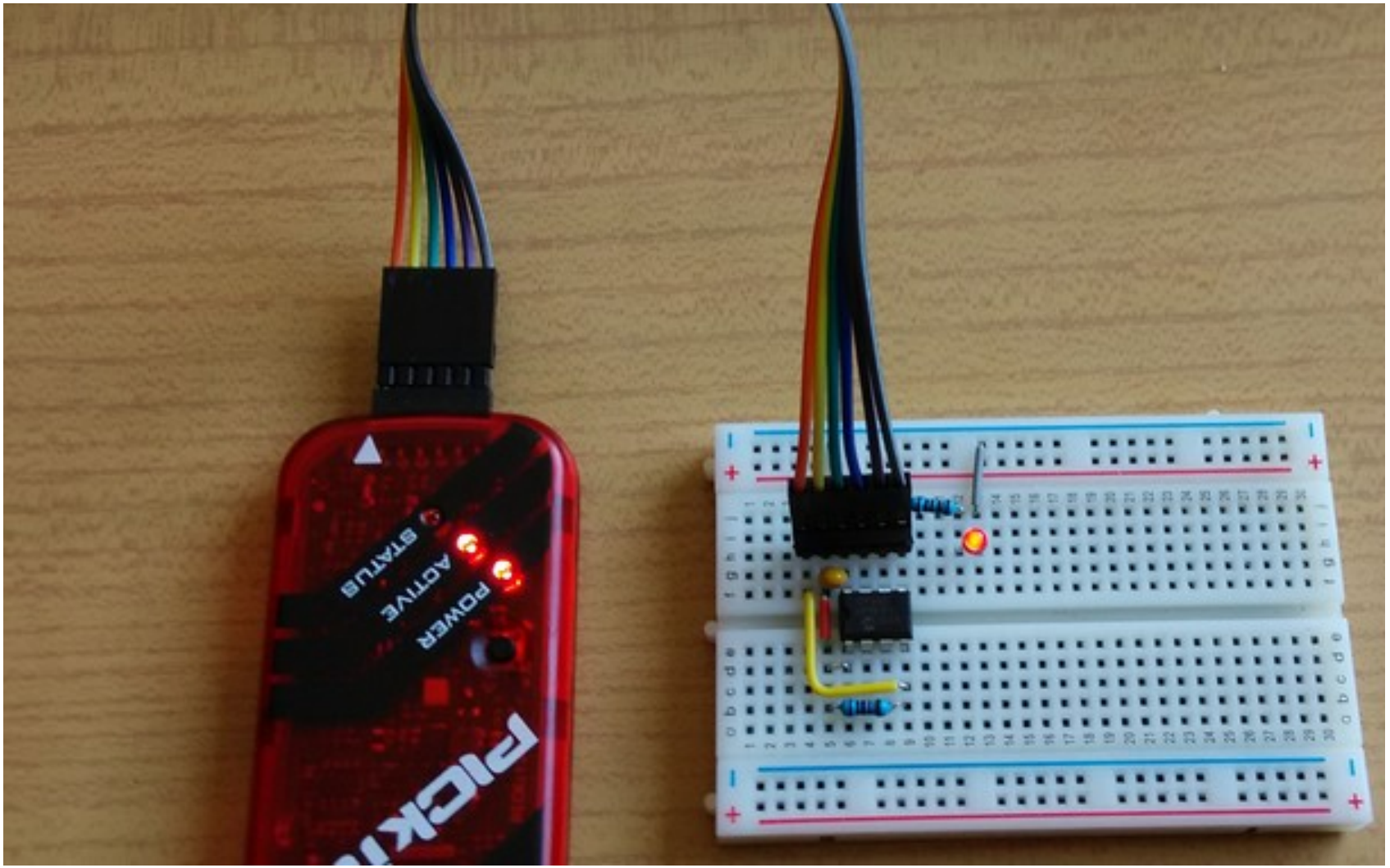






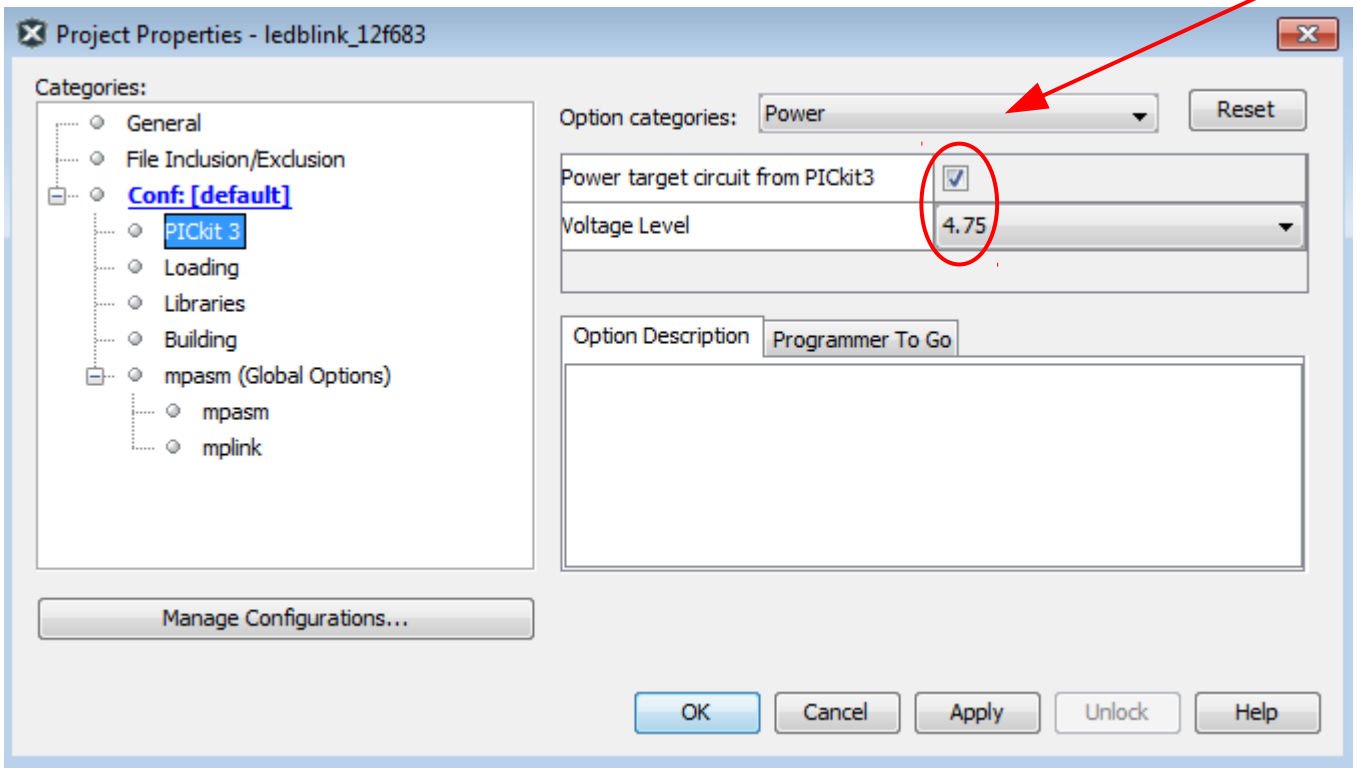
Programletöltés PICkit3 eszközzel

- Programletöltéshez kattintsunk a  gombra!



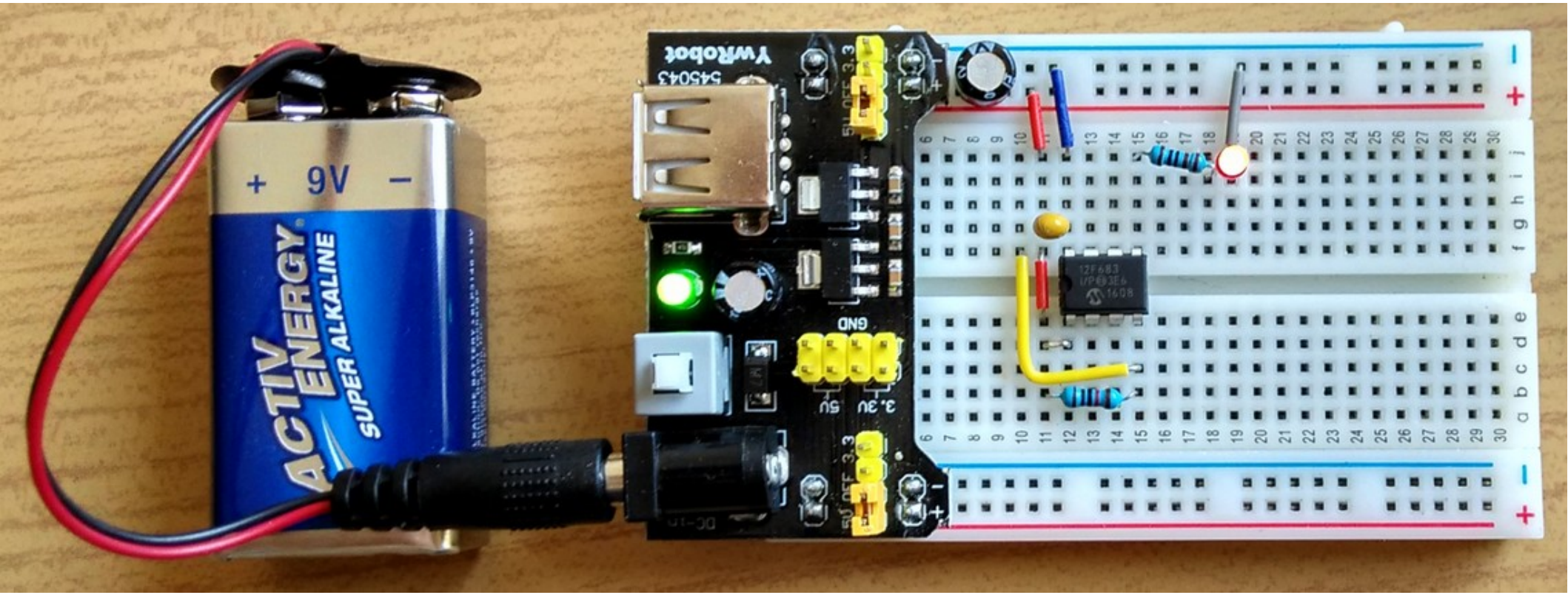
PICkit3 beállítása MPLAB X alatt

- Ha a mikrovezérlőt a **PICkit3**-ról akarjuk táplálni (max. 20 mA):
- **File → Project Configuration** menüpont választása
- A felbukkanó ablakban válasszuk ki a **PICkit3** szekcióban a **Power** kategóriát és végezzük el az alábbi beállítást



Tápellátás 9 V-os elemről

- Gondoskodnunk kell róla, hogy a PIC tápfeszültsége ne haladja meg a maximális 5,5 V-ot! Használjunk feszültségstabilizátort!
- A tápfeszültség és a GND sínek közé tegyünk szűrőkondenzátort! Esetünkben pl. egy 220 µF / 16V kondenzátort használunk.
- A mikrovezérlő VCC és GND lábai közé 100 nF kondenzátor kell!



A Microchip XC8 fordító

- A fordító ingyenes változata a Microchip honlapjáról tölthető le
  Link: www.microchip.com/mplab/compilers
- Ügyeljünk a telepítés sorrendjére: csak az MPLAB X IDE telepítése után fogjunk a fordító(k) telepítéséhez!

- A telepítési könyvtár a **C:\Program Files (x86)\Microchip\xc8** mappába kerül, itt nézhetünk utána a dokumentációnak és a fejléc állományoknak
- Minden program elején csatoljuk be az **xc.h** állományt!
  Ez automatikusan becsatolja az MPLAB környezetben beállított a mikrovezérlő specifikus fejléc állományát is.
- A PIC12F683 mikrovezérlőhöz az XC8 mintaprogramokat a Gooligum Electronics honlapjáról, az **Enhanced Mid-Range PIC tutorials** ingyenes tananyagból vettük.

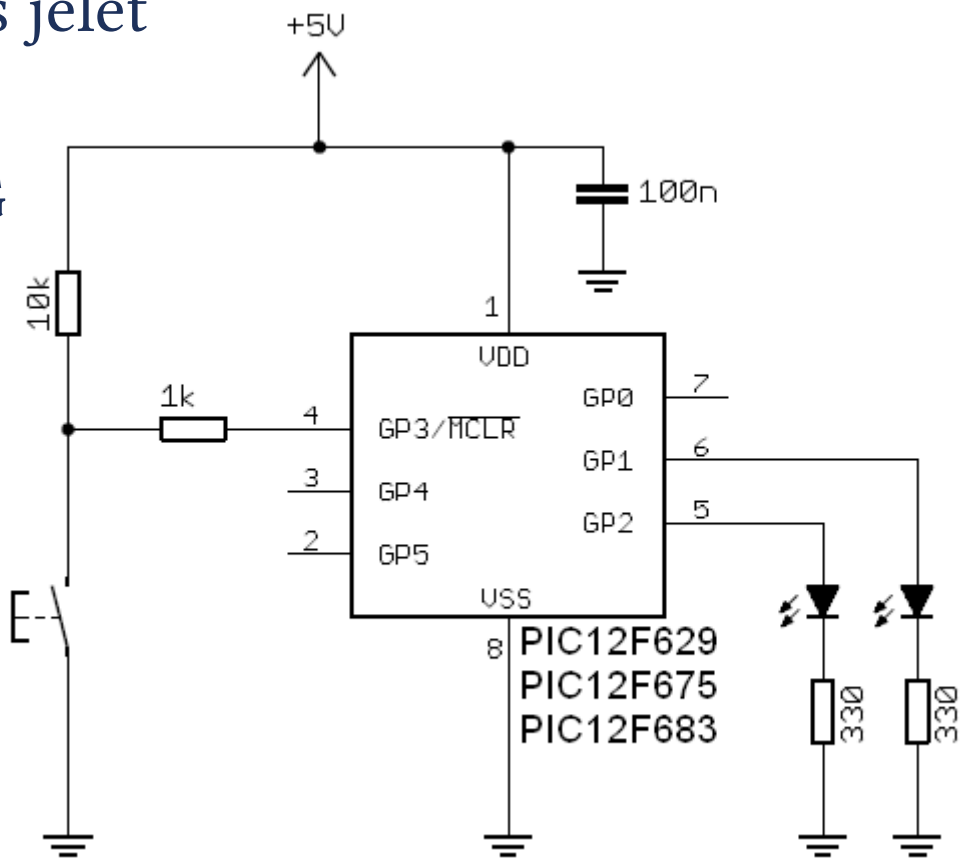
p12F683-ledflash-xc8

```
#include <xc.h>
#define _XTAL_FREQ  8000000     // Fosc frekvencia (Hz)

//--- KONFIGURÁCIÓS  BITEK ----
#pragma config FOSC = INTOSCIO  // Belső oszcillátor, I/O funkcióval)
#pragma config WDTE = OFF       // Watchdog Timer letiltva
#pragma config PWRTE = OFF      // Power-up Timer letiltva
#pragma config MCLRE = ON       // MCLR RESET funkció engedélyezve
#pragma config CP = OFF         // Flash memória kódvédelem letiltva
#pragma config CPD = OFF        // EEPROM memória adatvédelem letiltva)
#pragma config BOREN = OFF      // Brown-out Reset letiltva


//--- FŐPROGRAM -------------
void main() {
    //--- Inicializálás -------
    OSCCON = 0x70;              // Fosc legyen 8 MHz
    CMCON0 = 0x07;              // Analóg komparátor tiltás
    GPIO = 0;                   // GPIO kezdetben nulla legyen
    TRISIO = ~_GPIO_GP2_MASK;   // Csak GP2 legyen kimenet

    //--- Fő programhurok -----
    for (;;) {
        GPIO |= _GPIO_GP2_MASK; // LED (GP2) bekapcsolása
        __delay_ms(200);        // 200 ms várakozás
        GPIO &= ~_GPIO_GP2_MASK;// LED kikapcsolás (GP2 törlés))
        __delay_ms(800);        // 800 ms várakozás
    }                           // végtelenségig ismételjük...
}
```

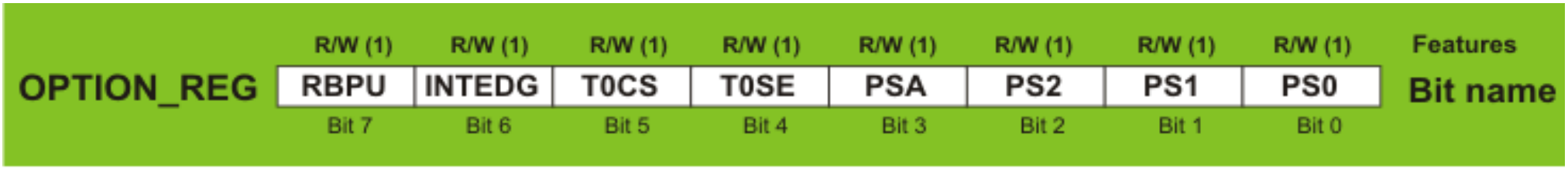
- LED villogtatás a GPIO2 kimeneten
- Tbe = 200 ms
- Tki = 800 ms
- Fosc = 8 MHz

p12f683-reaction-timer-cx8 projekt

- Reakcióidő mérése Timer0 segítségével: a GP2-re kötött LED felvillanásakor zárni kell a nyomógombot. Ha ez 200 ms-on belül sikerül, kigyullad a másik LED
- Fosc alapértelmezetten 4 MHz, Fcy = Fosc/4 = 1 MHz, ezt leosztjuk 32-szeresen, s ezt a 31 250 Hz-es jelet számlálja a Timer0 számláló
- Az előosztást az **OPTION_REG** regiszterben kell beállítani
- Amíg a 8-bites Timer0 250-ig számol, T = 250/31 250 = 8 ms idő telik el
- Ezeket a 8 ms-os „óraütéseket” szoftveresen számláljuk a reakcióidő méréséhez



Az OPTION_REG regiszter

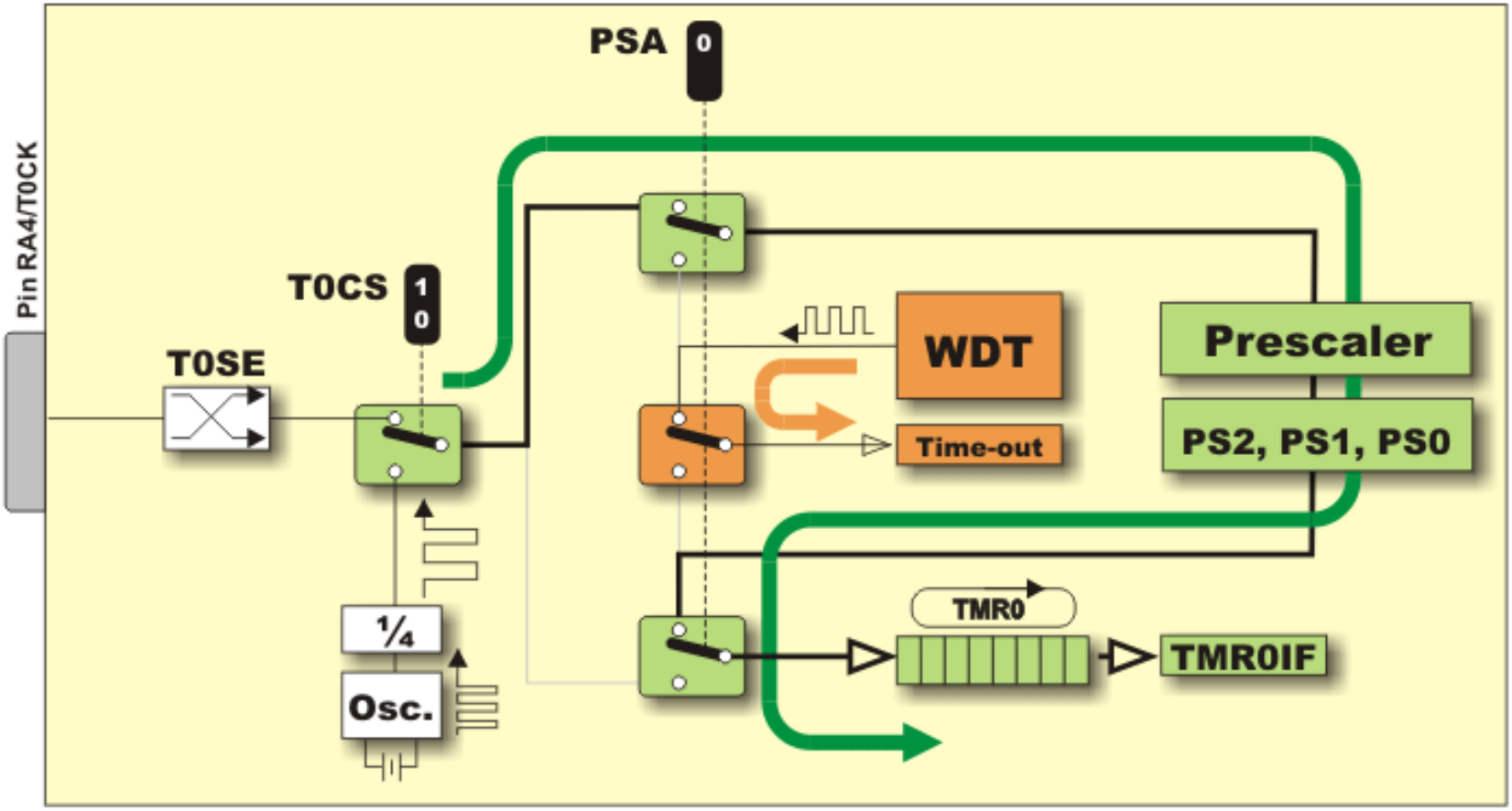


OPTION_REG	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	R/W (1)	Features
	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

- **RBPU** – belső felhúzások engedélyezése
- **INTEDG** – INT megszakításhoz élválasztás (0: lefutó élre, 1: felfutó élre)
- **T0CS** – Timer0 órajel választás (0: belső órajel (Fosc/4), 1: külső órajel)
- **T0SE** – Timer0 órajel élválasztás (0: felfutó, 1: lefutó élre számlál)
- **PSA** – Előszámláló hozzárendelés (0: TMR0 használja, 1: WDT használja)
- **PS[2:0]** – előosztási arány beállítása (2^{N+1} TMR0 esetén, illetve 2^{N} WDT esetén)

Timer0 előosztással

- Az előosztó vagylagosan kapcsolható Timer0-hoz vagy WDT-hez
- Timer0 8-bites számláló (0 – 255), túlcsorduláskor **TMR0IF** 1-be áll



p12f683-reaction-timer-cx8 (main.c)

```
#include <xc.h>
#include <stdint.h>
#define _XTAL_FREQ  4000000          // oscillator frequency for _delay()
#define START   GPIObits.GP2         // LEDs
#define SUCCESS GPIObits.GP1
#define BUTTON  GPIObits.GP3         // pushbutton
#define MAXRT   200                  // Maximum reaction time in ms

void main() {
    uint8_t     cnt_8ms;             // counter: increments every 8 ms
    TRISIO = 0b111001;               // configure GP1 and GP2 as outputs
    OPTION_REGbits.T0CS = 0;         // select timer mode
    OPTION_REGbits.PSA = 0;          // assign prescaler to Timer0
    OPTION_REGbits.PS = 0b100;       // prescale = 32 -> increment every 32 us
    for (;;)  {
        GPIO = 0;                    // initialyy both LEDs are off
        __delay_ms(2000);            // delay 2000 ms
        START = 1;                   // turn on start LED
        cnt_8ms = 0;                 // clear counter
        while (BUTTON == 1 && cnt_8ms < 1000/8) {
            TMR0 = 0;                // clear timer0
            while (TMR0 < 8000/32); // wait for 8 ms (32 us/tick)
            ++cnt_8ms;               // increment 8 ms counter
        }
        if (cnt_8ms < MAXRT/8)       // if time < max reaction time (8 ms/count)
            SUCCESS = 1;             //   turn on success LED
        __delay_ms(1000);            // delay 1000 ms
    }
}
```

Megszakítások kezelése

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE	PEIE	T0IE	INTE	GPIE	T0IF	INTF	GPIF

- Programmegszakítások használatához globálisan engedélyeznünk kell a megszakítást (INTCON GIE bit)
- Engedélyeznünk kell a megszakítást generáló eszköz megszakítását (INTCON INTE vagy T0IE bit)
- Megszakításkor a vezérlés a 0x0004 címre kerül (interrupt vektor), az itt elhelyezett kód
  - Elmenti a CPU regisztereit
  - Törli a megszakításkérő bitet (INTCON T0IF vagy INTF)
  - Elvégzi a kívánt műveleteket
  - Visszaállítja a CPU regisztereket és visszatér a megszakított programhoz
- Az itt nem tárgyal eszközök megszakítását az INTCON PEIE bitjével is engedélyezni kell, s egyedi bitjeik további regiszterekben találhatók

p12f683-ledflash-interrupt (main.c)

```
#include <xc.h>
#include <stdint.h>
#define sF_LED  sGPIO.GP2           // flashing LED (shadow)

volatile union {                    // shadow copy of GPIO
    uint8_t         port;
    struct {
        unsigned    GP0     : 1;
        unsigned    GP1     : 1;
        unsigned    GP2     : 1;
        unsigned    GP3     : 1;
        unsigned    GP4     : 1;
        unsigned    GP5     : 1;
    };
} sGPIO;

void main() {
    GPIO = 0;                       // start with all LEDs off
    sGPIO.port = 0;                 //   update shadow
    TRISIO = ~(1<<2);               // configure GP2 (only) as an output
    OPTION_REGbits.T0CS = 0;        // select timer mode
    OPTION_REGbits.PSA = 1;         // no prescaler -> increment every 1 us
    INTCONbits.T0IE = 1;            // enable Timer0 interrupt
    ei();                           // enable global interrupts

    for (;;) {
        GPIO = sGPIO.port;          // continually copy shadow GPIO to port
    }                               // repeat forever
}
```

Az árnyékregiszter koncepció

A GPIO adatkimeneti regiszter egy másolatát a memóriában tároljuk, ott piszkáljuk, s csak időnként másoljuk ki a fizikai kimenetre

union: a struktúrához többféle hozzáférést enged

A megszakítás kiszolgálása

- Az ***interrupt*** kulcsszó jelzi a fordítónak, hogy megszakítást kezelünk (a fordító gondoskodik a regiszterek mentését és visszaállítását végző utasítások beszúrásáról)
- A megszakításban törölni kell a megszakítást okozó jelzőbitet

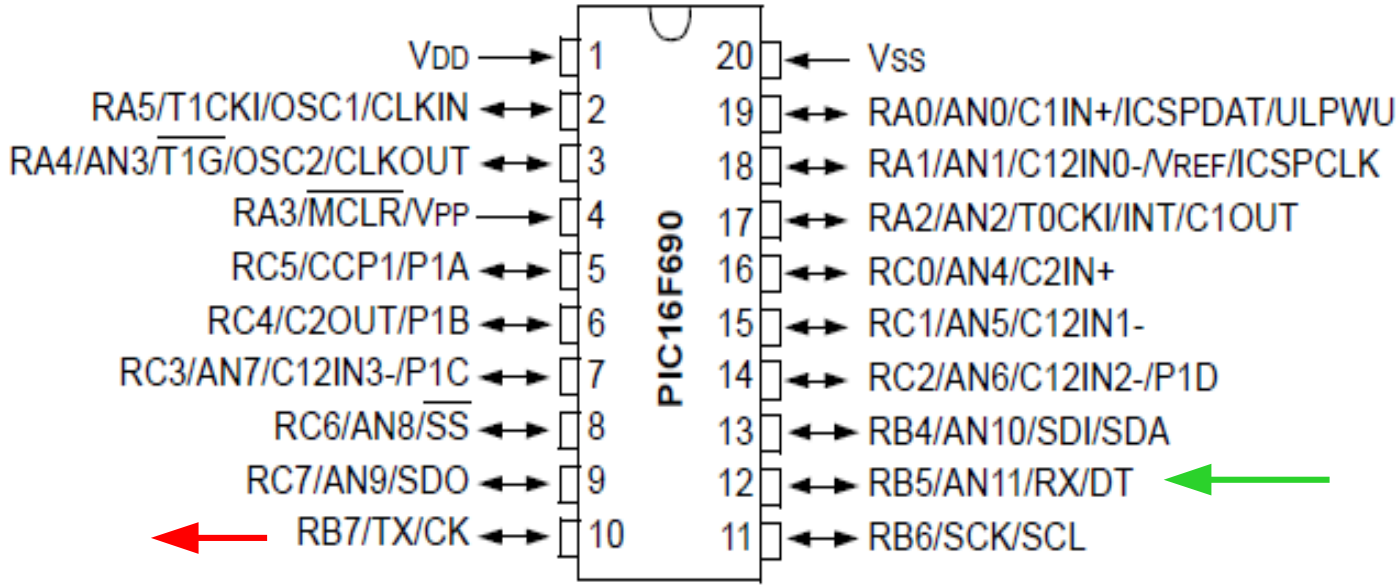
```
void interrupt isr(void) {          // Interrupt service
    //*** Service Timer0 interrupt
    //  TMR0 overflows every 250 clocks = 250 us
    //  Flashes LED at 1 Hz by toggling on every 2000th interrupt (every 500 ms)
    //  (only Timer0 interrupts are enabled)

    static uint16_t  cnt_t0 = 0;    // counts timer0 overflows
    TMR0 += 256-250+3;              // add value for overflow after 250 counts
    INTCONbits.T0IF = 0;            // clear interrupt flag
    ++cnt_t0;                       // increment interrupt count (every 250 us)
    if (cnt_t0 == 500000/250) {     // if count overflow (every 500 ms),
        cnt_t0 = 0;                 //   reset count
        sF_LED = ~sF_LED;           //   toggle LED (via shadow register)
    }
}
```

UART kommunikáció (PIC16F690)

- Az Arduinoval is programozható **PIC16F690** típus sok vonatkozásban a **PIC12F683** „nagytesójának” is tekinthetjük: UART perifériát is tartalmaz, ezt fogjuk most kipróbálni egy mintaprogram segítségével

A program forrása: bzzt.io/posts/understanding-the-usart-on-8-bit-pic-microcontrollers-using-xc8

- A mikrovezérlő UART RX és TX kivezetéseit egy USB- UART TTL átalakító segítségével kapcsolhatjuk össze a számítógéppel



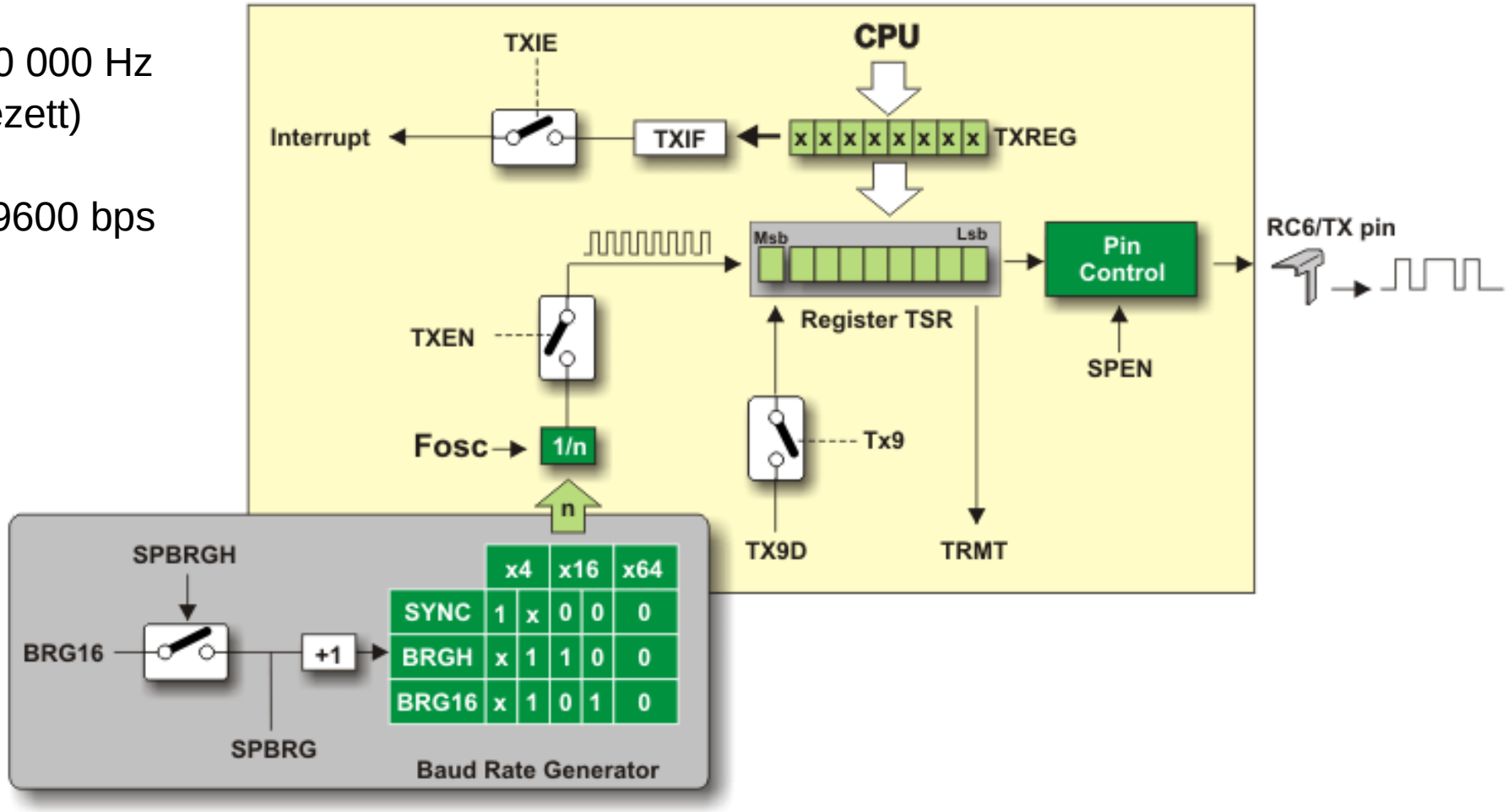
Aszinkron adó (TX) engedélyezése

```
SYNC = 0;                   // Aszinkron mód beállítása (UART)
SPEN = 1;                   // Soros port engedélyezése
TRISB7 = 1;                 // Ha az adatlap írja ...
TXEN = 1;                   // Adás engedélyezés
```

BRGH = 1 esetén SPBRG = Fosc / (16 * baudrate) – 1

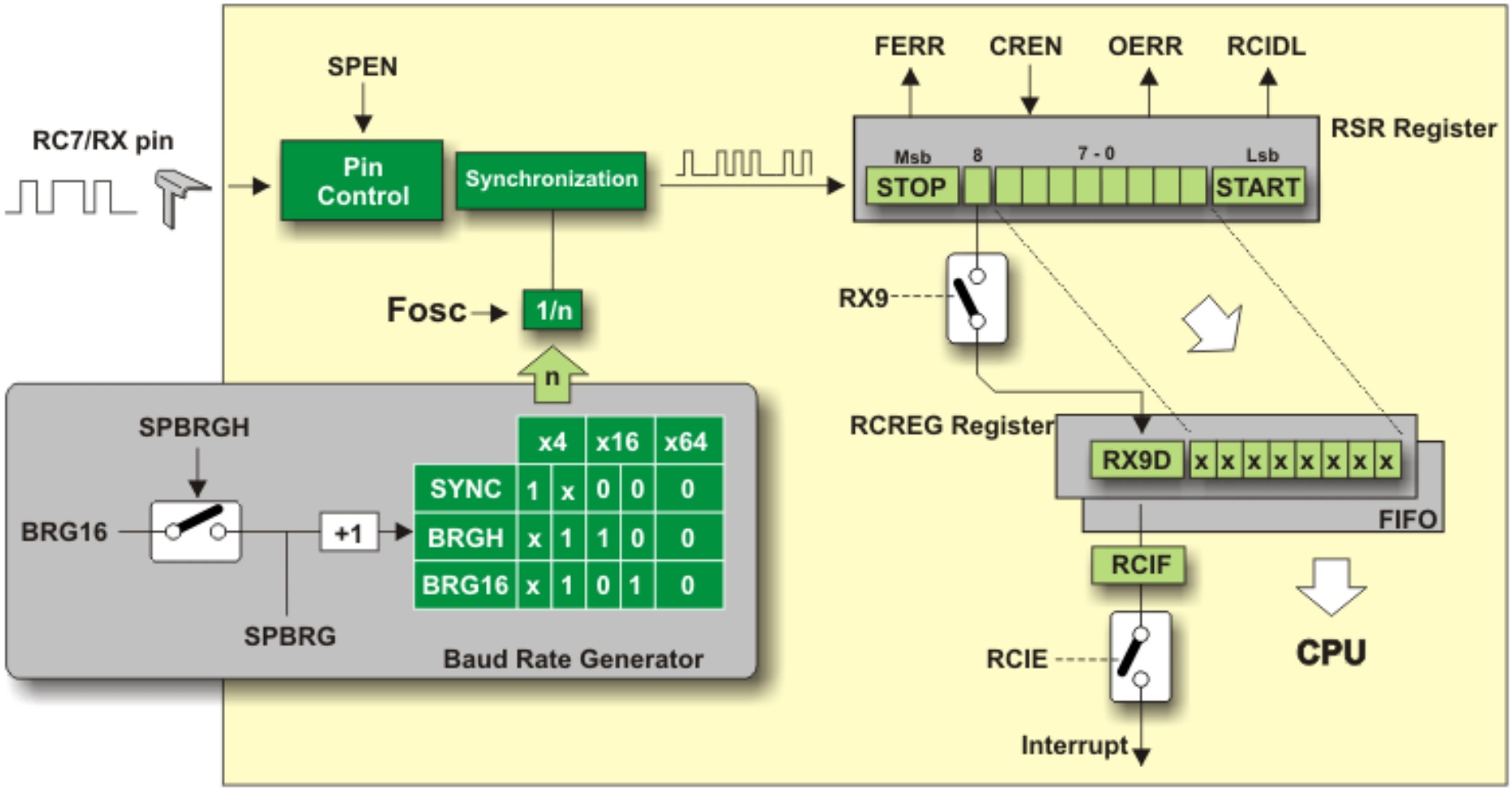
Fosc = 4 000 000 Hz
(alapértelmezett)

Baudrate = 9600 bps
SBRG = 25



Aszinkron vevő (RX) engedélyezése

```
SYNC = 0;                   // Aszinkron mód beállítása (UART)
SPEN = 1;                   // Soros port engedélyezése
TRISB5 = 1;                 // Ha az adatlap írja ...
CREN = 1;                   // Folyamatos vétel engedélyezés
```



UART-tal kapcsolatos regiszterek

- RCREG – innen olvassuk a vett karaktert
- TXREG – ide írjuk a kiküldeni kívánt karaktert
- RCSTA / TXSTA – vezérlő- és státusz regiszterek
- PIR1 – periféria megszakításkérő jelek

TABLE 12-1: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
BAUDCTL	ABDOVF	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	01-0 0-00	01-0 0-00
INTCON	GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF	0000 000x	0000 000x
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
RCREG	EUSART Receive Data Register								0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
SPBRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0	0000 0000	0000 0000
SPBRGH	BRG15	BRG14	BRG13	BRG12	BRG11	BRG10	BRG9	BRG8	0000 0000	0000 0000
TRISB	TRISB7	TRISB6	TRISB5	TRISB4					1111 ----	1111 ----
TXREG	EUSART Transmit Data Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010

Legend: x = unknown, – = unimplemented read as '0'. Shaded cells are not used for Asynchronous Transmission.

p16f690-uart-cx8 projekt: usart.h

Forrás: bzzt.io/posts/understanding-the-usart-on-8-bit-pic-microcontrollers-using-xc8

```
#define BAUD 9600
#define FREQUENCY 4000000L
#define MULTIPLIER 16UL             // muiltiplier is typically 64UL, 16UL or 4UL
#define HIGH_SPEED 1                // see in datasheet if you need high speed

#define NINE_BITS 0                 // Use 9bit communication?
#define RX_PIN TRISB5
#define TX_PIN TRISB7

#define DIVIDER ((int)(FREQUENCY/(MULTIPLIER * BAUD) -1))

#if HIGH_SPEED == 1
#define SPEED 0x4
#else
#define SPEED 0
#endif
#define RCSTA_DEFAULT 0x90          // SPEN = 1, CREN = 1
#define TXSTA_DEFAULT 0x20          // TXEN = 1, SYNC = 0

#define init_comms()\
    RX_PIN = 1; TX_PIN = 1; SPBRG = DIVIDER;\
    RCSTA = (NINE_BITS|RCSTA_DEFAULT);\
    TXSTA = (SPEED|NINE_BITS|TXSTA_DEFAULT)

void putch(unsigned char);
unsigned char getch(void);
unsigned char getche(void);
```

PIC16F690-hez igazítva

p16f690-uart-cx8 projekt: usart.c

Forrás: bzzt.io/posts/understanding-the-usart-on-8-bit-pic-microcontrollers-using-xc8

```
#include <xc.h>
#include <stdio.h>
#include "usart.h"

void putch(unsigned char byte) {
    /* output one byte */
    while(!TXIF);    /* set when register is empty */
    TXREG = byte;
}

unsigned char getch() {
    /* retrieve one byte */
    while(!RCIF);    /* set when register is not empty */
    return RCREG;
}

unsigned char getche(void) {
    /* retrieve one byte with echo */
    unsigned char c;
    putch(c = getch());
    return c;
}
```

p16f690-uart-cx8 projekt: main.c

Forrás: bzzt.io/posts/understanding-the-usart-on-8-bit-pic-microcontrollers-using-xc8

```
#pragma config FOSC = INTRCIO   // INTOSCIO oscillator: I/O function on RA4/RA5
#pragma config WDTE = OFF       // Watchdog Timer disabled
#pragma config PWRTE = ON       // Power-up Timer enabled
#pragma config MCLRE = ON       // MCLR Pin function is MCLR
#pragma config CP = OFF         // Code Protection is disabled
#pragma config CPD = OFF        // Data Code Protection is disabled
#pragma config BOREN = OFF      // Brown-out Reset is disabled
#pragma config IESO = OFF       // Internal External Switchover is disabled
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor is disabled

#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#include "usart.h"

void main() {
    unsigned char input;
    INTCON=0;                   // disable interrupts
    ANSELH&=~_ANSELH_ANS11_MASK;// configure RX for IO
    init_comms();               // Config UART port

    for(;;) {
        printf("\rPress a key and I will echo it back\n");
        input = getch();
        printf("\rIt was [%c]\n", input);
    }
}
```

PIC16F690-hez igazítva

p16f690-uart-cx8 projekt futási eredmény

- Csatlakoztassuk az RX, TX és GND kivezetéseket egy USB-UART TTL átalakítón keresztül a PC-hez!