

2. A PIC MIKROVEZÉRLŐK FELÉPÍTÉSE, FEJLŐDÉSE

A következőkben összefoglaljuk a PIC mikrovezérlők felépítését, ami az ismétlésben már bemutatott részekből áll:

- CPU
- Programmemória
- Adatmemória
- Perifériák

A PIC mikrovezérlőket két módon osztályozhatjuk:

- Az egyik szempont a mikrovezérlő utasításainak a bitekben mért szélessége: ez először 12 bit szélességű volt, majd 14, 16, 24, illetve 32 bit szélességűre változott. Az utasítás-szélesség pedig meghatározza az utasítások bonyolultságát, a közvetlenül kezelhető adat- és programmemória nagyságát.
- A másik osztályozási lehetőség az adatok szélessége: ez 8, 16 és 32 bit lehet. Ez – mivel az egy utasítással kezelhető adatbitek számát határozza meg –, alapvetően meghatározza a számítási teljesítményt.

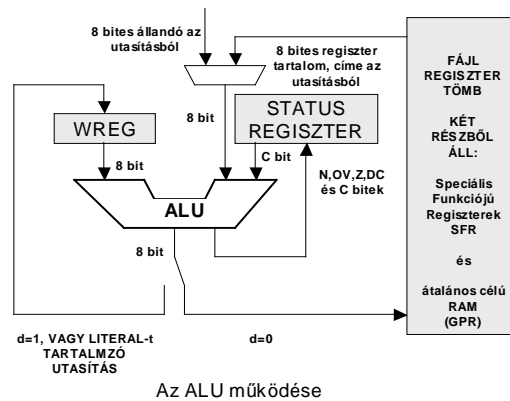
UTASÍTÁS HOSSZA	12 BIT	14 BIT	16 BIT	24 BIT	32 BIT
ADATMÉRET					
8 BIT	PIC10 PIC12 Baseline	PIC14 PIC16 Mid range	PIC18 High performance		
16 BIT	→			PIC24F PIC24H dsPIC30 dsPIC33	
32 BIT	→				PIC32

2.1. ábra
PIC mikrovezérlők

Szerencsés helyzetben vagyunk, mert a folyamatos fejlesztések a 8 és 16 bit adatszélességű PIC mikrovezérlőknél a felépítést alapvetően nem változtatták meg. A 32 bit adatszélességű mikrovezérlő processzormagja már nem Microchip-fejlesztés, ezért ott más megoldásokkal találkozhatunk. Ezért a következőkben ennek a figyelembevételével tárgyaljuk a felépítést.

2.1. ARITMETIKAI-LOGIKAI EGYSÉG (ALU)

Az utasítások általában a legtöbb művelethez két operandust használnak, amit érdemes úgy megoldani, hogy az egyik operandust az ALU-ban található speciális, kitüntetett regiszter tartalmazza, aminek a neve akkumulátor (A) vagy a Microchip szóhasználatában working regiszter (W). A másik operandus az utasításban szereplő című adatmemória-rekesz tartalma. Ezzel és az akkumulátor tartalmával végezzük el a műveletet. Célszerűen az eredmény is az egyik regiszterben keletkezik, amit a továbbiakban felhasználhatunk. Az ALU és a működés vezérlését végző, önállóan meg nem jelenő vezérlőegység együttesen alkotja a CPU egységet. Az ALU működése közben számos bitműveletet végez, ezért az utasítás-készletet ennek kezelésére is fel kell készíteni.



2.2. ábra
8 bites ALU

2.2. PROGRAMMEMÓRIA

A programmemória egyenként megcímezhető, utasításokat tartalmazó tároló regiszterek halmaza. Mérete határozza meg az adott típusba elhelyezhető program nagyságát. Az éppen végrehajtandó utasítás címe az utasításszámlálóban van. Az utasításszámláló bitszáma határozza meg a memória maximális méretét. Ez csupán egy lehetőség, általában fizikailag kisebb memóriaterület áll rendelkezésre.

A programmemória tartalmazza a mikrovezérlő programját. A programmemóriában lévő utasítás szélessége meghatározó, hiszen ebben van kódolva a művelet és az operandus(ok). Ezért a PIC-ek fejlődését, az utasítások növekvő szélességével is jellemezhetjük: kezdetben ez 12 bit volt, majd 14, 16, 24 és 32 bit szélességűre változott, ami egyre összetettebb utasítások kialakítását tette lehetővé. A programmemória nagysága („hosszúsága”) is változik, attól függően, hogy mekkora programot tudunk elhelyezni benne.

2.2.1. Programmemória-típusok

A programmemória legfontosabb jellemzője az, hogy a benne lévő tartalom esetleg csak az újraprogramozásával módosítható, ami fejlesztéskor gyakran, üzemszerű működés során ritkán (vagy soha) nem módosul. Ezért az első PIC processzorok minden típusa két formában volt elérhető:

- Kvarcüveg ablakos formában, aminek beprogramozott tartalmát ultrabolya fénnel törölhetjük (EPROM) – ez jó a programfejlesztéskor.
- Egyszer programozható kivitelben. Ilyenkor a tok csak egyszer programozható, és tartalma nem változtatható.

Gondolható, hogy a rendszerfejlesztés ilyen ablakos eszközzel nagyon nehézkes volt, a beírt programot a mikrovezérlőt a foglalatába helyezve próbálhattuk ki, és az esetleges hibás működés kiderítéséhez sokszor komoly nyomozómunkát kellett végezni. Igaz – mivel az ultrabolya fénnel való törlés elég időigényes volt –, elegendő idő állt rendelkezésre a „Hol hibáztam?” kérdés megválaszolására.

Az ilyen fejlesztés másik problémája az egyszer programozható eszközök programozásánál jelent meg. A programozó élete egy rettegés volt: „Ok. Kipróbáltam, működött, de minden lehetséges eseményt teszteltem?” Ez különösen akkor vált érdekessé, ha vagy nagy darabszámú termékbe került a tok, vagy esetleg néhány elkészült termék mondjuk Szibériában landolt...

2. fejezet: A PIC mikrovezérlők felépítése, fejlődése

Az első, elektromosan újraprogramozható EEPROM-ot tartalmazó tok a legendás **PIC16C84**-es típus volt. Aki tehette, ezt használta, mert ennek újraprogramozása nagyon egyszerű és gyors volt. Termékbe beépítve már nem okozott nagy gondot az esetleges újraprogramozás. (Persze a szibériai probléma továbbra is megmaradt...).

Igazi változást jelentett, mikor a Microchip bevezette a **PIC16F87x** családot. Ez már flash EEPROM programmemóriával rendelkezett, és megjelent a hibakereső (debug) lehetőség. *(A ma már jól ismert flash memóriára jellemző, hogy programozása nem bájtonként, hanem több szomszédos bájtból álló blokkokban történik.)*

Az ilyen tokkal történő fejlesztés során, kiegészítve az ICD1 hibakereső-programozó eszközzel, lehetővé vált a program futásának nyomon követése. Megállíthatjuk a program futását adott programszámláló értéknél (ez a töréspont), és a megállás után a regiszterek tartalmát áttöltjük az MPLAB IDE környezetbe, és megvizsgálhatjuk tartalmukat, pontosan úgy, mint szimulációnál.

Mivel a család változó lábszámú és változó programmemória-méretű tagokat tartalmazott, különféle perifériákat tartalmazó kiépítésben állt rendelkezésre, ezért lehetővé vált a PIC16 összes családtagjának fejlesztése. Az addig több lábon történő tokprogramozást felváltotta a két jelvezetékkel használatos megoldás. A **PIC16F87x** eszközökön kifejlesztett, tesztelt programokat könnyű volt az egyszerű programozható céleszközbe átvinni.

A PIC18-as család bevezetésétől kezdve többségben már csak EEPROM-alapú programmemóriát tartalmazó eszközök vannak; az új típusoknál ez természetes, míg a továbbra is népszerű és sok termékben megjelenő régebbi típusokat (PIC12Cxx, PIC16C5x, PIC16Cxx) is átterveztek flash memóriásra.

Ezek után nézzük meg, hogy milyen programmemória-típusokat biztosít a Microchip a fejlesztések optimalizálására.

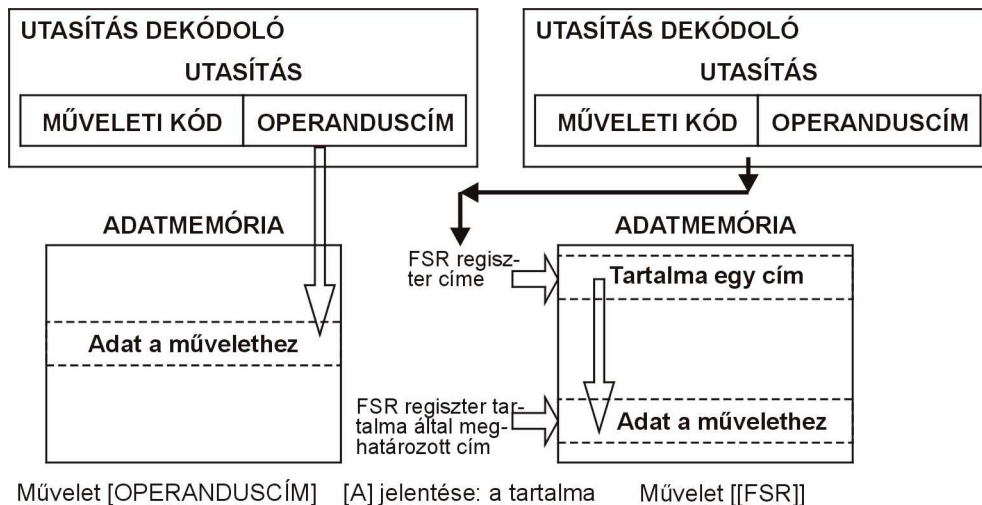
- **Flash (elektromosan újraprogramozható) memória:** a jelenleg legnépszerűbb megoldás. Flash PIC mikrovezérlők esetén a programmemóriát törölhetjük és újraprogramozhatjuk. Ez számos előnnyel jár: a felhasználók a gyártási folyamat végén vagy akár a helyszínen programozhatják be a termékeiket. Lehetséges a kód utólagos módosítása, paraméterek átírása. Fontos tudni, hogy egy memóriacella programozási ideje néhány ms. Ezért a memóriacellákat nem egyenként, hanem blokkonként írják, illetve törlik. Adott típusnál a blokkok mérete fontos katalógusadat. Megjelenik az önprogramozási lehetőség (*self programming*). Ez azt jelenti, hogy távolról, egy kommunikációs kapcsolaton keresztül frissítjük a flash programmemóriát. Ennek megvalósításához a tervező egyszerű letöltőprogramot helyez el a memória egy kódvédett területére. Egy biztonságos internetes, rádiós, infravörös fényt használó vagy telefonvonalon keresztül belépünk a programba a tok USART, I²C™ vagy SPI™ soros interfészén keresztül. A letöltő programmal ezután a vonalon érkező programadatokat újraírhatjuk a programmemóriát. Jelenleg a Microchip két flash memóriatípust használ:
 - *Továbbfejlesztett enhanced flash:* 40 éves adatmegőrzési idő, önprogramozhatóság 2 V–5,5 V között, ICSP (*In-Circuit Serial Programming*) 5 V vagy 12 V-on; adat EEPROM 1 millió törlés/írási ciklussal.
 - *Szabványos flash:* 10 000 törlés/írási ciklus, 40 éves adatmegőrzési idő, ICSP képesség csak 12 V-on.
- **Egyszer programozható (ONE TIME PROGRAMMING – OTP) memória:** az OTP memóriájú PIC mikrovezérlőket beírt program nélkül gyártják, és szállítják a felhasználóknak. Ez azoknak hasznos, akik gyorsan akarják a gyártmányukat piacra dobni, és a gyakran kell a gyártmányt tokcserével frissíteni. Az ilyen típusoknál lehetséges a mikrovezérlők beforrasztott állapotában történő programozása: *áramkörben történő programozás*. Ez a megoldás nagyon rugalmas, csökkenti a fejlesztés idejét, a gyártás hatékonyságát növeli, és lerövidíti a gyártmány piacon való megjelenését. Lehetséges a gyártott rendszer kalibrálása a gyártás alatt, és egyedi azonosító kódokat is adhatunk a termékeknek a gyártás során. Programozása mindössze két I/O lábat igényel a legtöbb eszköznél.

- **Gyorsan gyártható (QUICK-TURN PROGRAMMING - QTP):** van lehetőség a tokok felprogramozására a tokok gyártásának egy adott fázisában. Ez azoknak a megrendelőknek ideális, akik nem saját maguk akarják a viszonylag nagy mennyiségű tokot beprogramozni, valamint a beírandó kód már biztosan hibamentes és nem változik.
- **Sorszámmal ellátott tokokat eredményező gyors gyártás (SERIALIZED QUICK-TURN PROGRAMMING - SQTPSM):** ennél a megoldásnál a QTP megoldás mellett még minden programozott tok saját egyedi azonosítót, sorszámot is kap.
- **Maszkolt (MASKED) ROM:** nagy mennyiségű tok azonos programmal történő használatához a programból gyártási maszkot generálva állítják elő a gyártás során memóriába kerülő programot.

2.3. ADATMEMÓRIA

Az adatmemória az adatszélességgel azonos bitszámú (8, 16 vagy 32 bit) megcímezhető regiszterekből épül fel. A Microchip erre a memóriára a **fájlregiszterek** kifejezést használja. Címzése két módon lehetséges:

- **Közvetlen (direkt) címzés:** az utasításban van az a cím, amelynek tartalmával kell elvégezni a műveletet. Szimbolikusan: *MOV REG, W - REG tartalma W-be kerül.*
- **Közvetett (indirekt) címzés:** ilyenkor az utasításban egy címhez hasonló bitszóport szerepel. Az utasításdekódoló ezt érzékelve egy adott regiszter, az FSR (=File Select Register) tartalmát tekinti címnek, és az azon a címen található regiszter tartalmával végzi el a műveletet. Szimbolikusan: *MOV [REG], W - REG-ben lévő címen található regiszter tartalma kerül W-be.*



2.3. ábra
Közvetlen és közvetett címzés

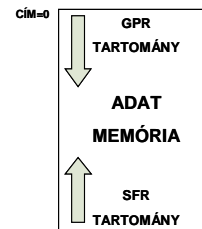
Az adatmemória két részre oszlik:

- a felhasználó által általánosan felhasználható adatregiszterekre, angol elvezése: **General Purpose Registers – GPR = általános célú regiszterek**, illetve
- a mikrovezérlő működéséhez és a perifériák kezeléséhez szükséges regiszterekre. Ennek angol neve: **Special Function Registers – SFR = speciális funkciójú regiszterek.**

2. fejezet: A PIC mikrovezérlők felépítése, fejlődése

8/12 **8/14** Mivel fizikailag ez a két adatmemória-tartomány nem különül el, ezért valamilyen módon szét kell osztani. A PIC18-as család megjelenéséig a tervezők úgy döntöttek, hogy a 0-s adatmemória-címtől helyezik el az SFR tartományt, majd felette helyezkedik el a GPR tartomány.

8/16 A PIC18-as családban már az adatmemória két végén helyezkednek el ezek az adatmemória-részek. A GPR terület a 0-s címen kezdődik, míg az SFR memória az adatmemória végén.



Illusztrációul a 12 és 14 bites PIC-eknél alkalmazott SFR kiosztás egy részletét láthatjuk a 2.4. ábrán. Ez a megoldás használható, de a perifériák számának – és azok kezelő regisztereinek – növekedése miatt egyre jobban eltolódik a GPR tartomány kezdete.

Address			
Indirect address	00h	PORTC	07h
TMR0	01h	PORTD	08h
PCL	02h	PORTE	09h
STATUS	03h	PCLATH	0Ah
FSR	04h	INTCON	0Bh
PORTA	05h	PIR1	0Ch
PORTB	06h		

2.4. ábra
Adatmemória-részlet

2.4. A PROGRAM- ÉS AZ ADATMEMÓRIA KAPCSOLATA

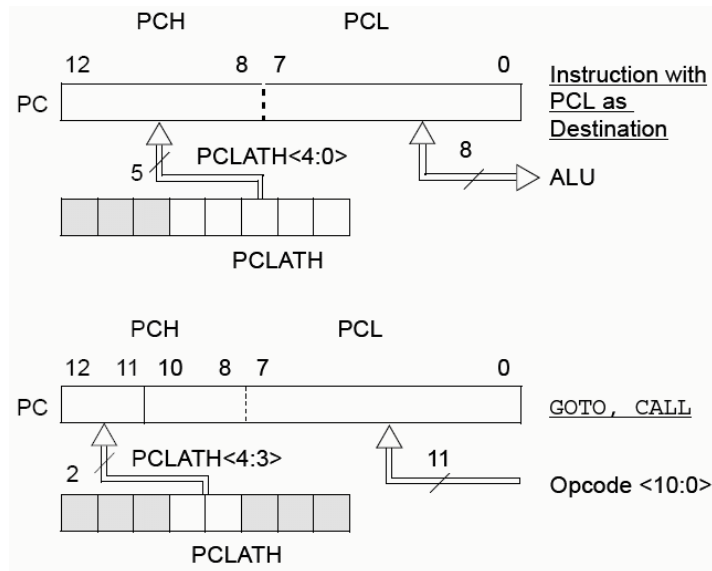
Sok esetben vannak olyan adataink, amelyek értékei nem változnak, például egy hőmérő kalibrációs táblázata: mit mérünk, és mi a tényleges hőmérséklet, amit ki kell jeleznünk. Ezek adatként viselkednek, úgy is használjuk ezeket, de változatlanóságuk miatt célszerű lenne nem változó, tartalmát megőrző memóriában tárolni. Három megvalósítás is lehetséges:

- 1) Kialakítunk egy **adat EEPROM memóriát**, és az adatokat itt tároljuk. Ehhez szükséges, hogy a kontroller rendelkezzen ilyen memóriával, és megfelelő méretű legyen. A tok felprogramozásakor visszük be ide azt a tartalmat, amit adatként használunk. Ez a megoldás a megadott feltételek mellett használható, de az elhelyezhető adatmennyiség erősen korlátozott, és ilyen esetben nem használjuk ki azt, hogy ez a memória programból írható is.
- 2) Megoldjuk, hogy az **adatokat a programmemóriában tároljuk** el, és gondoskodunk az így elhelyezett adatok kiolvasásáról.

8/14 **Első megoldásként** egy olyan utasítást definiáltak, aminek végrehajtásakor az utasításban operandusként szereplő állandó adat (konstans vagy más néven literál) a *W* munkaregiszterbe íródik, és így már adatként használható. Egyúttal az utasítás a szubrutinokat befejező és a szubrutin hívási helye utáni utasításra visszaugró RETURN utasítást is megvalósít. Az utasítás neve: **RETLW (Return with Literal in W)**. Ez a megoldás az adatsorok tárolására való, illetve a segítségével megadhatunk a szubrutin befejezésekor egy hibakódot is, ami jelzi a szubrutin sikerességét. Ezt az utasítást kompatibilitási okokból mind a 8, mind a 16 bites PIC mikrovezérlők utasításkészlete tartalmazza. Az ilyen utasítás használatához biztosítanunk kell, hogy az utasításslámlálóba be tudjuk írni az ezt az utasítást tároló memóriahely címét.

PIC mikrovezérlők alkalmazástechnikája

A 2.5. ábra szemlélteti ezt a megoldást, míg az ezt követő programrészlet a programban történő megvalósítást mutatja.



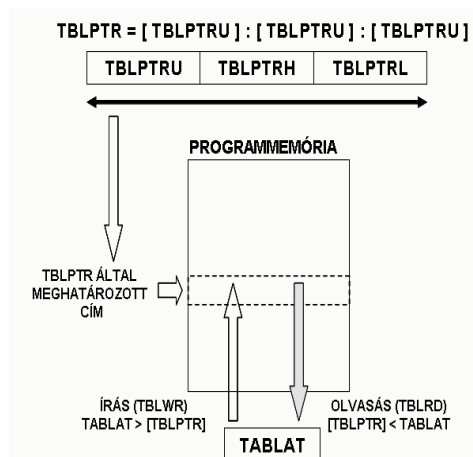
2.5. ábra
PIC16 táblakezelés

```

MOVLW LOW TABLE          ;W-BE TÖLTJÜK A CÍM ALSÓ 8 BITJÉT
ADDWF OFFSET,F            ;HOZZÁADJUK AZ ELTOLÁST OFFSET:=OFFSET+W
MOVLW HIGH TABLE        ;W-BE TÖLTJÜK A CÍM FELSŐ 5 BITJÉT
BTFSF STATUS,C           ;HA AZ ÖSSZEADÁS EREDMÉNYE
                           ;TÚLCSORDULT, ÁTLÉPTÜK A LAPHATÁRT
ADDLW 1                   ;HA IGEN FELSŐ CÍMET 1-GYEL NÖVELNI KELL
MOVWF PCLATH              ;A FELSŐ CÍM PCLATH-BA TÖLTÉSE
MOVF OFFSET,W             ;SZÁMÍTOTT ELTOLÁS W-BE ÍRÁSA
CALL TABLE               ;TÁBLAELEM KIOLVASÁS MEGHÍVÁSA
.
.
ORG 0X9FD
TABLE:
MOVWF PCL,F               ;SZÁMÍTOTT ELTOLÁS PCL-BE TÖLTÉSE
RETLW 'A'                  ;RETURN ASCII CHAR A
RETLW 'B'                  ;RETURN ASCII CHAR B
RETLW 'C'                  ;RETURN ASCII CHAR C
    
```

8/16 A másik, PIC18-as családnál megjelenő megoldás: a programmemória egy bájtját megcímező táblamutató bevezetése. A programmemória egybájtos memóriahelyét a **TBLPTR**-nek nevezett 22 bit szélességű regiszter tartalma címzi. Innen a megcímezett memóriabájt olvasásakor az adat a **TABLAT** elnevezésű regiszterbe kerül. Íráskor a TABLAT regiszter tartalma kerül a TBLPTR által megcímezett memóriahelyre (2.6. ábra). Fizikailag a TBLPTR regiszter három nyolcbites regiszterből épül fel:

TBLPTRU = TBLPTR[21:16] TBLPTRH = TBLPTR[15:8] TBLPTRL = TBLPTR [7:0]



2.6. ábra
PIC18 táblakezelés

Az adatmozgatás két utasítást igényel. Olvasáskor először kiolvassuk az értéket a TABLAT regiszterbe, utána az eredményt valahová tároljuk. Például:

```
TBLRD *           ;PROGMEM(TBLPTR) - > TABLAT
MOVFF TABLAT,REGI ;TABLAT TARTALMA REGI-BE
```

A hatékonyság érdekében vannak műveletek, ahol a táblamutató változtatása is automatikus, az utasításban összekapcsolódik a memóriatartalom-kezelő és a mutatót mozgó művelet:

TBLRD*,TBLWT*	TBLPTR tartalma nem változik
TBLRD*+,TBLWT*+	TBLPTR tartalma a művelet után eggyel nő
TBLRD*-,TBLWT*-	TBLPTR tartalma a művelet után eggyel csökken
TBLRD+*,TBLWT+*	TBLPTR tartalma a művelet előtt eggyel nő.

3) A programmemória azon része, amiben adatok vannak, az adatmemória egy adott címtartományában adatként „látszik”. Ezt részletesebben a 16 bites mikrovezérlőknél mutatjuk be.

2.5. MEGSZAKÍTÁS

Ha egy számítógépes rendszerben valamilyen esemény létrejöttét kívánjuk érzékelni, ezt szokásos módon kétféleképpen tehetjük meg. Az első módszernél a külső esemény létrejöttét egy bemenet változásának figyelésével érzékelhetjük.

Például ilyen megoldás alkalmazható, amikor egy billentyűzetről akarunk beolvasni. Bármelyik billentyű megnyomásakor a billentyűzet kimenetén lévő „adat érvényes” jel szintet vált. Ha ezt egy bemeneti port egyik bitjére kötjük, akkor az állapotának a programból való figyelése lehetővé teszi a billentyű megnyomásának az érzékelését, majd a kód beolvasását.

Ezt a módszert általánosan elterjedt kifejezéssel **pollingnak** (lekérdezésnek), azaz programozott átvitelnek hívják. Alkalmazása azonban lelassítja a rendszer tényleges működési sebességét, hiszen a mikroprocesszor az idejének nagy részét azzal tölti, hogy ciklikusan megvizsgálja a kijelölt bemeneti bit állapotát.

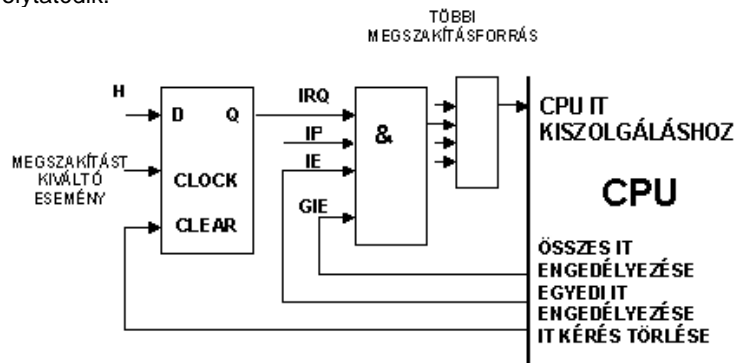
Sokkal szerencsésebb, ha az esemény maga jelzi a processzor számára állapotának megváltozását. Ez a megoldás a **megszakítás** vagy ismert angol kifejezéssel az *interrupt* (szokták IT-nek rövidíteni).

PIC mikrovezérlők alkalmazástechnikája

A megszakítás az eredetileg futó program utasításainak végrehajtását leállítja, és a processzor egy ún. megszakítási alprogramot (ISR – *Interrupt Service Rutin*) hajt végre, ami az esemény kezelését elvégzi, majd ennek befejeztével a processzor visszatér a megszakított program végrehajtására. Lényegében a programunk egy szubrutint hajt végre, majd visszatér folytatni az eredeti programot.

Az előbbi példánál maradva, a billentyű megnyomását jelző „adat érvényes” jel megszakítást okoz, a megszakítási alprogram elvégzi a lenyomott billentyűhöz tartozó kód beolvasását, majd utána folytatódik a megszakított program.

A processzor oldaláról a megszakítási lehetőség kialakítása azt kívánja meg, hogy legyenek olyan bemenetei, amelyek állapotainak megváltozásakor képes a processzor az éppen futó program utasításainak végrehajtását felfüggeszteni, a megszakított program programszámlálójának az értékét elmenteni, és helyébe a megszakítási alprogram kezdőcímét betölteni, és ilyen módon az alprogramot elindítani. A végrehajtás befejeztével (amit általában az utolsónak elhelyezett, speciális utasítás jelez) a programszámlálóba a megszakított program programszámlálójának elmentett értéke töltődik vissza, a megszakított program pedig folytatódik.



2.7. ábra
A megszakítás elve

A megszakítás olyan speciális szubrutinhívás, amelynél a hívás bekövetkezésének időpontját nem tudjuk.

Mivel több megszakítást alakítanak ki, a megszakítás kiszolgálásának első lépése a megszakítást kiváltó azonosítása. Ezért megszakításkor, egy ahhoz tartozó IR (*Interrupt Request*) bit – addig 0 állapotú – bit fog 1-re váltani. Ezek a megszakításokhoz tartozó IR bitek jelzik az adott megszakítás létrejöttét.

Ha a processzor több megszakítási vonallal rendelkezik, ezek mindegyikéhez egy-egy eseményt rendelhetünk hozzá. Olyan rendszerekben, ahol több esemény okozhat megszakítást, megtörténhet, hogy egyszerre egy időben két megszakítás is fellép. Ilyen esetben a megszakítások kiszolgálásának fontossági sorrendje – **prioritása** – dönti el a kiszolgálási sorrendet. Ezt az **IP** bit jelöli.

A program futása nem minden esetben szakítható meg káros következmények nélkül. Ezért a legtöbb rendszer biztosítja, hogy a megszakítások programból tilthatók, illetve engedélyezhetők legyenek, erre szolgálnak az **IE** (*Interrupt Enable*) bitek. Több megszakítás esetén a megszakításokat egyenként kell engedélyezni, illetve tiltani, ezért bevezették a **GIE** bitet, amivel egyedül az összes megszakítás engedélyezhető, illetve tiltható.

Ha több megszakításforrás van, akkor a PC-be be kell tölteni az adott megszakításhoz tartozó kiszolgálórutin (ISR – *Interrupt Service Routine*) kezdőcímét. Az így működő kialakítás a vektoros megszakítás. A **megszakításvektor** a megszakítás kiszolgálásakor betöltött rutin kezdőcíme.

2. fejezet: A PIC mikrovezérlők felépítése, fejlődése

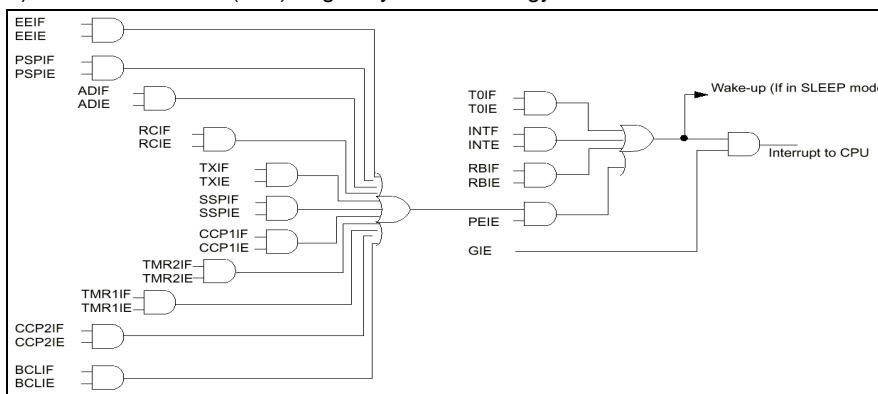
8/12 A legelső, **PIC16C5xx** család nem is rendelkezett megszakítással, minden eseményfigyelés csak lekérdezéssel (pollinggal) volt lehetséges. A következő típusokban megjelent a megszakítás, mindössze maximum két visszatérési címet tároló veremmel, ami azt jelentette, hogy vagy két egymásba skatulyázott szubrutinhívás, vagy egy megszakításban egy szubrutinhívás volt lehetséges. A megszakítás egyszintű, azaz egyszerre csak egy megszakítás kiszolgálása történhet, az egy időben bekövetkező megszakítás esetén a prioritást az ISR-ben való lekérdezési sorrend határozza meg. Több megszakításforrás lehetséges, de a megszakításoknak csak egy vektorcíme van (04h).

8/14 **PIC16-os** termékvonalnál a verem mélysége nyolcra növekedett.

A megszakítás létrejöttékor – mivel egy megszakítási cím van – még nem tudjuk, melyik forrás okozta a megszakítást. Ezért a megszakítási alprogram elején meg kell vizsgálni az egyes megszakításhoz tartozó IR bitek állapotát. Amelyik H állapotú, az a forrás okozta a megszakítást. Több forrás esetén a bitek lekérdezési sorrendje a prioritást is megadja, hiszen több IT egyidejű bekövetkezése esetén a lekérdezés sorrendje alapján az első IT-t okozó forrást szolgáljuk ki. A megszakítás kiszolgálása végén töröljük a D tárolót, amely az IT kérést tárolta a kiszolgálás alatt.

Több megszakítás a processzort a szundi (*sleep*) módból ébreszti, a megszakítás felismerése 3 utasításciklus, illetve 4 ciklus külső megszakításoknál. A megszakítás kiszolgálásakor fontos megőrizni a STATUS, a W regiszter, valamint a PCLATH regiszter tartalmát. Az egyes megszakításforrások engedélyezése-tiltása az INTCON, PIE1 és PIE2 regiszterek segítségével lehetséges. *A megszakítások globális engedélyezésére-tiltására a GIE bit szolgál.* A megszakítás kiszolgálásának kezdetekor a hardver törli a GIE bitet, és a visszatérési címet a verembe helyezi. Több perifériát tartalmazó típusoknál még az ezek megszakításait külön engedélyező PIE bit állítása is szükséges (háromszintű engedélyezés: egyedi – PIE – GIE).

A 2.8. ábra a **PIC16F87x** megszakításrendszerét mutatja. Az IF végződés a megszakításjelző bitet, az IE végzések a megszakítás engedélyezését jelölik. Látható az ábrán, hogy perifériák megszakításainak engedélyezéséhez az egyedi (xxIE), a közös periféria (PEIE) és az általános IT (GIE) engedélyező bitet is egybe kell állítani.



2.8. ábra
A PIC16F87x megszakításrendszere

8/16 A **PIC18Fxxx** család számos megszakítási lehetőséggel rendelkezik. Már két, eltérő prioritású megszakítási vektor van, ezért a minden megszakításhoz az eddigi, az adott megszakítást egyedileg engedélyező IE (*interrupt enable*) és a megszakításhoz tartozó esemény bekövetkezését jelző IF (*interrupt flag*) bitek mellé egy újabb bit társult. Ez a megszakítás prioritását jelző IP (*interrupt priority*) bit.

PIC mikrovezérlők alkalmazástechnikája

A megszakítások prioritásos kezelése csupán lehetőség, ha nem alkalmazzuk, akkor bármelyik megszakításakor a **0008h** cím töltődik be az utasításszámlálóba.

A prioritásos megszakításrendszer alkalmazásakor minden megszakításhoz a két prioritási szint valamelyikét rendeljük hozzá. Az alacsonyabb prioritású megszakítás bekövetkeztekor a **0018h** cím íródik be az utasításmutatóba, míg a magasabb prioritású megszakítások bekövetkezésekor a **0008h** cím töltődik. Úgy is fogalmazhatunk, hogy az IP bit értéke határozza meg azt, hogy melyik cím töltődik be az utasításszámlálóba.

Hogy működik a prioritásos megszakítási rendszer? Ha egy alacsonyabb szintű megszakítás kiszolgálása közben egy magasabb prioritású megszakítás következik be, akkor az alacsonyabb szintű megszakítás kiszolgálása felfüggesztődik, és a magasabb szinten lévő kerül végrehajtásra. Ennek befejeződése után a felfüggesztett alacsonyabb szintű megszakítás kiszolgálása folytatódik. Azonos szinten lévő megszakítások egymást nem szakítják meg.

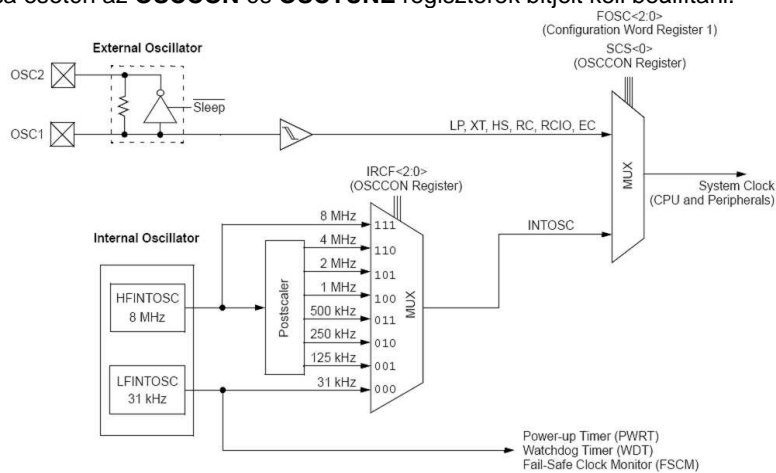
A 16 és 32 bites PIC mikrovezérlők megszakítás-kialakítását azok leírásánál ismertettjük.

2.6. RENDSZERELEMEK

Ebben a részben a mikrovezérlők működéséhez szükséges rendszerelemeket foglaljuk össze.

2.6.1. Oszcillátor, órajel

Az órajelgenerátor általános vázlatát a 2.9. ábrán látható. A külső oszcillátor-csatlakozás mellett egy belső oszcillátorblokk is megtalálható. Adott típusnál az adatlap tanulmányozása alapján tudhatjuk meg, hogy a konkrét esetben milyen megoldásokat valósítottak meg. Az órajel forrásának és típusának a beállítása a konfigurációs bitekkel lehetséges. Belső órajel választása esetén az **OSCCON** és **OSCTUNE** regiszterek bitjeit kell beállítani.



2.9. ábra
Órajelforrások

A működtető órajel előállítására számos lehetőség van, ezek a következők:

I. Külső kristály vagy kerámiarezonátor használata:

- 1) LP – *Low Power Crystal* – kis frekvenciájú (max 200 KHz) kristály vagy kerámiarezonátor
- 2) XT 0,4 – 4 MHz-es kristály vagy kerámiarezonátor