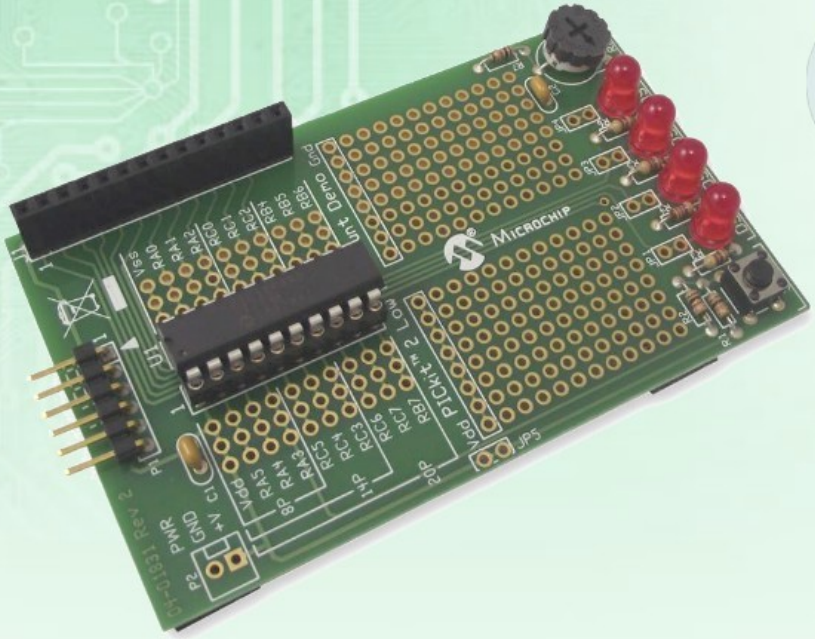


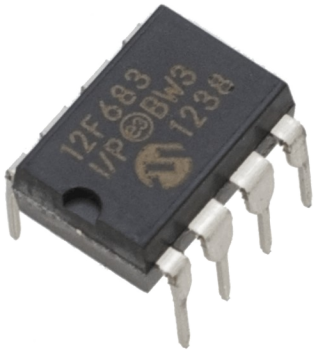
MICROCHIP

PIC mikrovezérlők

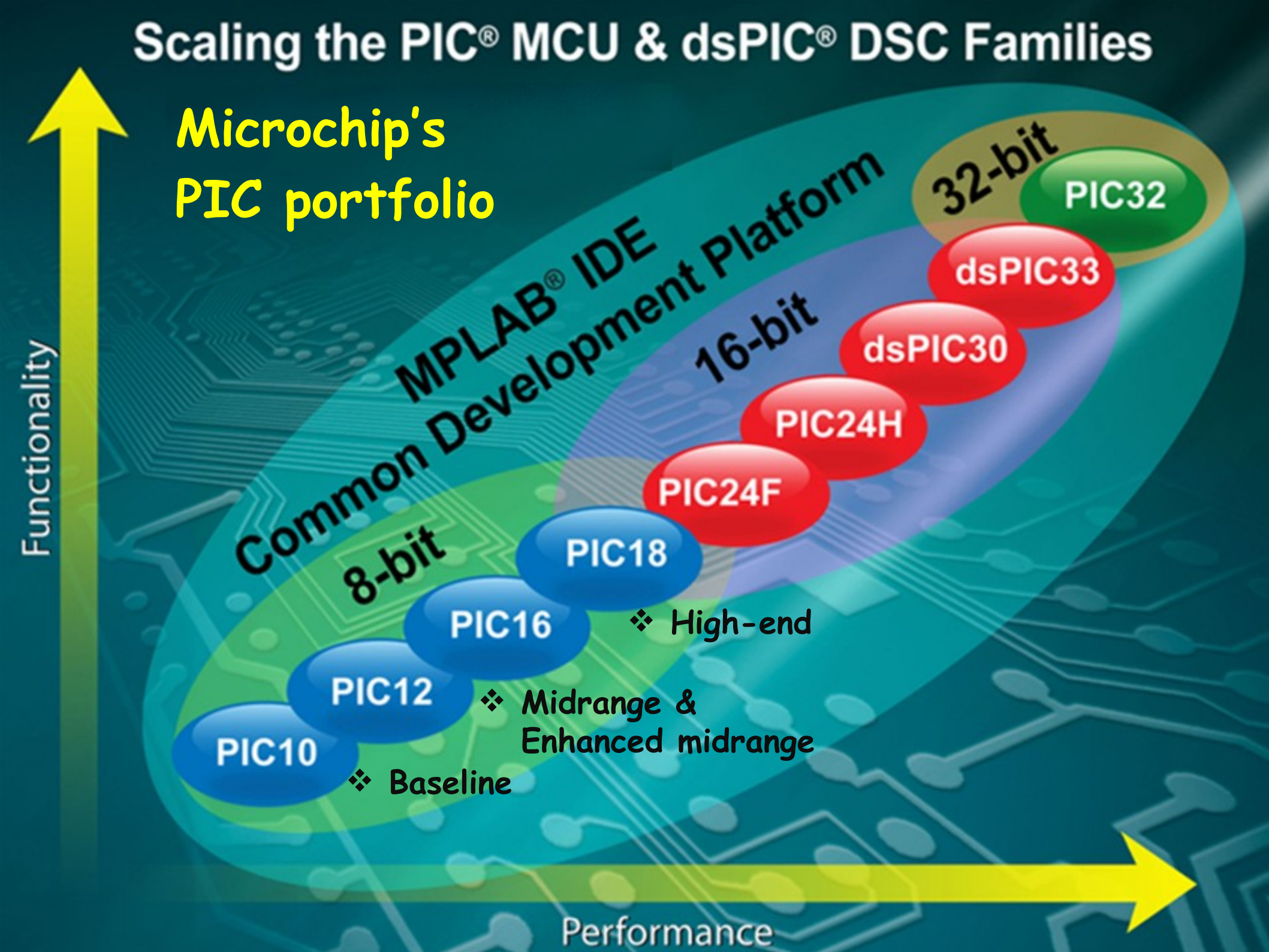
Felhasznált és ajánlott irodalom

- **Milan Verle:** PIC MicrocontrollersProgramming in Assembly
- **Microchip:** PICmicro Mid-Range MCU Family Reference Manual
- **T&T:** Közepes teljesítményű PIC mikrovezérlők Felhasználói Kézikönyv
- **SimulIDE Community:** SimulIDE Tutorials
- **The Jallib Team:**
  - Have fun with PIC microcontrollers, Jal v2 and Jallib
  - Jal v2 Compiler Documentation
  - Installing the JAL Visual Studio Code extension
- **Microsoft:** Visual Studio Code Docs

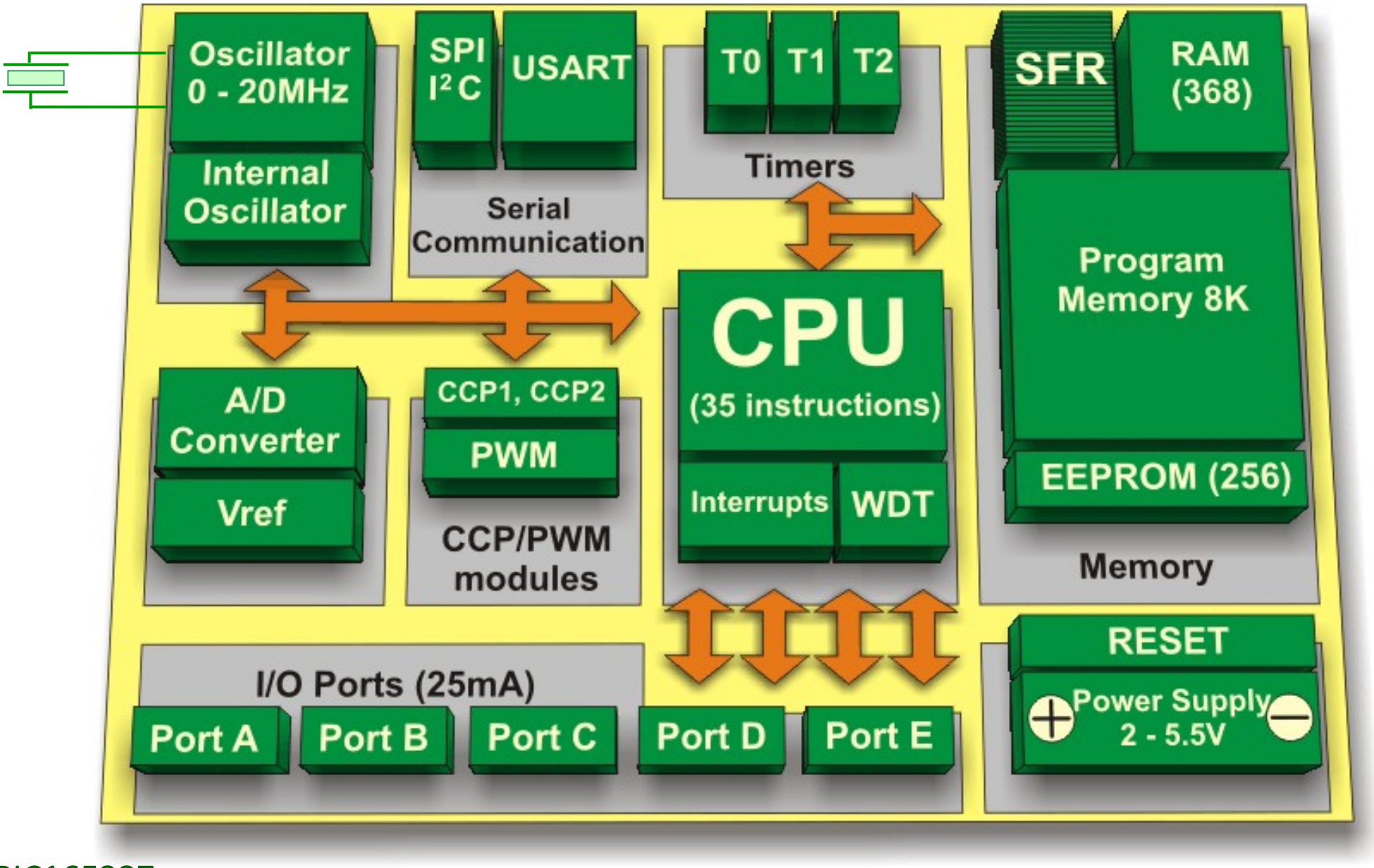
Adatlapok:



- **PIC12F683** adatlap és termékinfo
- **Microchip:** PICkit2 programmer User's Guide
- **Icircuit Technologies:** iCP02v2 USB PIC/EEPROM programmer manual


Scaling the PIC® MCU & dsPIC® DSC Families
Microchip's
PIC portfolio
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

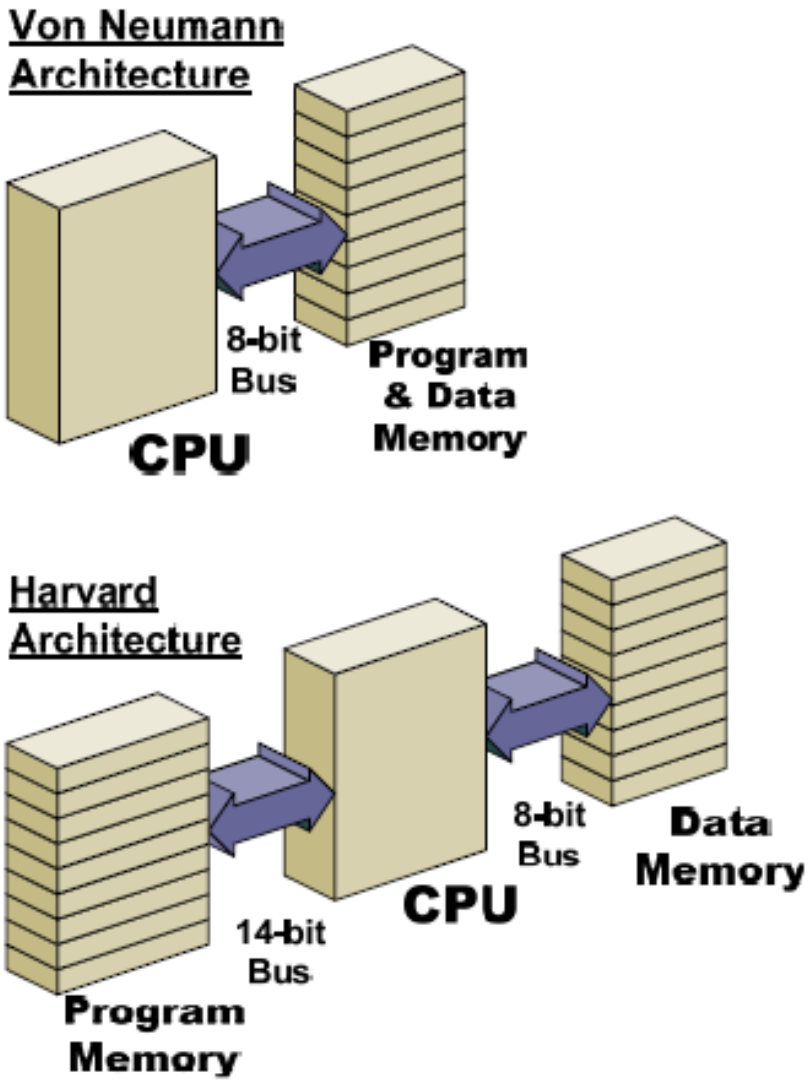
8-bites „mid-range” PIC felépítése



PIC16F887

Forrás: www.mikroe.com

Harvard felépítés



Neumann-felépítés

- Az utasításokat és az adatokat ugyanabban a memóriában tárolja
- Korlátozott sávszélesség

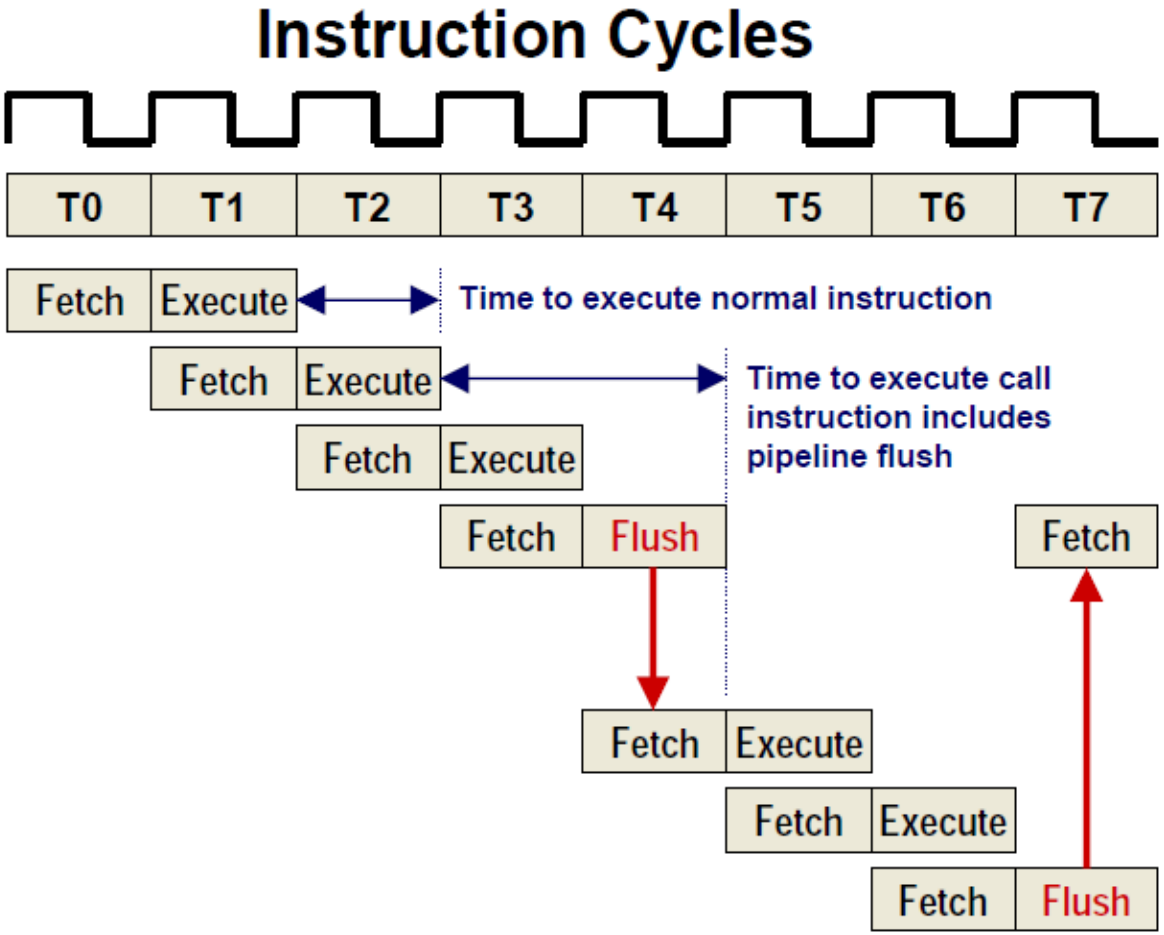
Harvard felépítés

- Az utasításokat és az adatokat külön-külön memóriában tárolja
- A párhuzamos működés miatt nagyobb a sávszélesség
- Lehetőség különböző szélességű kiépítésre (program memória 14-bites, adatmemória 8-bites)

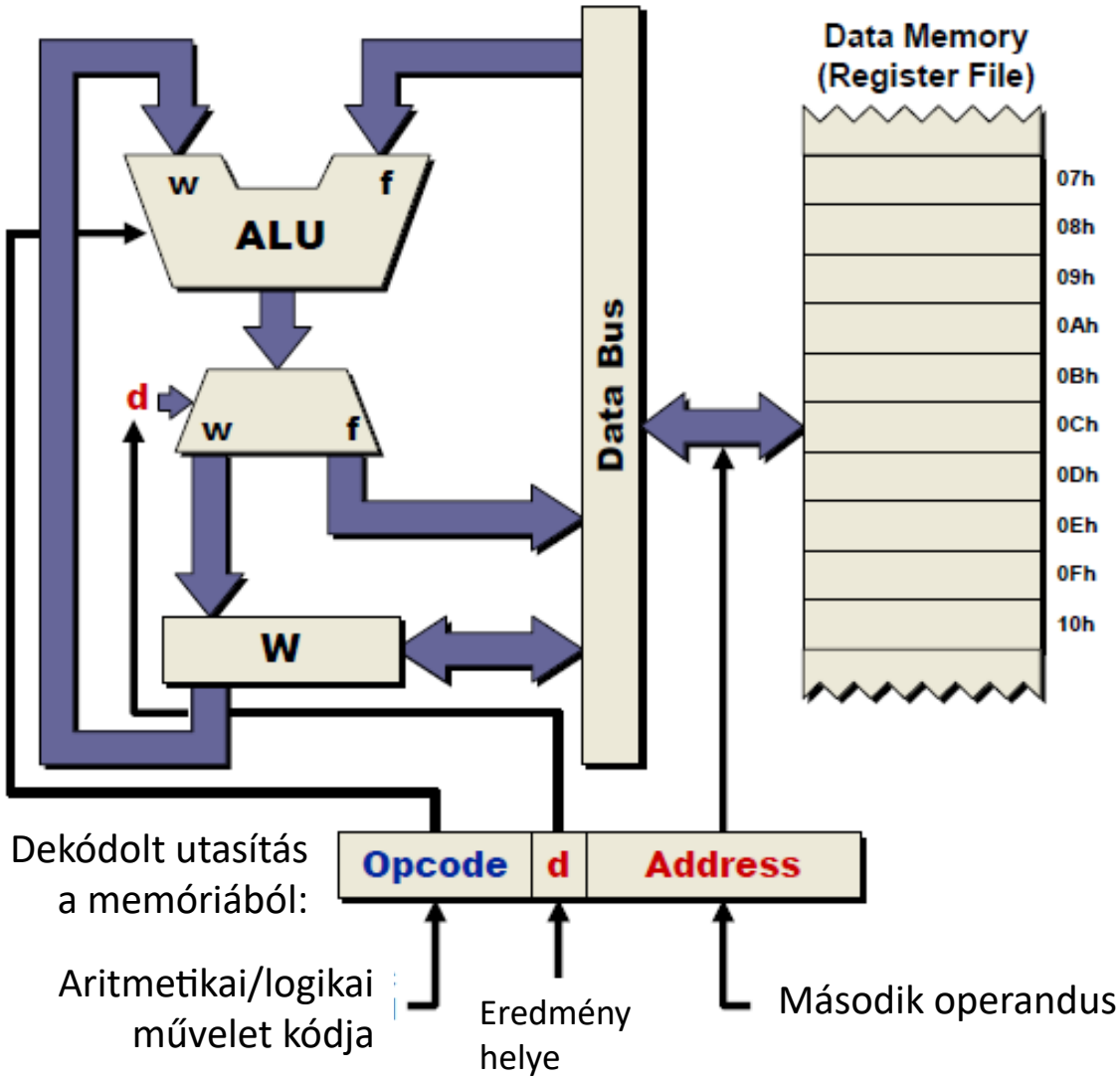
Utasítás futószalag

Az aktuális utasítás végrehajtásával párhuzamosan folyik a következő utasítás elővétele

```
1   MAIN  movlw   0x05
2         movwf   REG1
3         call    SUB1
4         addwf   REG2
          :
51  SUB1  movf    PORTB,w
52        return
53  SUB2  movf    PORTC,w
54        return
```



A regiszter fájl koncepció

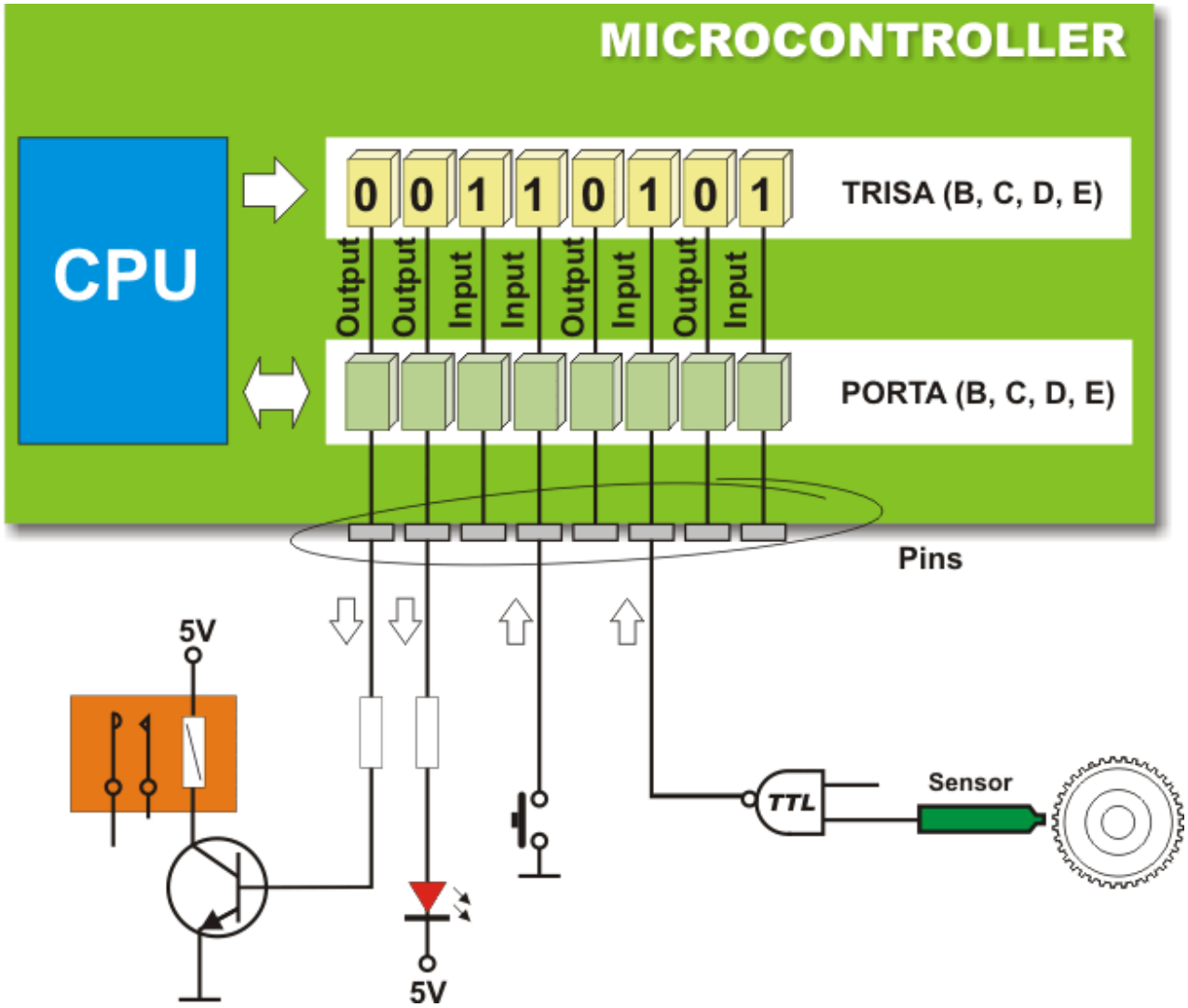


- **A teljes adatmemória a regiszterkészlethez tartozik,** tehát az adatok közvetlenül manipulálhatók.
- **Minden periféria regiszter** az memóriába van leképezve
- **Ortogonális utasításkészlet:** minden utasítás a memória bármely részét kezelheti.
- **A szószervezésű utasításokkal** közvetlenül címezhetjük a lapon belüli (7-bites címtartomány) regisztereket

A ***d*** bit beállításától függően az eredmény a regiszterfájlba, vagy a ***W*** munkaregiszterbe kerül.

Digitális I/O

Egy port általában 8 bites szervezésű, de bitenként is kezelhető, belelértve az adatáramlás irányának megválasztását is.

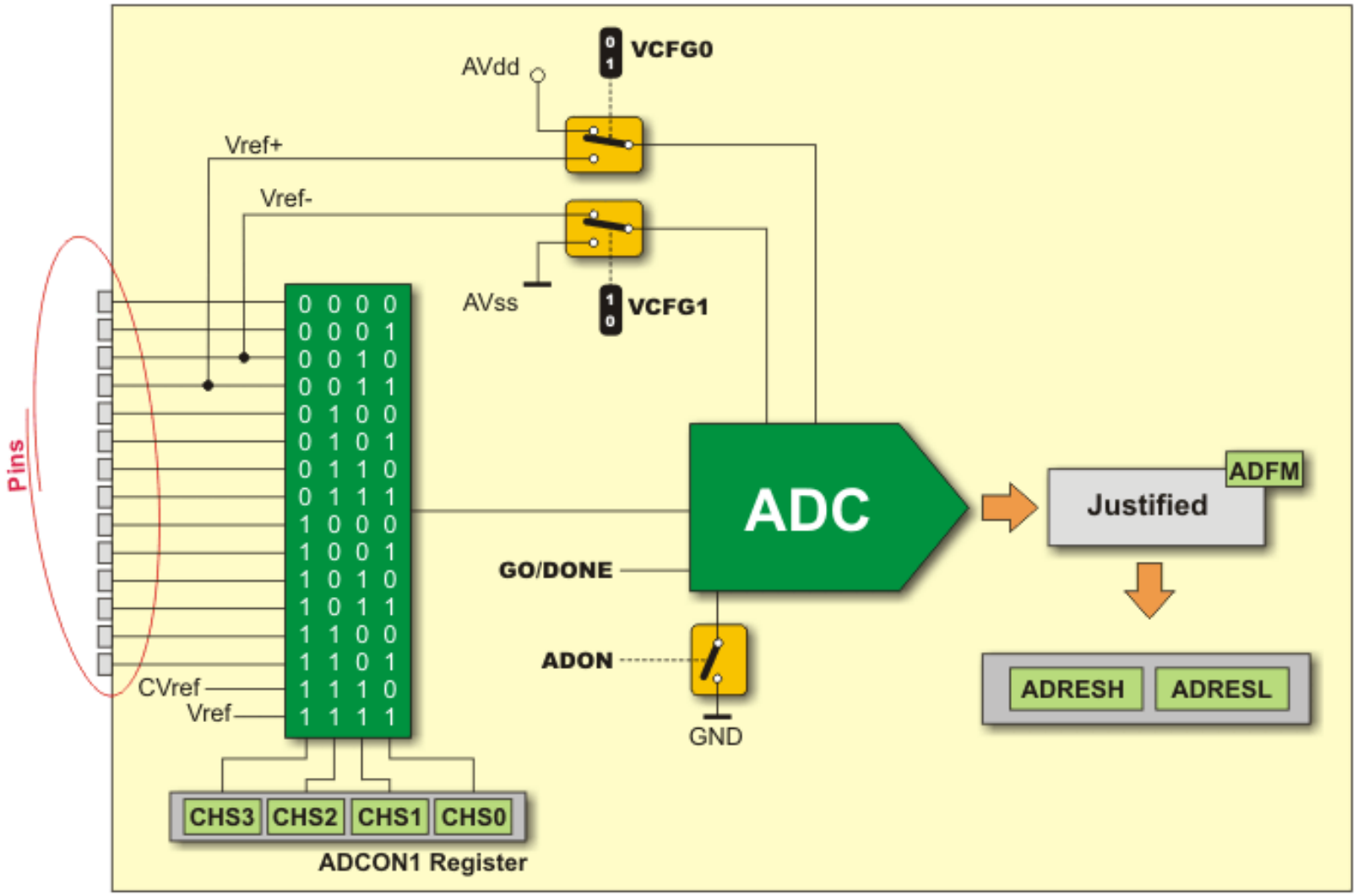


Forrás: www.mikroe.com

Analóg bemenetek (10-bites ADC)

Referencia: belső vagy külső
Felbontás: 10 bit (0 – 1023)

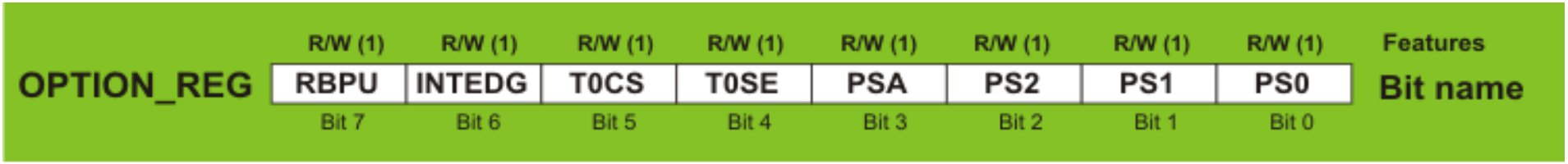
Felépítés: SAR (fokozatos megközelítés elvén)
Sebesség: 20 k – 1.1 M mintavétel/s (típustól függően)



Forrás: www.mikroe.com

OPTION_REG regiszter

Mielőtt a **Timer0** időzítő/számláló-val megismerkedünk, tisztáznunk kell az OPTION_REG regiszter szerepét!



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 élre számlál, **1:** lefutó élre számlál)

PSA – Előszámláló hozzárendelés (**0:** TMR0 használja, **1:** WDT használja)

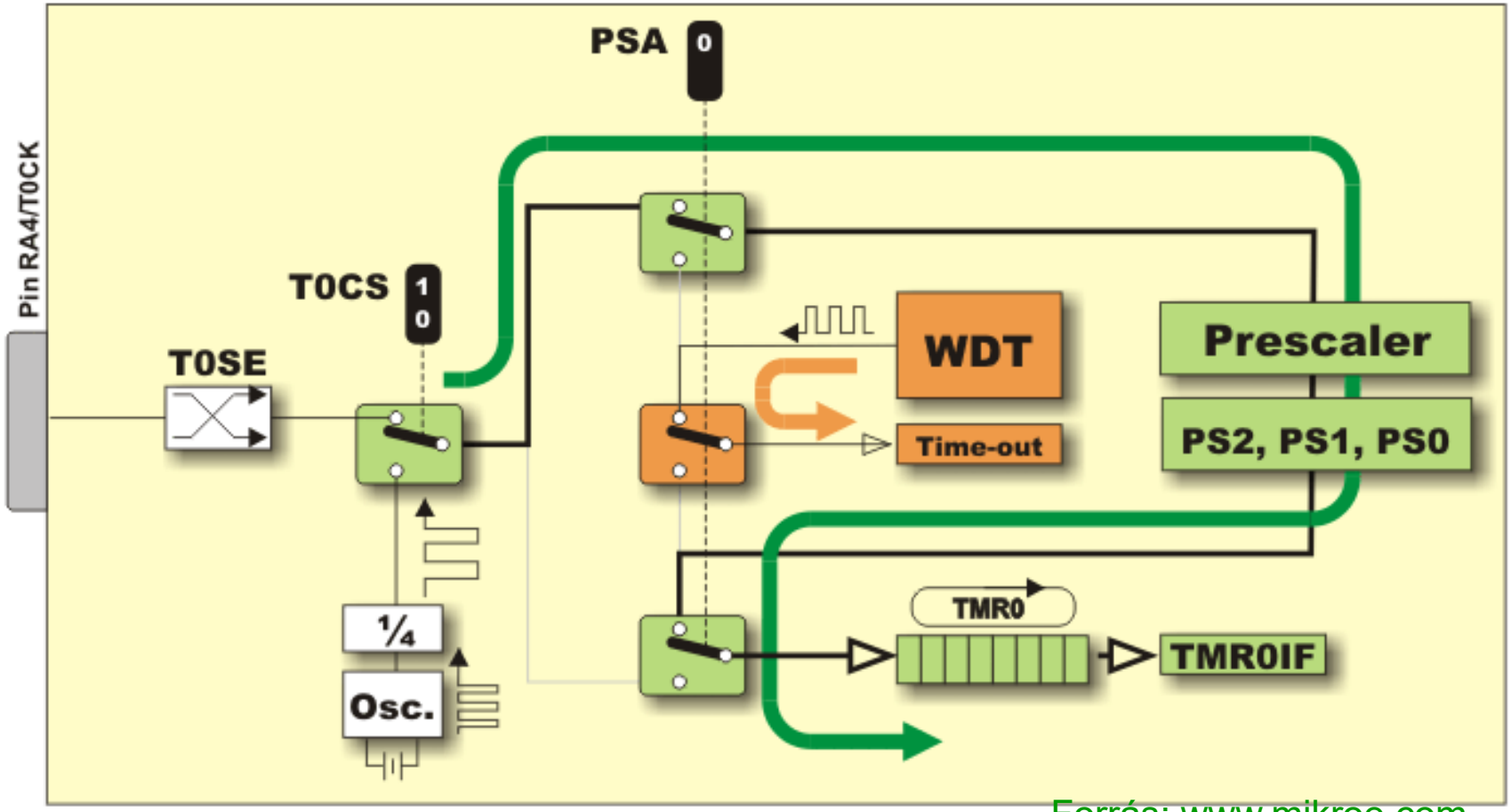
PS[2:0] – előosztási arány beállítása (2^{N+1} TMR0 esetén, illetve 2^{N} WDT esetén)

Forrás: www.mikroe.com

Timer0 8-bites időzítő/számláló

Timer0 előosztója vagylagosan az időzítőhöz, vagy a **Watchdog**-hoz rendelhető. Az ábrán látható esetben az **OPTION_REG PSA** bitje ’0’, ezért az előosztó **TMR0**-hoz kapcsolódik.

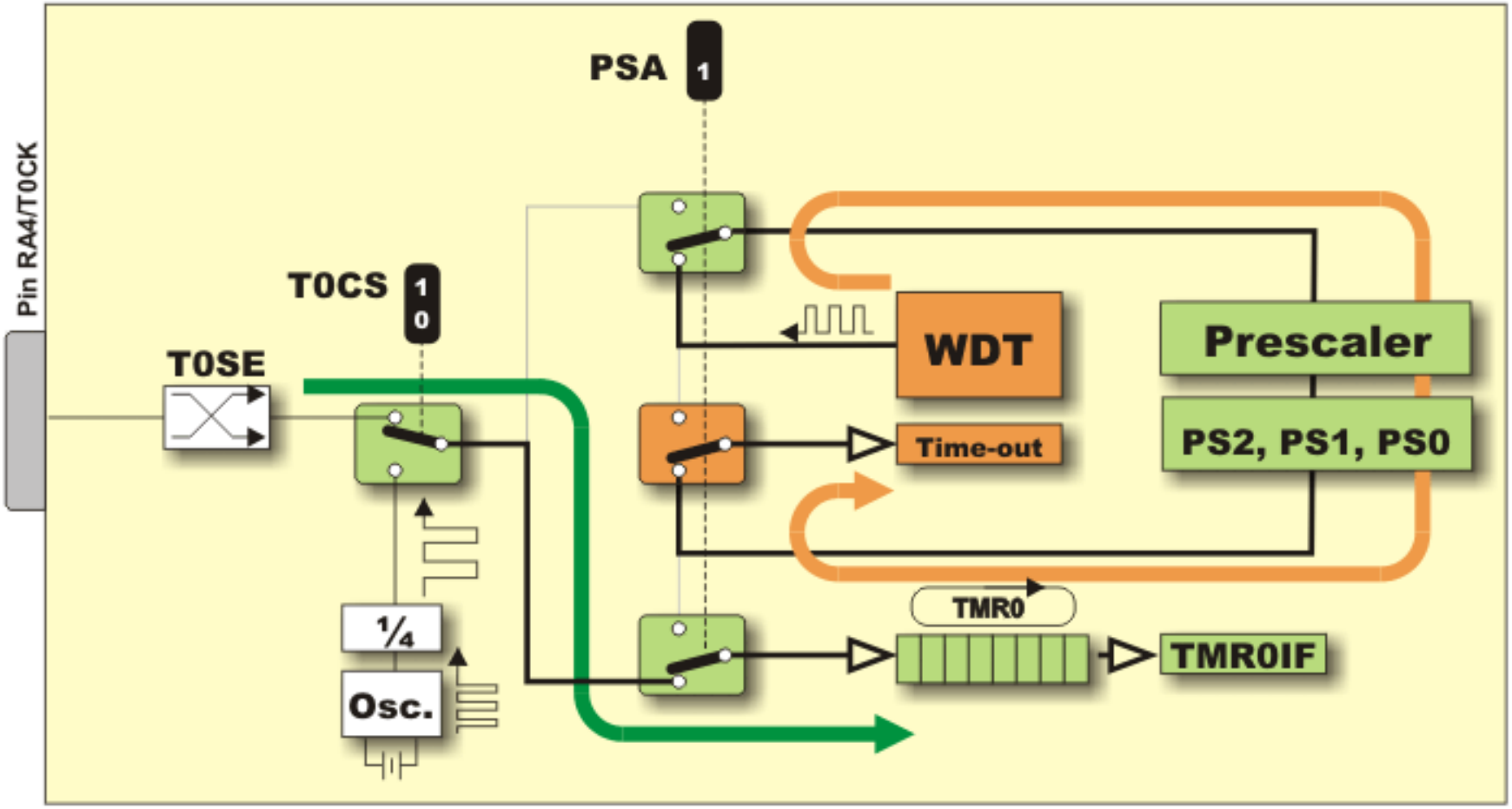
A **Watchdog** az elakadt programok utolsó menedéke – ha nem töröljük rendszeresen, túlcsorduláskor **RESET**-et okoz.



Forrás: www.mikroe.com

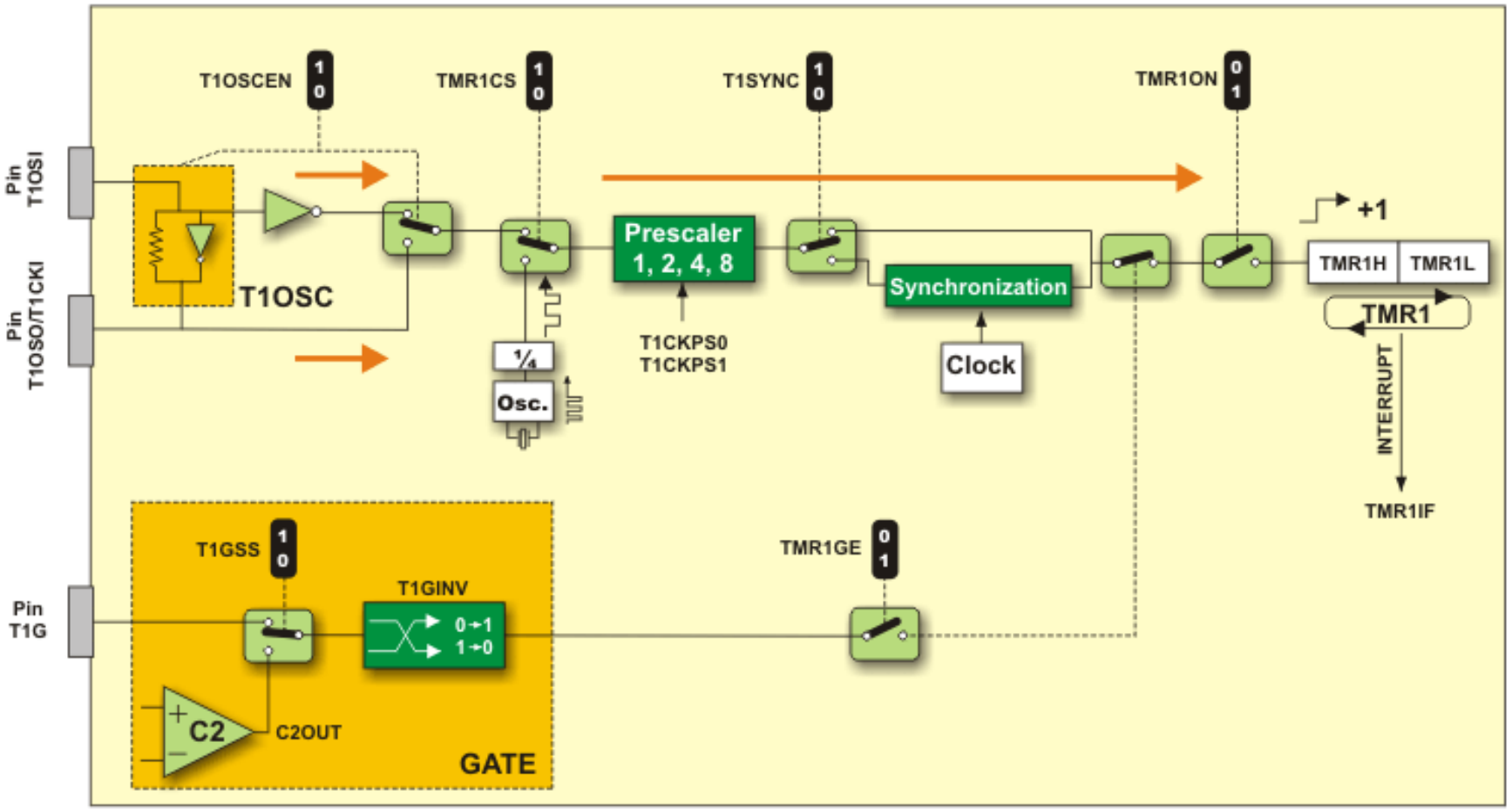
Timer0 8-bites időzítő/számláló

Az alábbi ábrán látható esetben az **OPTION_REG PSA** bitje ’1’, ezért az előosztó a **WDT**-hez kapcsolódik, **TMR0** pedig előosztó nélkül, közvetlenül a bejövő órajelet számlálja. Ebben az esetben a számláló legkésőbb 256 ciklusonként túlcsordul.



Forrás: www.mikroe.com

Timer1 16-bites időzítő/számláló



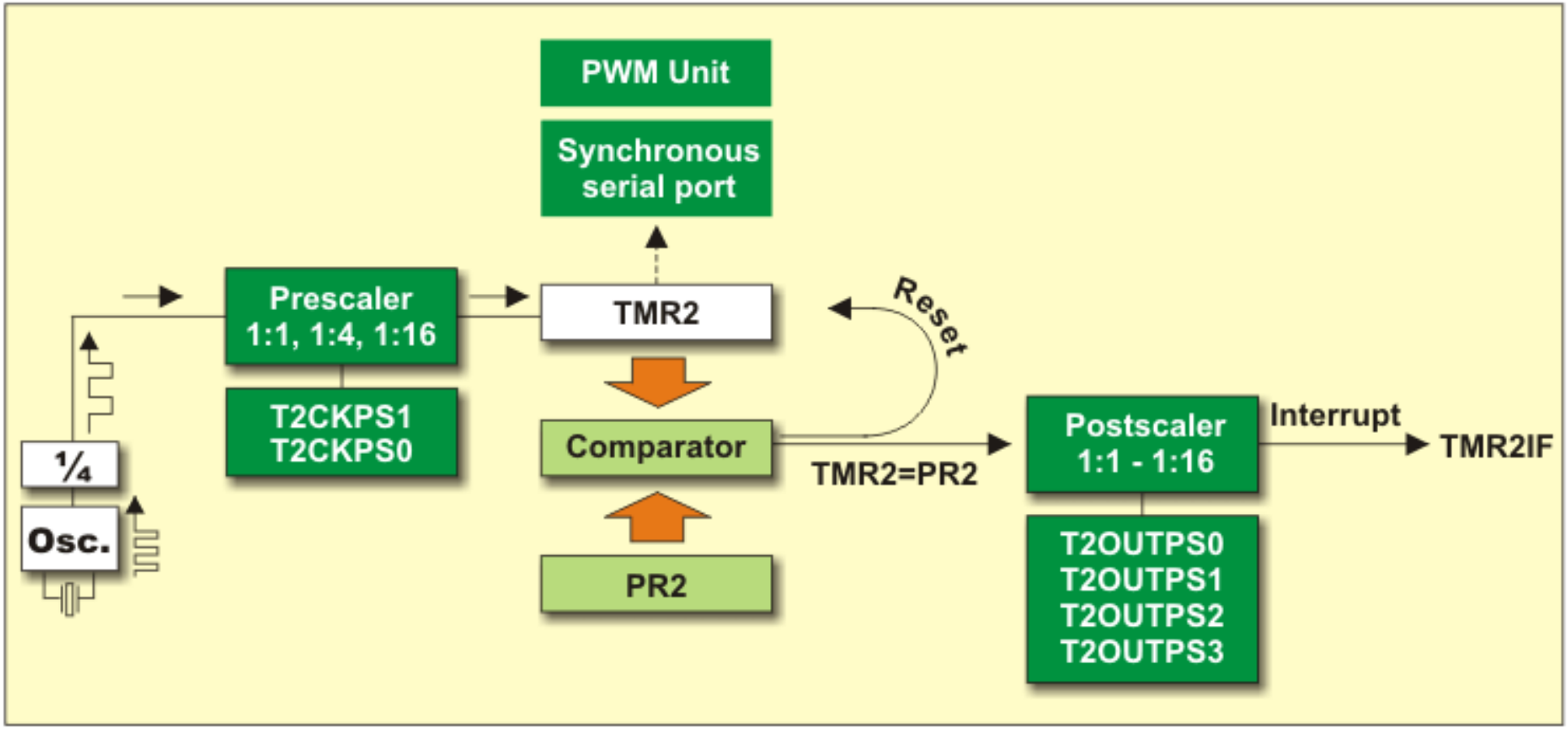
- Külső vagy belső jelforrás
- Szinkron/aszinkron mód
- Kapuzás külső jellel,vagy az analóg komparátor kimenetével

Előnyös tulajdonság, hogy a szinkronizálás az előszámláló kimenetén történik!

Forrás: www.mikroe.com

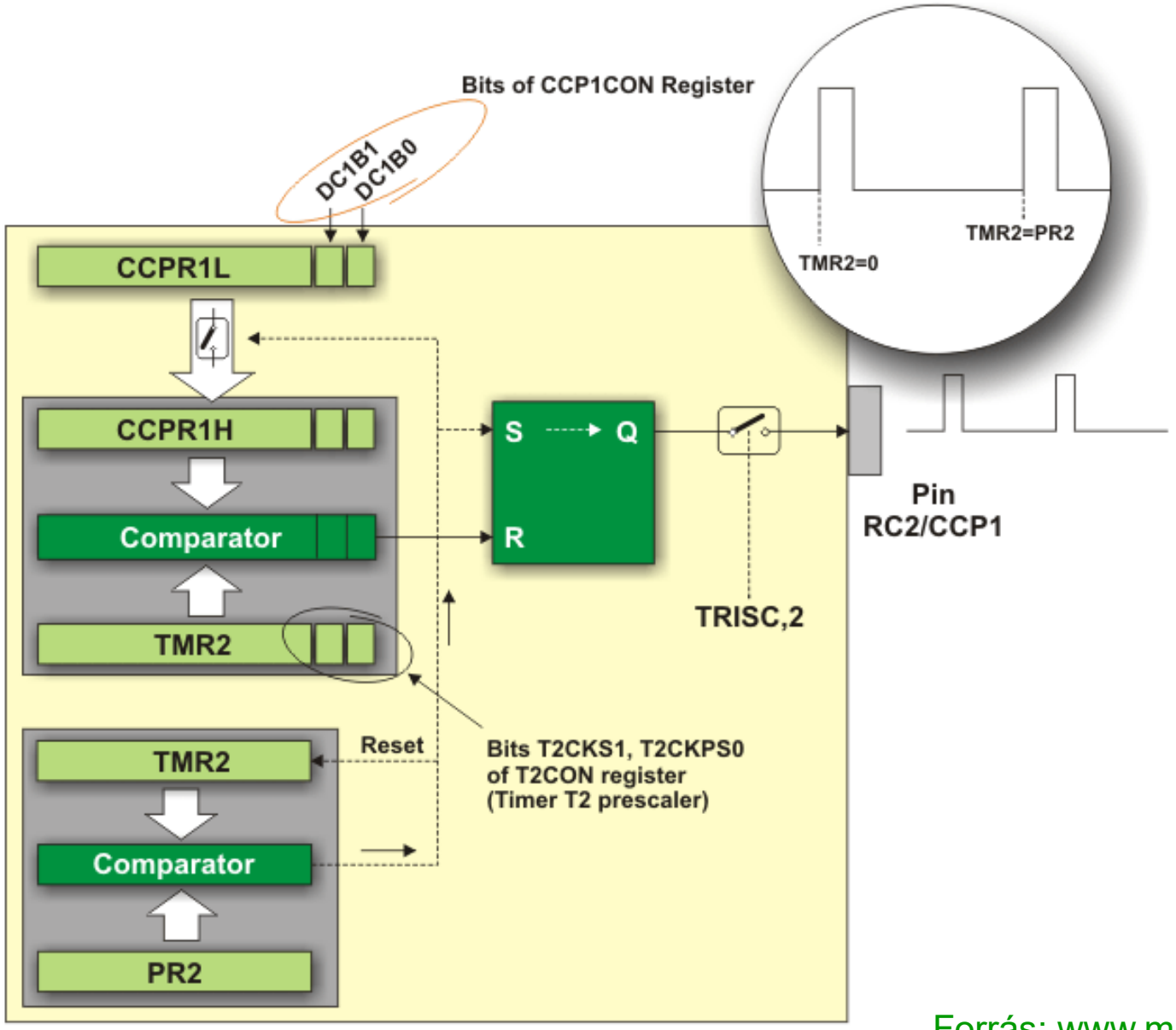
TMR2 8-bites számláló elő- és utóosztással

Általában periodikus időzítésre használjuk: rendszeres időközönkénti megszakításokhoz, vagy PWM jel előállításához (CCP modullal kombinálva).



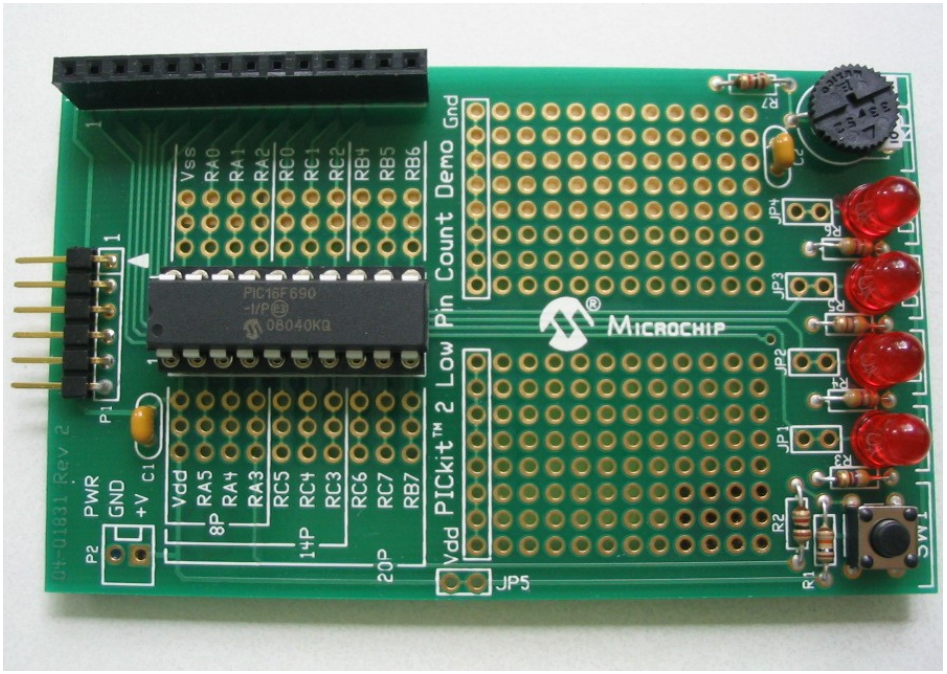
Forrás: www.mikroe.com

PWM jel generálás (CCP1+TMR2)



Forrás: www.mikroe.com

PICkit2 Starter Kit (2007)



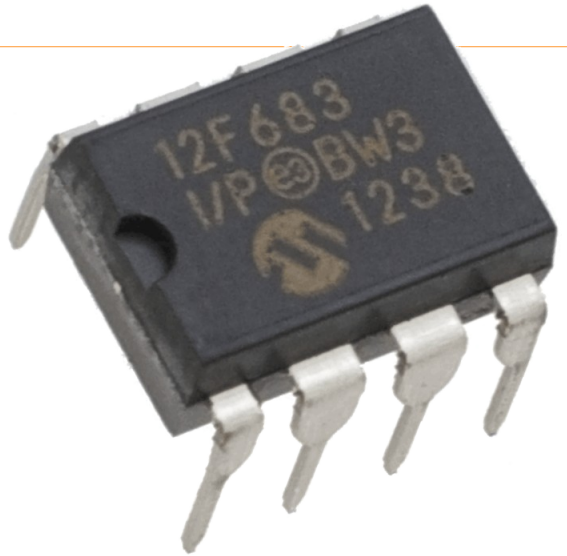
Low Pin Count Demo

- **PIC16F690**
- 4 db LED
- nyomógomb
- Potméter
- ICSP csatlakozó
- Protoboard felület
- PICtail bővítő csatlakozó

PICkit2

- Programozó
- Nyomkövető
- Logikai analizátor
- Soros kommunikátor

Egy konkrét típus: PIC12F683



Bővebb információ: cspista.hu/PICant/pic12f683.html

A CPU regiszterei

- **FSR (04h/84h)** – File Select Register, indirekt (adat) címzéshez
- **PCL (02h/82h)** – a 13 bites PC utasításszámláló alsó bitjei (bit7..bit0)
- **PCLATH (0Ah/8Ah)** – az utasításszámláló felső bitjei beírására szolgáló előkészítő tároló. Először ezt írjuk, majd a PCL regisztert
- **STATUS (03h/83h)** – az ALU jelzőbitjeit, a bankválasztó biteket, valamint a CPU állapotáról tájékoztató biteket tartalmazza
- **OPTION_REG (80h)** – TMR0/WDT előosztó, INT élválasztás, TMR0 bemenőjel választás, GPIO belső felhúzás konfigurálása
- **PCON (8Eh)** – Power control vezérlő regiszter: kisfogyasztású ébresztő, ill. feszültségesés detektálás engedélyezése, eseményjelzők
- **INTCON, PIR1, PIE1** – a megszakítási rendszerrel kapcsolatos regiszterek (engedélyező és megszakítási eseményt jelző bitek)

Bővebb információ: cspista.hu/PICant/a-cpu-regiszterei.html

A STATUS regiszter

	R/W (0)	R/W (0)	R/W (0)	R (1)	R (1)	R/W (x)	R/W (x)	R/W (x)	Features
STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	Bit name
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	

IRP – Regiszter bank választás kiegészítő bitje indirekt címzésnél.

RP1, RP0 – Regiszter bank választása direkt címzés esetén (lásd **BANKSEL**)

Megjegyzés: PIC12F683 esetén az **IRP** és **RP1** bitek nincsenek implementálva

TO – Időtúllépés jelzése (**0: WDT** túlcsorduláskor, **1:** egyébként).

PD – Power-down jelzése (0: SLEEP után, 1: egyébként)

Z – Zero bit (1: ha a művelet eredménye nulla)

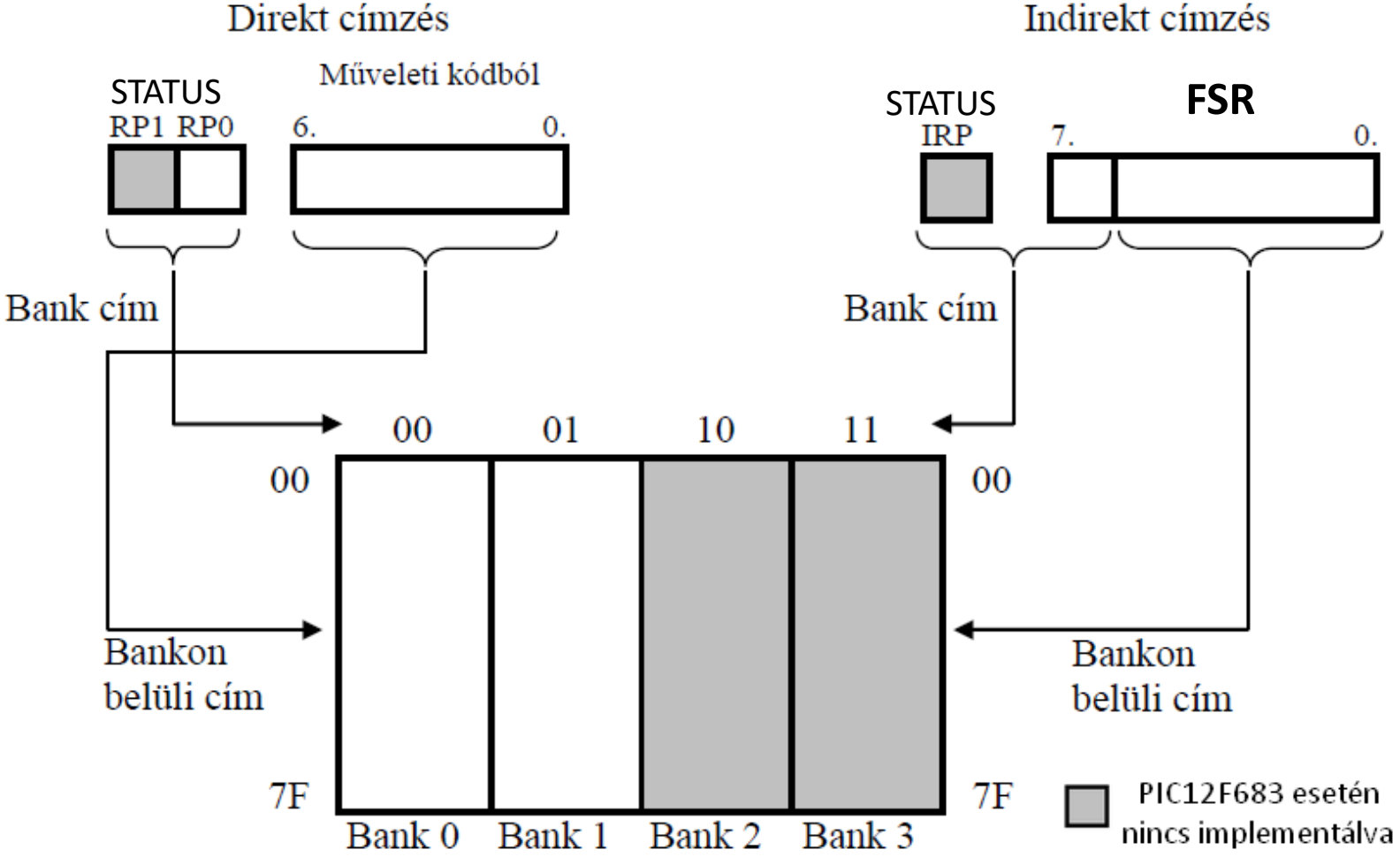
DC – számjegy túlcsordulás/áthozat bit (BCD összeadásnál, illetve kivonásnál)

C – túlcsordulás jelző bit (Carry)

Indirekt és direkt címzés

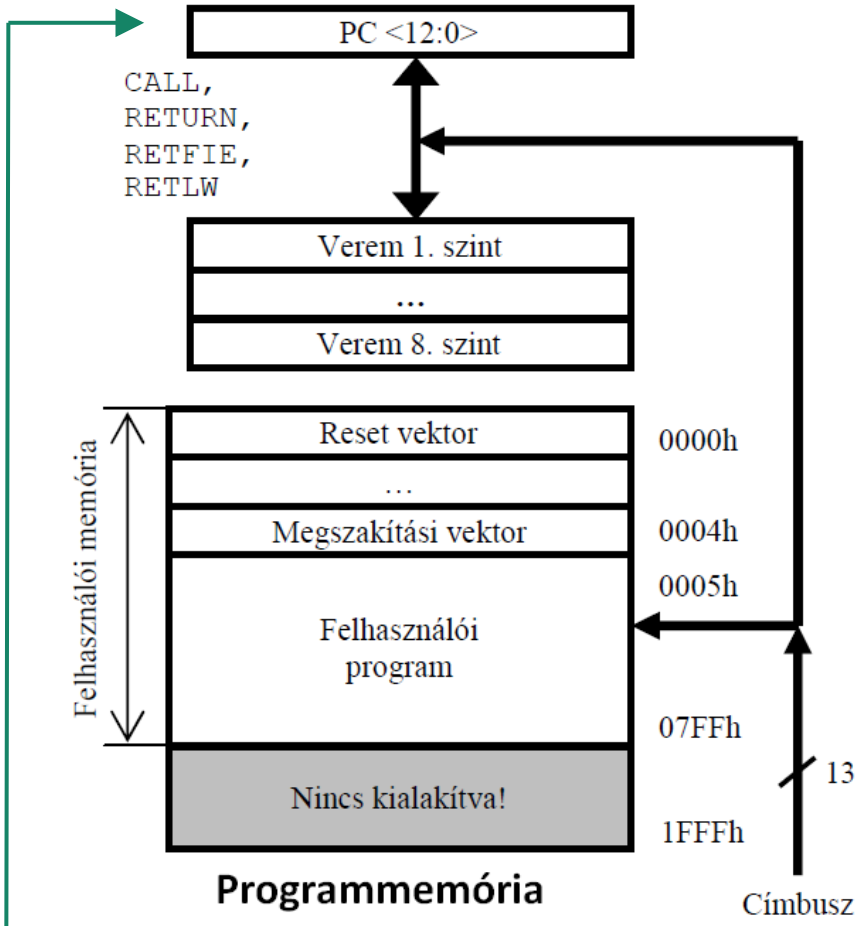
Adatmemória és az **SFR** regiszterek elérésére kétféle címzést használhatunk:

- **Direkt címzés** az utasítás kódjában tárolt címmel
- **Indirekt** (indexelt) **címzés** az **FSR** regiszter felhasználásával

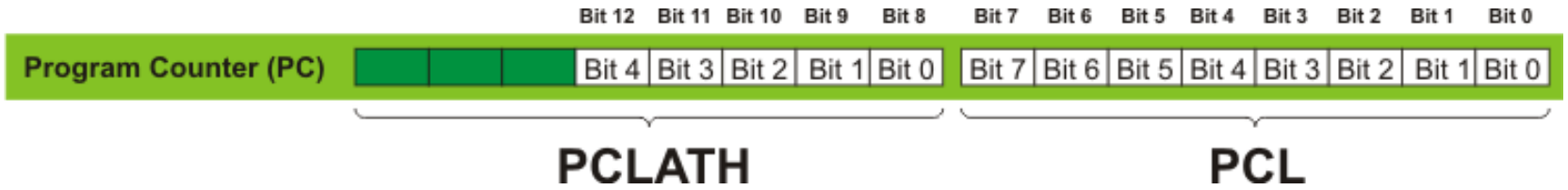


A **PIC12F683** esetében csak az első két bank van implementálva.

A programmemória címzése



- **PC** a programszámláló, 13 bites
- **A veremtár** 8 szintű, kezelése hardveresen, automatikusan történik, szoftveresen nem hozzáférhető.
  - **Betöltés: `CALL`**, illetve megszakításkor
  - **Visszatöltés: `RETURN`, `RETLW`,** vagy **`RETFIE`** hatására.
- A 2k-nál nagyobb memóriájú PIC midrange mikrovezérlők esetében a **GOTO** és **CALL** utasítások csak 2 K lapokban tudják kezelni a memóriát (**PIC12F683** esetén ez nem gond, mert csak 2 K a memória…)
- **PCL**-t manipuláló parancsoknál ügyelni kell, hogy nincs-e túlcsordulás (amikor **PCLATH** értékét növelni kell).



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

A hardver „viselkedését” befolyásoló, menet közben nem változtatható beállítások, a program beégetésekor kell megadni.

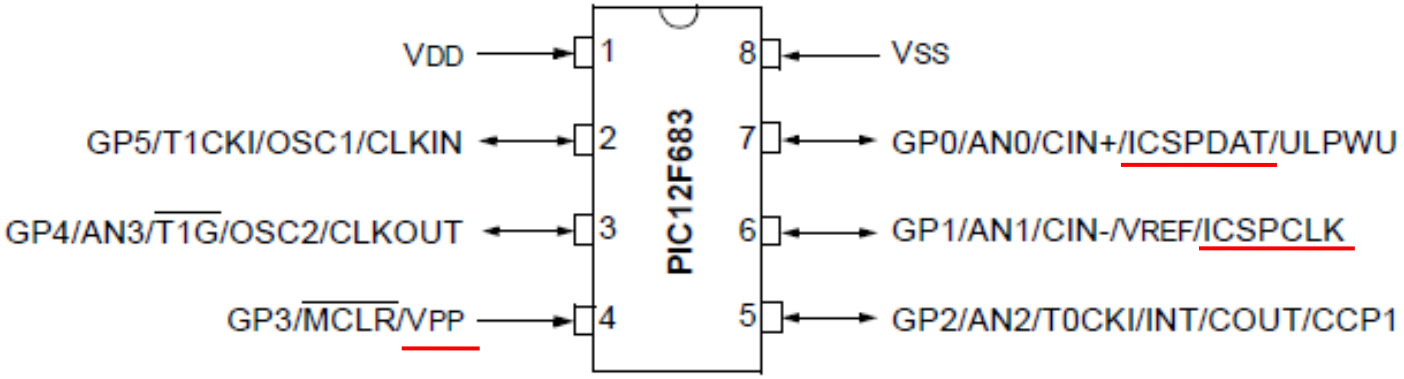
FCMEN: Fail-Safe Clock Monitor Enabled bit
IESO: Internal External Switchover enable
BOREN<1:0>: Brown-out Reset Selection
CPD: Data Code Protection bit
CP: Code Protection bit
MCLRE: GP3/MCLR pin function select bit (MCLR láb RESET vagy I/O legyen)
PWRTE: Power-up Timer Enable bit
WDTE: Watchdog Timer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

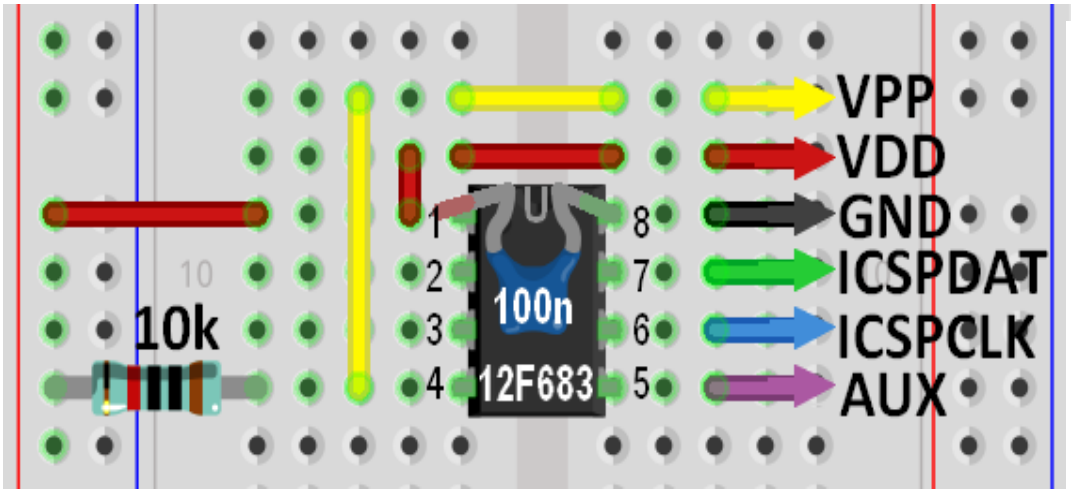
Programok letöltése

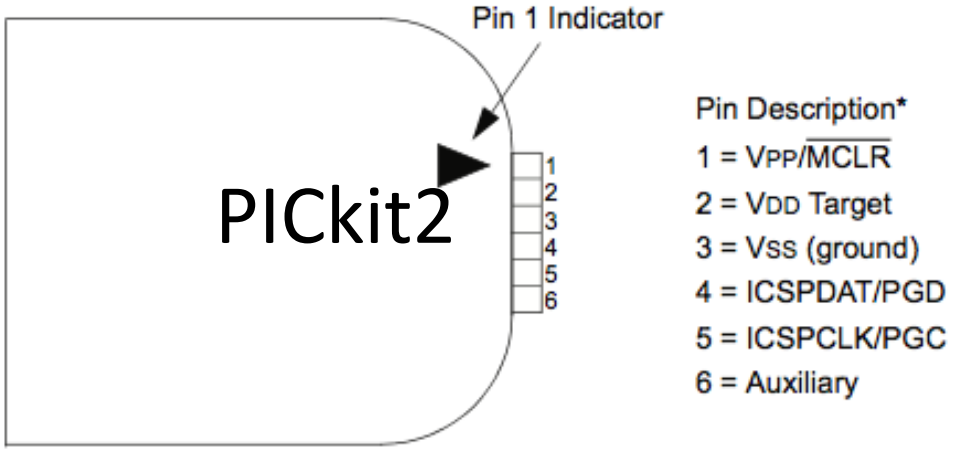
A programokat legegyszerűbben egy **PICkit2** programozóval tölthetjük le, amely saját programletöltő szoftver alkalmazással rendelkezik, tehát az **MPLAB IDE** telepítése nélkül is használható. Az ***AUX*** lábat esetünkben nem kell bekötni.

Figyelem! Programozás módban a VPP lábon 10 – 13 V-os feszültség jelenik meg!

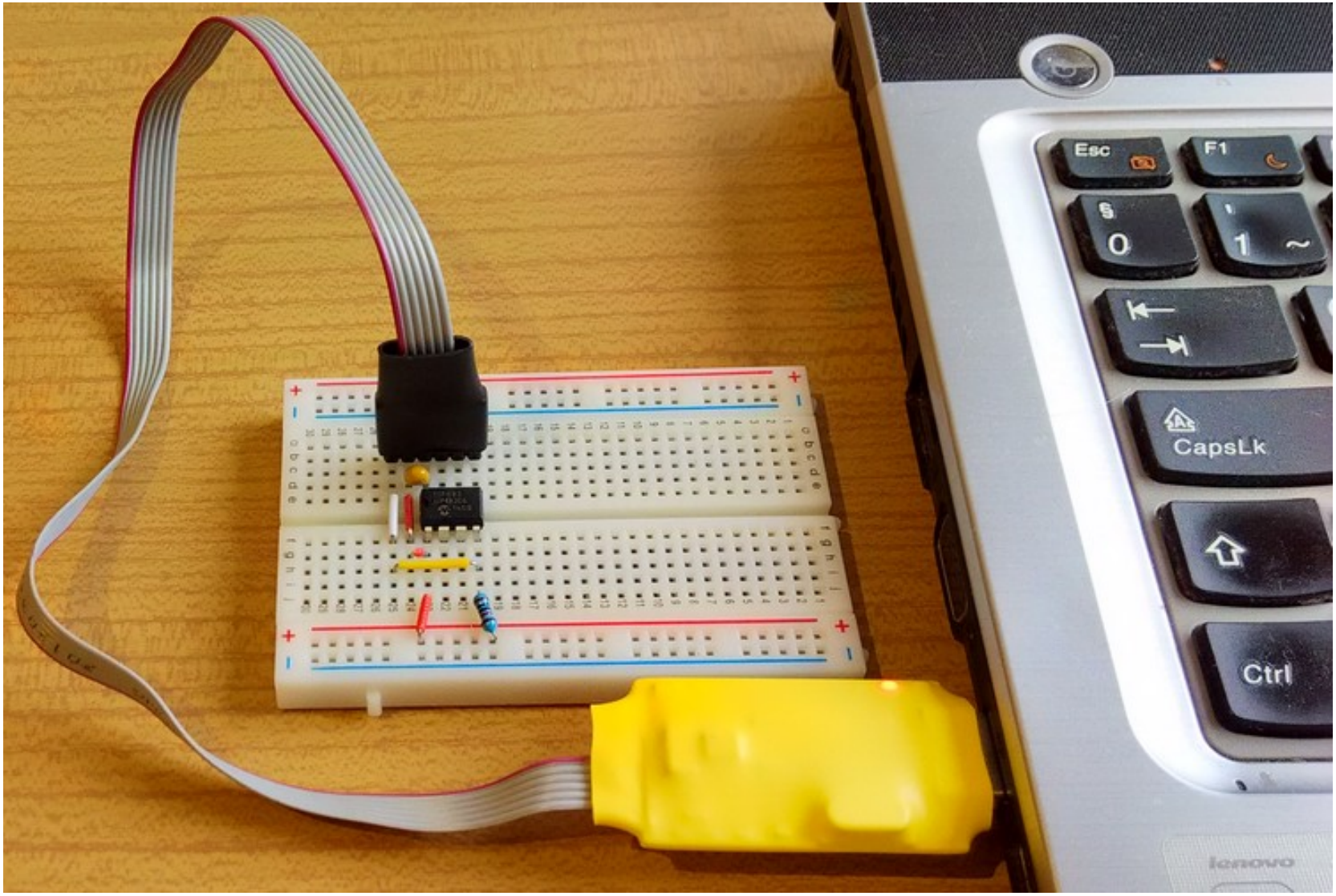
Programozás közben a kommunikáció az **ICSPDAT** és **ICSPCLK** lábakon folyik.







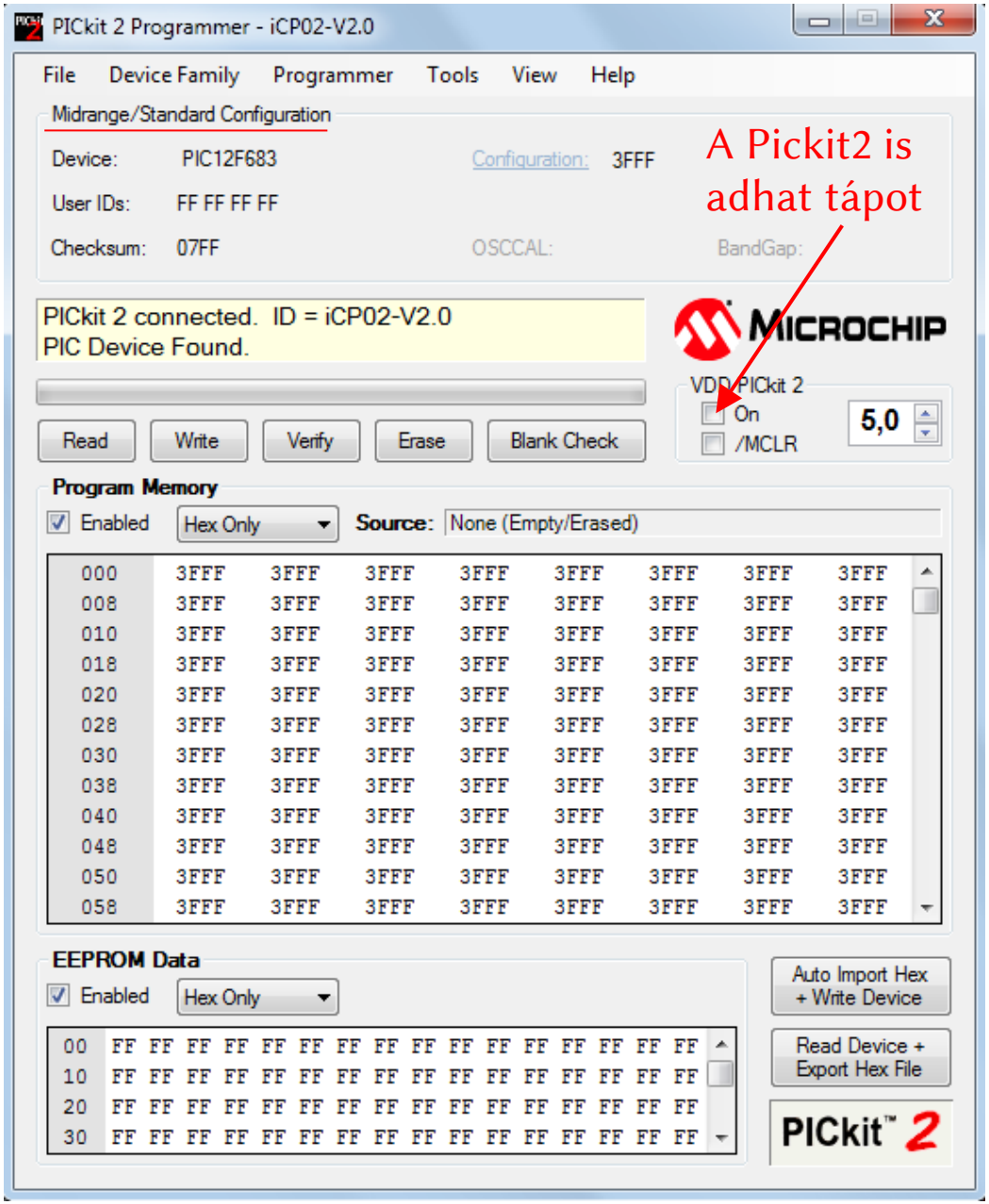
Programok letöltése

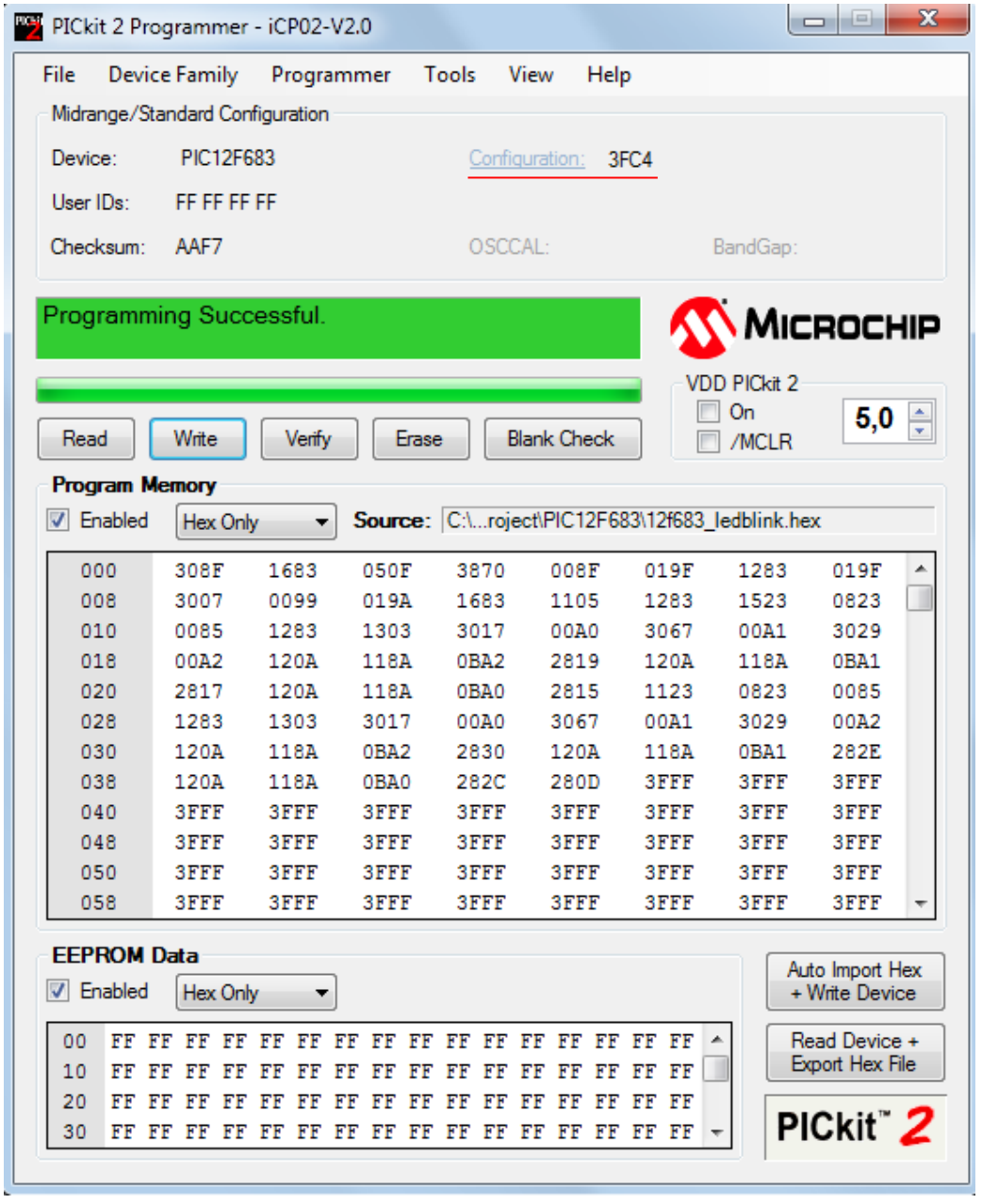


Programletöltés **ICP02v2 programozóval** (PICkit2 klón)

A PICkit 2V2 önálló alkalmazás ( )

Letöltés itt





A JAL programnyelv

- A **JAL** (Just Another Language) Pascal-szerű programnyelvet és a fordítóprogram első verzióját **Wouter van Ooijen** alkotta meg 2003-ban a Microchip PIC mikrovezérlőkhöz. 2006-ban **Kyle York** nekilátott a fordító átdolgozásának (JALv2).
- A **JALLIB** nemzetközi munkacsoport ezt tesztelte, kiegészítette perifériakönyvtárakkal és mintaprogramokkal. A szoftver ingyenesen használható, s a Microchip 8 bites mikrovezérlőit (baseline, midrange, enganced midrange és a PIC18F típusokat) támogatja
- Letöltések, dokumentáció: justanotherlanguage.org
  (a jallib_full-1.8.0.zip csomag a fordító mellett a JALEDIT IDE-t is tartalmazza)
- JALLIB Wiki: github.com/jallib/jallib/wiki
- JALLIB forráskód: github.com/jallib/jallib

LED villogtatás (JAL nyelven)

12f683_ledblink.jal

```
1  -- -------------------------------------------------
2  -- LED villogtatás Microchip pic12f683 mikrovezérlővel
3  -- -------------------------------------------------
4  include 12f683                                  -- PIC céláramkör
5
6  pragma target CLOCK      8_000_000              -- oszcillátor frekvencia
7  pragma target OSC        INTOSC_NOCLKOUT        -- belső oszcillátor 8MHz-en
8  pragma target WDT        disabled
9  pragma target PWRTE      enabled
10 pragma target MCLR       internal               -- MCLR láb GPIO legyen
11 OSCCON_IRCF    = 0b_111                         -- Fosc = 8 MHz beállítása
12
13 enable_digital_io()                             -- mindegyik GPIO digitális legyen
14 alias   led       is pin_A2                     -- GPIO2-re kötjük a LED-et
15 pin_A2_direction =  output                      -- GPIO2 legyen kimenet
16 --
17 forever loop                                    -- Végtelen ciklus
18    led = on                                     -- LED be (GPIO2 = 1)
19    _usec_delay(250_000)                         -- 250 ms várakozás
20    led = off                                    -- LED ki (GPIO2 = 0)
21    _usec_delay(250_000)                         -- 250 ms várakozás
22 end loop
```

Ez a sor csak a fordítónak szól

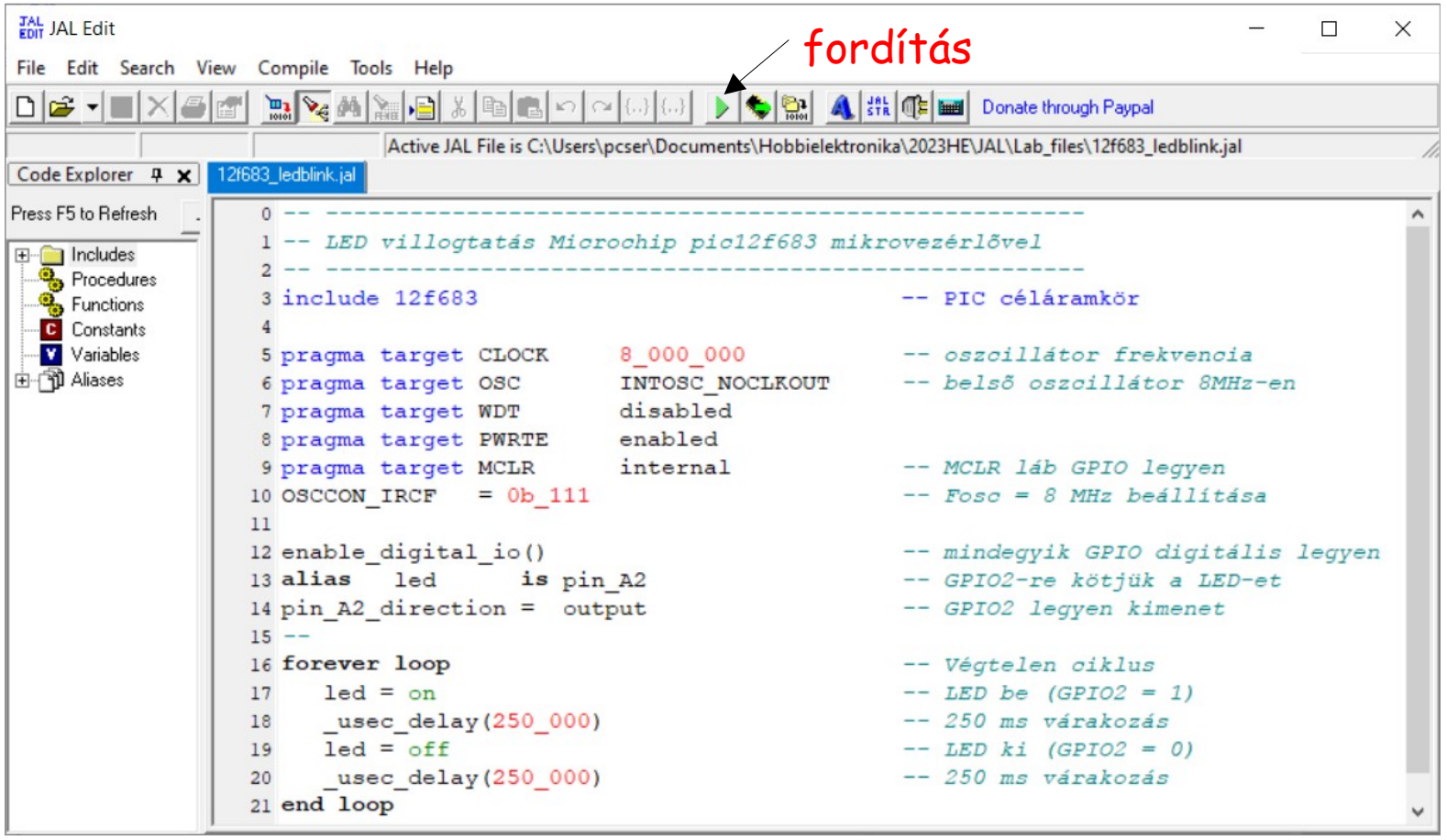
Konfigurációs bitek megadása

JAL programok lefordítása

- A jallib-1.8.0.zip vagy a jallib_full-1.8.0.zip csomag letöltése és pl. a C:\jallib1.8 könyvtárba történő kibontása után a fordítás parancssorból indítva így néz ki:

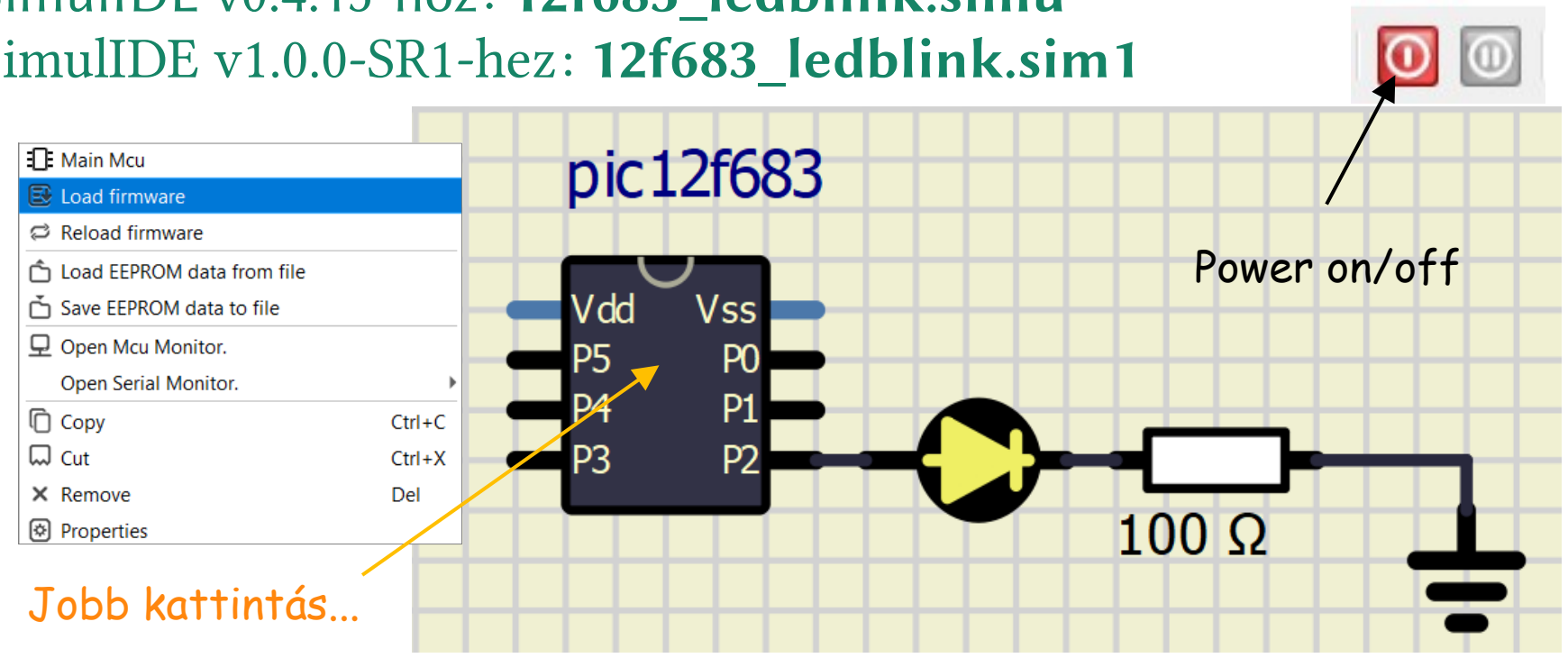
```
c:\jallib1.8\compiler\jalv2_64.exe -s c:\jallib1.8\lib 12f683_ledblink.jal
```

- A másik lehetőség a **JALEDIT** grafikus fejlesztői környezet használata, amelyhez a jallib_full-1.8.0.zip csomagot kell letölteni



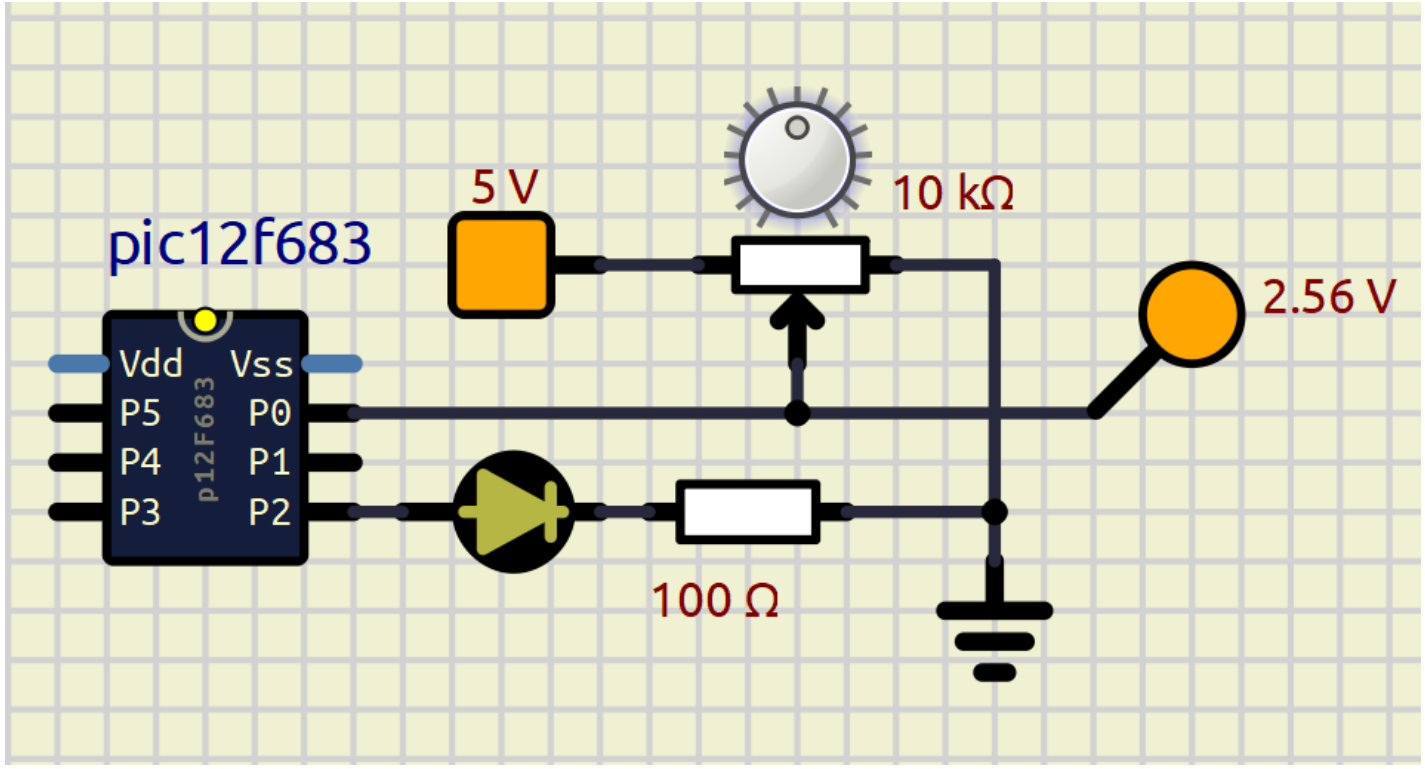
LED villogtatás (szimuláció)

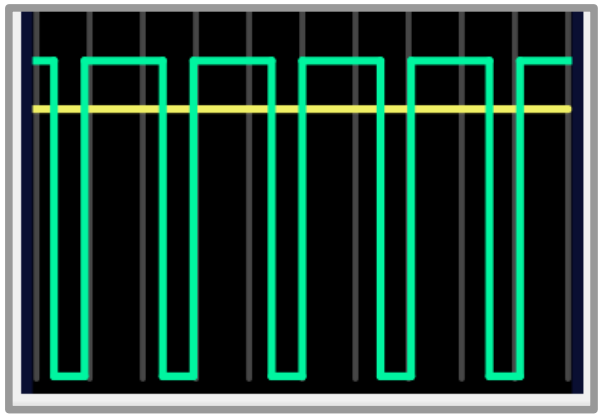
- Kössük egy LED anódját a **GPIO2** (PORTA 2. bit) kivezetésre!
- Kössük a LED katódját egy áramkorlátozó ellenálláson keresztül a **VSS** (GND) lábra!
- Töltsük be a mikrovezérlőbe a **12f683_ledblink.hex** programot!
- Indítsuk el a szimulációt és kapcsoljuk be a tápfeszültséget!
- A programfutást a SimulIDE szimulátorban is kipróbálhatjuk:
  - SimulIDE v0.4.15-höz: **12f683_ledblink.simu**
  - SimulIDE v1.0.0-SR1-hez: **12f683_ledblink.sim1**



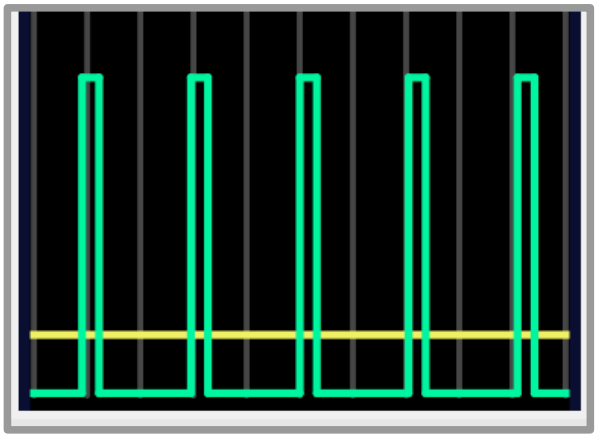
PWM jel szabályozása potméterrel

- A **GPIO0** (PORTA 0. bit) bemenetre kötött, potméterrel leosztott tápfeszültséget az ADC-vel megmérjük. A mért feszültséggel arányosan állítjuk be a **GPIO2** (PORTA 2. bit) kimeneten megjelenő, kb. 488 Hz-es PWM jel kitöltését
  - SimulIDE v0.4.15-höz: **12f683_pwm_adc.simu**
  - SimulIDE v1.0.0-SR1-hez: **12f683_pwm_adc.sim1**
  - Firmware: **12f683_pwm_adc.hex**





V = 3.6 V



V = 0.8 V

12f683_pwm_adc_2017.jal - 2/1.

- Megjegyzés: ez a 2017-es program a jallib 1.8-cal nem kompatibilis!

12f683_pwm_adc_2017.jal

```
-- -------------------------------------------------
-- Egycsatornás dimmer Microchip pic12f683 mikrovezérlővel
-- Rob Hamerling mintapéldája, apróbb módosításokkal
--
-- Leírás:
-- Ez a program egy egyszerű, egycsatornás dimmert valósít meg
-- egy ADC analóg bemenet és egy PWM kimenet felhasználásával.
-- Az MCU belső oszcillátorát használjuk alkatrész takarékossági okokból.
-- A PWM szabályozása 256 lépésben bőven elegendő ehhez a feladathoz,
-- ennek megfelelően 8 bites ADC felbontás is elegendő.
-- Fosc = 8 MHz és 16-os előosztás esetén a PWM frekvencia 488 Hz lesz.
-- -------------------------------------------------
include 12f683                                  -- PIC céláramkör

pragma target CLOCK     8_000_000               -- oszcillátor frekvencia
pragma target OSC       INTOSC_NOCLKOUT         -- belső oszcillátor
pragma target WDT       disabled
Progma target MCLR      internal
OSCCON_IRCF   = 0b_111                          -- Fosc = 8 MHz beállítása

enable_digital_io()                             -- GPIO digitális mód engedélyezés
```

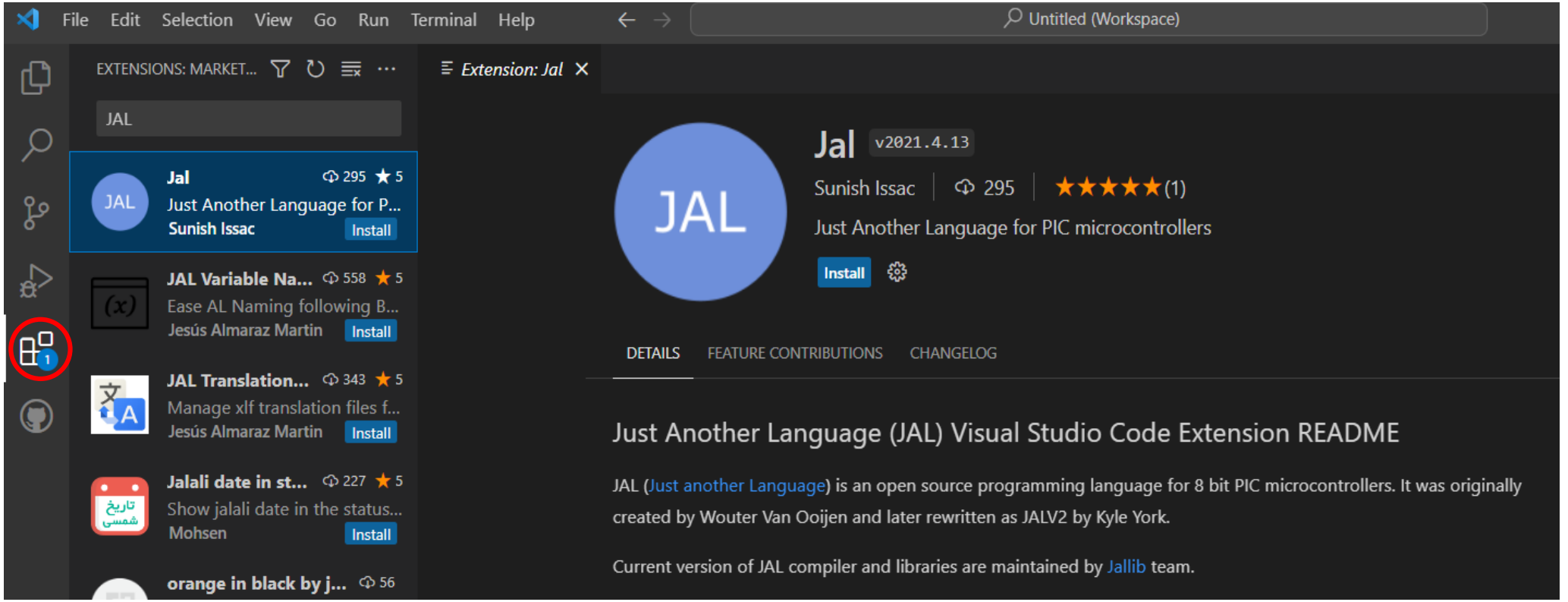
12f683_pwm_adc_2017.jal - 2/2.

- Megjegyzés: a megjelölt sorok a jallib 1.8-cal nem kompatibilisek!

```
-- ADC beállítás ----------
const byte ADC_NVREF = ADC_NO_EXT_VREF       -- nincs külső Vref
const ADC_RSOURCE = 10_000                   -- Bemenet: 10K potméter
const ADC_HIGH_RESOLUTION = FALSE            -- Kis felbontású ADC is elég!
include adc                                  -- ADC könyvtár becsatolása
adc_init()                                   -- ADC inicializálás

const  byte  ADC_CHANNEL = 0                 -- A potméter pin_AN0-hoz kötve
set_analog_pin(ADC_CHANNEL)                  -- a választott csatorna beállítása

-- PWM beállítás -----------
include pwm_hardware                         -- a PWM könyvtár becsatolása
pwm_max_resolution(16)                       -- Timer2 előosztási arány
pin_CCP1_direction = output                  -- a PWM-láb kimenet legyen
var byte measure                             -- ADC érték / PWM kitöltés

-- programhurok ------------
forever loop
   measure = adc_read_low_res(ADC_CHANNEL)   -- ADC kiolvasása
   pwm1_set_dutycycle(measure)               -- PWM kitöltés átírása
   _usec_delay(20_000)                       -- várunk egy kicsit...
end loop
```

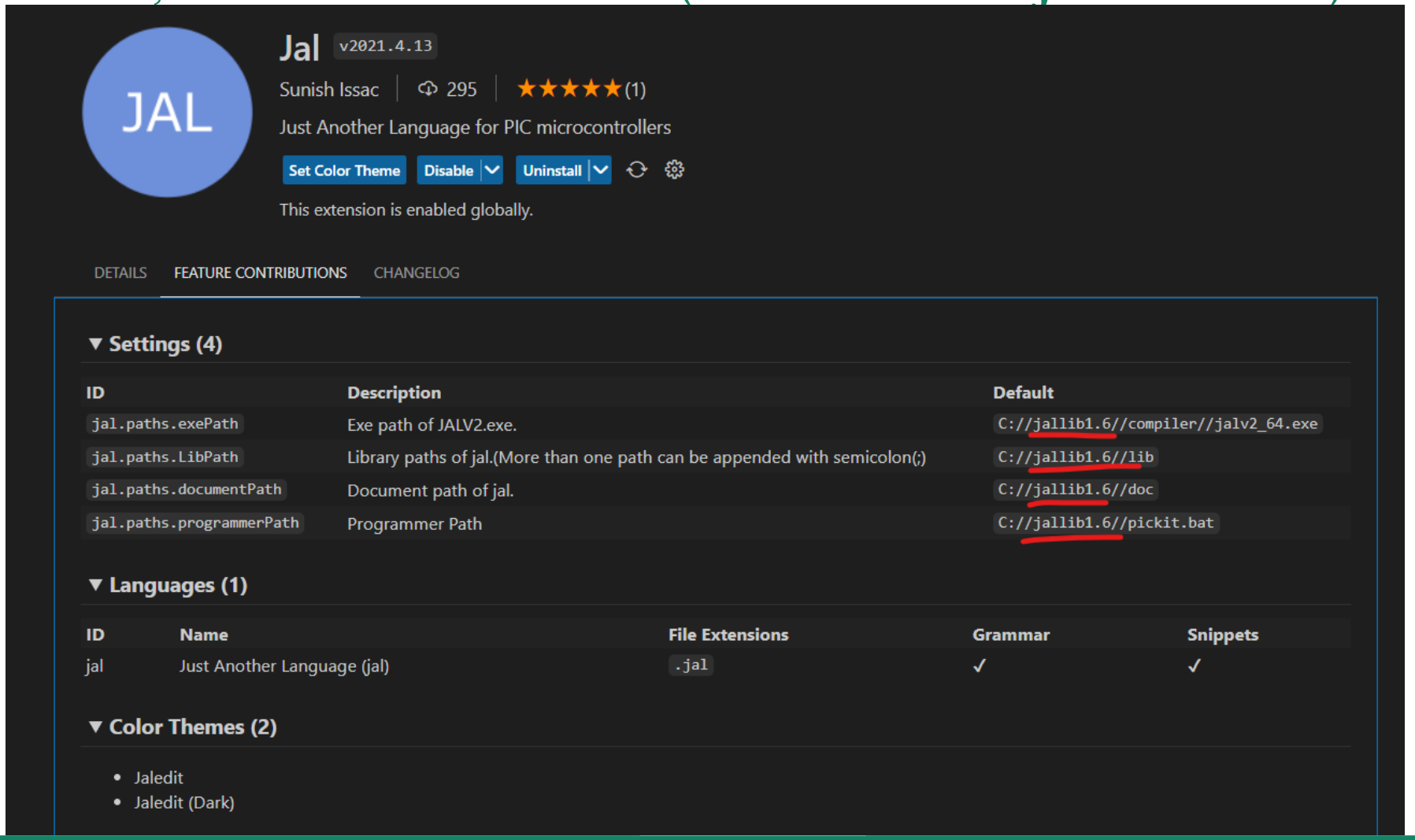
JAL programfejlesztés VS Code Editorral

- **Jaledit** helyett a korszerűbb **VS Code Editor**-t is használhatjuk **JAL** programok fejlesztéséhez, csak telepítenünk és konfigurálnunk kell hozzá a **VS Code JAL** bővítményét
- A tevékenység sávban a Bővítmények ikonra kattintunk, majd a keresőablakba írjuk be, hogy JAL, majd a felbukkanó JAL bővítménynél kattintsunk az **Install** gombra



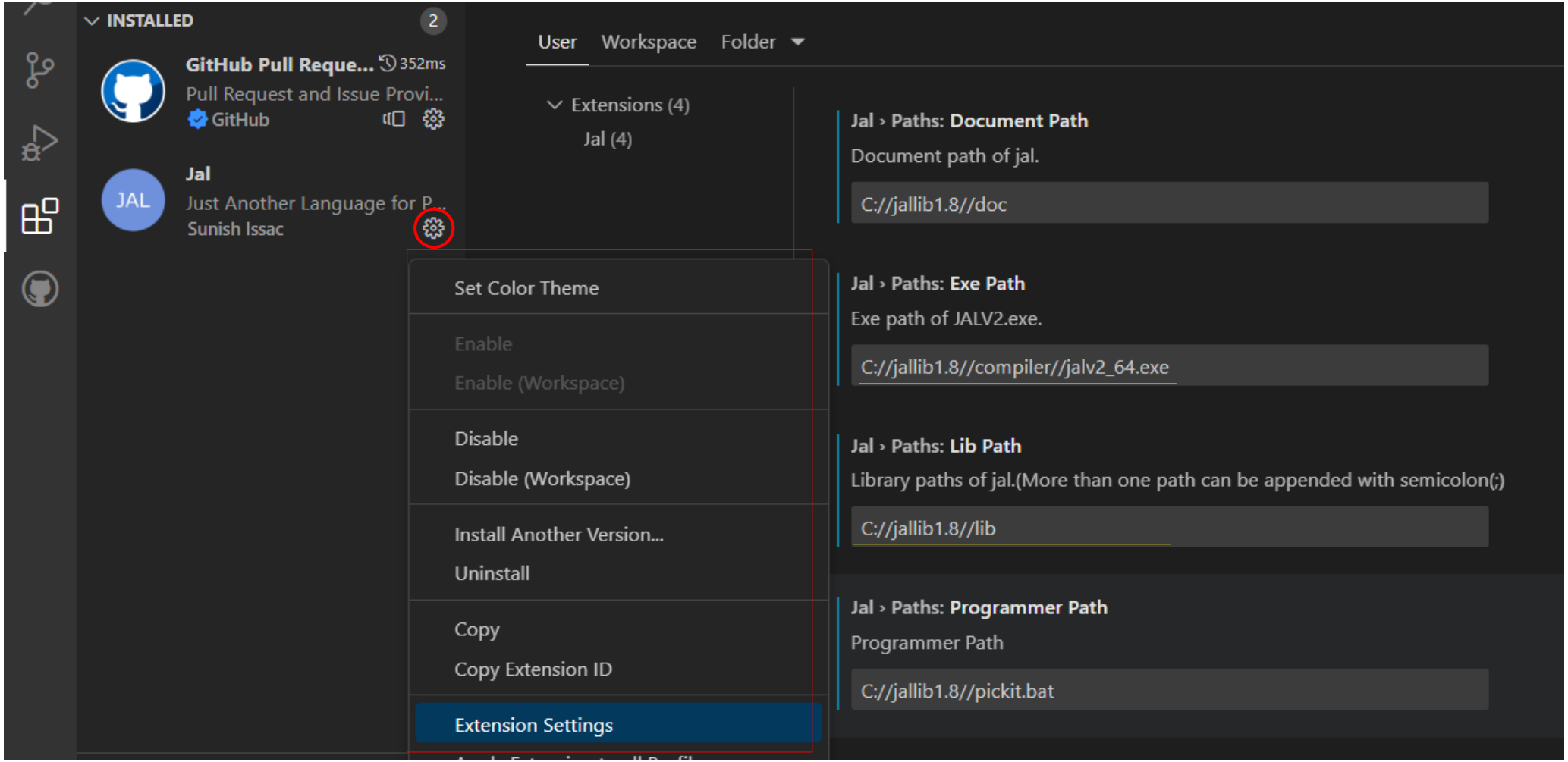
A JAL bővítmény konfigurálása

- A **Feature Contributions** fülre kattintva, azt látjuk, hogy az alapértelmezett **Jallib** telepítési könyvtár **C://jallib1.6,** nem oda mutat, ahová nekünk kellene (esetemben **C://jallib1.8** kell)



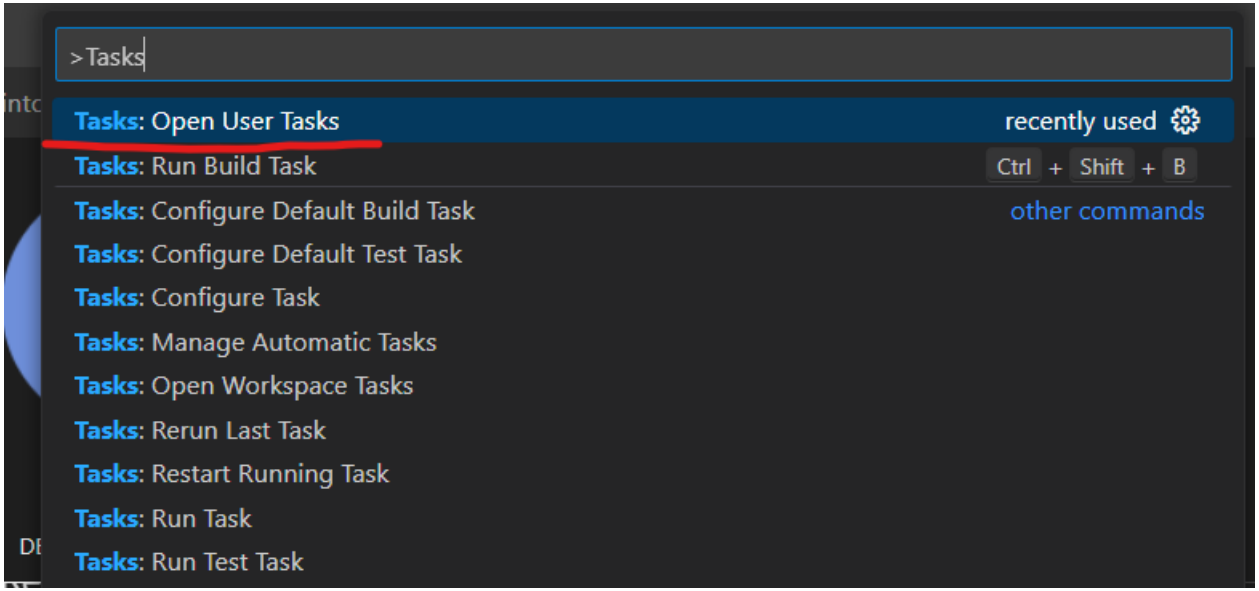
A JAL bővítmény konfigurálása

- Kattintsunk a pirossal jelölt fogaskerék ikonra, s a felbukkanó menüben válasszuk az **Extension Settings** menüpontot!
- A felbukkanó lapon át tudjuk írni az elérési útvonalakat

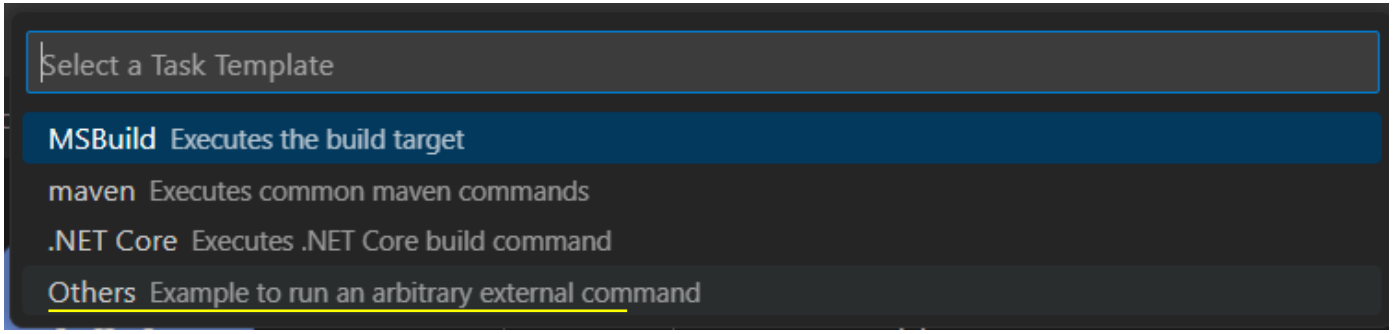


VS Code Tasks konfigurálása

- **Shift+Ctrl+P** nyomása után keressük a **Tasks: Open User Tasks**-ot!



- Kattintsunk a **Tasks: Open User Tasks** menüpontra, majd a felbukkanó listán válasszuk ki az **Others (Example to run arbitary command)** lehetőséget!



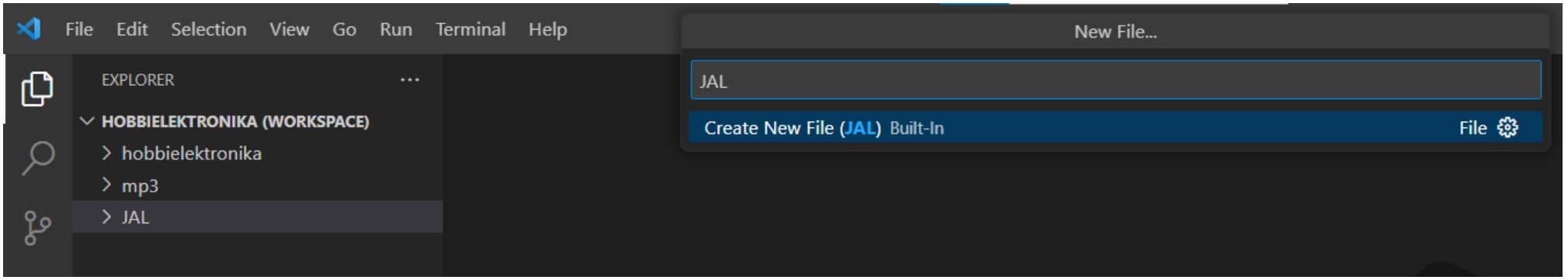
VS Code Tasks konfigurálása

- A megnyíló **task.json** ablakba másoljuk be az alábbiakat és mentsük el!

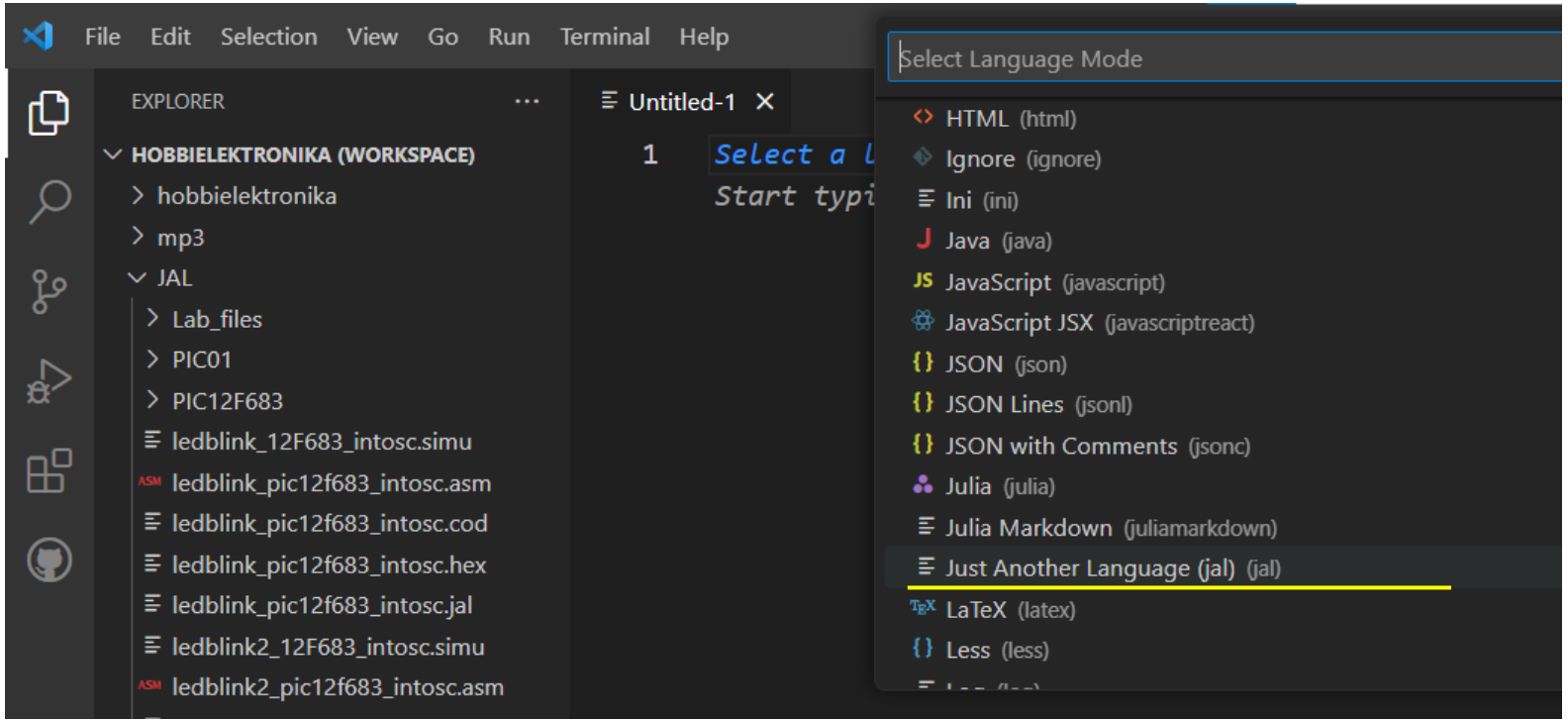
```
{   // prefilled tasks.json for compiling a JAL file
    "version": "2.0.0",
    "tasks": [
        {
            "label": "Compile JAL File",
            "type": "process",
            "command": "${config:jal.paths.exePath}",
            "args": [
                "${file}",
                "-s",
                "${config:jal.paths.LibPath}"
            ],
            "presentation": {
                "reveal": "always",
                "panel": "new"
            },
            "problemMatcher": [],
            "group": {
                "kind": "build",
                "isDefault": true
            }
        }
    ]
}
```

Új JAL fájl nyitása

- Egyik lehetőség: **File → New,** majd a **JAL** típus megadása



- Másik lehetőség: **Ctrl+N** majd **Ctrl+K** és **M** után típusválasztás



Aktualizáljuk a régi 12f683_pwm_adc.jal mintaprogramot az új Jallib-1.8 kiadáshoz!

12f683_pwm_adc_2023.jal

```
1   include 12f683                          -- PIC céláramkör
2   pragma target CLOCK     8_000_000       -- oszcillátor frekvencia
3   pragma target OSC       INTOSC_NOCLKOUT -- belső oszcillátor 8MHz-en
4   pragma target WDT       disabled
5   OSCCON_IRCF   = 0b_111                  -- Fosc = 8 MHz beállítása
6   enable_digital_io()                     -- mindegyik GPIO digitális legyen
7   -- ADC és AN0 konfigurálás ----------
8   const byte ADC_CHANNEL = 0              -- A potméter pin_AN0-hoz kötve
9   ANSEL_ANS0 = TRUE                       -- AN0 analóg móba állítva
10  pin_AN0_direction = input               -- AN0 bemenetre állítva
11  ADCON0_VCFG = FALSE                     -- VDD és VSS a referencia
12  ANSEL_ADCS = 0b011                      -- FRC legyen az ADC órajel
13  const ADC_RSOURCE = 10_000              -- Bemenet: 10K potméter
14  const ADC_HIGH_RESOLUTION = FALSE       -- Kis felbontású ADC is elég!
15  include adc                             -- ADC könyvtár becsatolása
16  adc_init()                              -- ADC inicializálás
17  -- PWM beállítás -----------
18  include pwm_hardware                    -- a PWM könyvtár becsatolása
19  pwm_max_resolution(16)                  -- Timer2 előosztási arány
20  pin_CCP1_direction = output             -- a PWM-láb kimenet legyen
21  var byte measure                        -- ADC érték / PWM kitöltés
22  -- programhurok ------------
23  forever loop
24     measure = adc_read_low_res(ADC_CHANNEL)  -- ADC kiolvasása
25     pwm1_set_dutycycle(measure)              -- PWM kitöltés átírása
26     _usec_delay(20_000)                      -- várunk egy kicsit...
27  end loop
```

set_analog_pin() helyett

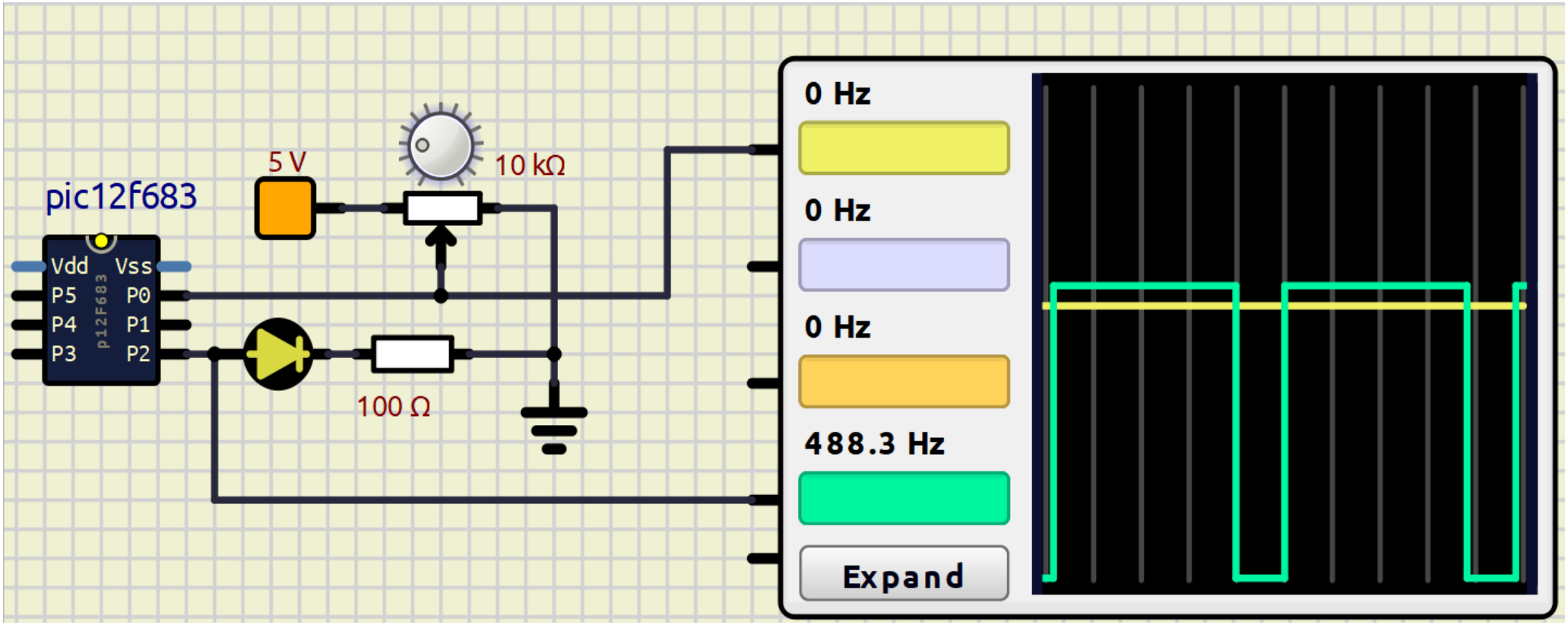
JAL programok fordítása VS Code-ban

- Ha a **JAL** forrásfájl szerkesztésre meg van nyitva, akkor **Ctrl+Shift+B** nyomása után megjelenik a **Compile JAL fájl** parancs, amit **Enter**-rel, vagy rákattintással lehet indítani
- Az üzenetek alul, a **Terminal ablakban** jelennek meg, mint például:

```
Executing task in folder hobbielektronika: C://jallib1.8//compiler//jalv2_64.exe C:\
Users\pcser\Documents\Hobbielektronika\2023HE\JAL\Lab_files\12f683_pwm_adc.jal -s
C://jallib1.8//lib

jal jalv25r7 (compiled Jan 29 2023)
generating p-code
574 tokens, 105669 chars; 2411 lines; 7 files
generating PIC code pass 1
generating PIC code pass 2
26 branches checked, 0 errors
133 data accesses checked, 0 errors
13 skips checked, 0 errors
writing result
C://jallib1.8//lib/pwm_ccp1.jal:124: warning: "pwm1_set_dutycycle() is deprecated,
.. use pwm1_set_dutycycle_percent() or pwm1_set_dutycycle_ratio()"
Code area: 229 of 2048 used (words)
Data area: 20 of 128 used
Software stack available: 73 bytes
Hardware stack depth 2 of 8
0 errors, 1 warnings
```

A puding próbája ...

- A **jallib-1.8.0**-val lefordított módosított program a várakozásnak megfelelően viselkedik a **SimulIDE 1.0.0 SR1** szimulátorban
  - SimulIDE v0.4.15-höz: **12f683_pwm_adc.simu**
  - SimulIDE v1.0.0-SR1-hez: **12f683_pwm_adc.sim1**
  - Firmware: **12f683_pwm_adc.hex**



12f683_pwm_adc futtatása