A SAB80515/80535-ÖS MIKROKONTROLLER

TARTALOMJEGYZÉK

1. A SAB80515-ÖS MIKROKONTROLLER HARDVER FELÉPÍTÉSE

1.1. Mikroszámítógépek felépítése

1.1.1. A Neumann-elv

Általánosan elterjedt tévhit, hogy a számítógépet Neumann János találta fel. Ez természetesen nem igaz, hiszen már az ókori Görögországban is építettek számítási feladatokat végző mechanikus szerkezeteket, mint pl. a számos fogaskereket tartalmazó **Antikythera-mechanizmus**. A szerkezetet 1900-ban találták meg egy elsüllyedt hajóban, de csak az utóbbi években tisztázódott, hogy csillagászati pozíciók kiszámítására használhatták. A 19. század elején az angol **Charles Babbage** tervezett egy hetedfokú polinomok kiszámítására szolgáló eszközt, amit **differenciálgépnek** nevezett el. Bár működőképes változata sohasem készült el, a 20. században Babbage tervei alapján készült replikája üzemképesnek bizonyult.

A 20. század elején már **elektromos** működtetésű, eleinte mechanikus, később **elektronikus** gépek is készültek. Ezeknek az volt a nagy hátránya, hogy minden egyes feladathoz újra és újra át kellett építeni, az adott feladatnak megfelelően újra kellett huzalozni a részegységek közötti kapcsolatokat, így programozták.

Időközben **Alan Turing** angol matematikus kidolgozta a végesautomaták matematikai modelljét, vagyis az "elméleti" számítógépet. Meghatározta, hogy milyen elemei legyenek és milyen műveleteket végezzen el.

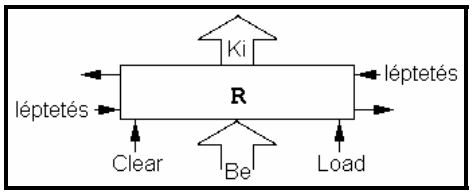
Neumann János érdeme, hogy a kezdeti zűrzavarban rendet teremtett. 1944-ben Los Alamosban **öt alapelvet javasolt** a számítógépek építéséhez:

1. A számítógép legyen teljesen elektronikus.
2. Használjon kettes számrendszert, vagyis legyen bináris.
3. Tárolja a memóriájában a számításokhoz szükséges adatokat.
4. Tárolja a memóriájában az elvégzendő műveleteket, vagyis a programot.
5. Legyen univerzális Turing-gép.

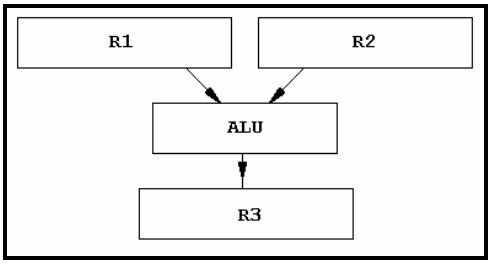
A fenti elvek alapján megvalósított számítógépek néhány alapvető **építőelemet** igényelnek: **aritmetikai-logikai egységet** (ALU), **memóriatömböt,** valamint **regisztereket** az utasítások és a számításokhoz használt operandusok átmeneti tárolására. Szükség van még mindezek összehangolt működésének biztosítására megfelelő **időzítésvezérlő** áramkörökre és a külvilág felé a kapcsolattartáshoz **perifériákra** (I/O eszközök).

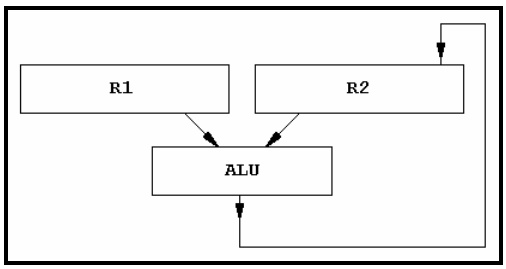
Egy univerzális számítógépben a tárolt utasításoknak nagyon sok bitből kell állni: tartalmaznia kell az elvégzendő művelet kódját, a címzésmódot, egy ún. címkonstanst, az első operandus címét, a második operandus címét, az eredmény címét és a következő utasítás címét. Az ilyen, **négycímes** utasításokat használó számítógép igen rugalmas, de nagyon bonyolult és drága, ezért ilyent soha nem építettek.

Az n bit tárolására alkalmas **regiszterek** is lehetnek **univerzálisak**: az adatokat beírhatjuk és kiolvashatjuk sorosan vagy párhuzamosan, léptethetjük jobbra vagy balra és törölhetjük is.



Egy aritmetikai-logikai művelet elvégzéséhez három regiszterre van szükség, kettő tárolja a memóriából kiolvasott két operandust, egy pedig a tárolandó eredményt. Csökkenthetjük a regiszterigényt, ha az eredményt az egyik bemeneti regiszterben tároljuk, de akkor ez a regiszter már nem univerzálisan használható, hanem speciális célú, ún. **dedikált** regiszter. Külön neve is van, **akkumulátornak** (Accumulator, **ACC**) hívják.





A négycímes rendszer egyszerűsítése több lépésben történt meg, míg eljutottak a ma már szinte kizárólagosan használt egycímes számítógéphez. Az egycímes rendszerben az utasításokat szigorúan a tárolásuk sorrendjében hajtjuk végre. Ehhez szükség volt egy új hardver elem, az **utasításszámláló** (Program Counter, **PC**) megalkotására. A PC mindig a következő utasítás címét tartalmazza. Ezenkívül számos olyan új utasításra is szükség volt, amiket a Turing-féle elméleti gép nem tartalmazott, pl. ugró- és elágazóutasítások.

A félvezetőtechnika megjelenése és fejlődése a 70-es évek elején lehetővé tette a mikroprocesszorok megjelenését. Ezek aritmetikai-logikai egységet, néhány regisztert és a külső memória ill. a perifériák kezelését lehetővé tevő cím- és adatbusz vezetékeket (lábakat) tartalmaztak.

A mikroprocesszorok által használt memóriarekeszek kapacitása (a memória szélessége) általában **8 bit**, amit **bájtnak** nevezünk. Így egy utasítás 1, 2 vagy akár 3 rekeszt is elfoglalhat. A processzor belső vezérlőáramkörének kell **"tudni"** hogy a következő utasítás eléréséhez a PC-t mennyivel kell megnövelni.

1.1.2. Mikroszámítógépek felépítése

A mikroszámítógépek olyan digitális áramkörökből álló rendszerek, amelyek adott feladatok végrehajtásának időbeli sorrendjét vezérlik. Az ilyen rendszerek néhány alapvető építőelemből állíthatók össze:

A rendszer magja egy ***mikroprocesszor***, amelyben az adatokat a számolómű (ALU : Arithmetic Logic Unit) dolgozza fel. Ebben az **akkumulátor** a legfontosabb adatregiszter, a műveletek eredményei többnyire itt tárolódnak. Az adatbitek száma processzorfüggő.

A mikroprocesszorban található ezenkívül egy kisebb-nagyobb további **regiszterkészlet,** amelyben az adatok (és címek) átmenetileg tárolhatók. Mivel a belső regiszterkészlet a változó adatok tárolására általában nem elegendő, a processzor képes a **címbuszon (egyirányú)** keresztül egy ***külső adatmemóriához*** hozzáférni és onnan az **adatbuszon (kétirányú)** adatokat beolvasni illetve oda adatokat tárolni. Az "írás" illetve "olvasás" vezérlésére az **vezérlőbusz (egyirányú)** szolgál.

Egy **„gépi kódú" (bináris)** program, amely utasítások sorozatából áll, meghatározza a mikroprocesszor számára, hogy mely adatokat hol, mikor és milyen módon kell feldolgozni. A végrehajtás úgy történik, hogy a mikroprocesszor az **utasításszámlálóban** (program counter) található címet kiadja a **címbuszra**, majd beolvassa a ***programmemória*** (ROM : Read Only Memory) adott címén található adatot az **adatbuszon** keresztül a **vezérlőműbe,** végül értelmezi és végrehajtja az utasítást szükség esetén további adatok beolvasásával. Ezután az utasításszámláló már a következő utasítás címét tartalmazza.

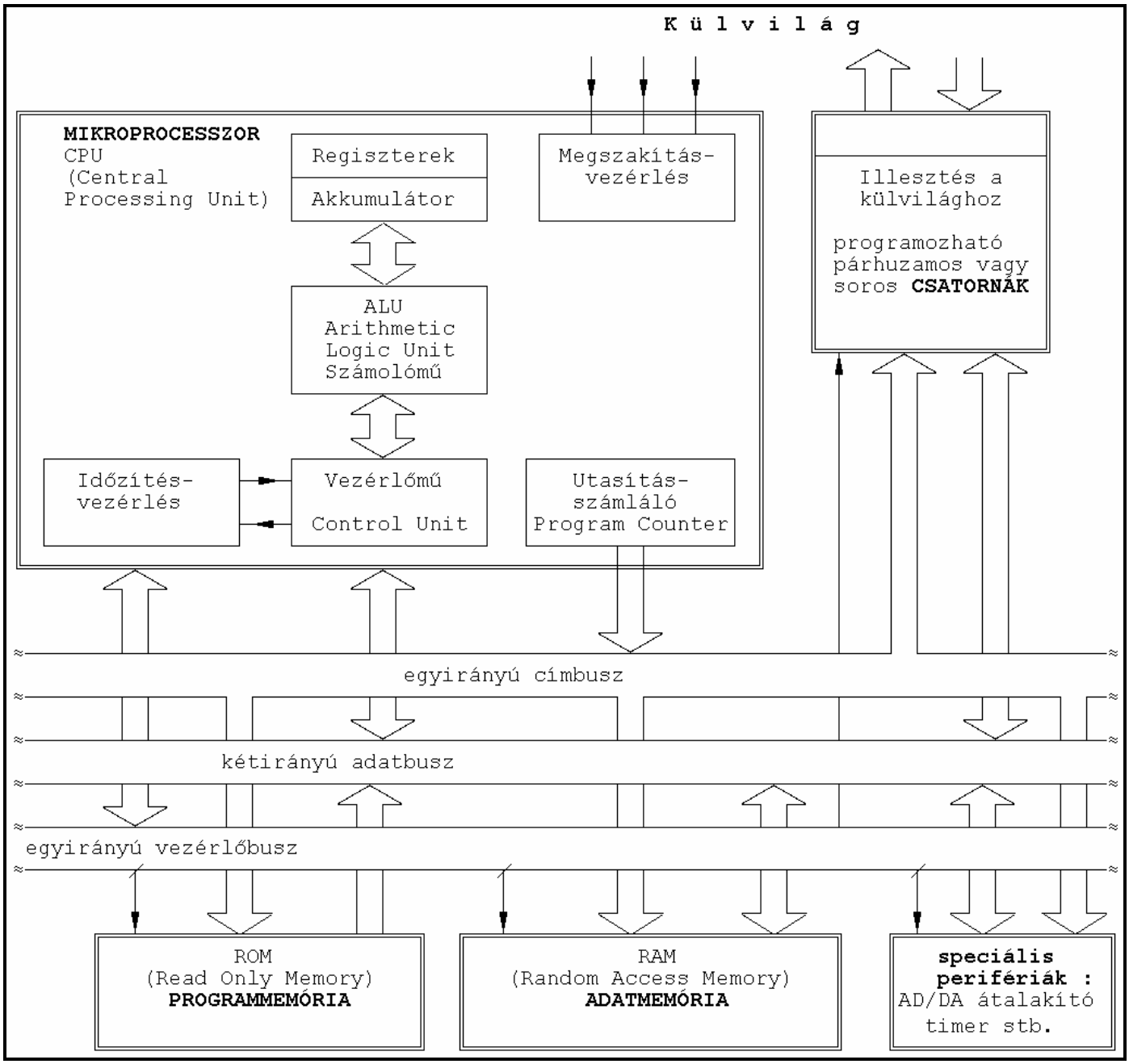
A mikroprocesszor **időzítés-vezérlése** gondoskodik valamennyi belső művelet helyes időbeli elvégzéséről, és vezérlőjeleket küld ("írás" vagy "olvasás") a külső építőelemeknek.

A mikroprocesszor további igen fontos tulajdonsága, hogy külső események hatására képes a pillanatnyi programvégrehajtást felfüggeszteni és az adott eseménynek megfelelő program végrehajtásával reagálni. Ezek az ún. **megszakítások** (interruptok) a processzor megfelelő bemeneteire kapcsolt aktív jelszinttel vagy éllel válthatók ki.

A mikroprocesszoros rendszerek további fontos építőelemei a programozható soros vagy párhuzamos **csatornák** a külvilág felé, a timerek időmérésre vagy eseményszámlálásra, az A/D és D/A átalakítók stb.

Az olyan mikroprocesszoros rendszereket, amelyek a fenti építőelemeket tartalmazzák, mikroszámítógépeknek nevezzük. Ha valamennyi építőelemet egy chip-be integrálják, akkor mikrokontrollerről (mikrovezérlő) beszélünk. A mikrokontroller chip tartalmaz még valamennyi adatmemóriát és többnyire programmemóriát is.

Mikroszámítógép felépítése :



1.2. A 80515/80535 mikrokontroller felépítése

-> CPU : 8 bites regiszterek és ALU; saját utasításkészlet

-> 128 járulékos regisztercím : (**SFR** : **S**pecial **F**unction **R**egister), 41 db
Akkumulátor A
Segédakkumulátor B
8 általános célú regiszter R0-R7, 4 regiszterbankban
Veremmutató
Utasításszámláló (16 bites)
különböző vezérlő- és kiszolgáló regiszterek

-> 256 bájt RAM : 256 memóriarekesz (8 bites)
NMOS : 40 memóriarekesz külső teleppel (VPD)

-> 8kB ROM : 8192 memóriarekesz (8 bites)
80535 : a 80515 ROM nélküli változata

-> 6 párhuzamos port : P0 - P5 (8 bites);
kétirányú, **nem** háromállapotú;
CMOS : P6 járulékos port, csak olvasásra

-> 1 soros port : Adás kimeneti shiftregiszterrel;
Vétel bemeneti shiftregiszterrel;
Baudrate-generátor(-> timerek)

-> 8 analóg bemenet : 8 bemenet analóg multiplexerrel,
mintavevő-tartó és A/D átalakító,
programozható referenciafeszültségek,
(alsó/felső határ, 16 fokozatban);
(CMOS : P6-on, csak olvasásra)

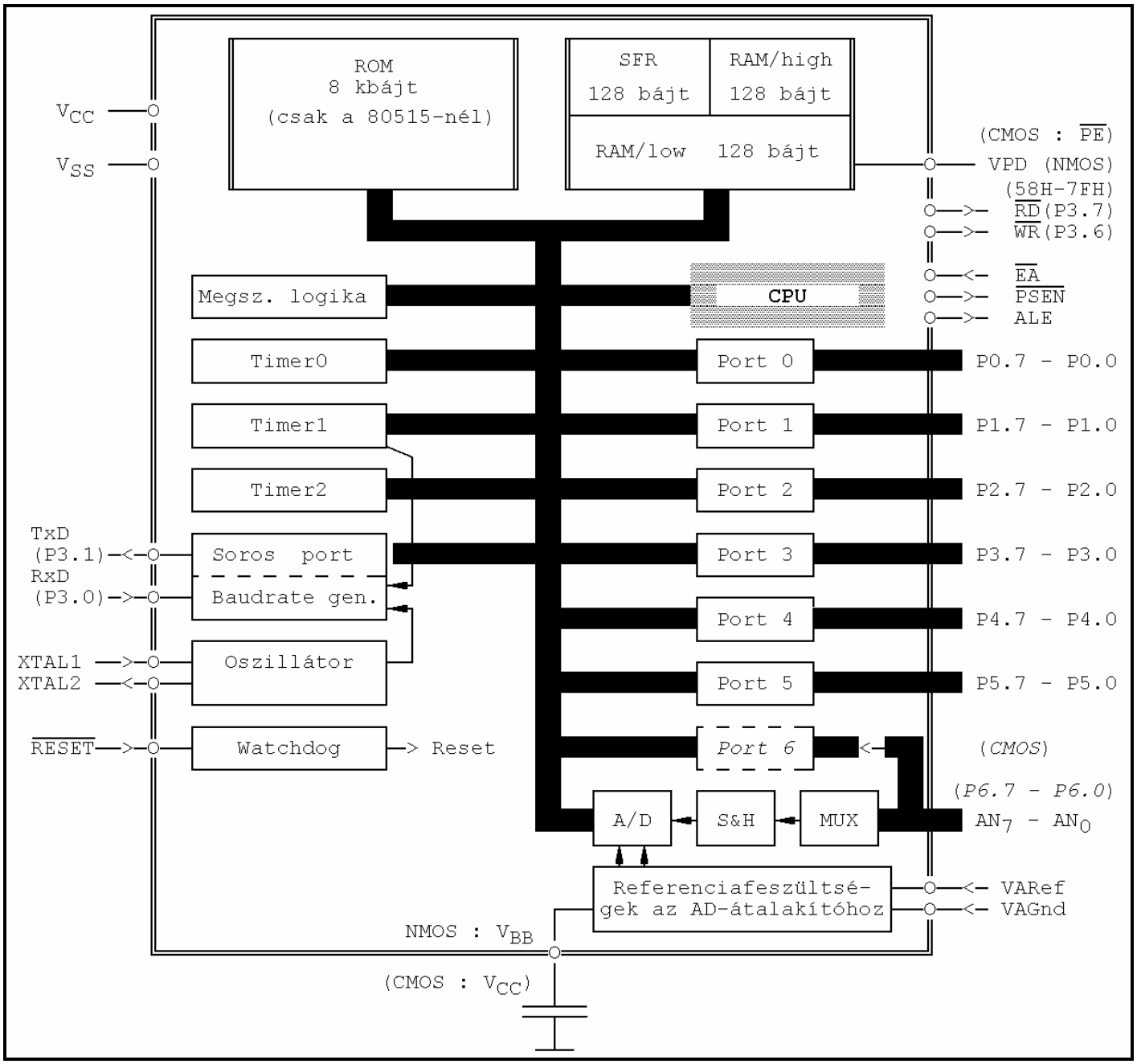
-> 3 számláló/időadó : 16 bites számlálók;
Időmérés;
Eseményszámlálás;
Időintervallum jelzés és mérés;
Baudrate-generátor a soros
csatorna számára
Impulzusszélesség moduláció (D/A átalakítás);

-> 1 watchdog : "lefagyás" elkerülésére

A 80535-ös vezérlőjelei külső memóriakezeléshez:

$\overline{EA}$	:	**E**xternal **A**ccess
ALE	:	**A**ddress **L**atch **E**nable
$\overline{PSEN}$	:	**P**rogram **S**torage **EN**able
$\overline{WR}$, $\overline{RD}$	:	**WR**ite- (írás) és **R**ea**D**- (olvasás) vezérlés (P3.6, P3.7)

A SAB80515/80535-ös mikrokontroller architektúrája:



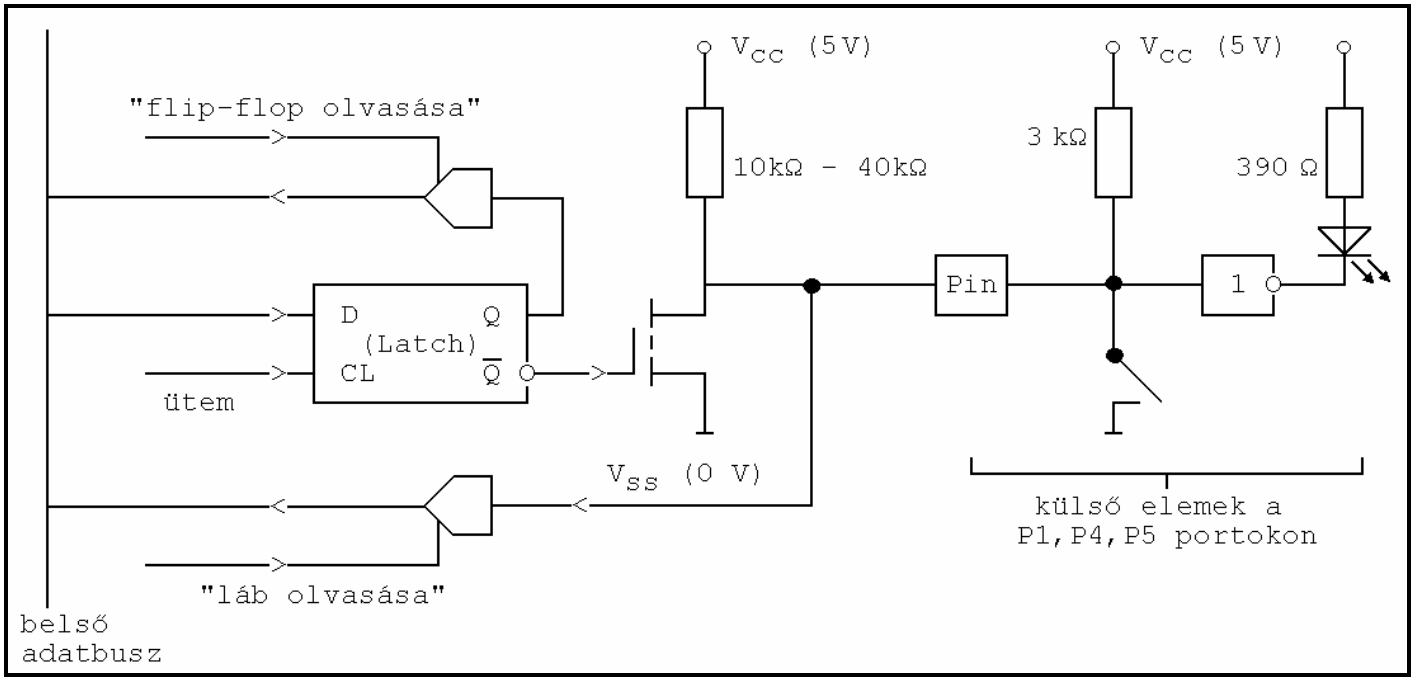
Többszörös portkihasználás :

Port	Külső memória-címzés	Megszakítás	Timer	Egyéb
P0.0 P0.1 P0.2 P0.3 P0.4 P0.5 P0.6 P0.7	AD0 AD1 AD2 AD3 AD4 AD5 AD6 AD7			
P1.0 P1.1 P1.2 P1.3 P1.4 P1.5 P1.6 P1.7		$\overline{\text{INT3}}$ INT4 INT5 $\overline{\text{INT6}}$ $\overline{\text{INT2}}$	CC0 CC1 CC2 CC3 T2EX T2	 clkout*
P2.0 P2.1 P2.2 P2.3 P2.4 P2.5 P2.6 P2.7	A8 A9 A10 A11 A12 A13 A14 A15			
P3.0 P3.1 P3.2 P3.3 P3.4 P3.5 P3.6 P3.7	 $\overline{\text{WR}}$ $\overline{\text{RD}}$	 $\overline{\text{INT0}}$ $\overline{\text{INT1}}$	 $\overline{\text{INT0}}$ $\overline{\text{INT1}}$ T0 T1	RxD** TxD**
	-> *ld. 1.4*	-> *ld. 3.8 / 3.9*		-> *ld. 3.11*

* Az ADCON regiszter CLK bitje engedélyezi.
** A soros adatátviteli csatorna használja.

1.3. A P1-P5 portok tulajdonságai

A P1-P5 portok felépítésének **egyszerűsített** ábrázolása, és az MVUS 535 gyakorlórendszer külső alkatrészei:



Logikai "0" a flip-flop Q kimenetén :

-> A csatlakozópont impedanciája alacsony (300 Ω), négy LS-TTL bemenet számára nyelő lehet. ($I_{lmax} \leq$ 1,6 mA) A port ***csak kimenetként*** működhet.

Logikai "1" a flip-flop Q kimenetén :

-> A csatlakozópont impedanciája magas (10kΩ - 40kΩ) négy LS-TTL bemenet számára forrás lehet.
($I_{hmax} \leq$ 120 μA; ha az áram eléri a 180 μA-t, a láb logikai "0" szintű lesz.)

-> A port ***bemenet vagy kimenet*** egyaránt lehet.
Következtetés: Ha egy portot bemenetként szeretnénk használni, 0FFH-t kell írni a portra. RESET után minden port bemenet.

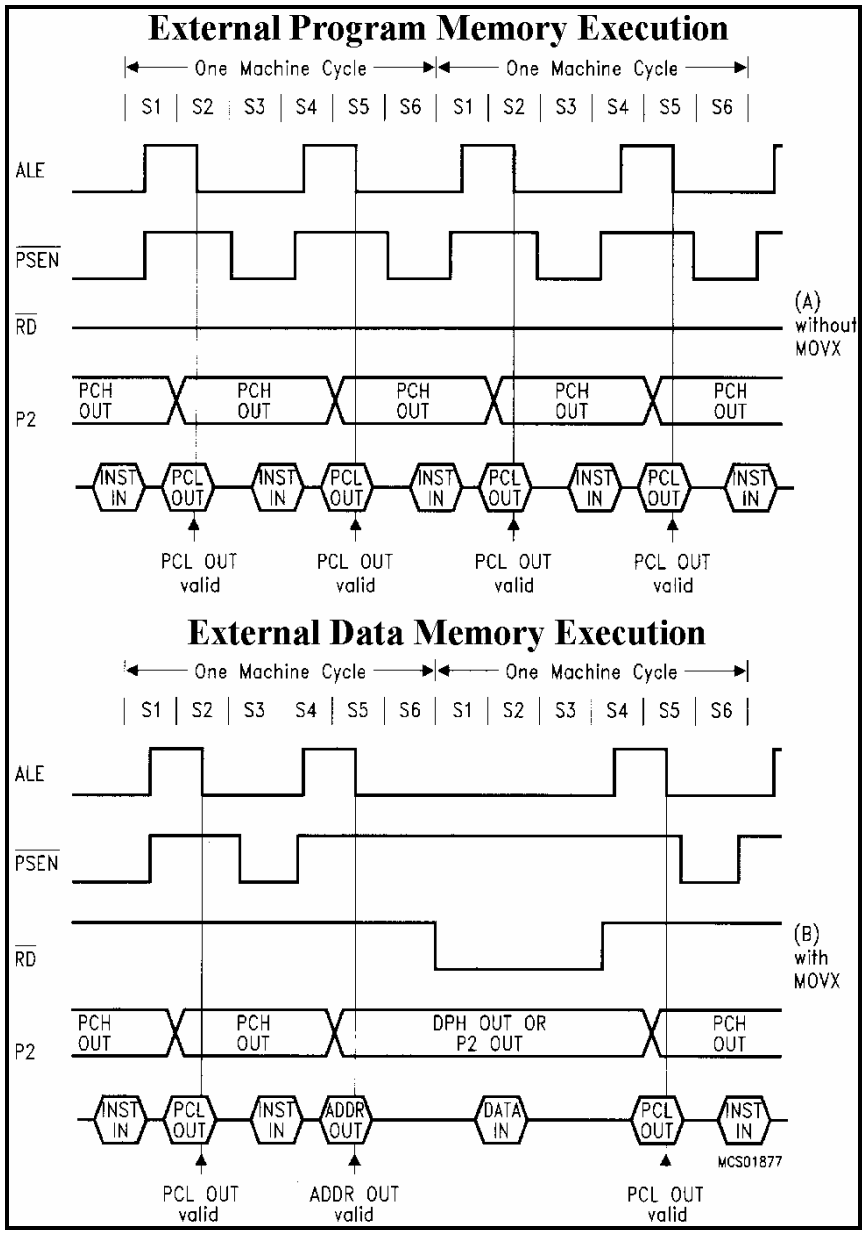
A P0 valódi háromállapotú kétirányú portként működik.

A P0-P3 portok további feladatokat is ellátnak (ld. "többszörös portkihasználás" az előző oldalon).

1.4. Hozzáférés a külső program- és adatmemóriához

A mikrokontroller chip a processzoron kívül tartalmaz programozható perifériákat, ROM-ot programmemória és RAM-ot adatmemória céljára. Ha nincs belső programmemóriánk (pl. az MVUS 535 gyakorlórendszer 80535-ös mikrokontrollerében) és/vagy a belső adatmemória kevés, lehetőségünk van külső program- és/vagy adatmemória csatlakoztatására. Az MVUS 535 gyakorlórendszer 64 kB EEPROM ***külső adatmemóriát*** tartalmaz, amelynek az alsó 32k címtartománya ***program- és adatmemória*** céljára egyaránt használható, és amelybe a lefordított gyakorlóprogramokat letölthetjük. Ezenkívül tartalmaz még EPROM programmemóriát, amelynek felső 32 kilobájtos részében található a rendszer működtető programja (az ún. **monitorprogram**). A programozó a programmemóriához csak olvasóutasításokkal férhet hozzá.

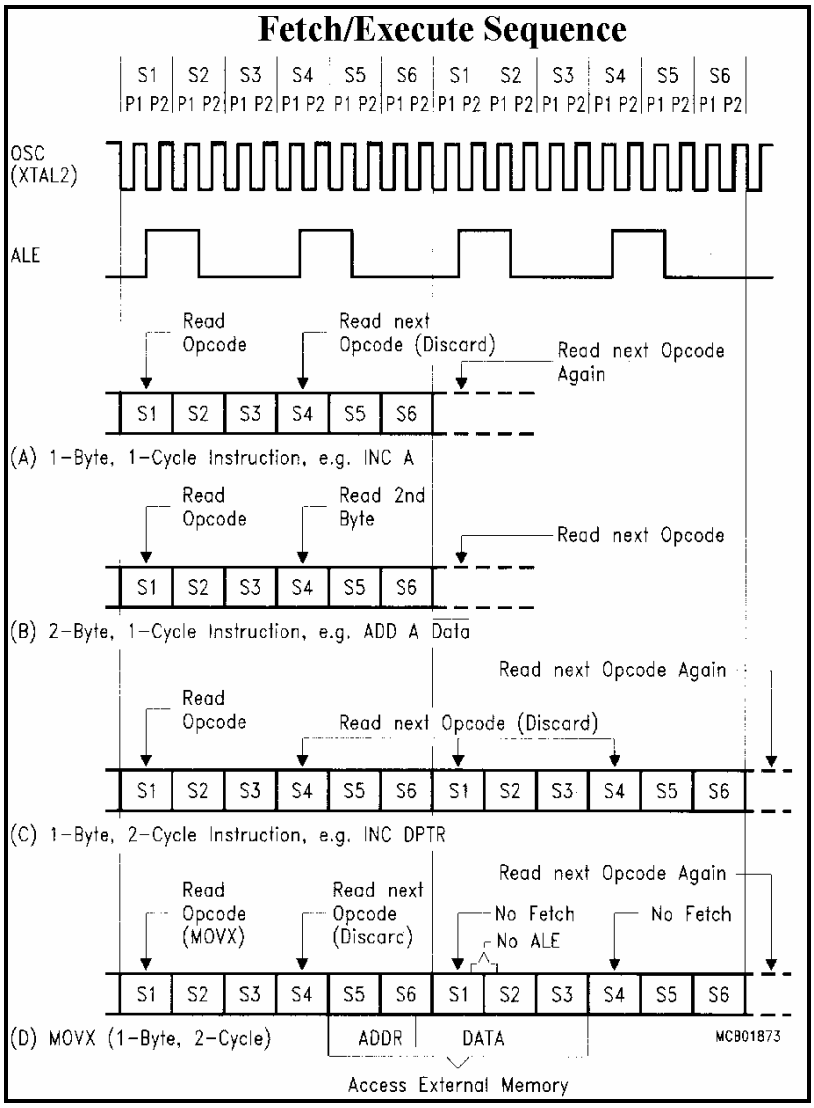
1.4.1. A CPU időzítése




```
1 gépi ciklus      =     6  állapot           : S1   - S6
1 állapot          =     2  fázis             : P1   - P2
1 gépi ciklus      =    12  fázis             : S1P1 - S6P2
1 fázis            =     1   órajel ciklus

Kvarcfrekvencia    :    fosz    = 12    MHz
1 fázis            :    1/fosz  = 1/12  µsec
1 gépi ciklus      :    12/fosz = 1     µsec
```

Hozzáférés a külső programmemóriához : A programmemória csak olvasható. A program feldolgozása **"várakozási sorban"** (**1 bájtos queue**) történik. "Átlapolt mód": mialatt a processzor a következő bájtot tölti ,feldolgozza az előzőt. A külső programmemória számára a $\overline{PSEN}$ jelenti az olvasójelet és nem eshet a külső adatmemóriát olvasó $\overline{RD}$ jellel azonos időre. Egy $\overline{PSEN}$ ciklus 6, egy $\overline{RD}$ ciklus 12 órajelig tart. A programmemóriából **egy gépi ciklus alatt két utasításbájt olvasása történik:** az idődiagram minden 3 állapota után a processzor további utasításbájtot olvas, de gépi ciklusonként maximum egy utasítást hajt végre. Ebből az alábbi feltételek következnek:



2. ábra

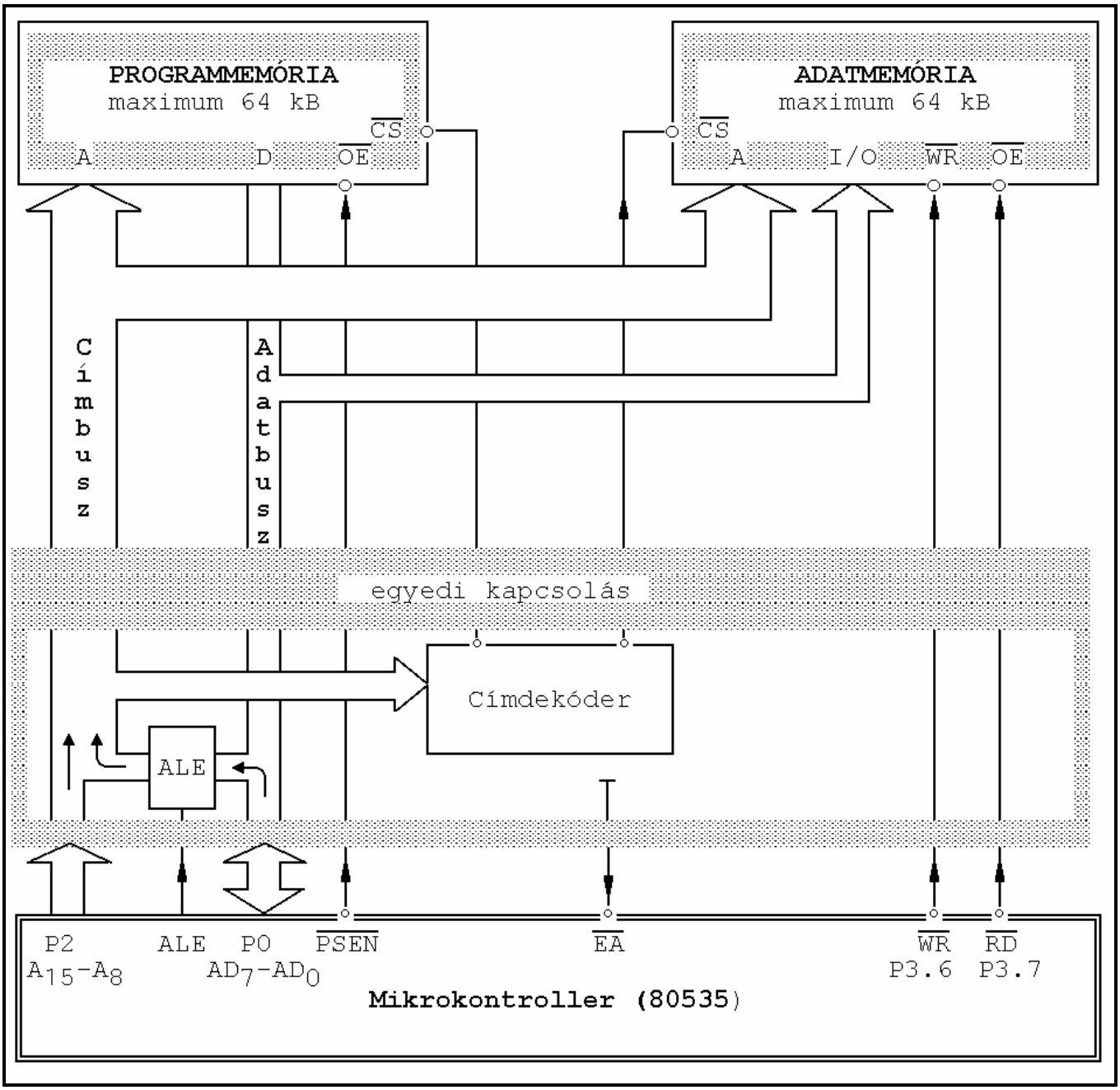
(A) sor:
1 bájtos, *1* ciklusig tartó utasítások (pl. INC A): a következő utasításbájtot (S4-nél) a processzor "eldobja", és a következő gépi ciklusban (S6 után) újra beolvassa.

(B) sor:
2 bájtos, *1* ciklusig tartó utasítások (pl. ADD A,#adat): egy gépi ciklus (S1-S6) pontosan elegendő.

(C) sor:
1 bájtos, *2* ciklusig tartó utasítások (pl. INC DPTR): a következő utasításbájtokat (S4-nél az első ill. S1-nél és S4-nél a második gépi ciklusban) a processzor "eldobja", ezért az ilyen utasítások lehívása és feldolgozása 2 gépi ciklusig tart.

(D) sor: a külső adatmemóriát író vagy olvasó utasítások
1 bájtos, *2* ciklusig tartó utasítások (pl. MOVX ...) : a következő utasításbájtot (S4-nél az első ciklusban) a processzor "eldobja"; a lehívást a második ciklus S1 és S4 állapotainál "kihagyja" (no fetch), mert a gépi ciklus 12 órajele alatt tud elvégezni egy a külső adatmemóriát író vagy olvasó ciklust.

1.4.2. Külső memóriaszervezés és buszstruktúra



Az azonos címtartományba eső ROM és RAM chipek egyidejűleg kapnak $\overline{CS}$ jelet a címdekódertől, de az $\overline{OE}$ jelet a ROM-oknak a PSEN, a RAM-oknak a $\overline{RD}$ jelenti .

Ha a mikrokontroller a külső programmemóriához fordul, a ***$\overline{PSEN}$*** (**P**rogram **S**torage **E**nable) ***vezérlőjelet*** minden gépi ciklusban kétszer adja ki, függetlenül attól, hogy utasításkódot vagy operandust olvas-e be. Az ***$\overline{EA}$*** (**E**xternal **A**ccess) ***bemenőjel*** jelzi a programmemória fizikai helyét.

$\overline{EA}$ = 0 : A mikrokontroller ***csak a külső programmemóriát*** érheti el. Ilyenkor a belső ROM nélküli mikrokontroller $\overline{EA}$ bemenetét a GND-re kell huzalozni. (Az MVUS 535 oktatórendszer is ilyen.)

$\overline{EA}$ = 1 : A cím < 8kB : csak a belső programmemória (ROM) elérhető. A program végrehajtása során a külső cím- és adatbusz használata csak a külső adatmemória eléréséhez lehet szükséges.
A cím > 8 kB : a mikrokontroller a programkódokat 8k felett a külső programmemóriából olvassa.

Külső címbusz (16 bit)**:** A felső címbájtot (A15-A8) a mikrokontroller a P2, az alsót

(A7-A0) az adatbusszal multiplexelve a P0 porton adja ki (AD7-AD0). Az ALE (**A**dress **L**atch **E**nable) vezérlőjel logikai 1-el jelzi, hogy az AD7-AD0 buszon pillanatnyilag az alsó címbájt található és az az ALE aktív élének hatására egy latch-ben tárolódik.

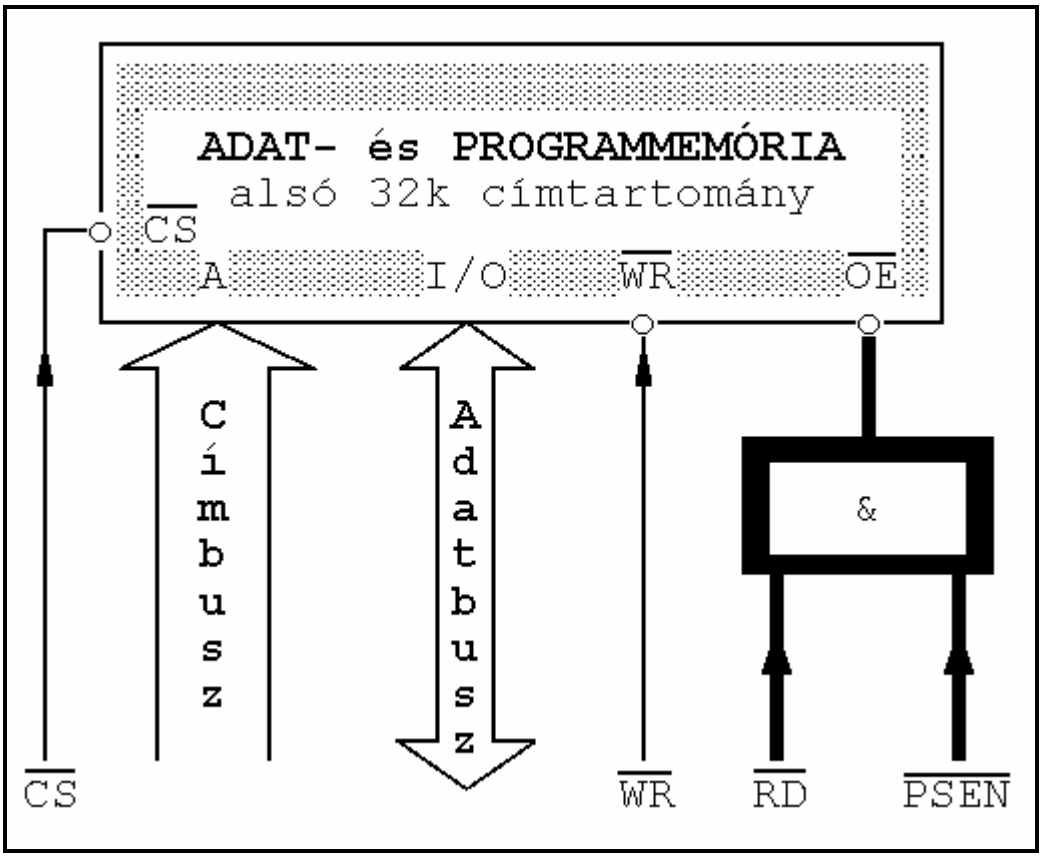
Külső adatbusz : Ha az ALE vezérlőjel nem címet jelez (logikai 0) az AD7-AD0 buszon, a P0 portot a mikrokontroller kétirányú adatbuszként használhatja. (pontosabban : ld. 1.4.1).

Hozzáférés a külső adatmemóriához :

A külső adatmemóriához is a fent leírt cím- és adatbuszokon keresztül lehet hozzáférni. Ha a P0 porton nem cím van ($\overline{ALE}$ = 0), a ***RD*** (Read) ***vezérlőjellel*** az adatok az adat-memóriából olvashatók illetve a ***$\overline{WR}$*** (Write) ***vezérlőjellel*** írhatók.

A külső adat- és programmemória az MVUS 535 oktatórendszerben átlapolódik.

Ahhoz, hogy az MVUS 535 oktatórendszerben a lefordított gyakorlóprogramok a külső RAM-ba letölthetőek legyenek, arra van szükség, hogy az adatmemória programmemóriaként is olvasható legyen. Ezt valósítja meg az alábbi (elvi) hardver kapcsolás:



A program- és adatmemória az alsó 32k címtartományban átlapolódik.

1.5. Perifériaelemek

1.5.1. A 8255-ös párhuzamos perifériaillesztő (PPI)

Az MVUS 535 oktatórendszerben a mikrovezérlőhöz illesztettek egy 8255-ös külső párhuzamos perifériaillesztőt is a használható párhuzamos csatlakozópontok számának növelésére. A 8255 egy vezérlőregisztert és három kétirányú adatregisztert (port) tartalmaz. Az A, B és C portok adatátviteli irányát a vezérlőregiszterbe írt bitkombináció határozza meg. A portbitek két csoportot képeznek: az A csoporthoz tartozik az A port és a C port felső négy bitje, a B csoporthoz a B port és a C port alsó négy bitje. Az MVUS 535 oktatórendszerben a 8255-ös regiszterei a külső adatmemóriába ágyazva érhetők el a DPTR segítségével az alábbi címeken:

```
F800H   A port
F801H   B port
F802H   C port
F803H   vezérlőregiszter
```

Az A csoport három, a B csoport két különböző üzemmódban működhet, de itt most csak a 0-ás módhoz tartozó vezérlőregiszter-beállítást ismertetjük. (Az információk az INTEL 1-333-tól 1-352-ig terjedő adatlapjaiból származnak).

A vezérlőregiszter egyes bitjei 0-ás módban az alábbiakat jelentik:

```
 7.   bit : 1=aktív
 6-5. bit : A csoport üzemmódválasztás, 0-ás módhoz 00
 4.   bit : A port adatirány; 1=bemenet, 0=kimenet
 3.   bit : C port 4-7 bitek adatiránya; 1=bemenet, 0=kimenet
 2.   bit : B csoport üzemmódválasztás, 0-ás módhoz 0
 1.   bit : B port adatiránya; 1=bemenet, 0=kimenet
 0.   bit : C port 0-3 bitek adatiránya; 1=bemenet, 0=kimenet
```

Az alábbi példaprogramban a vezérlőregiszterbe írt 8BH (10001011B) az A portot bemenetként, a B és C portokat kimenetként programozza:

```
LSTOUT -
$SFR80515.inc
LSTOUT +
          ORG       200H
CONTR     DEFINE    0F803H        ;Címkonstansok hozzárendelése.
PA        DEFINE    0F800H
PB        DEFINE    0F801H
          MOV       DPTR,#CONTR
          MOV       A,#8BH        ;0-ás mód, PA kimenet, PB és PC bemenet.
          MOVX      @DPTR,A
CIKLUS:   MOV       DPTR,#PB
          MOVX      A,@DPTR       ;PB tartalma az akkumulátorba.
          MOV       DPTR,#PA      ;PA címe a DPTR-be.
          MOVX      @DPTR,A       ;Az akkumulátor tartalma a PA-ra
          MOV       P5,A          ;és a  P5-re.
          LJMP      CIKLUS
          END
```

(A fenti program megértéséhez szükséges a 80515 utasításkészletének ismerete.)

1.5.2. További perifériák

Az MVUS 535 oktatórendszerben további külső perifériákat is illesztettek a mikrovezérlőhöz: alfanumerikus-grafikus LCD kijelzőt, soros aszinkron kommunikációs vezérlőt (UART) és D/A átalakítót. Az LCD kijelző kezeléséhez szükséges információkat külön segédlet tartalmazza.

2. A PROGRAMOZÁS ALAPJAI

2.1. A strukturált programozás alapfogalmai

A mikroprocesszorok programozására manapság gépközeli (assembly) és magasszintű nyelveket egyaránt használhatunk. Ez a jegyzet az assembly nyelvű programozást használja. Noha az assembly a strukturált programozást nem támogatja (ellentétben pl. azzal, ahogy a PASCAL-ból ismert), mégis szükséges az alapfogalmak összefoglalása. Egy assembly program elolvasáshoz és megértéséhez is szükséges, hogy lehetőleg ismerjük a strukturált programozás szabályait.

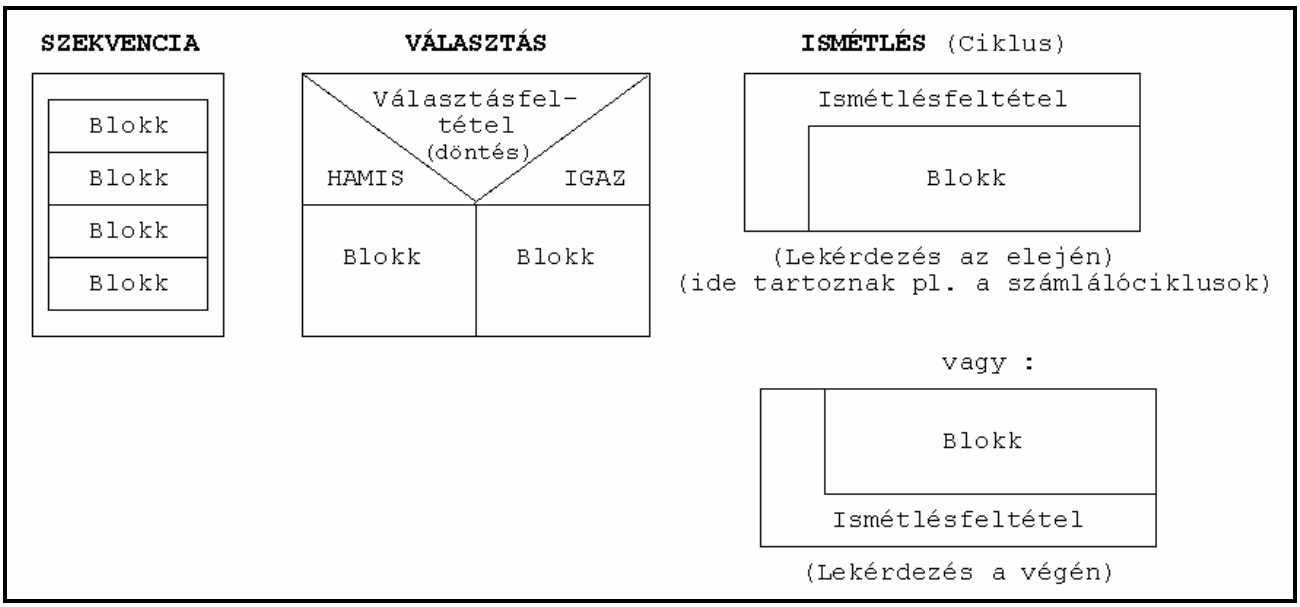
Programozási segédeszközök :

Az assembly programozáshoz a problémaorientált nyelvekkel megegyező segédeszközöket használhatunk:

1. **Program folyamatábra (pl. DIN 66001 szerint)**
2. **Struktogram (pl. DIN 66261 szerint)**
3. **Adatmozgási tervek**

Ezek közül az alábbiakban a struktogramot ismertetjük:

Mindössze három blokktípus megengedett : **SZEKVENCIA, VÁLASZTÁS, ISMÉTLÉS**
A struktúra egy blokkja egy részfeladatot realizál, és 50-100 utasítást tartalmaz (professzionális programokban).



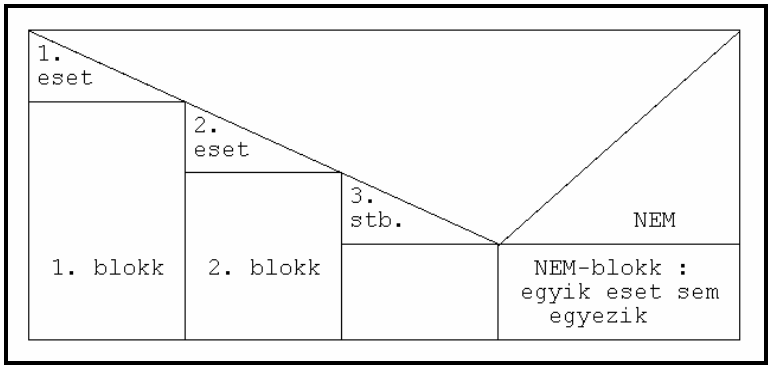
A ciklusok végrehajtása az ismétlésfeltétel vizsgálata alapján történik. Ha az ismétlésfeltétel a ciklus **elején** történő lekérdezéskor nem teljesül, a ciklust ***nyilvánvalóan egyszer sem*** hajtjuk végre, ellenben ha ezt a ciklus **végén** tesszük, ***legalább egyszer*** végre kell hajtani.

A struktúrablokkokra vonatkozó szabályok :

→ Minden blokknak **csak egy bemenete** és **egy kimenete** lehet;
→ A blokkon belüli elágazások **végpontja közös**;
→ A blokkon belüli többszörös elágazások (esetlekérdezés) csak **előre** és **közös végponttal megengedettek**;
→ Egy ciklus **idő előtti elhagyása tilos**.
→ Jóllehet a **blokkon belüli feltétel nélküli ugrások** megengedettek, **célszerű**
→ nyomban ellenőrizni hogy ez nem okozott-e **törést.**

Fenti szabályok betartása esetén a változó **ESETLEKÉRDEZÉS** és **TÖRÉSFELTÉTEL** blokként való használata megengedett :

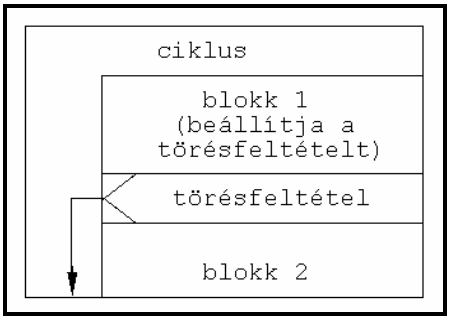
ESETLEKÉRDEZÉS :



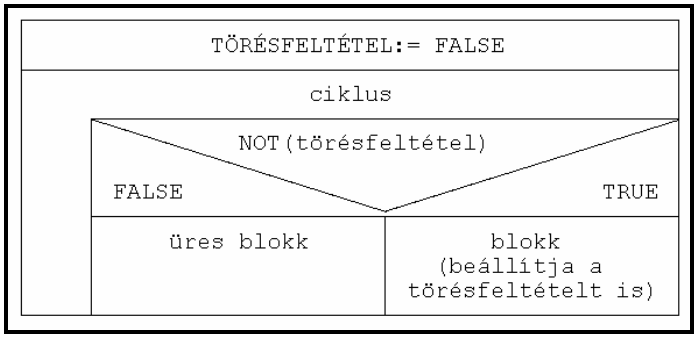
Csak az első **IGEN** válaszhoz tartozó blokk hajtódik végre. Ezért fontos, hogy az eseteket **prioritásuk** sorrendjébe rendezzük.

Ha nincs rá szükség, a **NEM** blokk elhagyható.

TÖRÉSFELTÉTEL:

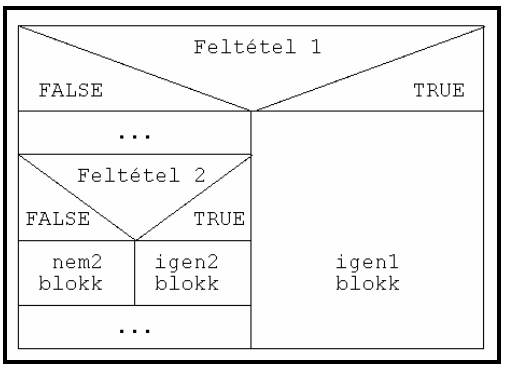


Az ismétlés-struktúra időelőtti elhagyásának csak **számlálóciklusokban** van értelme.
Ha a törésfeltétel teljesül, a ciklus **végére** kell mutatnia.
Figyelem :
Amíg a **ciklusszámláló végértékét** nem értük el, a **ciklusmag ismételten végrehajtandó**. Ez azt jelenti, hogy a blokk 1 újra végrehajtódik, és nyilvánvalóan újra beállítja a törésfeltételt is.
A ciklusszámláló megváltoztatása ciklusmagon belül nem megengedett.



A probléma egy lehetséges megoldása.

Blokkok egymásbaágyazása :



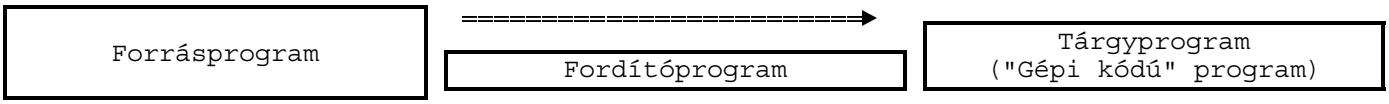
Blokkok egymásbaágyazása csak akkor megengedett, ha a belső blokk maradéktalanul végrehajtódik, mielőtt az őt tartalmazó blokk feldolgozását folytatnánk.

Alprogramtechnika : A strukturált programozásnak az a célja, hogy a programot részfeladatokra bonthassuk, amelyek egymástól függetlenül írhatók és tesztelhetők. A főprogram ilyenkor csak egy logikai "váz", amely a kívánt részfeladatokat megoldó alprogramok hívását végzi.

Az alprogramra vonatkozó előírások :

- → Mint bármely más blokknak **csak egy bemenete** és **egy kimenete** lehet ezért az alprogramon belüli ugrás vagy az őt hívó programnak a hívástól eltérő helyére való visszaugrás tilos.
- → Egy alprogram **egy önmagában zárt részfeladatot** hajt végre.
- → Az alprogramok legyenek **univerzálisak**, hogy különböző főprogramokból lehessen hívni őket. Így létrehozhatunk egy univerzális programkönyvtárat.
- → **Csak külső alprogramok** megengedettek, amelyekben a változók és címkék csak lokálisan (csak az alprogramon belül) érvényesek (assembly programban nem megoldható).

2.2. Forrás- és tárgyprogram



1. **problémaorientált nyelvek** ***COMPILER***
   (pl. PASCAL)
   (pl. BASIC) (vagy Interpreter)

2. **géporientált nyelvek** ***ASSEMBLER***
   (pl. 80515-ös
   ASSEMBLY)

3. hexadecimális ill. hexadecimális loader
   bináris bevitel

Bináris/hexadecimális bevitel **:**

A számítógépkódok direkt bináris bevitele ma már alig használt, a programozó számára nehezen áttekinthető és könnyen elhibázható. A hexadecimális formában rövidített kódok bevitele ezen csak kismértékben javít.

Assembly **:**

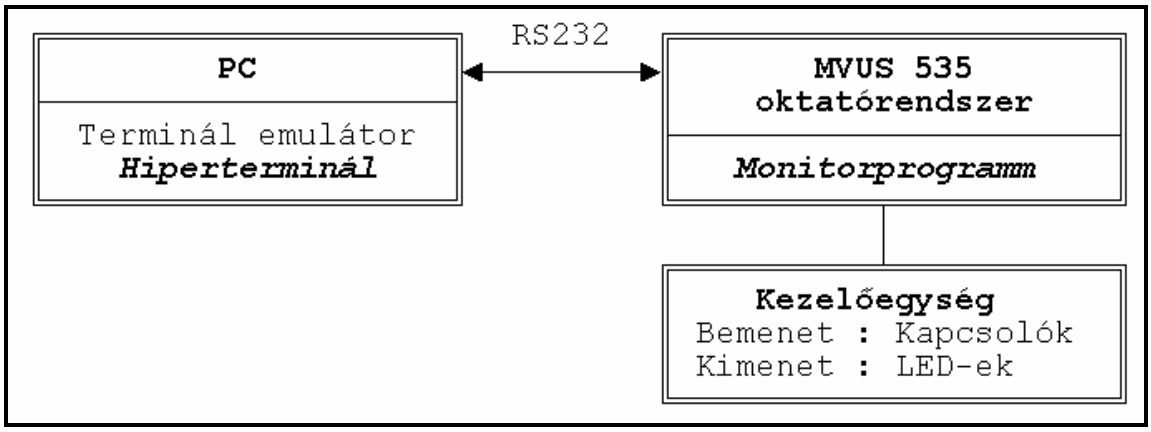
A forrásprogram **mnemonikus*** írásmódú utasításokból áll: az utasítást és az operandusait is néhány betűből álló, egyértelműen azonosítható rövidítések jelölik, a program ezáltal érthetőbb és kisebb a hibázás esélye. Mivel az utasítások géporientáltak, a processzor ill. a kontroller pontos ismerete (milyen regiszterei és portjai vannak, hogyan lehet azokat ill. a memóriát elérni stb.) feltétlenül szükséges. A különböző processzorcsaládok és számítógépek assembly nyelveinek nincs megkülönböztető nevük; egy adott nyelvet a felhasználási környezetével jellemzünk: pl. a 80515-ös mikrokontroller assembly nyelve. Az assembly forrásprogramot bináris kódokra az "Assembler" fordítja le.

Például : **MOV A,R0** (másold az R0 regiszter tartalmát az akkumulátorba)
(***** Mnemosyne: az emlékezet görög istennője. Mnemonik: emlékeztető)

Problémaorientált nyelvek**:**
A problémaorientált nyelveket azért fejlesztették ki, hogy a processzor pontos ismeretének súlyos követelményétől megszabaduljunk, és maradéktalanul a programozandó "problémára" koncentrálhassunk.
Például : **Összeg := Összeadandó1 + Összeadandó2;**
Ezek a nyelvek szabványosítottak és gépfüggetlenek; csak a compiler-ük gépfüggő. "Dialektusoknak" hívják az egy adott géptípus sajátosságaihoz igazított változataikat.

Egyes magasszintű nyelvek kifejezetten természettudósok és mérnökök számára készültek; ezek minden egyes "problémát" a matematika segítségével írnak le (pl. FORTRAN, BASIC, PASCAL, C), mások egyéb foglalkozási ágak, pl. közgazdászok számára (COBOL), használhatók.

2.3. Az oktatórendszer és a szoftver eszközkészlet felépítése



Az MVUS 535 oktatórendszer programjait a PC-n szövegszerkesztővel írhatjuk (DOS editor, Windows Commander stb.), használhatjuk a rendszer saját, a DOS editorral

gyakorlatilag megegyező szerkesztőprogramját, de **nem használhatunk** formátumszerkesztőt (pl. Word for Windows). A forrásprogramot **.S03** kiterjesztésű fájlban tároljuk és az ALL (Assembler, Linker, Librarian) batch program segítségével fordíthatjuk le. A lefordított un. tárgyprogram a forrásprogrammal azonos névvel, de .R03 kiterjesztéssel tárolódik ugyanabban az alkönyvtárban, ahol a forrásprogram található. A fordítás során egy fordítási lista is létrejön .LST kiterjesztéssel, ez természetesen tartalmazza a hibaüzeneteket is. Hiba esetén elindul a rendszer saját szerkesztőprogramja, amelybe automatikusan betöltődik a forrásprogram és a hibaüzenetek listája. Hibátlan fordítás esetén az .R03 kiterjesztésű fájl további feldolgozásra kerül, amelynek eredményeként létrejön egy **.HEX** kiterjesztésű, INTEL-HEXA formátumú, az oktatórendszerbe letölthető és futtatható **szövegfájl**.
Az oktatórendszer monitorprogramjával a PC-n futó terminál emulátor kommunikál. Ilyenkor a PC egyszerűen az oktatórendszer termináljaként működik.
A lefordított, futtatható programok a **szövegfájl küldése** parancs segítségével tölthetők le a mikrovezérlőbe a soros adatátviteli csatornán keresztül. Ez csak akkor lehetséges, ha a mikrokontroller parancsfogadási módban van. Végtelen ciklusban futó programot először a piros RESET gombbal meg kell szakítani.
A programot a "*G cím*" monitorparanccsal indíthatjuk és tesztelhetjük. Futtathatjuk a programot nyomkövetéses üzemmódban is a "*T lépésszám*" paranccsal, de gondoskodni kell arról, hogy az utasításszámláló "PC" a program kezdőcímére mutasson. Nyomkövetéses módban a mikrokontroller regisztereinek tartalma (PC, A, B, R0-R7, PSW) minden egyes utasítás végrehajtása után a képernyőn listázódik.
Ha a program nem a szándékunknak megfelelően működik, a monitorprogrammal a memóriatartalom vizsgálható és akár módosítható is. Célszerűbb azonban ehelyett a forrás-programot a PC-n szövegszerkesztővel átszerkeszteni, majd az ALL-lal újra lefordítani, letölteni és újra futtatni.

Az oktatórendszerhez tartozó ***programfejlesztési*** szoftvereszközök:

-> EDITOR (parancssorból: E)	:	Assembly forrásprogram szerkesztésére (más szöveg-szerkesztő is használható).
-> ASSEMBLER (A8051)	:	A forrásprogram lefordítására.
-> Csatoló (XLINK)	:	A futtatható program létrehozására. (Az ASSEMBLER és az XLINK hívását az ALL batch program végzi.)
-> Terminál emulátor	:	A PC és az oktatórendszer (mikrokontroller) közötti kapcsolattartásra.
-> MONITOR	:	A mikrokontroller "rendszerprogramja" a programozóval való kapcsolattartásra, a programok futtatására, tesztelésére, hibakeresésre.

EDITOR (Menüvezérelt, a DOS editorhoz hasonló):

Az EDITOR-ral megírt forrásprogramot szövegfájlban, .S03 kiterjesztéssel tároljuk. A fordítás során fellépő szintaxishibákat és a programfutás logikai hibáit szintén itt korrigáljuk. Az assembly forrásprogramot mezőkre osztjuk az alábbiak szerint :

Címkemező; Utasításmező Operandusmező ;Megjegyzésmező (opcionális)

```
;Példa forrásprogram felépítésére
;
             ORG      200H           ;A program kezdőcíme
             MOV      P5,#0FFH       ;A P5 port bemenetként működik
CIKLUS:      MOV      P4,P5          ;A P5 tartalmát a P4-re másolja
             LJMP     CIKLUS         ;végtelen ciklusban
             END
```

A **8051** assembler:

Az assembler fordítja le az .S03 kiterjesztésű forrásfájlt. Létrehozza az .R03 kiterjesztésű tárgyprogramfájlt és az .LST kiterjesztésű listafájlt a hibaüzenetekkel. Ezek a fájlok a Windows Commander F3 parancsával elolvashatók.

Különlegességei :

-> Megkülönbözteti a ***szimbolikus neveket*** és a ***címkéket***
-> A **címkék** kettősponttal végződnek

```
LOOP: .....
```

-> A **direkt bitcímeket** nem kell zárójelbe tenni

```
MOV C,20
MOV 14H,C     ; vagy MOV  22H.4,C
```

```
   ->  A használt számrendszer típusát jelezni kell:
                  decimális      :  10  (jelzés nélkül)
                  hexadecimális  :  0FFH (betűvel kezdődő szám elé 0-t kell írni)
                  bináris        :  01001100B
   ->  A konstansok elé #-t kell írni
```

A **pszeudo-utasítások** (assembler direktívák) befolyásolják a program fordítását, de a program futását **nem**:

```
              ORG       cím       ;A program kezdőcíme (elhelyezésszámláló). Értéke az
                                   MVUS 535 oktatórendszerben 0H<cím<7FFFH közé eshet.
név           EQU       érték     ;A név-hez számértéket rendel, nem definiálható át az
                                   adott programblokkon belül.
név           =         érték     ;Azonos az EQU-val.
név           DEFINE    érték     ;A név-hez számértéket rendel, de érvényessége az
                                   adott fájl végéig tart.
címke:        DS        érték     ;A program aktuális címétől memóriát foglal le.
címke:        DB        érték(ek) ;A megadott 8 bites érték(ek) kódja(i) beépülnek a
                                   memóriába a program adott helyétől kezdődően.
címke:        DW        érték(ek) ;A megadott 16 bites érték(ek) kódja(i) beépülnek a
                                   memóriába a program adott helyétől kezdődően.
címke:        DD        érték(ek) ;A megadott 32 bites érték(ek) kódja(i) beépülnek a
                                   memóriába a program adott helyétől kezdődően.
              EXTERN    név       ;Másik modult definiál.
              PUBLIC    név       ;Más modulok is használhatják ugyanazt a nevet
              LOW és HIGH         ;Egy 16 bites érték alsó ill. felső bájtját adja
              END                 ;A forrásprogram (és így a fordítás) végét jelzi
```

Figyelem: Az assembler a nevekben és a címkékben megkülönbözteti a kis- és nagybetűket!

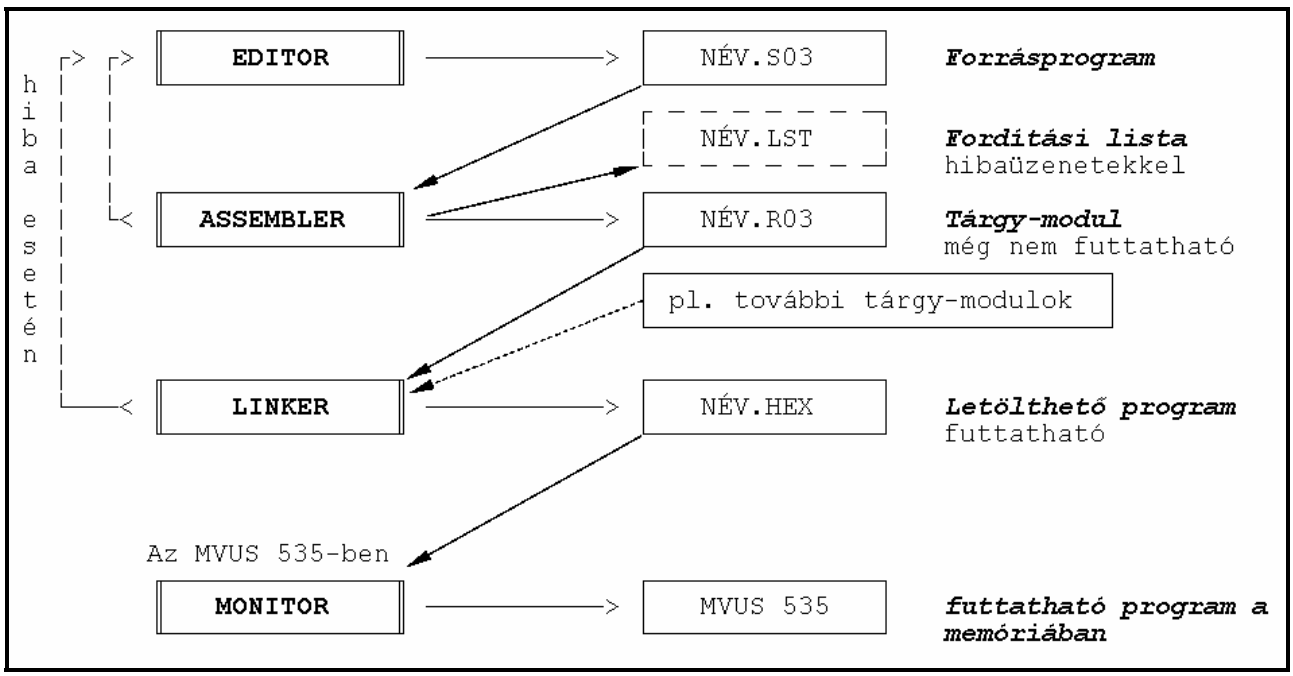
XLINK (csatoló):

A csatoló összeláncolja a különböző programmodulokat (ha vannak) és segédprogramokat, és kódolja az áthelyezhető (relokálható) .R03 programot egy letölthető, futtatható .HEX kiterjesztésű fájlba.

A terminál emulátor indításával kapcsolatot teremtünk a soros adatátviteli csatornán a PC és az oktatórendszer között. Innentől kezdve a ***monitorprogram*** veszi át a vezérlést. (A ***monitor*** egylépéses assemblerével is létrehozhatunk futtatható programokat. Monitor-parancsokkal szerkeszthetjük a memóriatartalmat. -ld. a fejezet végén)

Összefoglalva :

A programfejlesztés menete :



A programfejlesztés lépései :

1. Szimbolikus (assembly) nyelvű forrásprogram írása valamilyen szövegszerkesztővel.
2. Fordítás az ALL batch program segítségével (hiba esetén vissza a szerkesztőbe).
3. Terminál emulátor indítása : Kapcsolatfelvétel a monitorprogrammal.
4. Szövegfájl küldése : Programátvitel az oktatórendszerbe.
   G cím : Programfuttatás.
   RESET az oktatórendszeren.

Programpélda :

```
MONITOR  EQU     8000H          ; A monitorprogram kezdőcíme
         ORG     200H           ; A program kezdőcíme (elhelyezésszámláló)
         MOV     P4,#0FFH       ; P4 portot bemenetként használjuk
         MOV     P5,#0          ; P5 portot nullázzuk
         MOV     (0AH),#0       ; A ciklusszámlálót a (0AH) címen nullázzuk
HUROK:   LCALL   KESL           ; Késleltető szubrutin hívása
         DEC     P4             ; P4 dekrementálása
         INC     P5             ; P5 inkrementálása
         DJNZ    (0AH),HUROK    ; A ciklusszámlálót 256-szor csökkenti
         LJMP    MONITOR        ; és ha nulla, a monitorprogramra ugrik
KESL:    MOV     R0,#0          ; Kb. 256 x 256 x 2 ms
KESL1:   MOV     R1,#0
KESL2:   DJNZ    R1,KESL2       ; 256-szor a KESL2-re ugrik
         DJNZ    R0,KESL1       ; 256-szor a KESL1-re ugrik
         RET                    ; Visszatérés a hívás utáni utasításra
         END                    ; Vége a fordításnak
```

A monitorprogram :

A monitorprogram egyes funkciói néhány betűből álló parancsokkal aktiválhatók:

Memóriaparancsok:
- D : Memóriatartalom megjelenítése (display)
- E : Memóriatartalom megváltoztatása (enter, edit)
- FILL : Feltöltés konstans értékkel

Önmagukban nem használhatók, a memóriatípus betűjelével ki kell őket egészíteni (pl. ED, EX, FILLD, FILLI)

Címmegadás : hexadecimális formában (de a számrendszert jelző H nélkül)
- belső címtartomány : 0 - FF
- külső címtartomány : 0 - FFFF

Címtartomány : kezdet, vég ; display és fill parancsoknál

Memóriatípus :
- D : direkt címezhető belső memória
- I : indirekt címezhető belső memória
- X : indirekt címezhető külső adatmemória
- B : bitcímezhető belső memória
- C : külső programmemória

Mnemonikus assembly lista bevitele	:	A kezdőcím
Disassemblálás	:	U kezdőcím, végcím
Regisztertartalmak kijelzése	:	X
Regisztertartalom módosítása	:	X regiszternév
Programfuttatás	:	G kezdőcím
Nyomkövetéses programfuttatás	:	T [lépésszám]
Nyomkövetéses programfuttatás	:	P [lépésszám] (A szubrutinhívás egy lépés.)

Bármelyik funkcióból egy pont (.) begépelésével lehet kilépni.

A monitor tartalmaz egy **egylépéses assemblert**, amellyel a forráskódot soronként vihetjük be és fordíthatjuk le. A soronkénti fordítás miatt szimbolikus címek használata nem lehetséges. Amennyiben egy címre hivatkozunk, azt numerikusan, hexadecimális formában kell megadnunk (H jelzés nélkül).
(A monitorprogram részletes leírását külön segédlet tartalmazza.)

3. A 80515-ÖS MIKROKONTROLLER PROGRAMOZÁSA

3.1. Bevezetés

A mikrokontroller megkülönbözteti a gépi kódú programok tárolására szolgáló ***programmemóriát*** (ROM : **R**ead **O**nly **M**emory) és a változó adatok tárolására szolgáló ***adatmemóriát*** (RAM : **R**andom **A**ccess **M**emory). A 80515-ös egy-chipes kontroller minkettőt tartalmazza.

A ***belső adatmemória*** 256 bájt RAM-ból és a 128 rekeszes SFR (SFR : **S**pecial **F**unction **R**egister) tartományban található 41 db járulékos regiszterből áll. Ha ez nem elegendő, a mikrokontroller maximum 64k ***külső adatmemóriát is használhat.*** Az MVUS 535 oktatórendszerben az alsó 32k külső adatmemória programmemóriaként is címezhető.

A ***belső programmemória*** (a 80515-ben) 8 kilobájtos. ***Külső programmemória szintén*** maximum 64k-ig használható. Az MVUS 535 rendszerben a mikrokontroller ROM nélküli változata (80535) található, ezért külső programmemória használata szükséges. (Az oktatórendszer 64 kilobájt EEPROM külső adatmemóriát és 64 kilobájt EPROM külső programmemóriát tartalmaz).

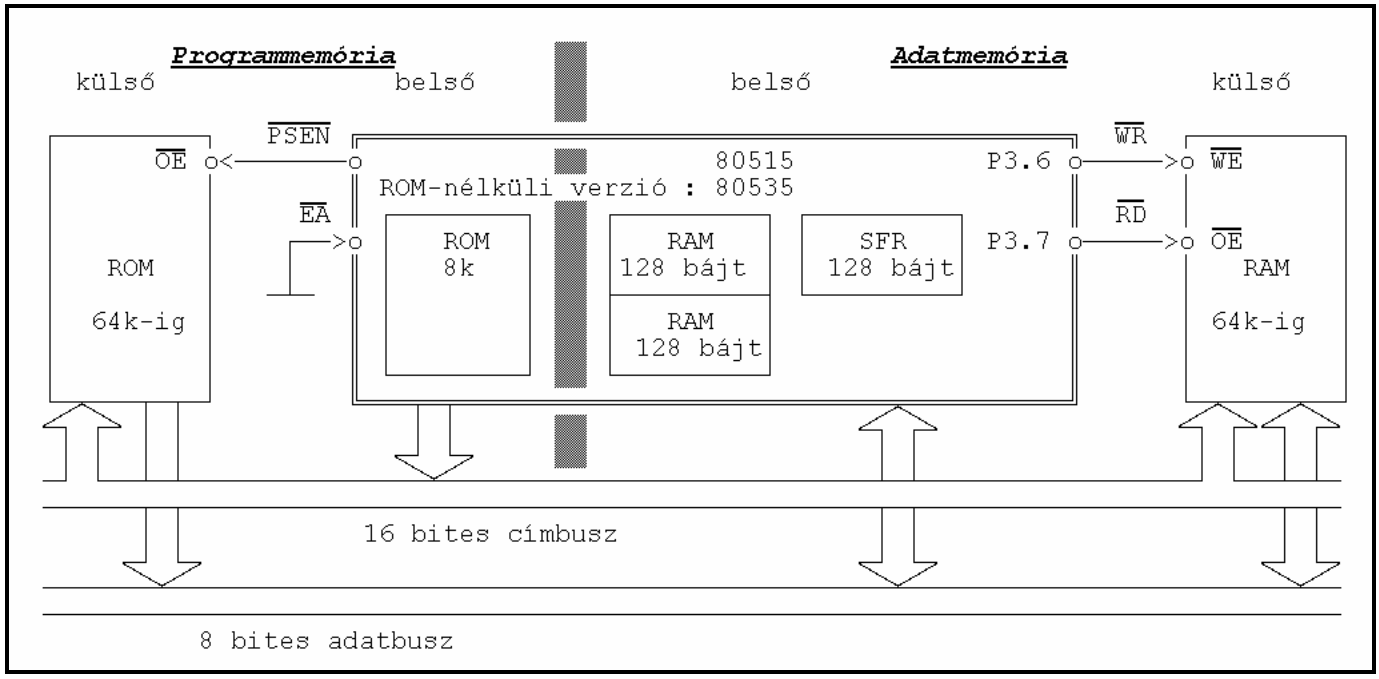
A 80515-ös mint Boole-processzor a 8 bites adatok mellett 1 bites adatok kezelésére is képes. Erre a célra speciális bitműveleti utasításai vannak.

Egyes adatműveletek megváltoztathatják a mikrokontroller belső állapotjelzőit, az un. **flag-eket**. Például a CARRY-flag bebillen, ha egy aritmetikai művelet során átvitel (carry) keletkezik. A flagek állapota vezérlésátadás feltételeül szolgálhat.

A mikrokontroller utasításkészlete száznál több utasításból áll, melyek négy csoportra oszthatók:

adatmozgatások
aritmetikai műveletek
logikai műveletek
programvezérlések (ugrások, szubrutinhívások, megszakítások)

A külső és belső program- és adatmemória áttekintése:



3.2. A mikrovezérlőn belüli adatmozgatások címzésmódjai példák alapján

Kétirányú ***adatmozgatás*** a 80535-ös rendszerben csak a belső adatmemória, az SFR tartomány és a külső adatmemória között lehetséges. A külső programmemória adatai csak olvashatóak.
Az adatmozgató utasítások a PSW (**P**rogram **S**tatus **W**ord, flagregiszter), ***egyik jelzőbitjét*** (flag) ***sem befolyásolják***, két eset kivételével:

1. Ha a PSW-t felülírjuk.
2. A ***paritásbit*** változhat, ha az adatmozgatás célja az akkumulátor.

Általános szintaxis : **MOV cél,forrás**

Az adatmozgatáshoz öt ***címzésmódot*** használhatunk:

direkt címzés	(direct addressing)
indirekt címzés	(register indirect addressing; a belső adatmemória számára)
regisztercímzés	(register addressing)
csatolt címzés	(immediate addressing)
relatív indirekt címzés	(base-register plus index-register indirect addressing; a külső adatmemória számára)

3.2.1. Direkt címzés (direct addressing)

Általános szintaxis : **MOV cím, cím**

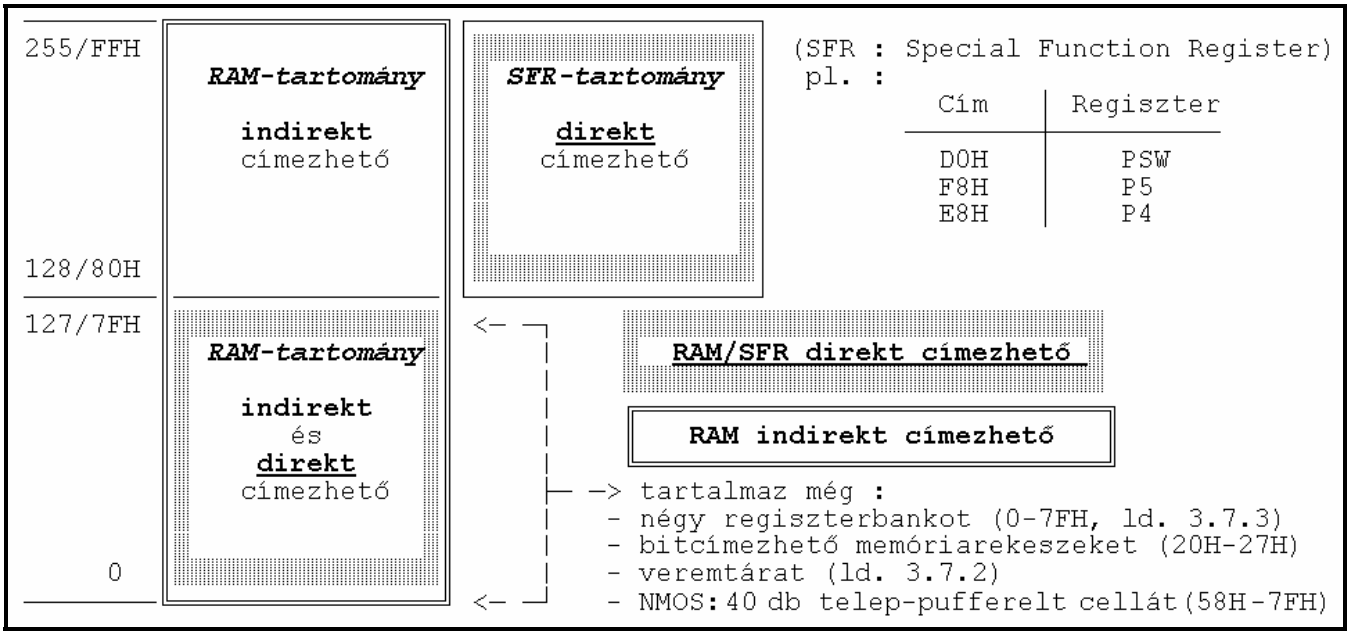
Célként és forrásként a ***bájtcímezhető memória*** **direkt címét** kell megadni; ha a cél és a forrás az SFR tartományban van , elegendő a regiszter nevét használni (lista és magyarázat a következő oldalon).

Belső direkt cím az akkumulátor is lehet :

Általános szintaxis: **MOV A, cím** vagy **MOV ACC, cím**
MOV cím, A vagy **MOV cím, ACC**

A MOV A, ACC utasítás viszont nem megengedett.

RAM- és regisztertartomány a mikrokontrolleren belül :



(SFR : Special Function Register)
pl. :

Cím	Regiszter
D0H	PSW
F8H	P5
E8H	P4

Az SFR tartomány direkt címezhető regiszterei. A regiszterek címe helyett elég a nevüket ismerni és használni:

Szimbolikus név		Special Function Register (SFR)	Cím
P0	*	Port 0	80H
SP		Stack Pointer	81H
DPL		Data Pointer Low Byte	82H
DPH		Data Pointer High Byte	83H
PCON		Power Control Register	87H
TCON	*	Timer Control Register	88H
TMOD		Timer Mode Register	89H
TL0		Timer 0 Low Byte	8AH
TL1		Timer 1 Low Byte	8BH
TH0		Timer 0 High Byte	8CH
TH1		Timer 1 High Byte	8DH
P1	*	Port 1	90H
SCON	*	Serial Port Control Register	98H
SBUF		Serial Port Buffer Register	99H
P2	*	Port 2	A0H
IEN0	*	Interrupt Enable Register 0	A8H
IP0		Interrupt Priority Register 0	A9H
P3	*	Port 3	B0H
IEN1	*	Interrupt Enable Register 1	B8H
IP1		Interrupt Priority Register 1	B9H
IRCON	*	Interrupt Request Control Register	C0H
CCEN		Compare/Capture Enable Register	C1H
CCL1		Compare/Capture Register 1 Low Byte	C2H
CCH1		Compare/Capture Register 1 High Byte	C3H
CCL2		Compare/Capture Register 2 Low Byte	C4H
CCH2		Compare/Capture Register 2 High Byte	C5H
CCL3		Compare/Capture Register 3 Low Byte	C6H
CCH3		Compare/Capture Register 3 High Byte	C7H
T2CON	*	Timer 2 Control Register	C8H
CRCL		Compare/Reload/Capture Register Low Byte	CAH
CRCH		Compare/Reload/Capture Register High Byte	CBH
TL2		Timer 2 Low Byte	CCH
TH2		Timer 2 High Byte	CDH
PSW	*	Program Status Word	D0H
ADCON	*	A/D Converter Control Register	D8H
ADDAT		A/D Converter Data Register	D9H
DAPR		D/A Converter Program Register	DAH
ACC	*	Accumulator	E0H
P4	*	Port 4	E8H
B	*	B Register	F0H
P5	*	Port 5	F8H

* : Az SRF-tartomány olyan regiszterei, melyek bitjei egyenként címezhetők (ld. 3.2.5)

Példák a direkt címzésre :

```
MOV   1,0         ; A 0-ás cím tartalma az 1-es címre másolódik.
MOV   0,0D1H      ; A D1H  cím tartalma az 0-ás címre másolódik.
MOV   0,P4        ; A P4 port tartalma a 0-ás címre másolódik.
MOV   P5,P4       ; A P4 port tartalma a P5 portra másolódik.
```

Az SFR tartomány néhány fontos regiszterének nevét a 8051-es assembler ismeri és le is tudja címként fordítani. Az összes többit a merevlemez egy adott könyvtárában található összerendelési lista (un. include fájl) tartalmazza. Ez a listát be kell építeni a forrásprogramunkba az alábbi módon:

```
LSTOUT-                 ; Listázás letiltása
$[path]SFR80515.INC     ; Definíciók hozzáfűzése a programhoz.
LSTOUT+                 ; Listázás engedélyezése.
```

Az SFR80515.INC nevű fájl is egy assembly programrészlet, de csak assembler direktívákat tartalmaz (szimbolikus nevekhez értéket rendel). A fenti három sor közül az első arra utasítja az assemblert, hogy a fordításról innen kezdve ne készítsen listát a .LST fájlba. A második sor azt jelenti, hogy a megadott programmodult is be kell fordítani a program adott helyére, a harmadik sor pedig ismét engedélyezi a listázást.

3.2.2. Indirekt címzés (register indirect addressing)

Indirekt címzésre a nyolc általános célú regiszter (R0 – R7) közül az R0 és az R1 használható.

Általános szintaxis :
```
MOV   @Ri, cím    ; i = 0 vagy 1
MOV   @Ri, A
MOV   @Ri, #adat
MOV   cím, @Ri
MOV   A, @Ri
```

Az **R0 vagy R1 indexregiszter** a belső indirekt címezhető memória egy rekeszének 8 bites címét tartalmazza, amelyik a forrás tartalmával feltöltődik ill. amelynek tartalma a célba másolódik. A forrás ill. a cél lehet egy belső direkt bájtcím (**cím)** vagy az **akkumulátor**. A forrás 8 bites konstans (#adat) is lehet.

Korlátozás : csak az **egyik** operandus lehet indirekt címzésű.

1. példa :

```
      MOV   @R0,5     ; Az 5-ös cím tartalma az R0-ban található címre másolódik.
```

2. példa :

```
      MOV   R1, #147  ; A 147-es memóriacímre
      MOV   @R1, #12H ; 12H konstans érték töltődik a két utasítás hatására.
      MOV   A, @R1    ; Az akkumulátorba 12H konstans érték töltődik.
      MOV   A, R1     ; Az akkumulátorba decimális 147 konstans érték töltődik
                      ; 1 bájtos, 1 ciklusig tartó utasítás.
      MOV   ACC, R1   ; 2 bájtos, 2 ciklusig tartó utasítás.
```

3.2.3. Regisztercímzés (register addressing)

Általános szintaxis:
```
MOV   Rn, A       ; n = 0-7
MOV   A, Rn
```

Cél és forrás az **R0 - R7** regiszterek egyike vagy az **akkumulátor** lehet.

Korlátozás : A cél **és** a forrás **is** nem lehet az R0-R7 regiszterek egyike. Ezen regiszterek közötti adatátvitel csak egy memóriarekesz vagy az akkumulátor közbeiktatásával lehetséges (minimum 2 utasítás). Az R0-R7 regiszterek az alsó 128 bájt belső RAM egyik regiszterbankjából választhatók (ld 3.8.3).

```
Példák : MOV   R7,A    ; Az akkumulátor tartalma az R7-be másolódik.
         MOV   A,R0    ; Az R0 tartalma az akkumulátoron keresztül
         MOV   R1,A    ; (korlátozás!) az R1-be másolódik.
```

Egy másik lehetőség, ha az akkumulátor helyett egy belső direkt címezhető memóriarekeszt használunk :

Általános szintaxis:
```
MOV   cím, Rn; n = 0-7
MOV   Rn, cím
```

Példa : `MOV  P4,R7  ; Az R7 regiszter tartalma (regisztercímzés) a P4 portra másolódik.`

3.2.4. Csatolt címzés (immediate addressing)

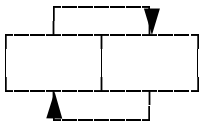
Általános szintaxis:
```
MOV   cím, #adat
MOV   Rn, #adat    ; n = 0-7
MOV   A, #adat
```

A forrás egy 8 bites (bináris, hexadecimális, decimális vagy ASCII) **konstans** (#adat). A cél egy belső direkt bájtcím , az **R0**-**R7** regiszterek egyike vagy az akkumulátor.

```
Példák : MOV   A,#0     ; Az akkumulátorba 0-t töltünk.
         MOV   R5,#'A'  ; Az R5 regiszterbe az A betű ASCII kódja (41H) töltődik.
         MOV   P4,#0FFH ; A P4 port minden bitje 1-es lesz.
```

További, az akkumulátorra vonatkozó utasítások :

Az akkumulátor alsó és felső bitnégyeseinek (nibble) cseréje : **SWAP A**



Regisztertartalom-csere az akkumulátor és az R0-R7 regiszterek egyike, egy belső direkt címezhető vagy az **R0** ill. **R1** idexregiszterekkel indirekt címzett memóriarekesz tartalma között :

```
                    XCH     A, Rn      ; n = 0-7
                    XCH     A, cím
                    XCH     A, @Ri     ; i = 0 vagy 1

        Valamint    XCHD    A, @Ri     ; azonos az előzővel, de csak az alsó
                                         négy bitet (nibble) cseréli.
```

Összefoglalás : A mikrokontrolleren belüli adatmozgatások
(8 bites MOV utasítások)

Lehetséges címzésmódok :
- -> direkt címzés : A cím 0 - 255 közé esik.
- -> indirekt címzés : A cím az R0-ban vagy az R1-ben.
- -> regisztercímzés : R0 - R7
- -> csatolt címzés : Feltöltés konstanssal.

Korlátozások :

- -> Csak az egyik oparandus lehet indirekt címzésű.
- -> Csak az egyik oparandus lehet regisztercímzésű.
- -> Az indirekt és a regisztercímzés nem kombinálható.
- -> A B segédakkumulátor (F0H az SFR tartományban) csak direkt címezhető.
- -> Konstans természetesen csak forrás lehet (cél nem).

3.2.5. Bitmozgatások és bitműveletek

A 80515 a következő adatméretekkel dolgozik :

- -> 8 bites: bájt
- -> 16 bites: kettősbájt (word, szó)
- -> 4 bites: félbájt (halfbyte, nibble)
- -> **bit**

Összesen 256 direkt címezhető bitpozíció létezik (ld. a következő oldalon) :

-> *128 1 bites rekesz az alsó RAM-tartományban*

Címmegadás :
=> direkt bitcímek: (**0** - **7FH**)
vagy => direkt bájtcímek a bitpozíció megadásával: (**20H.0** - **2FH.7**)
(Lista a következő oldalon.)

-> *128 1 bites rekesz az SFR tartományban*

Minden olyan regiszter bitjei, amelyek címe 8-al maradék nélkül osztható.
Címmegadás :
=> A bájtok szimbolikus neve és bitpozíciója:
(**P0.0** - **P5.7**). (Lista a következő oldalon.)
vagy => A bitek szimbolikus neve: (**IT0** - **BD**)
(Lista a következő oldalon.)
=> Szimbolikus név nélkül bitek bitcímei: (80H-0FFH)

A belső adatmemória bitcímezhető pozíciói

	(MSB)						(LSB)	
7FH								127
30H								48
2FH	7F	7E	7D	7C	7B	7A	79	78 47
2EH	77	76	75	74	73	72	71	70 46
2DH	6F	6E	6D	6C	6B	6A	69	68 45
2CH	67	66	65	64	63	62	61	60 44
2BH	5F	5E	5D	5C	5B	5A	59	58 43
2AH	57	56	55	54	53	52	51	50 42
29H	4F	4E	4D	4C	4B	4A	49	48 41
28H	47	46	45	44	43	42	41	40 40
27H	3F	3E	3D	3C	3B	3A	39	38 39
26H	37	36	35	34	33	32	31	30 38
25H	2F	2E	2D	2C	2B	2A	29	28 37
24H	27	26	25	24	23	22	21	20 36
23H	1F	1E	1D	1C	1B	1A	19	18 35
22H	17	16	15	14	13	12	11	10 34
21H	0F	0E	0D	0C	0B	0A	9	8 33
20H	7	6	5	4	3	2	1	0 32
1FH	Regiszterbank 3							31
18H								24
17H	Regiszterbank 2							23
10H								16
0FH	Regiszterbank 1							15
08H								8
07H	Regiszterbank 0							7
00H								0

Az SFR tartomány bitcímezhető pozíciói

F8H	FFH	FEH	FDH	FCH	FBH	FAH	F9H	F8H	P5
F0H	F7H	F6H	F5H	F4H	F3H	F2H	F1H	F0H	B
E8H	EFH	EEH	EDH	ECH	EBH	EAH	E9H	E8H	P4
E0H	E7H	E6H	E5H	E4H	E3H	E2H	E1H	E0H	ACC
	BD	**CLK**	**-**	**BSY**	**ADM**	**MX2**	**MX1**	**MX0**	
D8H	DFH	DEH	DDH	DCH	DBH	DAH	D9H	D8H	ADCON
	CY	**AC**	**F0**	**RS1**	**RS0**	**OV**	**F1**	**P**	
D0H	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H	PSW
	T2PS	**I3FR**	**I2FR**	**T2R1**	**T2R0**	**T2CM**	**T2I1**	**T2I0**	
C8H	CFH	CEH	CDH	CCH	CBH	CAH	C9H	C8H	T2CON
	EXF2	**TF2**	**IEX6**	**IEX5**	**IEX4**	**IEX3**	**IEX2**	**IADC**	
C0H	C7H	C6H	C5H	C4H	C3H	C2H	C1H	C0H	IRCON
	EXEN2	**SWDT**	**EX6**	**EX5**	**EX4**	**EX3**	**EX2**	**EADC**	
B8H	BFH	BEH	BDH	BCH	BBH	BAH	B9H	B8H	IEN1
B0H	B7H	B6H	B5H	B4H	B3H	B2H	B1H	B0H	P3
	EAL	**WDT**	**ET2**	**ES**	**ET1**	**EX1**	**ET0**	**EX0**	
A8H	AFH	AEH	ADH	ACH	ABH	AAH	A9H	A8H	IEN0
A0H	A7H	A6H	A5H	A4H	A3H	A2H	A1H	A0H	P2
	SM0	**SM1**	**SM2**	**REN**	**TB8**	**RB8**	**TI**	**RI**	
98H	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H	SCON
90H	97H	96H	95H	94H	93H	92H	91H	90H	P1
	TF1	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**	
88H	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H	TCON
80H	87H	86H	85H	84H	83H	82H	81H	80H	P0

LSB: **L**east **S**ignificant **B**it - legkisebb helyiértékű bit
MSB: **M**ost **S**ignificant **B**it - legnagyobb helyiértékű bit

Bitmozgató utasítások :

Általános szintaxis : **MOV C, bitcím**
MOV bitcím, C

Minden bitmozgatás a carry-n keresztül történik.. A bitcímezhető memóriapozíció tartalma bemásolódik a carry-be illetve átveszi a carry tartalmát.

```
Példák :        MOV   C, P4.0     ;A P4 port 0-ás bitje a carry-be.
                MOV   20H.7, C    ;A carry a RAM 20H című rekeszének 7-es bitjére.
                MOV   7H, C       ;A carry a 7-es bitcímre (ua. mint az előző
sor).
```

Bitpozíciók beállítása, törlése és komplementálása :

```
Általános szintaxis : SETB bitcím    ; bit = 1 (CY is)    ,2 bájtos utasítás
                      SETB C         ;   C = 1            ,1 bájtos utasítás
                      CLR bitcím     ; bit = 0 (CY is)    ,2 bájtos utasítás
                      CLR C          ;   C = 0            ,1 bájtos utasítás
                      CPL bitcím     ; bit = bit (CY is)  ,2 bájtos utasítás
                      CPL C          ;   C = C
```

(A **JBC bitcím,rel** feltételes ugróutasítás is törli a feltételbitet, ha az ugrást végrehajtja. ld. 3.4.3.1)

Programpélda :

```
;A P1.0, P1.1 kapcsolóinak állapotát a P1.2, P1.3-ra másolja végtelen ciklusban
          ORG       0200H     ; A program kezdőcíme
H1        EQU       P1.0      ; A P1.0 - P1.3 bitcímeinek hozzárendelése
H2        EQU       P1.1      ; a H1 - H4 szimbolikus nevekhez.
H3        EQU       P1.2
H4        EQU       P1.3
          SETB      H1        ; H1, H2 logikai 1 (bemenetként fogjuk használni, ld 1.3).
          SETB      H2
HUROK:    MOV       C,H1      ; A H1 kapcsoló beolvasása.
          MOV       H3,C      ; H3 kimeneti bit írása.
          MOV       C,H2      ; A H2 kapcsoló beolvasása.
          MOV       H4,C      ; H4 kimeneti bit írása.
          LJMP      HUROK     ; Ismétlés (csak RESET-tel megszakítható)
          END
```

3.3. Hozzáférés a külső program- és adatmemóriához

3.3.1. A külső adatmemória adatmozgatásai

Általános szintaxis : **MOVX A, @DPTR**
MOVX @DPTR, A

A külső adatmemóriához (ld. 1.4) csak *indirekt módon, a Data-Pointer (DPTR)* segítségével lehet hozzáférni. A DPTR egy olyan (16 bites) regiszter, amely a külső adatmemória egy rekeszének címét tartalmazza. A regiszter *az SFR tartományban* található, és DPL (Data-Pointer Low, 82H) ill. DPH (Data-Pointer High, 83H) néven a szokásos utasításokkal elérhető. A külső adatmemória egy 256 bájtos lapján belül az R0-R1 8 bites indexregisztereket is használhatjuk indirekt címzésre, ha előzőleg a cím felső bájtját (lapsorszám) a P2 portra kiadtuk.

Általános szintaxis : **MOVX A, @Ri** ; i = 0 vagy 1
MOVX @Ri, A

A DPTR-re vonatkozó utasítások :

A DPTR feltölthető egy 16 bites konstanssal, amelyet az assembler a külső adatmemória címeként értelmez és fordít:

MOV DPTR, #adat ; 16 bites konstans

A DPTR inkrementálása : **INC DPTR**

A DPTR dekrementálására nincs utasítás, de a DPL és DPH külön-külön dekrementálásával programból megoldható:

```
          DEC  DPL
          JNC  TOVABB
          DEC  DPH
TOVABB:   stb.
```

3.3.2. A külső programmemória adatmozgatásai

A külső programmemóriát csak olvasni lehet (mivel ROM) :

MOVC A, @A+DPTR

Az utasítás a külső programmemória azon címének tartalmát olvassa, mely cím az A akkumulátor és a DPTR előjelnélküli tartalmának összege (*bázisrelativ indirekt címzés; base-register plus index-register addressing*).

vagy : **MOVC A, @A+PC**

amikor a DPTR helyett a PC-t (Program Counter) használjuk.

Példa :

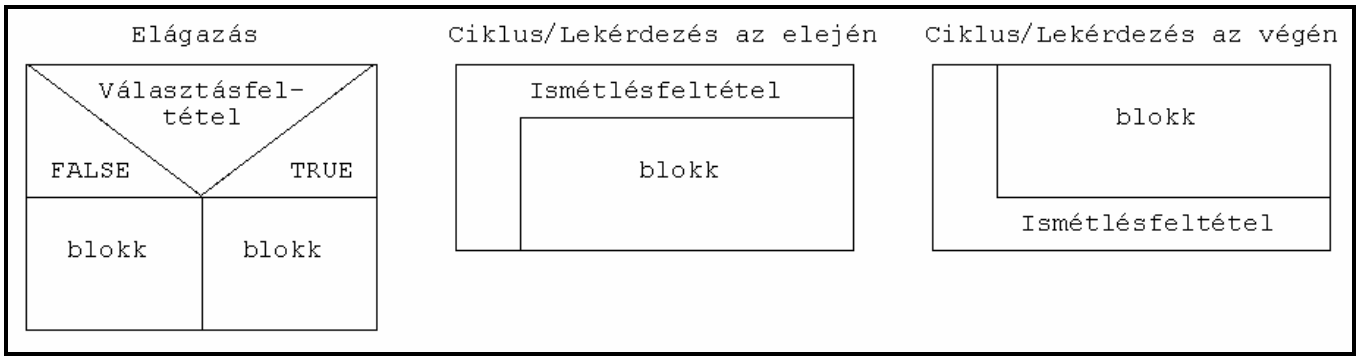
```
;A program saját magát listázza a P4 porton.
;
H1        EQU   P1.0                  ; A H1 szimbolikus névhez a P1.0 bitcímének
                                      ; hozzárendelése.
MONITOR   EQU   8000H                 ; A monitorprogram belépési címe.
          ORG   200H                  ; A program kezdőcíme.
START:    MOV   DPTR,#START           ; A DPTR-be a program kezdőcímének betöltése.
          SETB  H1                    ; H1 bemenet.
HUROK:    CLR   A                     ; Az akkumulátor törlése. A MOVC utasítás
          MOVC  A,@A+DPTR             ; a DPTR-ben lévő címet használja.
          MOV   P4,A
VAR1:     JB    H1,VAR1               ; Várakozás a H1 kikapcsolására,
VAR2:     JNB   H1,VAR2               ; majd újrabekapcsolására.
          INC   DPTR                  ; Következő cím.
          MOV   A,#HIGH(VEGE)         ; A VEGE címkével jelzett utasítás címének
                                      ; felső bájtja (az assembler számítja ki).
          CJNE  A,DPH,HUROK           ; A HUROK címkével jelzett sorra ugrik, ha a
                                      ; VEGE és a DPTR felső bájtjai nem egyeznek.
          MOV   A,#LOW(VEGE)          ; A VEGE címkével jelzett utasítás címének
                                      ; alsó bájtja (az assembler számítja ki).
          CJNE  A,DPL,HUROK           ; A HUROK címkével jelzett sorra ugrik, ha a
                                      ; VEGE és a DPTR felső bájtjai nem egyeznek.
          LJMP  MONITOR
VEGE:     END                         ; Ez már nem része a programnak.
```

(Az MVUS 535 oktatórendszerben 64 kilobájt EEPROM adatmemória található, amelynek alsó 32 kilobájtja programmemóriaként is elérhető.)

3.4. Elágazó és ugróutasítások

3.4.1. Elágazások és ciklusok

Amint az a magasszintű nyelvekben szokásos, az assembly programokban is lehetséges elágazások és ciklusok használata. Elágazást alkalmazunk, ha egy ugrás a program vége felé mutat. A ciklus egy adott programrészlet ismételt végrehajtását jelenti a programkezdet irányába végrehajtott ugrás által. A ciklusismétlés befejezése az ismétlésfeltétel megváltozásával lehetséges. A gépközeli programozás során a feltételes ugrások végrehajtását a flag-ek vagy a belső bitcímezhető memória egyes bitpozícióinak állapota vezérelheti.



Az ugróutasításokat az ***ugrási feltétel*** alapján két csoportra oszthatjuk :

- -> ***Feltétel nélkül*** végrehajtott ugrások.
  Ide tartoznak az *abszolút*, az *indirekt* és a *relatív ugrások*.
- -> Valamilyen ***feltétel megléte estén*** végrehajtott ugrások.
  A *feltételes ugrások* mellett ide tartoznak a *kombinált összehasonlító- és ugróutasítások*. Ezek valamennyien relatív ugrások.

Az ugróutasításokat az ***ugrási cél*** megadási módja szerint is megkülönböztethetjük :

- -> Az ugrási cél ill. célcím direkt megadása. Ilyenkor csak abszolút ugrás használható.

-> Az ugrási cél ill. célcím egy ***regiszterben*** található. Ezek indirekt ugrások.

-> Az ugrási cél ***távolságát*** adjuk meg a PC pillanatnyi értékéhez képest. Ide tartoznak a *relatív* ugrások és az *összehasonlítás* eredményétől függő feltételes ugrások.

3.4.2. Feltétel nélküli ugrások

3.4.2.1. Abszolút ugrás

Általános szintaxis : **LJMP cím** A **cím** 16 bites, 00H és 0FFFFH közé eshet.

Feltétel nélkül az operandusban megadott címre ugrik. (Long jump; 3 bájtos utasítás.)

Példa : LJMP 8000H ; Ugrás a monitorprogramra.

Létezik egy 2 bájtos abszolút ugrás is, kisebb távolságra:

Általános szintaxis : **AJMP cím** A **cím** 11 bites.

2 bájtos utasítás, a PC inkrementálódik, majd a cím a PC utolsó 11 bitpozícióját állítja be, az első három bit az első utasitásbájtban tárolódik. Csak egy 2 kilobájtos lapon belüli címzésre használható. (A fordító hibaüzenetet küld, ha a cím laphatáron kívülre mutat.)

3.4.2.2. DPTR relatív indirekt ugrás

Általános szintaxis : **JMP @A+DPTR**

Az ugrási cél címe az akkumulátor és a DPTR előjel nélküli tartalmának összege.

Példák :

```
Cím  | Utasítás |            Forrásszöveg
-----+----------+-------------------------------------------------------------
                          ORG  200H           ; A program kezdőcíme
0200 | 74 04    |        MOV  A,#4
0202 | 90 02 05 |        MOV  DPTR,#START     ; A DPTR-be a START címkével jelölt
                                              ; utasítás címe töltődik.
0205 | 73       | START: JMP  @A+DPTR         ; 1 bájtos utasítás
0206 | 02 03 00 |        LJMP EGY             ; 3 bájtos utasítás
0209 | 02 04 00 |        LJMP KETTO
020C | 02 05 00 |        LJMP HAROM
020F |          |        ----
                          ----
0300 |          | EGY:                        ; Programrészlet EGY
                          ----
0400 |          | KETTO:                      ; Programrészlet KETTO
                          ----
0500 |          | HAROM:                      ; Programrészlet HAROM
                          ----
```

A tényleges cél a KETTO programrészlet.

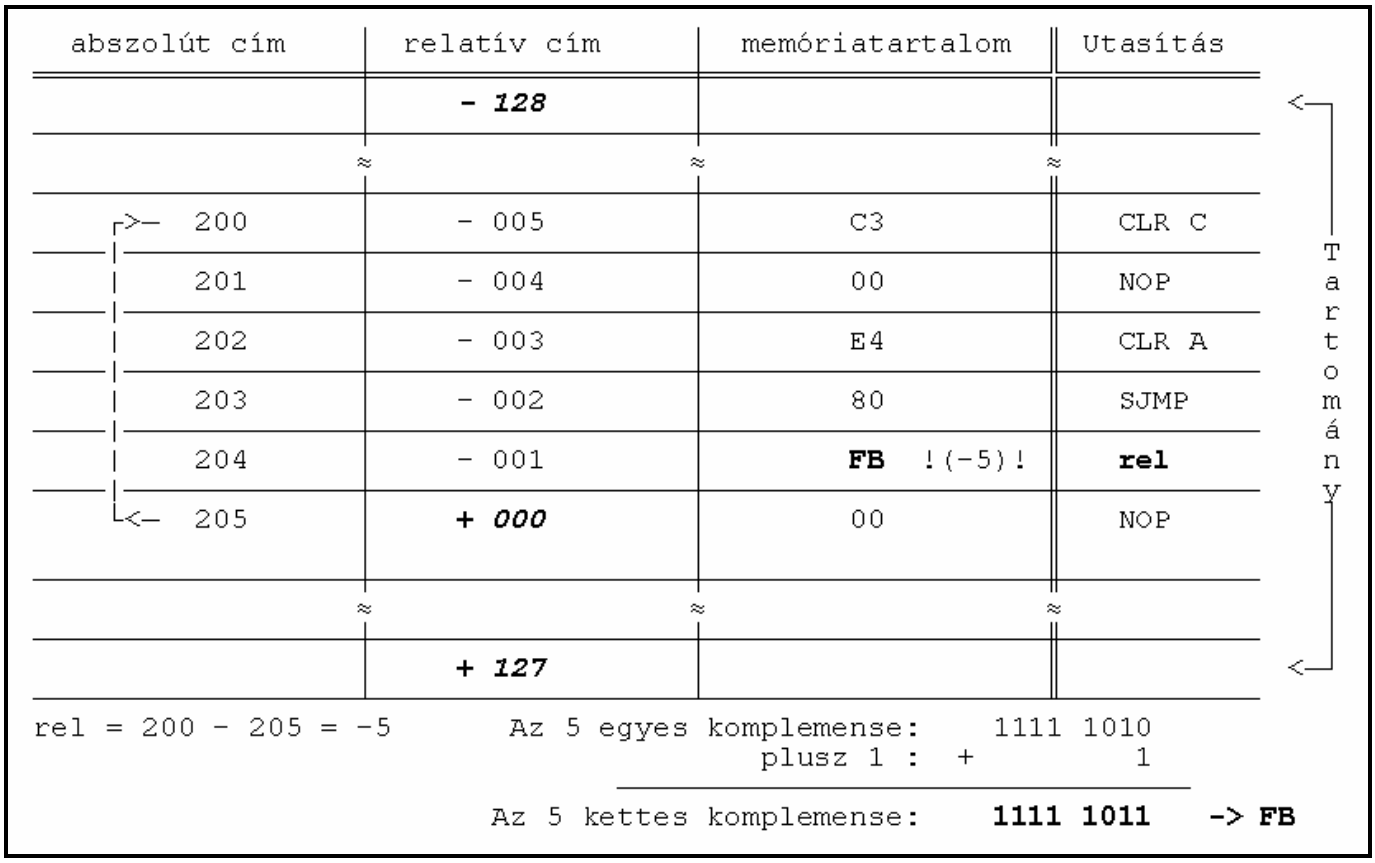
3.4.2.3. Relatív ugrás

Általános szintaxis : **SJMP rel**

A **rel** operandus az ugrási távolság, a tényleges cím a PC pillanatnyi értékéből számítható ki. A távolság *előjeles* 8 bites érték, amit kettes komplemens kódúnak kell tekinteni, ha az ugrás visszafelé mutat (negatív távolság). Az ugrási távolság tehát -128 és +127 közé eshet. *Az utasításszámláló már az aktuális (dekódolt) utasítás végrehajtása előtt a következő utasítás címét tartalmazza.*

Címszámítás : rel = az ugrási cél címe mínusz az SJMP-t követő utasítás címe. Az assembler a **rel** helyén az ugrási cél szimbolikus címét várja és ebből számítja ki a távolságot; ha a megengedett tartományon kívül esik, hibaüzenetet küld.

Példa :



3.4.3. Feltételes ugrások

3.4.3.1. Feltételes relatív ugrások

A feltételes ugrások mindig **relatívak**, azaz az ugrás távolságát adjuk meg. Az utasítások megvizsgálják egy adott bit állapotát, és végrehajtják az ugrást, ha a feltétel teljesül. Feltételbit az alsó RAM tartomány és az SFR tartomány 256 bitcímezhető pozíciójának valamelyike lehet (ld. 3.2.5).

Feltételbitként használható továbbá a carry-flag a PSW (**P**rogram **S**tatus **W**ord) regiszterben és a zéró állapotjelző (ld. 3.6).

```
Általános szintaxis :  JB   bitcím, rel ; Ugrik, ha a bit = 1
(3 bájtos utasítások)  JNB  bitcím, rel ; Ugrik, ha a bit = 0
                       JBC  bitcím, rel ; Ugrik, ha a bit = 1, de törli is a bitet

Általános szintaxis :  JC    rel        ; Ugrik, ha a carry = 1
(2 bájtos utasítások)  JNC   rel        ; Ugrik, ha a carry = 0
                       JZ    rel        ; Ugrik, ha az akkumulátor tartalma 0
                       JNZ   rel        ; Ugrik, ha az akkumulátor tartalma nem 0
```

A **carry flaget** az aritmetikai utasítások (ld. 3.5.3), az összehasonlító utasítások, (ld. 3.4.3.3), a carryt is használó forgatóutasítások (ld. 3.5.2) és a carryt mozgató utasítások befolyásolhatják.

1. példa : Várakozás egy kapcsoló kikapcsolására.

```
          ORG       200H      ; A program kezdőcíme. Ha nem adjuk meg, az assembler
                              ; 0H-nak veszi.
H1        EQU       P1.0      ; A H1 szimbolikus névhez a P1.0 hozzárendelése.
MONITOR   EQU       8000H     ; A monitorprogram címe.
          SETB      H1        ; H1 bemenet.
                    --------  ; stb..
LOOP0:    JB        H1,LOOP0  ; Vár a H1 kikapcsolására (logikai 0-ra)
LOOP1:    JNB       H1,LOOP1  ; Vár a H1 bekapcsolására (logikai 1-re)
                    --------  ; stb.
          LJMP      MONITOR
```

2. példa : A P4 tartalmát a P5-re másolja, ha nulla, a monitorprogramra ugrik.

```
          ORG       200H      ; A program kezdőcíme.
          MOV       P4,#0FFH  ; P4 bemenet.
START:    MOV       A,P4
          JZ        KESZ
          MOV       P5,A
          LJMP      START
KESZ:     LJMP      8000H     ; Ugrás a monitorprogramra.
          END
```

3. példa : P4 inkrementálása vagy dekrementálása a H1 és H2 kapcsolók állapotától függően.

1. megoldás : két egy másutáni inkrementálás ill. dekrementálás közötti időt késleltető-program állítja be.

```
H1        EQU       P1.0      ; A H1 és H2 szimbolikus nevekhez a
H2        EQU       P1.1      ; P1.0 és P1.1 portbitek hozzárendelése.
          ORG       200H      ; A program kezdőcíme
          MOV       P1,#0FFH  ; P1 bemenet.
HUROK:    JNB       H1,INK    ; Ugrik, ha H1 = 0
          JNB       H2,DEK    ; Ugrik, ha H2 = 0
          LJMP      VEGE      ; Az elágazásblokknak csak egy kimenete lehet
                              ; (ld. strukturált programozás)
INK:      INC       P4
          LJMP      VEGE
DEK:      DEC       P4
VEGE:     LCALL     KESL      ; Késleltetőrutin hívása.
          LJMP      HUROK
KESL:     MOV       R7,#0FFH  ; Késleltetés.
KESL1:    MOV       R6,#0FFH
KESL2:    DJNZ      R6,KESL2
          DJNZ      R7,KESL1
          RET
          END
```

2. megoldás : két egymás utáni inkrementálás ill. dekrementálás közötti egy kapcsolóra várakozik.

```
H1        EQU       P1.0      ; A H1 és H2 szimbolikus nevekhez a
H2        EQU       P1.1      ; P1.0 és P1.1 portbitek hozzárendelése.
          ORG       200H      ; A program kezdőcíme.
HUROK:    JB        H1,DEK    ; Ugrik, ha H1 ≠ 0, talán H2.
H1VAR:    JNB       H1,H1VAR  ; Vár a H1 visszakapcsolására.
          INC       P4
DEK:      JB        H2,VEGE   ; Ugrik, ha H2 ≠ 0, Talán H1.
H2VAR:    JNB       H2,H2VAR  ; Vár a H2 visszakapcsolására.
          DEC       P4
VEGE:     LJMP      HUROK
          END
```

3.4.3.2. Feltételes relatív ugrás dekrementálás után

Általános szintaxis : **DJNZ cím,rel**
DJNZ Rn,rel ; n = 0 - 7

A **cím**-en található direkt címzésű belső memóriarekesz, ill. valamely regiszter tartalmát eggyel csökkenti, majd ha az eredmény nem nulla, végrehajtja az ugrást a **rel** operandussal megadott távolságra.

Példa :

```
          ORG       200H      ; A program kezdőcíme.
          MOV       P4,#0FFH  ; A P4 minden bitje 1.
CIKL:     DJNZ      P4,CIKL   ; P4 dekrementálása amíg az eredmény nem nulla.
          LJMP      8000H     ; Ugrás a monitorprogramra.
```

3.4.3.3. Feltételes relatív ugrás összehasonlítás alapján

A processzor a második operandust kivonja az elsőből, de eredmény nem képződik, csak a flagek állítódnak. Ha a két operandus különbözik, végrehajtja az ugrást a **rel** operandussal megadott távolságra.

Általános szintaxis :

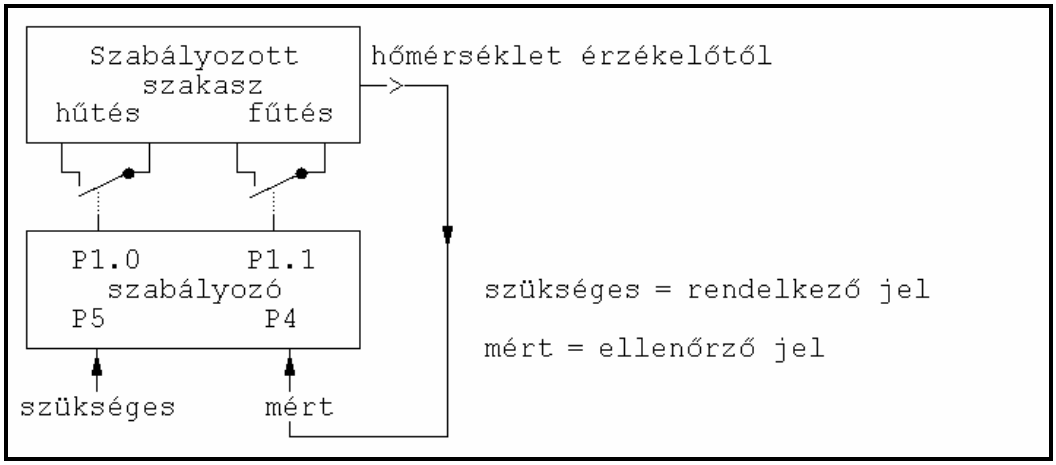
```
CJNE   A, cím, rel
CJNE   A, #adat, rel
CJNE   Rn, #adat, rel     ; n = 0 - 7
CJNE   @Ri, #adat, rel    ; i = 0 vagy 1
```

Ha az első operandus előjelnélküli értéke kisebb mint a másodiké, az összehasonlítás a PSW carry bitjét 1-be állítja, egyébként törli. Az ugrási címen lévő programrészlet a carry értékéből eldöntheti, hogy az első operandus kisebb volt-e a másodiknál.

1. példa : Egy ciklus befejezése, ha a P4 értéke 127.

```
          ORG       0               ; A program kezdőcíme.
          MOV       P4,#0FFH        ; P4 bemenet.
HUROK:    MOV       A,#127
          CJNE      A,P4,FOLYT      ; FOLYT címre ugrik, ha 127-P4≠0
          LJMP      8000H           ; A monitorprogramra ugrik.
FOLYT:    -----------               ; Ciklusmag.
          LJMP      HUROK
          END
```

2. példa : Egy terem hőmérsékletének szabályozása: a P5 porton beállított szükséges értéket összehasonlítja egy hőmérsékletérzékelőnek a P4-re adott 8 bites értékével és az eredménynek megfelelően bekapcsolja a fűtést vagy a hűtést.




```
KELL      EQU       P5              ; Szükséges érték a P5-ön.
VAN       EQU       P4              ; Mért érték a P4-en.
HUTES     EQU       P1.0            ; Hűtéskapcsoló.
FUTES     EQU       P1.1            ; Fűtéskapcsoló.
          ORG       0               ; A program kezdőcíme.
          MOV       KELL,#0FFH      ; A KELL (P5 port) bemenet.
          MOV       VAN,#0FFH       ; A VAN (P4 port) bemenet.
HUROK:    MOV       A,KELL
          CJNE      A,VAN,ELTERO    ; A szükséges és a mért érték összehasonlítása.
          CLR       HUTES           ; A szükséges és a mért érték azonos: hűtés
          CLR       FUTES           ; és fűtés kikapcsol,
          LJMP      LOOPEND         ; és új ciklusvégrehajtás.
ELTERO:   JC        HUTENI          ; Nem azonosak, és a szükséges érték kisebb a
                                    ; mértnél:
          SETB      FUTES           ; fűtés bekapcsol,
          CLR       HUTES           ; hűtés kikapcsol
          LJMP      LOOPEND         ; és új ciklusvégrehajtás.
HUTENI:   SETB      HUTES           ; Egyébként a hűtés bekapcsol, a
          CLR       FUTES           ; fűtés kikapcsol
LOOPEND:  LJMP      HUROK           ; és új ciklusvégrehajtás.
          END
```

3.5. Logikai és aritmetikai műveletek

3.5.1. Logikai műveletek

Az eredmény az első operandusban képződik.

```
Általános szintaxis:   ÉS  :    ANL    A, Rn          ; n = 0 - 7
                                ANL    A, cím
                                ANL    A, @Ri         ; i = 0 vagy 1
                                ANL    A, #adat
                                ANL    cím, A
                                ANL    cím, #adat

                       VAGY :   ORL    A, Rn
                                ORL    A, cím
                                ORL    A, @Ri
                                ORL    A, #adat
                                ORL    cím, A
                                ORL    cím, #adat

                  KIZÁRÓ VAGY : XRL    A, Rn
                                XRL    A, cím
                                XRL    A, @Ri
                                XRL    A, #adat
                                XRL    cím, A
                                XRL    cím, #adat
```

Egyes logikai kapcsolatok bitműveletekben is alkalmazhatók. Az eredmény a carryben képződik.

```
                    ÉS   :      ANL    C, bitcím
                    VAGY :      ORL    C, bitcím
```

Bitműveletekben a kizáró-vagy kapcsolat nem használható.

A logikai műveletek egy adott bájt egyes bitjeit maszkolják, vagyis a biteket pillanatnyi értéküktől függetlenül állíthatjuk logikai 1-be vagy 0-ba.

Példák bitmaszkolásra : MOV A, #01010101B

	ANL A,#0F0H	ORL A,#0F0H	XRL A,#0F0H
1. Maszkolandó operandus, **A** :	01010101B	01010101B	01010101B
2. Maszk (**konstans**) :	11110000B	11110000B	11110000B
Eredmény (az első operandusban) :	01010000B	11110101B	10100101B

További logikai utasítások :

Az akkumulátor törlése és egyes komplemens képzése (invertálása) :

```
                    CLR   A   ; A = 0   , 1 bájtos utasítás
                    CPL   A   ; A = Ā   , 1 bájtos utasítás
```

Példa : A P4.0 és P4.1 bitek összehasonlítása :

Ha a két bit egyenlő, a P4.7-re logikai 1-et, egyébként 0-át kell kiadni. A rutin a bitekre vonatkozó nem létező kizáró-vagy műveletet valósítja meg.

```
BIT1      EQU  P4.0          ; BIT1 bemenet a P4.0
BIT2      EQU  P4.1          ; BIT2 bemenet a P4.1
EXOR      EQU  P4.7          ; Eredménykimenet a P4.7
          ORG  200H          ; A program kezdőcíme.
          MOV  P4,#00000011B ; Bemenetek konfigurálása.
BEOLVAS:  MOV  C,BIT1        ; C ← BIT1
          JNB  BIT2,KESZ     ; A KESZ címre ugrik, ha BIT2 = 0 ,
          CPL  C             ; egyébként : BIT1 = BIT1
KESZ:     MOV  EXOR,C        ; Eredmény a kimenetre.
          LJMP BEOLVAS       ; Ismétlés végtelen ciklusban.
          END
```

3.5.2. Inkrementálás, dekrementálás és forgatás

```
Általános szintaxis:  INC   A        DEC   A
                      INC   Rn       DEC   Rn   ; n = 0 - 7
                      INC   cím      DEC   cím
                      INC   @Ri      DEC   @Ri  ; i = 0 vagy 1
                      INC   DPTR     ---        ; az egyetlen 16 bites inkrementálás
```

---> Ezek az utasítások nem befolyásolják a flagbiteket, kivéve a paritásbitet, ha a céloperandus az akkumulátor.

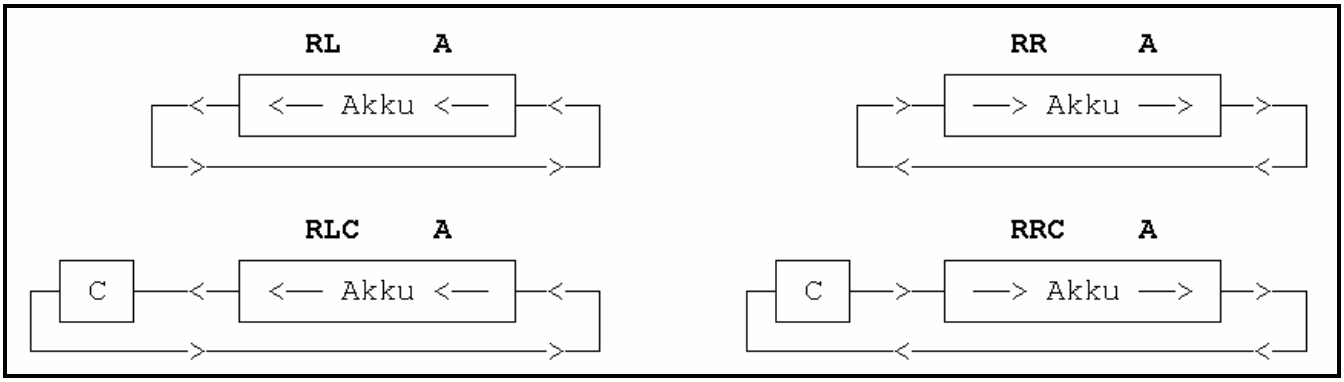
---> A zéró állapotjelző megváltozhat, ha a céloperandus az akkumulátor.

---> Ezek az utasítások a portok **flip-flopjait** olvassák (nem a csatlakozópontokat).

```
Példák :    INC   P5
            INC   R0     ; Az R0 tartalmát inkrementálja.
            INC   @R0    ; Az R0 regiszterrel megcímzett belső memóriarekesz tartalmát
                         ; inkrementálja.
```

Forgatóutasítások **:**

A forgatás **csak az akkumulátorban** történhet. A carry flag módosulhat, ha a forgatásban kilencedik bitként használjuk.



Az RLC A és RRC A utasítások a carryn kívül a paritásbitet is befolyásolhatják.

Egy bájt előjelnélküli értéke duplázódik, ha egy hellyel balra toljuk, illetve feleződik (div 2), ha egy hellyel jobbra toljuk (shifteljük). Forgatás esetén ez csak akkor igaz, ha a carry bitet is használjuk és a forgatás előtt 0-ba állítjuk.

3.5.3. Aritmetikai műveletek

Aritmetikai műveletek esetén először a feldolgozandó adatok **típusát** kell tisztáznunk. Megkülönböztetünk előjeles, előjelnélküli, BCD és lebegőpontos számokat. A mikrokontroller csak a 8 bites előjeles és előjelnélküli egész számok és a BCD számok használatát támogatja. A lebegőpontos számok feldolgozására külön programot kell írnunk, assembly utasítások e célra nincsenek.

8 bites adattípusok :

egész számok		**BCD számok**	Lebegőpontos számok
előjelnélküli	**előjeles**	**előjelnélküli**	mantissza és exponens
0 - 255	**-128 - +127**	**00 - 99**	a mikrokontroller nem támogatja
ADD ——> CY ADDC ——> CY SUBB ——> CY	ADD ——> OV ADDC ——> OV SUBB ——> OV	ADD, DA A ——> CY ADDC, DA A ——> CY | | AC	
MUL ——> OV DIV ——> OV			

A kontroller az összeadás és kivonás aritmetikai műveleteket ismeri, valamint 8 bites bináris *egész* számok szorzását és osztását.

Összeadás ***carry nélkül***: a második operandussal megadott értéket az akkumulátorhoz adja.

```
Általános szintaxis :        ADD     A, Rn      ; n : 0 - 7, regisztercímzés
                             ADD     A, @Ri     ; i : 0 vagy 1, indirek címzés
                             ADD     A, cím     ; direk címzés a belső RAM-ban
                             ADD     A, #adat   ; 8 bites konstans
```

Összeadás ***carryvel*** : a carryt és a második operandussal megadott értéket az akkumulátorhoz adja

```
Általános szintaxis :        ADDC    A, Rn      ; a carry az LSB-hez adódik
                             ADDC    A, @Ri
                             ADDC    A, cím
                             ADDC    A, #adat
```

Kivonás ***csak carryvel*** létezik : a carry és a második operandussal megadott érték összegét kivonja az akkumulátorból.

```
Általános szintaxis :        SUBB    A, Rn      ; a carry az LSB-ből kivonódik
                             SUBB    A, @Ri
                             SUBB    A, cím
                             SUBB    A, #adat
```

Az összeadás és a kivonás eredménye az akkumulátorban keletkezik.

```
Szorzás                :     MUL     AB         ; A x B -> B,A

Osztás                 :     DIV     AB         ; A / B -> A (maradék : B)
```

Az összeadásnál és kivonásnál :

--> A 8 bites *egész számok* lehetnek *előjelesek* vagy *előjelnélküliek.* Az ALU az összeadást és a kivonást *mindig binárisan* végzi. A *megengedett számtartomány* előjeles és előjelnélküli számok esetén különböző. Azt, hogy egy aritmetikai művelet eredménye ezen a tartományon kívül esik-e (az akkumulátorban hamis eredmény van) a PSW *különböző bitjei* jelzik.

Előjelnélküli bináris számok összeadása és kivonása :

-> Megengedett számtartomány 0 - 255.
-> Ha az aritmetikai művelet eredménye a tartományon kívül esik, a carry flag bebillen.
-> A negyedik és ötödik bit közötti átvitelt az **auxillary carry** jelzi. Értékének csak ***BCD számoknál*** van jelentősége és a DAA utasítás (**D**ecimal **A**ccu **A**djust) használja.

Előjeles bináris számok összeadása és kivonása :

-> Megengedett számtartomány -128 - +127. Az első pozitív szám a nulla. A negatív számok kettes komplemens kódúak, jelzésükre a legmagasabb súlyú bitpozícióban álló "1" szolgál.
-> Ha az aritmetikai művelet eredménye a tartományon kívül esik, az overflow flag bebillen.

--> Az aritmetikai műveletek eredménye az akkumulátorban képződik.

--> Az összeadás (ADD) eredményét a carry nem befolyásolja. Ha az összeadáshoz a carry-t is használjuk (ADDC), a legkisebb súlyú bithez (LSB) hozzáadódik. *A kivonás mindig felhasználja a carry bitet is.* Carry nélküli kivonáshoz a művelet előtt a carry-t nullázni kell (CLR C).

Összeadási (ADD) példák, amelyeknél az eredmény minden bináris adattípus esetén a megengedett tartományon kívül esik :

```
                    előjeles          előjelnélküli            bináris

                       -61                195                 11000011
                     + -86              +  170               +  10101010
                     --------           ---------            ----------
                      -147                365                101101101
                        |                  |                   |    |
            Flag-ek : OV = 1             C = 1-----------------+    0 = AC

Az akkumulátor tartalma: :   109               109
```

A szorzásnál és osztásnál :

--> A műveletben használt 8 bites egész számok ***előjelnélküliek***. A számtartomány tehát 0 - 255. Az eredmény szintén előjelnélküli egész szám (szorzásnál 16, osztásnál 8 bites.
--> *A két operandust az akkumulátorba és a B segédakkumulátorba kell beírni.*

Szorzás **:**
-> Két 8 bites szám szorzásakor eredményként egy 16 bites értéket várunk. *A **B** segédakkumulátorban kapjuk meg az eredmény felső bájtját, az **A** akkumulátorban pedig az alsót.*
-> Ha az eredmény nagyobb 255-nél, azaz a B segédakkumulátorban nullánál nagyobb érték található, az overflow flag bebillen.
-> A **carry flag** mindig törlődik.

Osztás **:**
-> A művelet az akkumulátor tartalmát osztja a B segédakkumulátor tartalmával.
-> Az osztás után az *akkumulátorban található az eredmény és a B segédakkumulátorban a maradék (mindkettő egész szám!).*
-> Az **overflow** és a **carry** bitek törlődnek.
-> Az **overflow** flag bebillen, ha *nullával való osztást* akarunk végezni, vagyis ha a B segédakkumulátor értéke nulla volt. Az eredmény ilyenkor nem értelmezhető.

DIV AB az alábbi két PASCAL utasításnak felel meg :

X := A; **A := X DIV B; B := X MOD B;**

Szorzási példa (MUL) :

előjelnélküli bináris

```
     195                  11000011   : akkumulátor
  *  170               *  10101010   : B segédakkumulátor
-------------------------------------
   33150              10000001 01111110
                        |       |
                        B       A
                      129     126
```

Flag-ek : **OV = 1 C = 0**

Osztási példák (DIV) :

előjelnélküli bináris

```
     195                  11000011   : akkumulátor
   /  170               /  10101010   : B segédakkumulátor
-------------------------------------
        1 (maradék 25)  00011001 00000001
                          |       |
                          B     Akku
                          25      1
```

Flag-ek : **OV = 0 C = 0**

Számolás BCD számokkal :

A mikrokontroller csak binárisan tud számolni. Kétjegyű BCD számokkal csak ***összeadás*** végezhető, és az is csak úgy, hogy *rögtön utána* végrehajtjuk a decimális korrekció utasítást (**DA A**) is. Az utasítás megvizsgálja az auxillary flaget, és ha az bebillent, vagy az akkumulátor alsó négy bitjének tartalma nagyobb 9-nél akkor az eredményhez hozzáad 6-ot. Ha a carry értéke az összeadás előtt vagy után 1, vagy a felső négy bit értéke nagyobb kilencnél, akkor még hozzáad 60H-t. Az akkumulátor és a carry együttesen tartalmazza a korrigált, helyes eredményt.

3.6. A program állapotszó - PSW

A PSW az SFR tartományban található a RAM-ban a 0D0H címen. Egyes bitjeihez bitcímeken (0D0H - 0D7H) illetve a szimbolikus bájtnév és bitpozíció (PSW.0 - PSW.7) vagy a szimbolikus bitnév (pl. P vagy CY) használatával férhetünk hozzá. (Lista a 27. oldalon).

7	6	5	4	3	2	1	0	
CY	AC	F0	RS1	RS0	OV	F1	P	(D0H)
D7	D6	D5	D4	D3	D2	D1	D0	← bitcímek

CARRY CY : bebillen, ha a művelet eredménye az előjelnélküli számok tartományán kívül esik (a hetedik bitről történt átvitel).

AUXILIARY CARRY AC : bebillen, ha az akkumulátor D3 és D4 bitjei között átvitel keletkezik BCD számokkal végzett műveleteknél. BCD számolási módban a DAA (**D**ecimal **A**ccu **A**djust) használata szükséges;

OVERFLOW OV : bebillen, ha a művelet eredménye az előjeles számok tartományán kívül esik (a hatodik bitről történt átvitel).

PARITY P : Az akkumulátor logikai 1-eseinek számát a paritásbit páros számra egészíti ki;

FLAGS F0,F1 : Állapotjelzőként használható bitpozíciók, melyeket a processzor nem használ.

REGISTERBANK-SELECT RS0,RS1 : Választóbitek, amelyek kijelölik, hogy az R0- R7 regiszterek melyik regiszterbankban találhatók;(ld.3.7.3)

Figyelem :
A paritásbit kivételével valamennyi flaget az ALU állítja. A paritásflaget az akkumulátor közvetlenül vezérli.

A **CARRY-flaget** csak az aritmetikai (ld. 3.5.3), összehasonlító (ld. 3.4.3.3) és a carry-t is használó forgatóutasítások (ld. 3.5.2) befolyásolhatják. A flag állapota programelágazások feltételeként lekérdezhető (ld. 3.4.3).

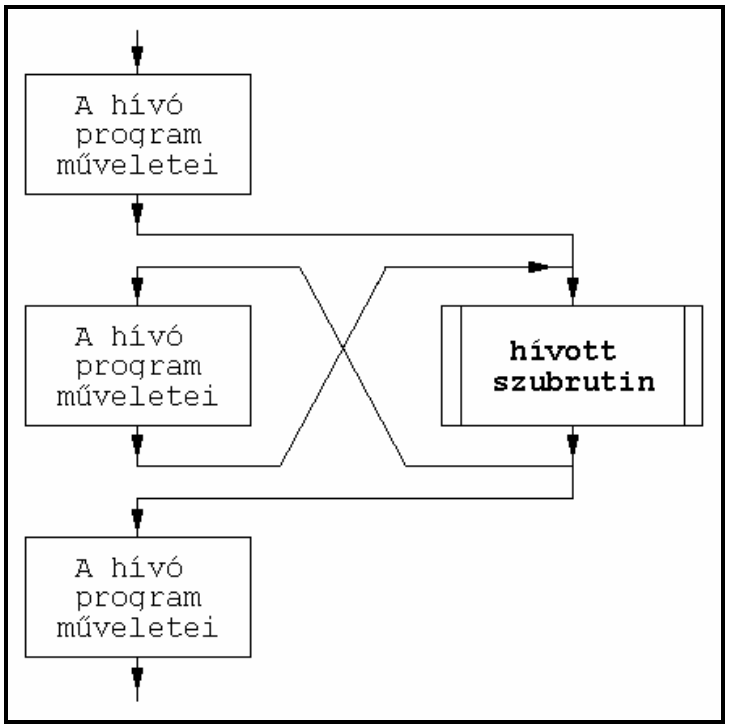
A **PARITÁS-flaget** minden olyan adatművelet befolyásolhatja, amelynek célja az akkumulátor, kivéve a carry-t nem használó forgatásokat és a SWAP A utasítást.

3.7. Szubrutintechnika

3.7.1. Áttekintés

Ha egy kódsorozat (programrészlet) használata egy programban többször szükséges, takarékoskodhatunk a programmemóriával, ha szubrutint (alprogramot) írunk. A szubrutinokkal nem csak memóriát spórolhatunk, de programunk áttekinthetőbb és könnyebben olvasható is lesz. Arra is lehetőségünk van, hogy a szubrutinnak adatokat adjunk át regisztereken vagy adatmemória címeken keresztül (paraméterátadás). Ezeket az adatokat **globálisan**, a program bármelyik részében (és nem csak **lokálisan**, az illető szubrutinban) változtathatjuk meg. Egyes magasszintű programnyelvek paraméterátadásra a veremtárat használják. A szubrutinok további szubrutinokat hívhatnak, hívhatják saját magukat is (**rekurzív hívás**), ha biztosított, hogy a hívó programhoz visszatalálnak. A **veremtároló** szolgál arra a célra, hogy egy alprogram végrehajtása után az őt hívó program megfelelő pontjára visszatérhessünk.

Folyamatábra :



→ A szubrutinoknak **általánosan használhatóknak** és **jól dokumentáltaknak** kell lenniük, hogy egy "könyvtárban" tárolhassuk és későbbi programokban is használhassuk őket.

→ Egy szubrutin **áthelyezhető** (relokálható), ha a programmemóriában bárhol elhelyezhető, tehát nem tartalmaz abszolút címeket.

→ Egy szubrutin **újrabelépő** (re-entrant), ha egy megszakítás kiszolgálása után folytatni lehet még akkor is, ha a megszakítást kiszolgáló program éppen azt a szubrutint hívja meg, amelynek végrehajtását megszakította.

→ Egy szubrutin **rekurzív**, ha saját magát is hívhatja, de akkor újrabelépőnek is kell lennie, hogy a hívó programhoz visszataláljon.

Általános szintaxis :

Hívás (a hívó programban) : **LCALL cím (16 bit)** 3 bájtos utasítás
vagy **ACALL cím (11 bit)** 2 bájtos utasítás
Visszatérés (a hívott program végén) : **RET**

Assembly forrásprogramban szubrutinhíváshoz szimbolikus címet használunk.

12 MHz-es órajel mellett mindkét utasítás végrehajtása 2 µs-ig tart.

3.7.2. A verem felhasználása

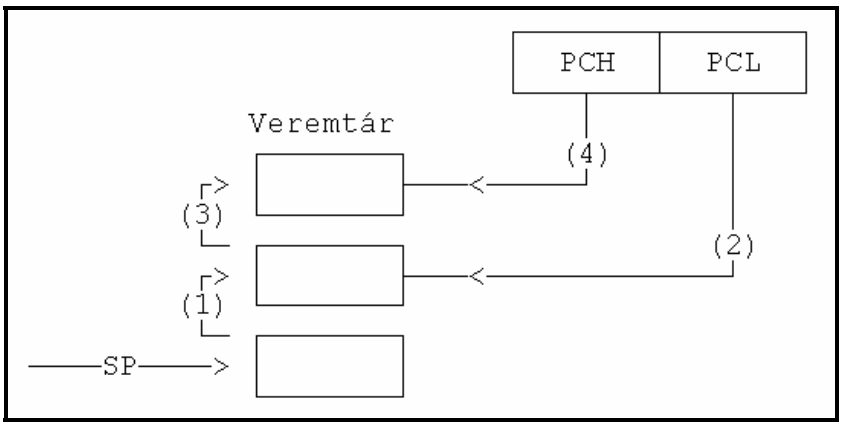
A **verem** (**stack**, **veremtároló**) egy belső RAM-tartomány, amely a szubrutinok és megszakítás-kiszolgálások végrehajtása utáni ***visszatérési címeket*** tárolja, illetve ahová ***belső direkt címezhető*** rekeszek tartalmát elmenthetjük (PUSH). Minden bájt, amit a verembe teszünk eggyel növeli, minden bájt amit kiveszünk eggyel csökkenti a veremmutatót. A veremmutató (**Stack-Pointer : SP)** az SFR tartományban található (címe : 81H). A verem kezdőcíme RESET után 07H a belső indirekt címezhető RAM-ban. Mivel az első bájt verembe töltésekor először a veremmutató nő eggyel, és csak utána tárolódik az adat, a verem gyakorlatilag 08H címen kezdődik. A veremtárat a programunknak máshová (feljebb) kell áthelyeznie, ha a második (vagy további) regiszterbank és/vagy bitcímezhető memóriapozíciók használatára van szükségünk az alsó RAM-tartományban:

```
pl.:    MOV   SP,#02FH    ; A verem a 30H címen kezdődik.
```

Mivel a verem a belső RAM-ban található, a mérete korlátozott. A programozás során meg kell győződni arról, hogy csak kevéssé van igénybevéve. A programozón múlik, hogy a verem méretét és helyét kézben tudja-e tartani. A veremmutató 8 bites, így kihasználhatja a teljes belső indirekt címezhető memóriát, beleértve a regiszterbankok és a bitcímezhető pozíciók tartományát is (ha az SP túlcsordul, a 0-ás címtől kezdi tárolni az adatokat, a program "elstackeli" magát).

A visszatérési cím mentése szubrutinhívás (LCALL) esetén :

Egy szubrutinhívás végrehajtása során először a veremmutató nő (1), majd erre a címre a visszatérési cím alsó bájtja kerül (2), azután a veremmutató újra nő eggyel (3) és az új címre kerül a felső bájt (4). A **RET** utasítás hatására mindez fordított sorrendben játszódik le.

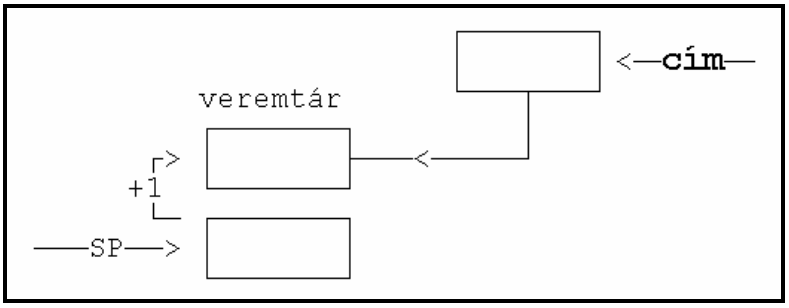


Egy ***megszakítás kiszolgáló*** program indulásakor (ld. 3.8) a visszatérési cím (a megszakított program következő utasításának címe) ugyanígy mentődik a verembe. A program végét jelző RETI utasítás hatása is ugyanaz mint itt a RET utasításé.

Belső direkt címezhető memóriatartalom mentése :

Általános szintaxis : **PUSH cím** ; a verembe másol
POP cím ; a veremből visszahív

A PUSH utasítás végrehajtása során először a veremmutató nő, majd a **cím**-en tárolt bájt a verembe íródik. A visszaírás fordított sorrendben hajtódik végre. (**cím**: a direkt címezhető belső RAM egy rekeszének címe).



Az R0-R7 regiszterek tartalma a PUSH és POP utasításokkal csak fizikai címükön keresztül menthető, így azt is tudnunk kell, hogy melyik regiszterbankban találhatók. Jobb módszer az R0-R7 mentésére a regiszterbank átkapcsolása (ld. 3.7.3).

Tartalomcsere két memóriarekesz között : pl. 10H és 20H

```
        PUSH    10H         ; A 10H cím tartalma a verembe.
        MOV     10H,20H     ; A 20H tartalma a 10H-ba másolódik.
        POP     20H         ; Veremtartalom a 20H-ba (a 10H korábbi tartalma)
                            ; (ld. még "XCH....")
```

A szubrutin keretezése : Minden olyan regisztert (és minden olyan direkt címezhető memóriatartalmat) a verembe kell menteni, melyek értékét a hívó program számára meg kell őrizni, és amelyeket a hívott szubrutin megváltoztat. Ezek általában az akkumulátor és a PSW. Szükséges lehet továbbá az R0-R7 regiszterek mentésére a regiszterbank átkapcsolása (ld. 3.7.3). A veremtár korlátozott mérete miatt fontos, hogy a PUSH utasítást csak a legszükségesebb esetekben használjuk. Rekurzív szubrutinhívások esetén természetesen csak ezt használhatjuk, vagy a külső memóriában kell stack-et kialakítanunk saját program segítségével.

Példa :

```
SZUBR:  PUSH    ACC     ; Az akkumulátor mentése.
        PUSH    PSW     ; A PSW mentése.
        SETB    RS0     ; Az R0-R7 a regiszterbank 1-ben használható, de akkor a
                        ; verem nem kezdődhet a 08H-n (SP átállítandó!).
        ----------      ; Maga a szubrutin.
                        ; Visszaállítások fordított sorrendben:
        CLR     RS0     ; A regiszterbank visszaállítása.
        POP     PSW     ; A PSW visszaállítása
        POP     ACC     ; Az akkumulátor visszaállítása
        RET             ; Visszatérés a hívó programba.
```

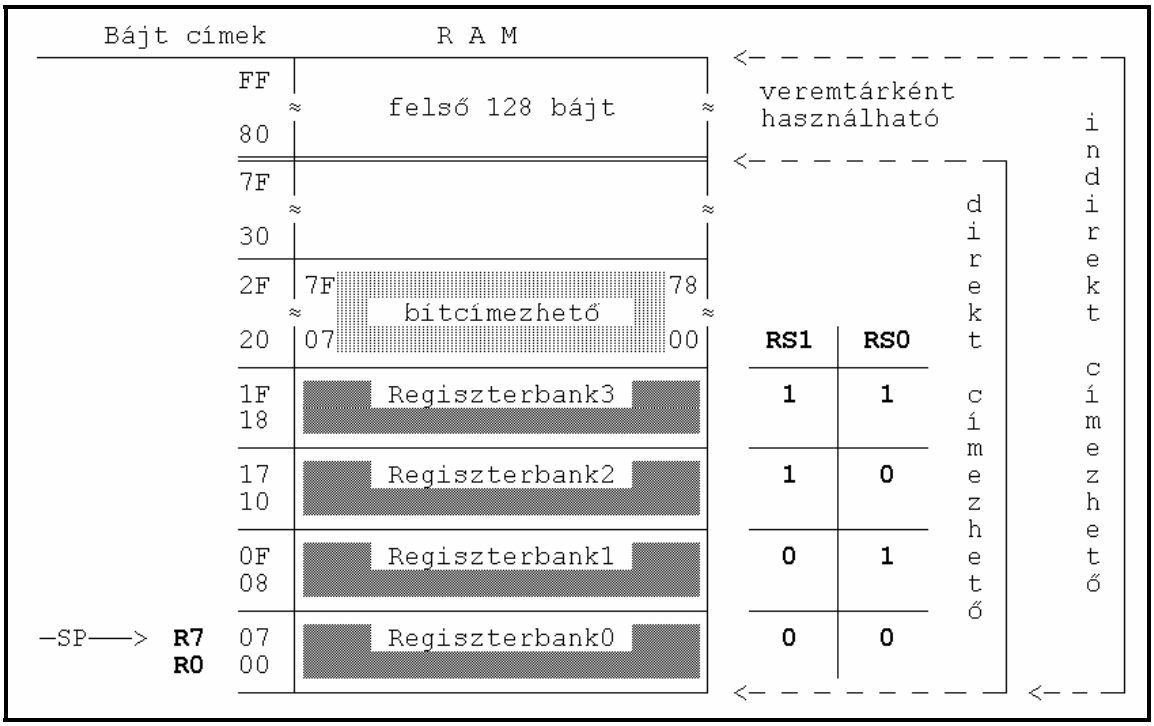
3.7.3. Regiszterbankok a belső RAM-ban

A belső memóriacímek tartalmának verembe mentése mellett lehetséges egyes fontos adatok vagy címek megőrzése az R0-R7 regiszterekben azok áthelyezésével egy másik memóriatartományba. A processzor négy különböző 8 bájtos címtartományt, azaz négy regiszter-bankot használhat erre a célra. A regiszterbankok a 0H címen kezdődnek. A nyolc általános célú regiszter (R0-R7) növekvő sorrendben foglalja el a rendelkezésre álló nyolc címet egy regiszterbankon belül.
A PSW-ben található RS1 és RS0 bitek határozzák meg, hogy az R0-R7 regisztereket a négy regiszterbank közül melyikben használjuk. Az átkapcsolás után a régi regiszterbank tartalma nem változik. RESET után a 0-ás regiszterbank lesz aktív és a veremmutató a 07H címre mutat. Regiszterbank átkapcsolás után az R0-R7 regiszterek szimbolikus nevei megmaradnak, de a címük megváltozik.

PSW				**RS1**	**RS0**				(D0H)
				0	0	Regiszterbank 0			
				0	1	Regiszterbank 1			
				1	0	Regiszterbank 2			
				1	1	Regiszterbank 3			

Az RS0 , RS1 kapcsolóbitek az SETB ill. CLR utasításokkal írhatók ill. törölhetők.

A belső RAM és a négy regiszterbank memóriatérképe :



Mikor kell a regiszterbankokat átkapcsolni?

-> Ha a programunk csak a főprogramból és néhány szubrutinból áll (és nem használunk megszakítást), használjunk mindegyikben külön regiszterbankot (ha szükséges és lehetséges).

-> Ha a program néhány megszakítás kiszolgálást is tartalmaz, a főprogram és a szubrutinok használják az első (0-ás) regiszterbankot, a megszakítások pedig a következőket csökkenő prioritási sorrendben. Megszakítás kiszolgálás után azt a regiszterbankot kell visszaállítani, amelyet eredetileg használtunk, ezért a kiszolgálórutin elején el kell mentenünk, a végén vissza kell állítanunk az átkapcsolóbiteket (is) tartalmazó PSW-t (PUSH PSW, POP PSW).

3.8. Megszakítás-kiszolgálás

3.8.1. Bevezetés

A **megszakítás** (interrupt) egy program **végrehajtásának felfüggesztése** egy előre nem látott időpontban. A ***megszakításkérés*** olyan eseményt jelez, amely **elsőbbségi** kezelést kíván.

Példa : Mosógépvezérlés

Ha a fedelet felnyitják (a megszakításkérés aszinkron a futó programhoz képest), a motort azonnal le kell állítani és a vízbevezető szelepet el kell zárni (megszakítás kiszolgálás).

További példák :

-> Adatok mentése áramszünet esetén.
-> Egy gép vészleállítása.
-> A számológép billentyűzete adatbevitelt jelez.
-> Mérési adatok átvitele egy mérőműszerből egy mikroszámítógépbe.

Megkülönböztetünk maszkolható (programból tiltható) és nem maszkolható(NMI, programból nem tiltható) megszakításokat. A 80515-ös csak maszkolható megszakításokat használ.
A mikrokontroller 14 megszakításforrás kiszolgálására képes. Ebből 7 valamilyen külső esemény hatására következik be, amit a mikrokontroller meghatározott bemenőpontjain megjelenő aktív szint vagy él jelez. Belső megszakításkérést a beépített perifériák (A/D átalakító, timerek, soros csatorna) adott állapotának elérése válthat ki. A megszakításforrásokhoz négy prioritási szint rendelhető.
Megszakításkérés érzékelése után az aktuális utasítás végrehajtását még befejezi a processzor, és csak ezután függeszti fel a programvégrehajtást. A következő végrehajtandó utasítás címét (visszatérési cím) a mikrokontroller a verembe menti (ahogy a szubrutinhívásnál), ezután az utasításszámláló a beérkezett megszakításkéréstől függő címre mutat és a következő utasítás lehívása innen történik. Ezzel elindult egy megszakítás-kiszolgáló rutin, melynek végét az RETI utasítás jelzi.

Hatására a visszatérési cím a veremből a PC-be töltődik (mint a szubrutinból való visszatérésnél), és a felfüggesztett program végrehajtása a következő utasítással folytatódik. Ha egyidejűleg több megszakításkérés érkezik, a mikrokontroller **prioritásuk** alapján dönti el, hogy melyiket kell először kiszolgálni.
A különböző megszakítások kiszolgálásának belépési címei a 80515-ös mikrokontrollerben rögzítettek. (Táblázat a következő oldalon). Mivel a belépési címek közötti távolság a programok számára túl kicsi, minden megszakítás-kiszolgálás általában a programmemóriában elhelyezett tényleges kiszolgálóprogram kezdőcímére való feltétel nélküli ugrással kezdődik.

Programozási eljárás :
A megszakítás-kiszolgáló programok a szubrutinokkal megegyező módon írhatók, de RET helyett a RETI utasítással végződnek. Ha egy megszakítás kiszolgálása alatt egy nála magasabb prioritású engedélyezett megszakításkérés érkezik, annak kiszolgálását azonnal el kell kezdeni. A processzornak tudnia kell, milyen prioritású megszakítás mikor fejeződik be, ezért egy adott megszakítás végén eltérően a szubrutintól a RETI utasítást használjuk. Megszakítás-kiszolgálás esetén is el kell menteni minden olyan regisztertartalmat, amit a kiszolgálóprogram megváltoztat és amelyek eredeti tartalmára a megszakított programban szükség van. Általában át kell kapcsolni a regiszterbankokat is (ha a kiszolgálóprogram igényli az R0-R7 általános célú regiszterek használatát). A legmagasabb prioritású megszakításhoz a 1-es, a legalacsonyabb prioritásúhoz a 3-as regiszterbankot rendeljük, a 0-ás regiszterbankot pedig megosztva használhatja a főprogram és a szubrutinok. A különböző szintű megszakítások kiszolgálásának befejezésekor csak akkor tudjuk, hogy beérkezésük előtt melyik regiszterbankot használtuk, ha a kiszolgálórutin elején a PSW-t elmentettük, a végén pedig visszaállítjuk (PUSH PSW, POP PSW).

Megszakítás-kiszolgálási címek :

Cím	Sorsz.	Jelző-bit	Csatlakozó	Név	A kiváltás módja
0003H	0	**IE0**	**P3.2**	**$\overline{\text{INT0}}$**	**beállítható : -> logikai 0, szintvezérelt -> lefutóéllel vezérelt**
000BH	1	TF0		Timer0	Túlcsordulás
0013H	2	**IE1**	**P3.3**	**$\overline{\text{INT1}}$**	**beállítható : -> logikai 0, szintvezérelt -> lefutóéllel vezérelt**
001BH	3	TF1		Timer1	Túlcsordulás
0023H	4	RI+TI		soros csatorna	Vételi puffer megtelt, vagy adáspuffer üres.
002BH	5	TF2+**EXF2**	**P1.5 $\overline{\text{T2EX}}$**	Timer2	Túlcsordulás, vagy **externer reload trigger.**
0033H	6				nem használt
003BH	7				nem használt
0043H	8	IADC		ADW	A/D átalakítás befejezése.
004BH	9	**IEX2**	**P1.4**	**$\overline{\text{INT2}}$**	**beállítható : -> felfutóéllel vezérelt -> lefutóéllel vezérelt**
0053H	10	**IEX3**	**P1.0**	**$\overline{\text{INT3}}$**	**beállítható : -> felfutóéllel vezérelt -> lefutóéllel vezérelt**
005BH	11	**IEX4**	**P1.1**	**INT4**	**felfutóéllel vezérelt**
0063H	12	**IEX5**	**P1.2**	**INT5**	**felfutóéllel vezérelt**
006BH	13	**IEX6**	**P1.3**	**INT6**	**felfutóéllel vezérelt**

xxxx ill. a **félkövér** szedés: a külső megszakításokat jelzi.
(ld. többszörös portkihasználás.)

Problémaorientált nyelvekben az egyes megszakításokat a sorszámukkal jelölik (assembly programban nem!). A címszámítás módja : cím = Sorsz. x 8 + 3

Az egyes megszakításkérések jelzőbitjeit a processzor minden egyes gépi ciklus S5P2 fázisában lekérdezi (ld. 1.4.1). Egy megszakításkérést csak akkor tekint érvényesnek, ha még a következő gépi ciklusban is fennáll. A szintvezérelt megszakításkérések aktív szintjének illetve az élvezérelt megszakítások aktív élét követő szintnek ezért legalább egy gépi cikluson keresztül fenn kell maradnia. Ahhoz azonban, hogy a hardver a következő két gépi ciklusban egy LCALL hívást generáljon a megszakításvektorra (belépési címre), még további három feltételnek kell teljesülnie:

- nem folyik azonos vagy magasabb szintű megszakítás kiszolgálása.
- az adott gépi ciklusban az aktuális utasítás feldolgozása éppen befejeződik.
- nincs folyamatban egy RETI utasítás feldolgozása, vagy az IEN0, IEN1, IEN2, IP0,IP1 regiszterek bármelyikének írása.

A külső és belső megszakítások prioritása :

Több megszakításkérés egyidejűsége esetén kiszolgálásuk prioritásuk sorrendjében történik. A 80515-ös a megszakításokat párokba rendezi. Páron belül az első megszakításforrásnak elsőbbsége van a másodikkal szemben és az adott párosnak is az utána következőkkel szemben. RESET után valamennyi megszakításpárhoz a 0-ás prioritási szint tartozik (alapbeállítás).

Megszakításpáron belüli prioritás		**Megszakítás-párok közötti prioritás**
magasabb	alacsonyabb	
$\overline{\text{INT0}}$ (IE0)	ADW (IADC)	magasabb
Timer0 (TF0)	$\overline{\text{INT2}}$ (IEX2)	
$\overline{\text{INT1}}$ (IE1)	$\overline{\text{INT3}}$ (IEX3)	
Timer1 (TF1)	INT4 (IEX4)	
soros csat. (RI+TI)	INT5 (IEX5)	
Timer2 (TF2+EXF2)	INT6 (IEX6)	alacsonyabb

Az egyes megszakításpárokhoz az **IP0**-**IP1** (**I**nterrupt-**P**riority **0** és **1**) regiszterek beállításával tudunk prioritási szinteket rendelni, és ezzel a fenti, hardver által meghatározott megszakítás kiszolgálási sorrendet megváltoztatni. **Ezek a regiszterek *nem* bitcímezhetők.**

Megszakításpárok		**Prioritásbitek az IP0,IP1 regiszterekben** IP1	IP0
$\overline{\text{INT0}}$ (IE0)	ADW (IADC)	IP1.0	IP0.0
Timer0 (TF0)	$\overline{\text{INT2}}$ (IEX2)	IP1.1	IP0.1
$\overline{\text{INT1}}$ (IE1)	$\overline{\text{INT3}}$ (IEX3)	IP1.2	IP0.2
Timer1 (TF1)	INT4 (IEX4)	IP1.3	IP0.3
soros csat. (RI+TI)	INT5 (IEX5)	IP1.4	IP0.4
Timer2 (TF2+EXF2)	INT6 (IEX6)	IP1.5	IP0.5
Prioritási szintek			
0 Prioritási szint		0	0
1 Prioritási szint		0	1
2 Prioritási szint		1	0
3 Prioritási szint		1	1

→ Az IP1.6/0.6 ill. IP1.7/0.7 biteknek nincs hatásuk a prioritási szintekre.

→ Magasabb szintű megszakításkérések az alacsonyabb szintűek kiszolgálását megszakíthatják.

→ Alacsonyabb szintű megszakításkérések a magasabb szintűek kiszolgálását nem szakíthatják meg.

→ Egyidejűleg fellépő azonos szintű megszakításkérések egymás kiszolgálását nem szakíthatják meg.

→ Egyidejűleg fellépő azonos szintű megszakításkérések kiszolgálása a fenti, hardver által meghatározott sorrendben történik.

3.8.2. Az $\overline{\text{INT0}}$ - INT6 külső megszakítások

A processzornak **hét** külső megszakítás-bemenete van. Erre a P1 és P3 portok egyes bitjeit használhatjuk (ld. 1.2, többszörös portkihasználás), ezért ezeket a portbiteket bemenetként kell konfigurálni.

					P1.3	P1.2	P1.1	P1.0	
P1 :	T2	clkout	T2EX	**$\overline{\text{INT2}}$**	**INT6** CC3	**INT5** CC2	**$\overline{\text{INT4}}$** CC1	**$\overline{\text{INT3}}$** CCO	(90H)
P3 :	$\overline{\text{RD}}$	$\overline{\text{WR}}$	T1	T0	**$\overline{\text{INT1}}$**	**$\overline{\text{INT0}}$**	TxD	RxD	(B0H)

Az oktatórendszerben a megszakítások a P1.0-P1.3 bitekre kötött kapcsolókkal is kiválthatók:

$\overline{\text{INT3}}$:(P1.0) $\overline{\text{INT4}}$:(P1.2) INT5:(P1.1) INT6:(P1.3)

Forrás	Jelzőbit	Kiváltás módja :	**Belépési cím**
$\overline{\text{INT0}}$ (P3.2) **$\overline{\text{INT1}}$** (P3.3)	IE0 IE1	beállítható : logikai 0, szintvezérelt vagy lefutóéllel vezérelt	0003H 0013H
$\overline{\text{INT2}}$ (P1.4) **$\overline{\text{INT3}}$** (P1.0)	IEX2 IEX3	beállítható : felfutóéllel vezérelt vagy lefutóéllel vezérelt	004BH 0053H
INT4 (P1.1) **INT5** (P1.2) **INT6** (P1.3)	IEX4 IEX5 IEX6	felfutóéllel vezérelt felfutóéllel vezérelt felfutóéllel vezérelt	005BH 0063H 006BH

A külső megszakítások kiváltási módjának beállítása :

Az $\overline{\text{INT0}}$, $\overline{\text{INT1}}$, $\overline{\text{INT2}}$ és $\overline{\text{INT3}}$ megszakítások szintvezérelt ill. élvezérelt kiváltási módját a **TCON** (**T**imer **CON**trol) és **T2CON** (**T**imer**2** **CON**trol) regiszterekkel állíthatjuk be.

$\overline{\text{INT0}}$ és $\overline{\text{INT1}}$ megszakítások :

TCON :	TF1	TR1	TF0	TR0	IE1	**IT1**	IE0	**IT0**	(88H)

Megszakítás lefutóél hatására : 0
Megszakítás logikai 0 hatására : 1

IT0 : **I**nterrupt **T**imer**0**
IT1 : **I**nterrupt **T**imer**1**

$\overline{\text{INT2}}$ és $\overline{\text{INT3}}$ megszakítások :

T2CON :	T2PS	**I3FR**	**I2FR**	T2R1	T2R0	T2CM	T2I1	T2I0	(C8H)

0 : Megszakítás lefutóél hatására
1 : Megszakítás felfutóél hatására

I2FR : **I**nterrupt **2** **F**alling **R**ising
I3FR : **I**nterrupt **3** **F**alling **R**ising

A külső megszakítások engedélyezése (maszkolás) :

A $\overline{\text{RESET}}$ minden megszakítást letilt. Az **IEN0** és **IEN1** (**I**nterrupt **EN**able **0** ill. **1**) regiszterekkel lehet a külső megszakításokat egyrészt globálisan (EAL), másrészt egyenként (EX0-EX6) engedélyezni.

IEN0 :	**EAL**	WDT	ET2	ES	ET1	**EX1**	ET0	**EX0**	(A8H)
IEN1 :	EXEN2	SWDT	**EX6**	**EX5**	**EX4**	**EX3**	**EX2**	EADC	(B8H)

$EX_{0/6}$ = 1 : Az $INT_{0/6}$ megszakítások egyenkénti engedélyezése (**EX**ternal Interrupt **0/6**)

EAL = 1 : Az összes külső és belső megszakítás globális engedélyezése:
(**E**nable **AL**L). Az egyes megszakítások egyenkénti engedélyezésének már ezelőtt meg kell történnie. Azonos prioritású kiszolgálások egymást nem szakíthatják meg.

Példa :

```
        MOV  IEN0,#0H  ; Tilt minden külső megszakítást.
        SETB EX1       ; Engedélyezi az INT1 megszakítást.
        SETB EAL       ; Globális engedély (de csak az INT1 bit aktív).
                       ; Fenti 3 sorral egyenértékű : MOV   IEN0,#84H
```

Fennálló külső megszakításkérések jelzése:

A fennálló megszakításkéréseket a **TCON** (**T**imer **CON**trol) és **IRCON** (**I**nterrupt **R**equest **CON**trol) regiszterekben az IE0 és IE1, ill. IEX2-IEX6 bitek logikai 1-es szinttel jelzik. ***Élvezérelt*** megszakítások jelzőbitjei a kiszolgálásuk után visszabillennek. A ***szintvezérelt*** megszakítások jelzőbitjei addig maradnak logikai 1-es szinten, amíg a megszakításkérés fennáll.

Egy adott jelzőbithez tartozó kiszolgálóprogram akkor is elindul, ha a bitet programból billentettük be (**szoftver-interrupt**).

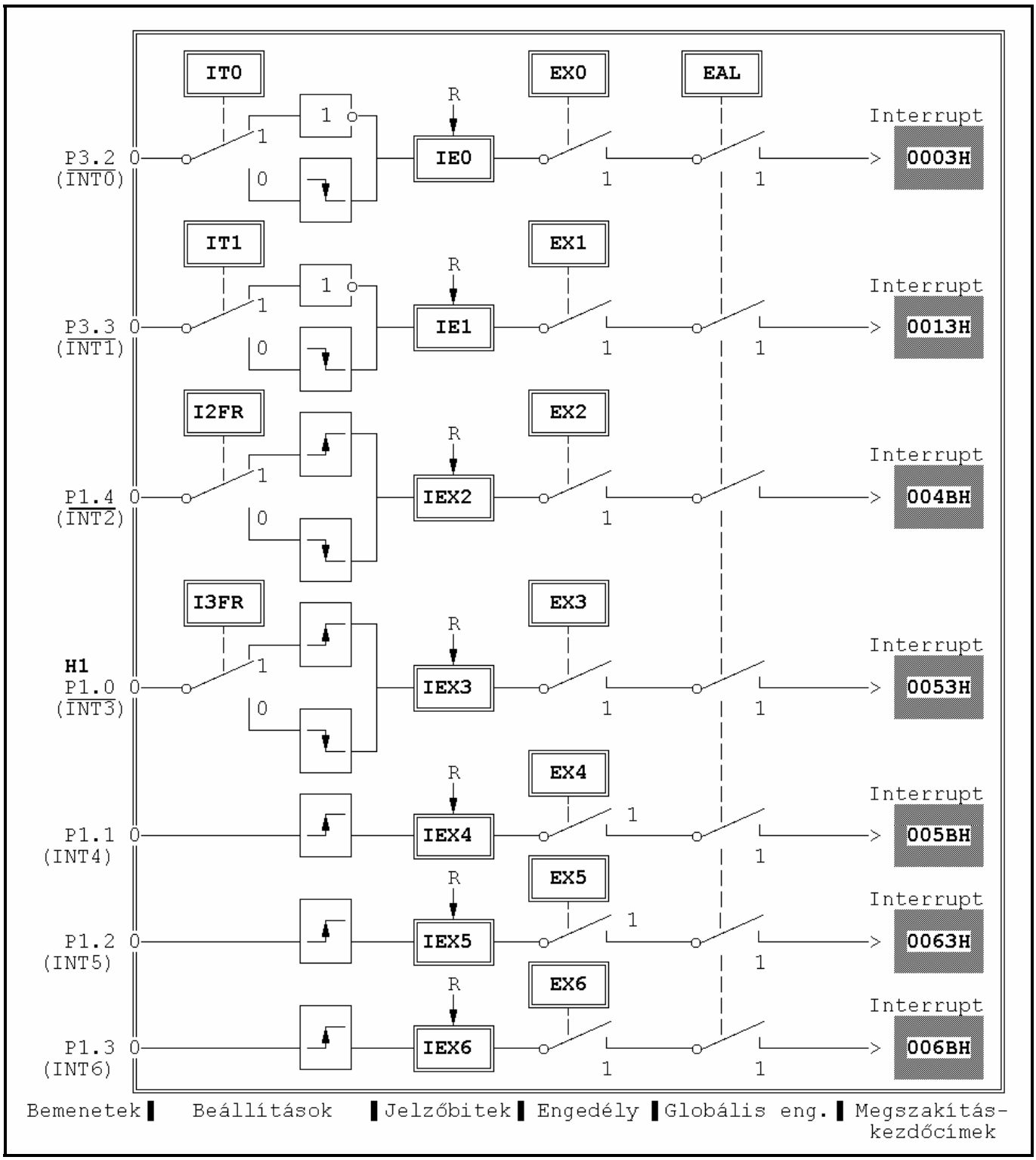
Az $\overline{\text{INT0}}$ és $\overline{\text{INT1}}$ jelzőbitjei :

TCON :	TF1	TR1	TF0	TR0	**IE1**	IT1	**IE0**	IT0	(88H)

Az $\overline{\text{INT2}}$ - INT6 jelzőbitjei:

IRCON :	EXF2	TF2	**IEX6**	**IEX5**	**IEX4**	**IEX3**	**IEX2**	IADC	(C0H)

A külső megszakítások összefoglalása :



1. példa: külső megszakításokat (INT3,INT4) kezelő program fejrésze.

```
         ORG      53H       ; Az INT3 belépési címe.
         LJMP     EX3SERV   ; A végrehajtás átirányítása.
         ORG      5BH       ; Az INT4 belépési címe.
         LJMP     EX4SERV   ; A végrehajtás átirányítása.
         ORG      200H      ; Főprogram.
         MOV      P1,#0FFH  ; P1.0 - P1.7 bemenet.
         SETB     EX3       ; Az INT3 engedélyezése.
         SETB     EX4       ; Az INT4 engedélyezése.
         SETB     I3FR      ; Az INT3 felfutóéllel vezérelt.
         SETB     EAL       ; Globális engedély.
         ----------
EX3SERV: ----------
         RETI
EX4SERV: ----------
         RETI
         END
```

2. példa :

A P1.0 bemeneten érkező $\overline{\text{INT3}}$ megszakítás komplementálja a P4 portot, a P1.1 bemeneten érkező INT4 pedig a P5-öt. A P1.2 kapcsolóval a monitorprogramra ugrunk.

```
MONITOR  EQU      8000H
         ORG      53H           ; INT3 belépési címe.
         LJMP     EX3SERV       ; Átirányítás.
         ORG      5BH           ; INT4 belépési címe.
         LJMP     EX4SERV       ; Átirányítás.
         ORG      0             ; RESET belépési címe.
         LJMP     START         ; Átirányítás.
         ORG      200H          ; Főprogram.
START:   MOV      P1,#0FFH      ; P1 minden bitje bemenet.
         SETB     EX3           ; Az INT3 engedélyezése.
         SETB     I3FR          ; Felfutóél vezérelt.
         SETB     EX4           ; Az INT4 engedélyezése
         SETB     EAL           ; Globális engedély.
HUROK:   JB       P1.2,HUROK    ; Kapcsolóra vár.
         LJMP     MONITOR       ; Ugrás a monitorprogramra.
EX3SERV: MOV      A,P4          ; P4
         CPL      A             ; komplementálása.
         MOV      P4,A
         RETI                   ; Engedélyezi az azonos vagy alacsonyabb
                                ; prioritású megszakításokat.
EX4SERV: MOV      A,P5          ; P5
         CPL      A             ; komplementálása.
         MOV      P5,A
         RETI
         END
```

3.8.3. Belső megszakítások

Belső megszakításkérést vált ki, ha valamelyik, a mikrokontrolleren belüli építőelem műveletei egy meghatározott esemény bekövetkeztét eredményezik:

→ Ha egy A/D átalakítás befejeződött.
→ Ha a három számláló egyike (Timer0 - Timer2) túlcsordult.
→ Ha a soros csatorna valamelyik állapotjelzője (RI, TI) bebillent.

Forrás	Jelzőbit	**Kiváltás oka :**	**Kezdőcím**
A/D átalakító	IADC	Az átalakítás befejeződött	0043H
Timer0	TF0	Túlcsordulás.	000BH
Timer1	TF1	Túlcsordulás.	001BH
soros csatorna	RI+TI	vételi puffer megtelt, vagy adási puffer üres	0023H
Timer2	TF2+EXF2	Túlcsordulás.	002BH

(ld. 3.9 : A számlálók, 3.10 : Az analóg-digitál átalakító, 3.11 : Soros adatátvitel)

3.9. A számlálók (timerek)

A mikrokontroller számlálói egy előzőleg betöltött kezdőértéktől a számlánc végértékéig számolnak, elérésekor túlcsordulásjelzést adnak, amit pl. egy megszakítás-kiszolgáló rutinnal feldolgozhatunk. A timerek a mikrokontroller egyéb műveleteitől függetlenül dolgoznak, de előzetesen beállítást igényelnek.

A számlálókat felhasználhatjuk :

--> **Időadónak**, amikor egy *belső* állandó frekvenciájú ütemjelet számolunk.
--> **Eseményszámlálónak**, ha egy *külső* (változó) ütemmel vezéreljük.
--> **Időintervallum-mérőnek**, amikor egy belső ütemjel számolását a P3.2 ill. P3.3 portbitekre vezetett külső jellel kapuzzuk.

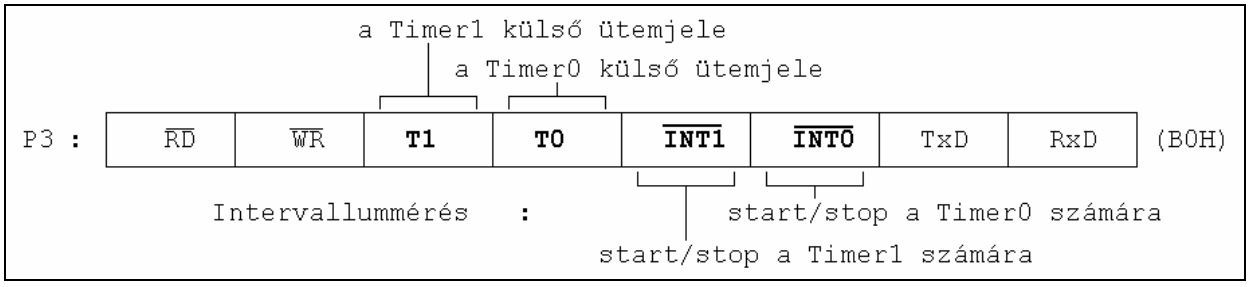
A számlálókat ***láncba*** is programozhatjuk; az így kapott intervallummérőt használhatjuk pl. a soros adatátvitel ***baud-rate generátorának*** ütemezésére. Megfelelő feltételek betartásával felhasználhatjuk őket impulzusszélesség modulációra, vagyis egyfajta ***digitál-analóg*** átalakításra.

3.9.1. A Timer0 és a Timer1

A Timer0 és a Timer1 16 bites előreszámlálók, amelyek két 8 bites regiszterből állnak:

Timer0 : TH0, TL0 (**T**imer **H**igh/**L**ow **0**)
Timer1 : TH1, TL1 (**T**imer **H**igh/**L**ow **1**)

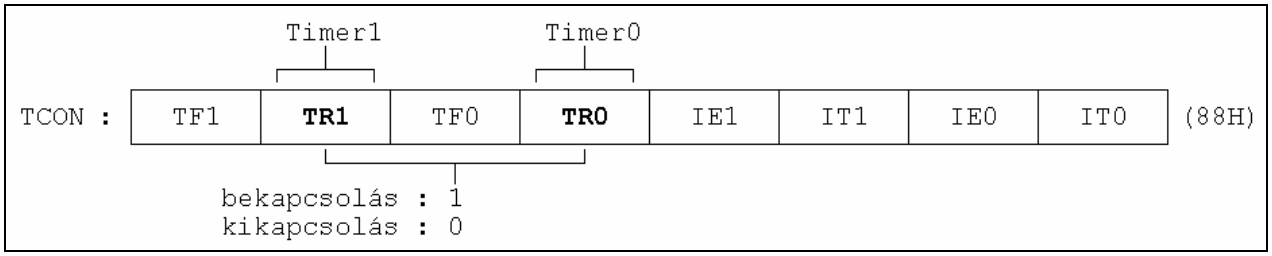
Felhasználhatjuk őket **időadónak** (belső ütemjel számolása: $f_{osz/12}$ = 1 MHz), **eseményszámlálónak** (külső események ***lefutóéleinek*** számlálása: maximum $f_{osz}/24$-ig, ami itt 500 kHz) és **időintervallum-mérőnek**. A Timer0 külső ütemjelét a P3.4 portbitre (T0), a Timer1-ét pedig a P3.5-re (T1) kell kötni. Intervallummérés estén a Timer0 startját és stopját a P3.2 portbit ($\overline{INT0}$), a Timer1-ét a P3.3 ($\overline{INT1}$) vezérli.



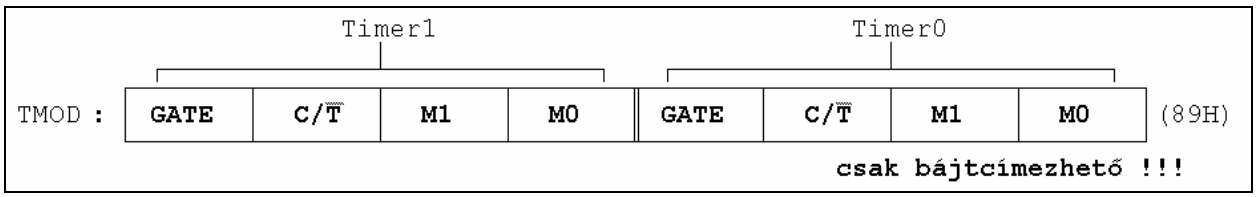
A **túlcsordulás megszakítást** vált ki, ha az ET0 ill. ET1 (**E**nable **T**imer**0/1**) engedélybit és az IEN0 regiszter globális engedélybitje (EAL, **E**nable **AL**L) is 1-es.

A Timer0 és a Timer1 be- és kikapcsolása:

A timereket a **TCON** (**T**imer **CON**trol) regiszter **TR0**-**TR1** (**T**imer **R**un **0/1**) bitjeivel kapcsolhatjuk be ill. ki.



A **TMOD** (**T**imer/counter **MOD**e) regiszter **C/$\overline{T}$** (**C**ounter/**T**imer) vezérlőbitjeivel választhatunk a timerek belső időadó vagy külső eseményszámláló funkciói közül. A **GATE** bit teszi lehetővé a timerek ki- és bekapcsolását egy külső jellel (intervallummérés; Timer0: P3.2; Timer1: P3.3). Az **M1** és **M0** bitek állítják be timerek üzemmódját.



GATE Az intervallummérés be-/kikapcsolása külső jellel:
0 : ***kikapcsolás***
1 : ***bekapcsolás***
a bemenetek : INTO (P3.2); Timer0
$\overline{INT1}$ (P3.3); Timer1

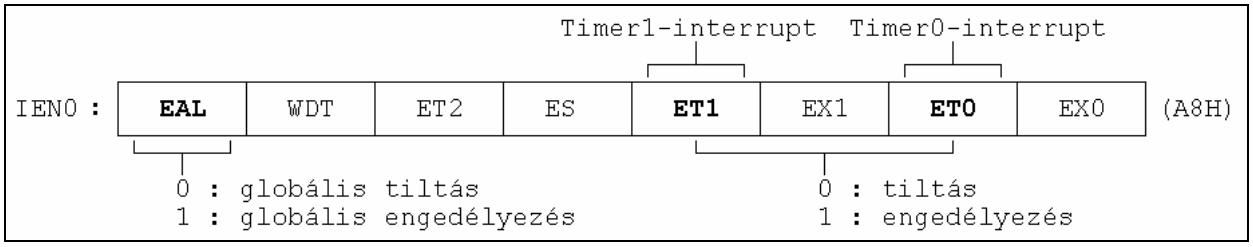
C/$\overline{T}$ időadó/eseményszámláló átkapcsolóbit:
0 : ***Időadó***; belső ütemjelet számol ($f_{OSZ}/12$ = 1 MHz)
1 : ***Eseményszámláló***; külső események *lefutóéleit* számolja ($\leq f_{OSZ}/24$ = 500 KHz)
Bemenetek :
T0 (P3.4); Timer0
T1 (P3.5); Timer1
A külső ütemjel maximális frekvenciája $f_{OSZ}/24$-nél nagyobb nem lehet, mert a bemenet logikai szintjét a processzor minden gépi ciklus S5P2 fázisában kérdezi le, ezért a lefutóél felismeréséhez minimum 2 gépi ciklus szükséges.

A Timer0 és a Timer1 üzemmódválasztása:

M1	M0	Üzemmód	
0	0	0	8 bites **időadó/eseményszámláló** a TH0/1-ben; 5 bites előosztó a TL0/1-ben.
0	1	1	16 bites **időadó/eseményszámláló.**
1	0	2	Saját magát újratöltő 8 bites **időadó/eseményszámláló** (auto reload)
1	1	3	Timer1 **:** kikapcsolva Timer0 **:** TL0 mint 8 bites **időadó/eseményszámláló** a Timer0 kontrollbitekkel vezérelve. TH0 mint 8 bites **időadó,** a Timer1 kontrollbitekkel vezérelve.

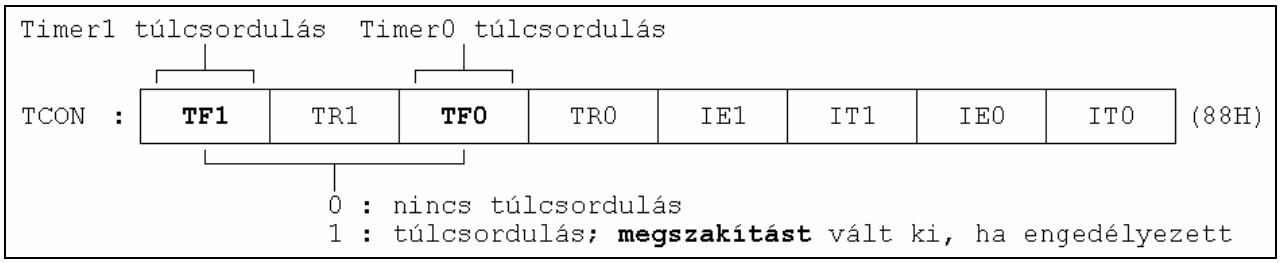
A Timer0 és Timer1 megszakítások engedélyezése :

A $\overline{RESET}$ minden megszakítást letilt. Az **IEN0** (**I**nterrupt **EN**able **0**) regiszter **ET0** és **ET1** (**E**nable **T**imer**0/1**) bitjeivel lehet a Timer0 és Timer1 megszakításokat egyenként, az EAL bittel pedig globálisan (az $\overline{INT0}$-INT6-tal együtt) engedélyezni.

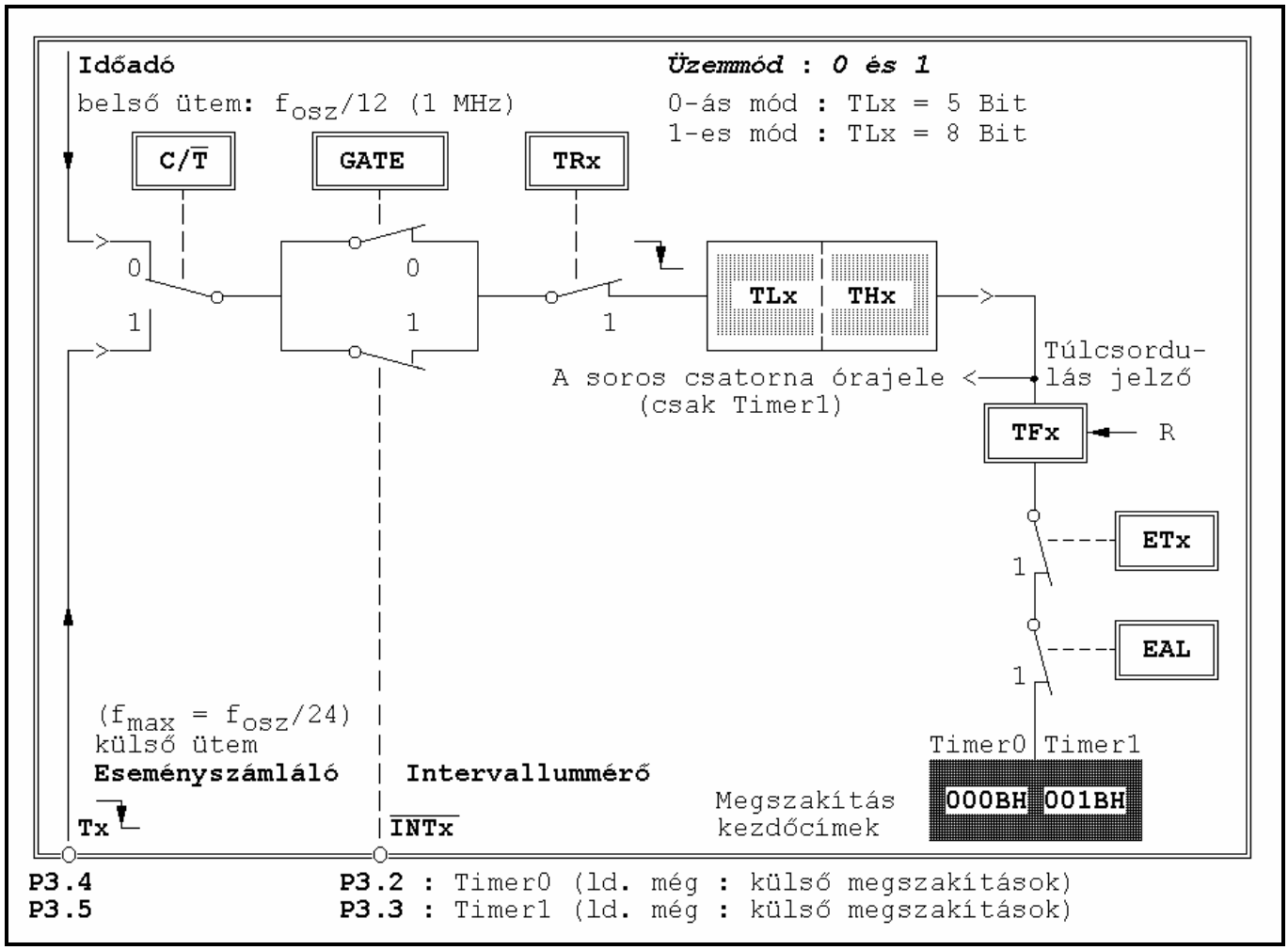


A Timer0 és a Timer1 túlcsordulásának jelzése :

A **TF0** és **TF1** bitek jelzik a timerek túlcsordulását, és megszakítást váltanak ki, ha az engedélyezett. **A megszakítás-kiszolgáló rutin elindulásakor a bitek automatikusan visszabillennek.** Szoftver úton is kiváltható egy megszakítás-kiszolgálás elindulása a hozzátartozó jelzőbit bebillentésével, ha az adott megszakítás egyébként engedélyezett.



A Timer0 és a Timer1 0-ás és 1-es üzemmódjainak összefoglalása:

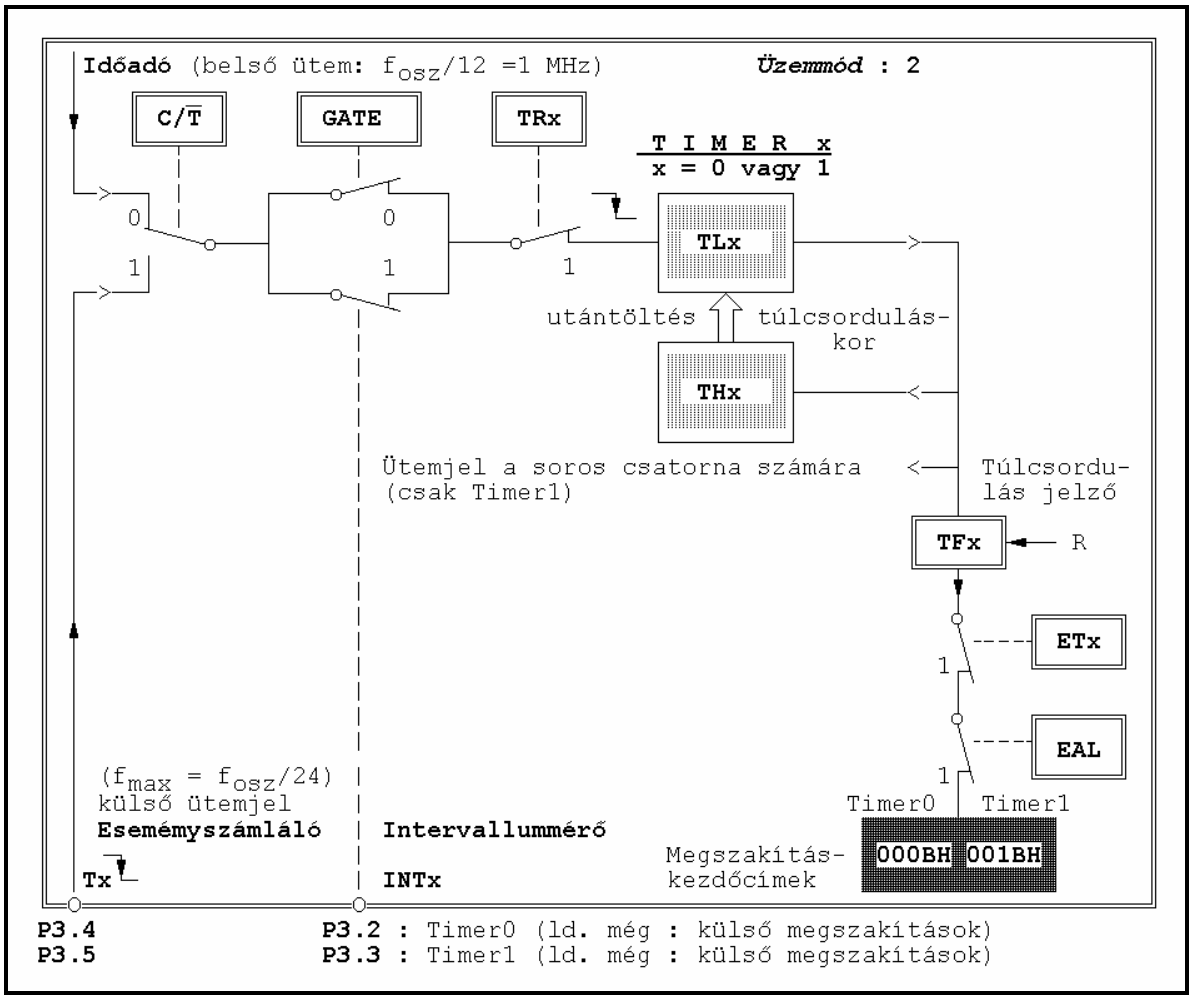


A Timer0 és a Timer1 mint saját magát újratöltő 8 bites számláló (frekvenciagenerátor, auto reload) :

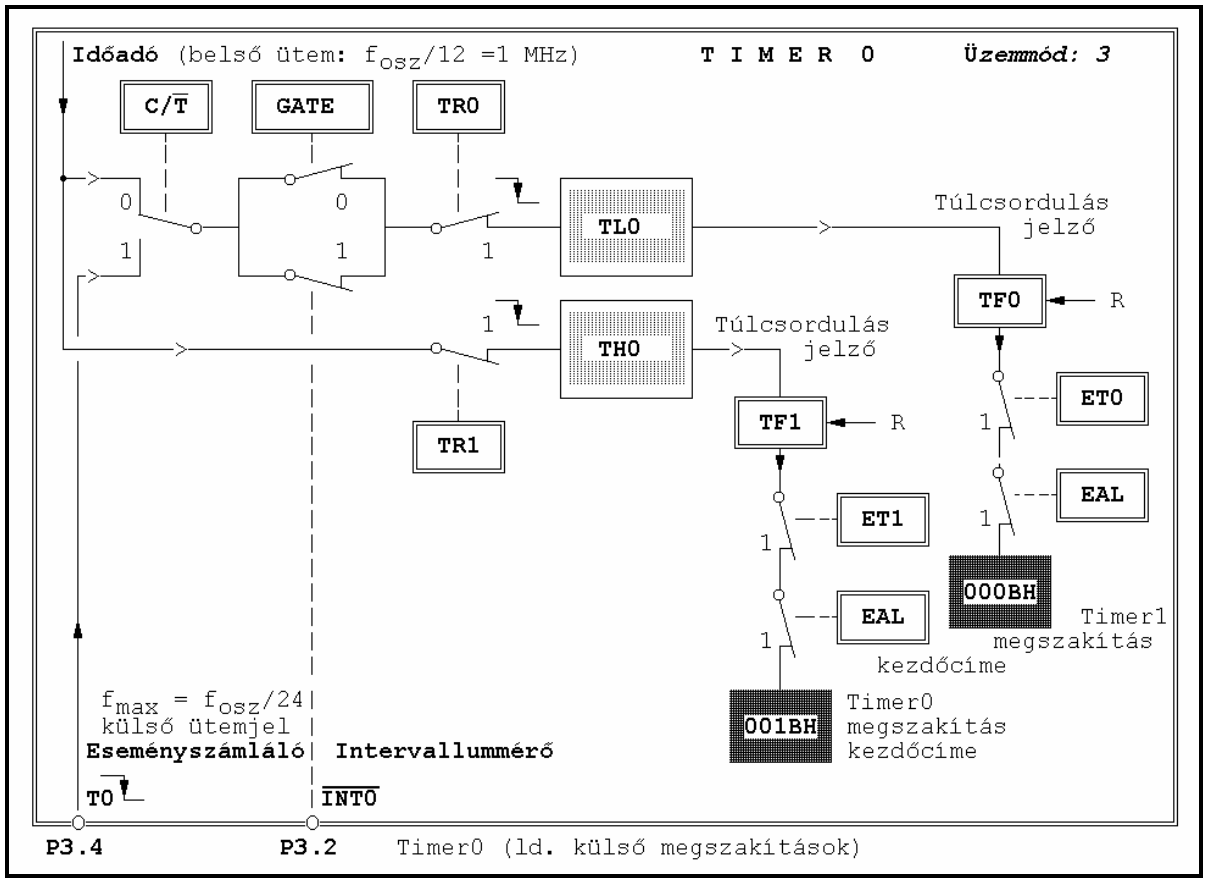
Amikor a TL0/1 255-ből 0-ba fordul (túlcsordul), a TL0/1 átveszi TH0/1-ben tárolt értéket (auto reload). Ezáltal a számlálóciklus hossza 1µs-tól (töltési érték : 255) 256 µs-ig (töltési érték : 0) kvarcpontossággal beállítható.

Ez az azonos időközönként ismétlődő túlcsordulás a 0-ás és 1-es módokhoz hasonlóan felhasználható például a soros adatátvitel ütemezésére (ld. 3.11.2).

A Timer0 és a Timer1 2-es üzemmódjának összefoglalása:

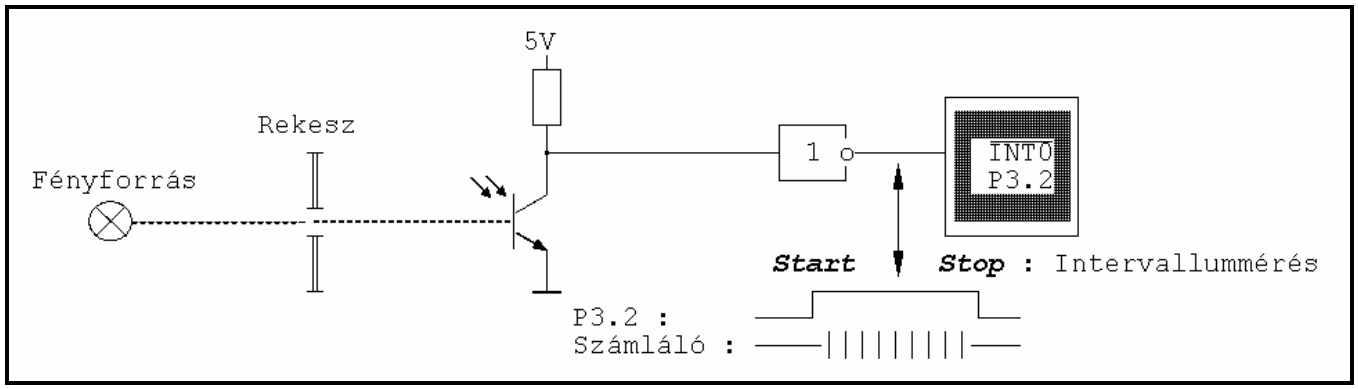


A Timer0 3-as üzemmódjának összefoglalása (Timer1 kikapcsolva) :



1. példa : Időintervallum-mérés; Fényképezőgép rekeszidejének mérése:(16 bites eseményszámláló. Üzemmód 1 :)

A P3.2 (Timer0) bemenet vezérli a számláló startját és stopját. (**Időintervallum-mérés**). Ha hosszabb időt kívánunk mérni, amire a 16 bites számláló nem elegendő, meg kell számolni a túlcsordulások (megszakítás-kiszolgálások) számát is.




```
          ORG  0BH            ; Timer0 megszakítás kezdőcíme.
          LJMP TOINT          ; A kiszolgálás átirányítása.
SZAML     EQU  (0)            ; A túlcsordulások számlálója (≡ R0).
          ORG  200H           ; Főprogram.
          MOV  SZAML,#0       ; Túlcsordulás-számláló nullázása.
          MOV  TH0,#0         ; Timer0 nullázása
          MOV  TL0,#0         ; bájtonként (csak így lehet).
          CLR  TF0            ; Túlcsordulás jelzőbit törlése.
          MOV  TMOD,#9        ; Timer0 : 16 bites kapuzott számláló (GATE=1).
          SETB ET0            ; Timer0 megszakítás engedélyezése.
          SETB TR0            ; A Timer0 bekapcsolása.
          SETB EAL            ; Globális megszakítás-engedély.
START:    JNB  P3.2,START     ; Az intervallummérés start élére vár.
VEGE:     JB   P3.2,VEGE      ; Az intervallummérés stop élére vár.

EREDM:    MOV  P5,SZAML       ; Eredménykijelzés: impulzusszám=SZAML*2^16 + TH0*2^8 + TL0
          MOV  P4,TH0         ; (a TL0 kijelzésére nincs szabad port.)
          LJMP 8000H          ; Ha kész, a monitorprogramra ugrik.
TOINT:    INC  SZAML          ; Túlcsordulások számolása.
          RETI
          END
```

2. példa: A P1.0 portbiten 5 kHz-es, a P1.1-en 6,25 kHz-es 1:1-es kitöltésű impulzussorozat kiadása. Félperiódusok: 5,00 KHz: 100 µs (Timer0); 6,25 KHz: 80 µs (Timer1)

```
          ORG  0BH              ; Timer0 megszakítás kezdőcíme.
          LJMP T0INT
          ORG  1BH              ; Timer1 megszakítás kezdőcíme.
          LJMP T1INT            ;
KI1       EQU  P1.0             ; 5 kHz-es kimenet
KI2       EQU  P1.1             ; 6,25 kHz-es kimenet
          ORG  200H             ; Főprogram.
          MOV  TH0,#(256-100)   ; Utántöltési értékek.
          MOV  TH1,#(256- 80)
          MOV  TL0,#(256-100)   ; A TLx regiszterek kezdőértéke megegyezik az után-
          MOV  TL1,#(256- 80)   ; töltésivel: az első ciklus azonos a továbbiakkal.
          MOV  TMOD,#00100010B  ; Időadó, reload mód.
          SETB ET0              ; Timer0 megszakítás engedélyezés.
          SETB ET1              ; Timer1 megszakítás engedélyezés.
          SETB TR0              ; A Timer0 bekapcsolása.
          SETB TR1              ; A Timer1 bekapcsolása.
          SETB EAL              ; Globális megszakítás-engedély.
HUROK:    LJMP HUROK            ; Végtelen ciklus, csak megszakítások futnak.
T0INT:    CPL  KI1              ; Kimenőjel invertálása.
          RETI
T1INT:    CPL  KI2              ; Kimenőjel invertálása.
          RETI
          END
```

Ha a programot elindítjuk, 400 µs-onként a **két megszakításkérés ütközik** (egyszerre lépnek fel).

→ Azonos prioritású megszakításkérések közül az előbb fellépőt kell kiszolgálni. A következőnek meg kell várnia az első kiszolgálórutin befejezését.

→ Példánkban ez azt jelenti, hogy a Timer0 és Timer1 megszakításkérések egyidejűsége esetén a Timer0-át a Timer1 előtt kell kiszolgálni. (ld. 3.8.1). A Timer1 megszakításkérése tárolódik és csak a Timer0 kiszolgálása után indul el.

→ A két Timer megszakítás mindig periodikus marad, függetlenül attól, hogy a hozzájuk tartozó kiszolgálórutinok mikor hajtódnak végre.

→ Ha a Timer megszakítások prioritását az IP0-IP1 regiszterekkel különböző szintűre állítjuk be, a magasabb szintű az alacsonyabb végrehajtását megszakíthatja.

3.9.2. A Timer2

A Timer2 a 16 bites számlálón kívül tartalmaz további 4 db 16 bites regisztert: CC0-CC3 compare-capture regiszterek. Ezeket két olyan üzemmódban használja, amelyek a Timer0 és a Timer1 esetében nem léteztek:

Capture (befogás) **módban** a programozott CCx regiszter átveszi a Timer2 16 bites tartalmát belső (szoftver), vagy külső (hardver) vezérlés hatására. A külső vezérlések (P1.0-P1.3) megszakításkérést is generálnak (IEX3-IEX6).

Compare (összehasonlítás) **módban** kimenőjelet (P1.0-P1.3) és megszakításkérést (IEX3-IEX6) generál, ha a Timer2 és a beprogramozott CCx regiszter 16 bites tartalmai megegyeznek.

Reload módban a Timer2 utántöltése a 16 bites CC0 regiszterből történik, ezért ezt szokás CRC (**C**ompare- **R**eload- **C**apture-) regiszternek is nevezni.

A Timer2 alkalmazásai :

A Timer2 lehetővé teszi a Timer0 és Timer1 esetében megismert felhasználásokon kívül az **időmérést** és a **digitál-analóg átalakítást** (impulzusszélesség moduláció formájában) is.

--> **Időadó** *belső* ütemjellel.
--> **Eseményszámláló** a P1.7-re (T2) kötött *külső* ütemjellel.
--> **Időintervallum-mérő** *belső* ütemjellel, a P1.7 (T2) portbittel kapuzva.
--> **Frekvenciagenerátor** (belső vagy külső reload vezérléssel). Az utántöltési érték 16 bites: CRC (CRLH, CRLL). Ezáltal nagyobb frekvenciatartomány érhető el, mint a Timer0 és Timer1 esetében.
--> **Időmérő** (Capture mód) : a Timer2 tartalma négy 16 bites regiszterben tárolható.
--> **Impulzusszélesség-modulátor** (PWM) : a kitöltési tényező compare módban megváltoztatható.
--> Megszakítás kiváltása túlcsorduláskor (minden üzemmódban).

A P1 port P1.7 (T2) bitje a Timer2 külső ütemjelbemenete, P1.5 (T2EX) bitje pedig a Timer2 utántöltésének külső vezérlőjel-bemenete. Mindkettő lefutóéllel aktív. A P1 port *capture módban* négy külső vezérlőbemenetet tartalmaz : P1.0 (CC0) - P1.3 (CC3). Hatásukra a Timer2 16 bites értékét átveszi a CC0-CC3 regiszterek egyike. *Compare módban* ez a négy bit kimenet, a Timer2 és a megfelelő compare-regiszter tartalmának azonosságát jelzi.

P1 :	**T2**	clkout	**T2EX**	$\overline{\text{INT2}}$	INT6 **CC3**	INT5 **CC2**	$\overline{\text{INT4}}$ **CC1**	$\overline{\text{INT3}}$ **CC0**	(90H)

A Timer2 üzemmódjainak beállítása:

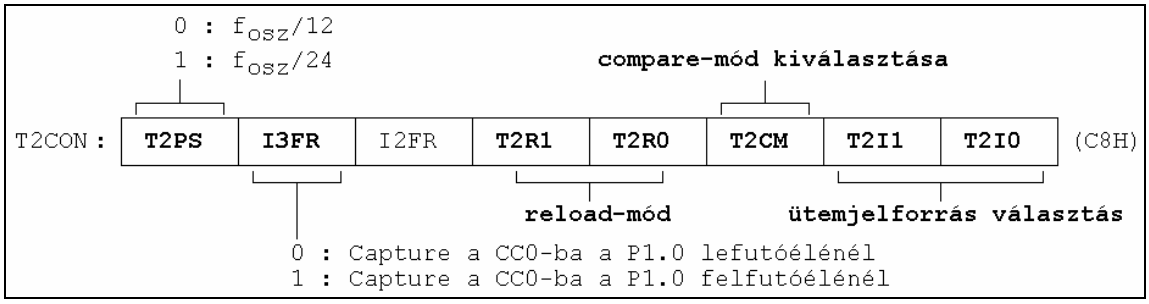
A különböző üzemmódok beállítására a **T2CON** (**T**imer**2** **CON**trol) és a **CCEN** (**C**ompare/**C**apture **EN**able) regiszterek szolgálnak.

A T2CON regiszter:

A **T2PS** (**T**imer**2** **P**re**S**caler) bit határozza meg, hogy a Timer2 belső ütemjele az oszcillátorfrekvencia 1/12 vagy 1/24 része legyen. Ha a Timer2 a P1.7-re (T2) kötött külső ütemjelet számolja, ennek a bitnek nullának kell lennie.

A **T2CM** (**T**imer**2** **C**ompare **M**odus) bit állítja be a kétféle compare módot, a **T2I1** és **T2I0** (**T**imer**2** **I**nput **0/1**) bitekkel választhatunk ütemjel-bemenetet, a **T2R1** és **T2R0** (**T**imer**2** **R**eload **0/1**) vezérli az utántöltést (reload).

Az **I3FR** (**I**nterrupt **3** **F**alling **R**ising; ld. 3.9.2) bittel állítható be hogy a **CC0** (**C**ompare and **C**apture Register **0**) regiszter *capture módban* a P1.0 felfutó- vagy lefutóélére vegye át a Timer2 tartalmát, illetve *compare módban* a P1.0 felfutó- vagy lefutóélével jelezze az azonosságot.



Az ütemjelforrás kiválasztása :

T2I1	T2I0	Jelentése
0	0	Stop, a Timer2 nem fut.
0	1	Belső ütem; időadó; A bemenőfrekvencia $f_{osz}/12$ ill. $f_{osz}/24$ a T2PS-től függően
1	0	Külső ütem a P1.7-en (T2); eseményszámláló
1	1	Intervallummérés a belső ütemjel külső kapuzásával. Start/stop vezérlés a P1.7-tel (T2).

A reload-mód beállításai :

T2R1	T2R0	Jelentése
0	x	Utántöltés kikapcsolva.
1	0	Utántöltés túlcsordulásnál (**belső reload**).
1	1	Utántöltés a P1.5 (T2EX) lefutóélére (**külső reload**).

A compare mód kiválasztása :

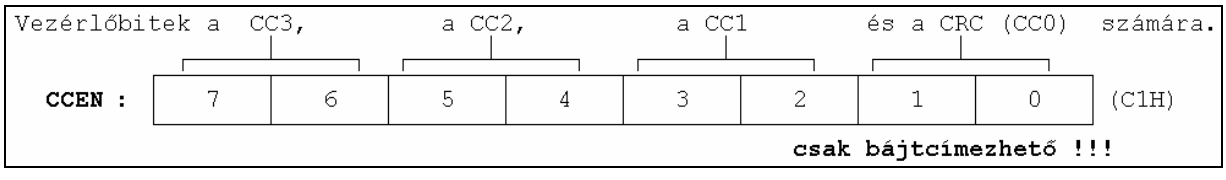
T2CM	0 = 0-ás compare-mód, 1 = 1-es compare-mód

A belső ütemjel frekvenciájának átkapcsolása :

T2PS	0 : ütemf = $f_{osz}/12$ 1 : ütemf = $f_{osz}/24$ (T2PS-nek külső ütemjel esetén 0-nak kell lennie.)

A CCEN regiszter :

A **CCEN** (**C**apture and **C**ompare **EN**able) 2-2 bittel vezérli a CC0/CRC, CC1, CC2 és CC3 regiszterek használatát reload, capture és compare módban.



high bit	low bit	Jelentése	CRC CC0	CC1	CC2	CC3
0	0	**Compare és capture kikapcsolva,** csak reload mód (P1.5 vagy Timer2 túlcsordulás hatására, T2R1-T2R0-tól függően)	CRC	----	----	----
0	1	**Capture felfutóélre** : (vagy lefutóélre, de csak a CC0-ba) :	P1.0 I3FR	P1.1 ----	P1.2 ----	P1.3 ----
1	0	**Compare bekapcsolva.**				
1	1	**Capture a regiszter írásakor** :	CRCL	CCL1	CCL2	CCL3

--> Reload mód a CRC (CRCH, CRCL) regiszterrel:

Reload módban a Timer2 átveszi a **CRC** (**C**ompare **R**eload **C**apture) regiszter tartalmát a **T2CON** (**T**imer**2** **CON**trol) regiszter **T2R1** és **T2R0** (**T**imer**2** **R**eload **1/0**) bitjeitől függően vagy a Timer2 túlcsordulását követően (belső reload), vagy a P1.5 (**T2EX** = **T**imer**2** **EX**tern) portbit lefutóélére (külső reload).

--> Capture mód a CC0 - CC3 regiszterekkel:

Capture módban a Timer2 16 bites tartalma a CC0-CC3 regiszterek egyikébe másolódik :

Külső vezérléssel: a P1.0 portbit a CC0-ba, P1.1 a CC1-be, a P1.2 a CC2-be, a P1.3 pedig a CC3-ba másolást váltja ki (***hardvere capture***). A vezérlés felfutóéllel történik, de a P1.0 az I3FR bittel lefutóélre is átkapcsolható. A bemenetek aktív élének hatására a hozzájuk rendelt IEX3-IEX6 megszakításkérés jelzőbitek bebillennek és elindul a megszakítás-kiszolgálás, ha engedélyezett.
Belső vezérléssel: bármilyen adatmozgató utasítás hatására, amelyben a CC0-CC3 regiszterek valamelyikének alsó bájtja a cél (***szoftver capture***).

Ha a Timer2-t programból a
```
MOV     R1,TH2
MOV     R0,TL2
```
utasításokkal 8 bitenként olvassuk, a TH2 értéke két gépi ciklussal (2 µs) "idősebb" a TL2-nél és ezért hamis. Capture módban a Timer2 16 bites tartalmát egyszerre veszi át valamelyik 16 bites capture regiszter.

Példa ***szoftver capture****-ra*:

```
MOV  CCEN,#00000011B    ; CRC regiszter szoftver capture előkészítése.
stb.
MOV  CRCL,A             ; Bármilyen adatmozgató utasítás hatására, amelynek a
                        ; CCLx regiszter a célja, a CCx regiszter átveszi a
                        ; Timer2 tartalmát.
```

A CCx regiszterek csatlakozókhoz és megszakításokhoz rendelése ***hardver capture*** esetén:

Regiszter	Bemenet	A compare vagy capture által kiváltható külső megszakítás
CC0 ≡ CRC	P1.0	$\overline{\text{INT3}}$ (0053H)
CC1	P1.1	INT4 (005BH)
CC2	P1.2	INT5 (0063H)
CC3	P1.3	INT6 (006BH)

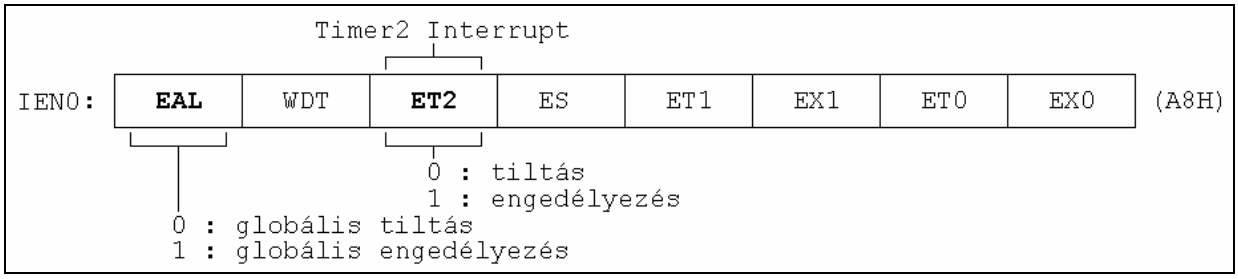
--> Compare mód a CC0 - CC3 regiszterekkel:

Compare módban, ha a Timer2 tartalma megegyezik a CCEN regiszterben kijelölt CCx regiszter tartalmával, a hozzárendelt kimeneten (P1.0-P1.3) kimenőjelet generál. A T2CON regiszter T2CM bitje határozza meg a kimenőjel időbeli lefolyását. Egyezés esetén a CCx regiszterhez rendelt megszakításkérés jelzőbit bebillen, és elindul a megszakítás-kiszolgálás, ha engedélyezett.

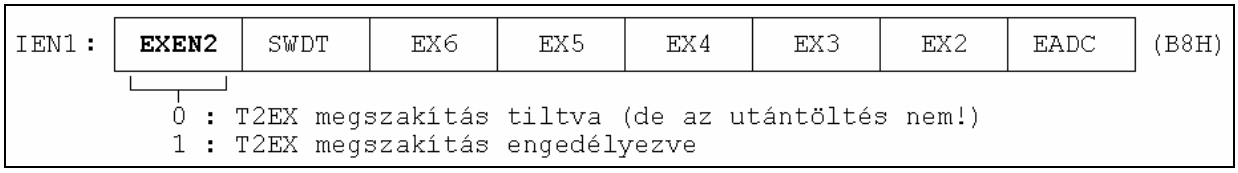
T2CON regiszter: T2CM = 0 : **0-ás compare mód**:
A kimenőjel a Timer2 túlcsordulásakor alacsony szintre, azonosságot adó összehasonlításkor magas szintre billen.
T2CON regiszter: T2CM = 1 : **1-es compare mód** :
Az azonosságot adó összehasonlításkor az adott kimenethez tartozó port-latch tartalma megjelenik a csatlakozóponton. A latch bármikor írható. Ha ez az egyezés kijelzése alatt történik, a beírt érték azonnal megjelenik a csatlakozóponton.
Impulzusszélesség-moduláció (PWM) esetén az ***utántöltési érték*** meghatározza a **periódusidőt**. A három compare regiszter (**CC1-CC3)** három PWM kimenőjelet generálhat.
Ha a **CRC**-t reload helyett szintén összehasonlításra használjuk, négy PWM jelet kaphatunk. Ebben az esetben a periódusidőt **T2CON** (**T**imer**2** **CON**trol) regiszter **T2PS** (**T**imer**2** **P**re**S**caler) bitje határozza meg: T2PS = 0 [$f_{OSZ}/12$] : 65535 óraütem = 65535 µs;
T2PS = 1 [$f_{OSZ}/24$] : 131070 óraütem = 131070 µs)

A Timer2 megszakítás engedélyezése :

A $\overline{\text{RESET}}$ után minden megszakítás tiltott. A Timer2 megszakítást az **IEN0** (**I**nterrupt **EN**able **0**) regiszter **ET2** (**E**nable **T**imer**2**) bitjével tudjuk egyedileg, az **EAL** (**E**nable **AL**L) bittel pedig globálisan engedélyezni.

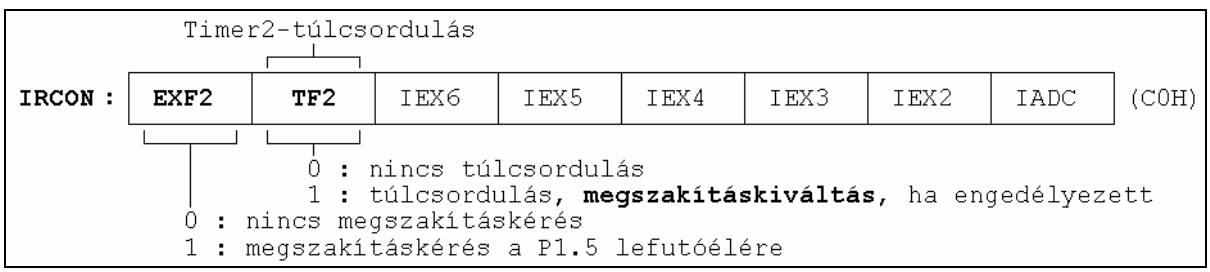


Az **IEN1** (**I**nterrupt **EN**able**1**) regiszter **EXEN2** (**EX**tern **EN**able Timer**2**) bitje engedélyezi a Timer2 megszakításával azonos hatást kiváltó külső megszakítást a P1.5/T2EX (**T**imer**2** **EX**tern) lefutóélének hatására. A lefutóél vezérelheti a Timer2 utántöltését is, ha a T2R0 és T2R1 bitek beállítása lehetővé teszi.

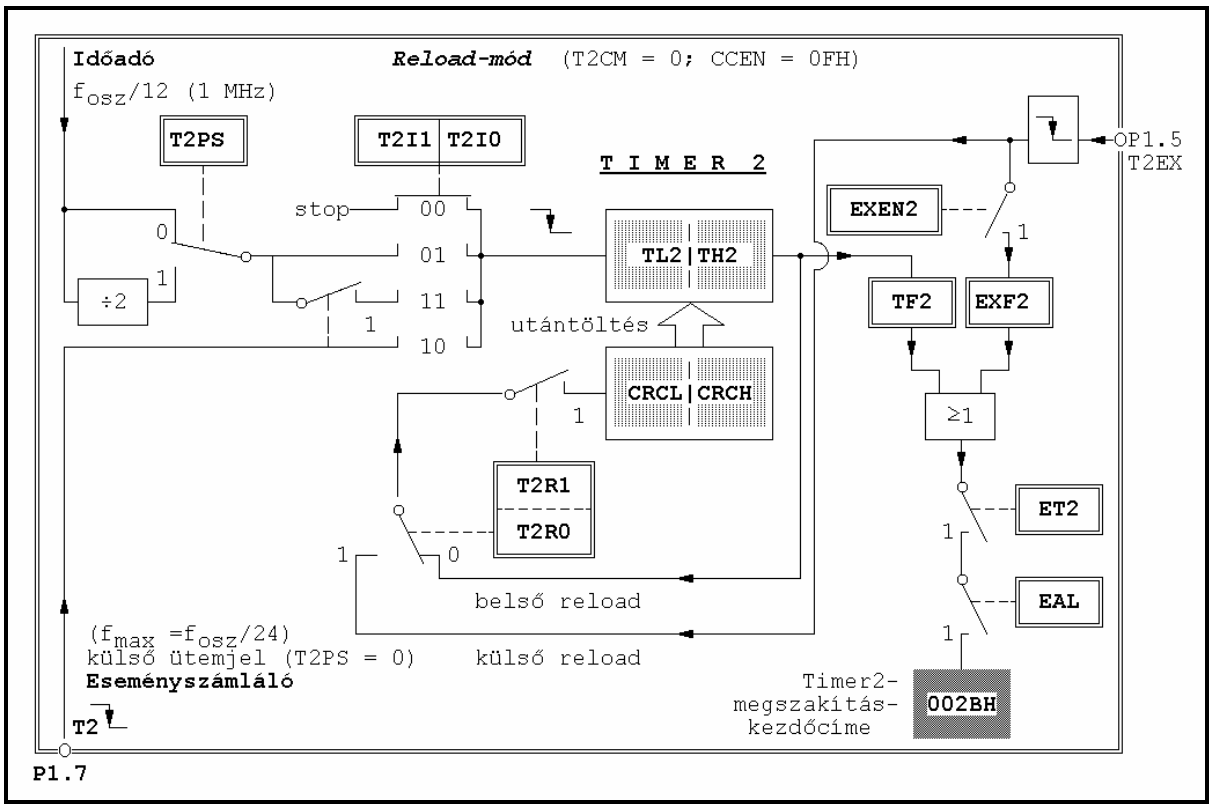


A fennálló megszakításkérések jelzése :

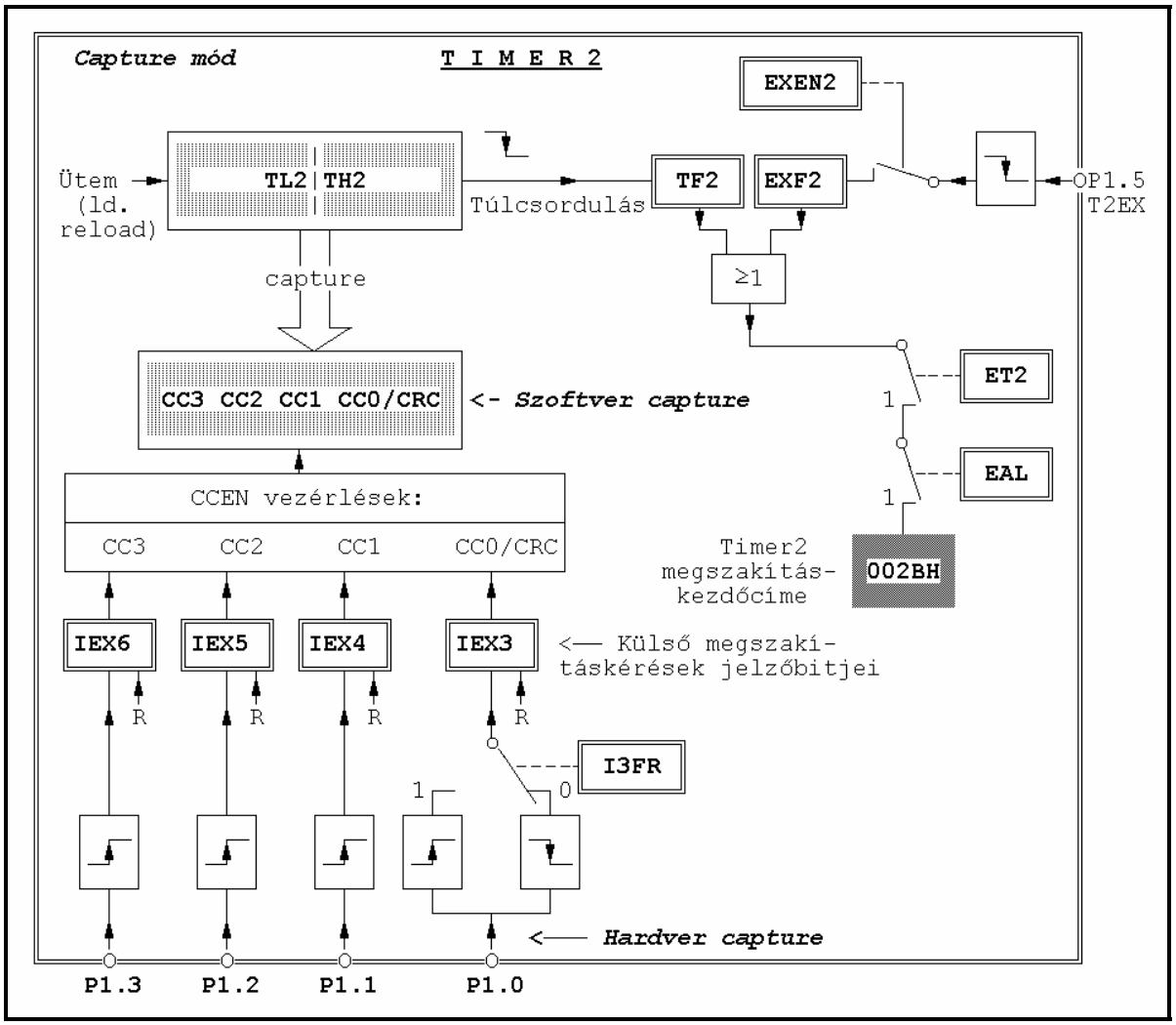
Az **IRCON** (**I**nterrupt **R**equest **CON**trol) regiszter **EXF2** bitje megszakításkérést jelez a P1.5 (T2EX) lefutóélének hatására, ugyanakkor jelzi a Timer2 utántöltésének külső kezdeményezését is (ha a T2R0 és T2R1 beállítása megengedi). A **TF2** bit a Timer2 túlcsordulását jelzi, hatására megtörténik a belső utántöltés (ha a T2R0 és T2R1 megengedi), és megszakítást vált ki, ha engedélyezett. Mindkét bitet programból kell törölni.



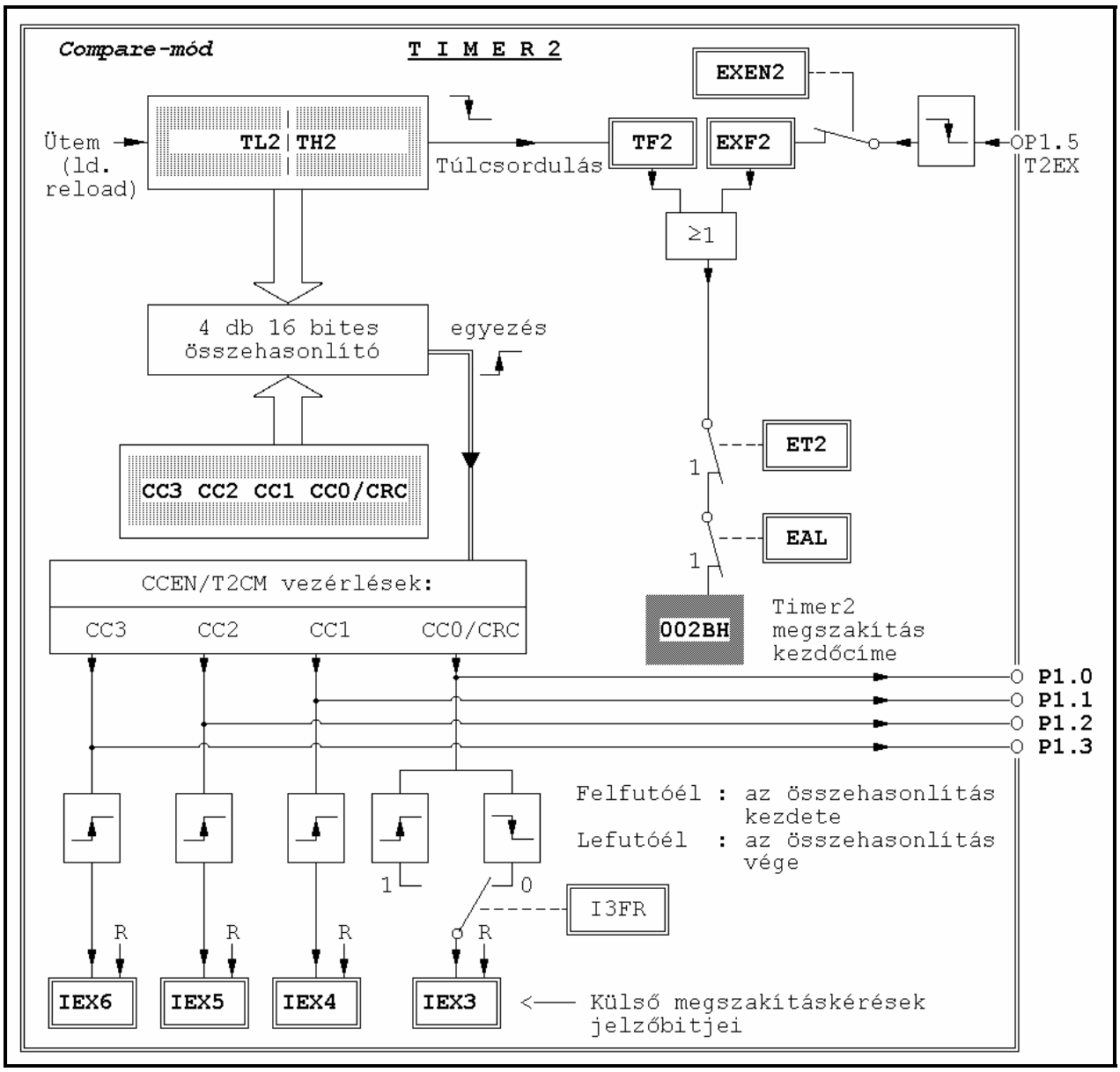
A reload-mód összefoglalása:



A capture-mód összefoglalása:



A compare-mód összefoglalása:



1. példa : **Impulzusszélesség-moduláció konstans frekvenciával** (0-ás compare mód)

A P1.0-ra és P1.1-re kötött LED-ek fényerővezérlése a P1.2 és P1.3 kapcsolókkal.
P1.2 aktív : a P5, P4 tartalmát átvenni a CC0-ba (komparálási érték)
P1.3 aktív : a P5, P4 tartalmát átvenni a CC1-be (komparálási érték)

P5-öt és P4-et kimenetként programozni		
A Timer2-t a T2CON regiszterben időadó módba állítani		
A CC0 és CC1 számára a CCEN regiszterben compare módot beállítani		
	P1.2 kikapcsolva ?	
	igen	nem
	P5-P4-et a CC0-ba másolni	
	P1.3 kikapcsolva ?	
	igen	nem
	P5-P4-et a CC1-be másolni	
ismétlés		

```
        ORG  200H               ; A program kezdőcíme.
        MOV  P5,#0FFH           ; P5 és
        MOV  P4,#0FFH           ; P4 bemenetek.
        CLR  T2PS               ; A Timer2 belső ütemjele f_OSZ/12 (1 µs-os ütem).
        CLR  T2R0               ; Nincs auto reload
        CLR  T2R1               ; a Timer2 túlcsordulásakor.
        CLR  T2CM               ; 0-ás compare mód.
        SETB T2I0               ; Időadó.
        CLR  T2I1
        MOV  CCEN,#00001010B    ; Compare mód a CC0 és a CC1 számára.
ISM:    JNB  P1.2,PWM0          ; Ugrik, ha P1.2 kikapcsolva.
        JNB  P1.3,PWM1          ; Ugrik, ha P1.3 kikapcsolva.
        LJMP KESZ
PWM0:   MOV  CRCH,P5            ; Komparálási érték a CRC/CC0-ba.
        MOV  CRCL,P4            ; (azonosság jelzése a P1.2 kimeneten)
        LJMP KESZ
PWM1:   MOV  CCH1,P5            ; Komparálási érték a CC1-be.
        MOV  CCL1,P4            ; (azonosság jelzése a P1.3 kimeneten)
KESZ:   LJMP ISM
        END
```

2. példa : **Frekvencia és impulzusszélesség változtatása :**
Impulzussorozat előállítása a P1.1-en. A frekvenciát a P1.3, az impulzusszélességet pedig a P1.2 hatására a P5-P4 portok tartalma határozza meg.

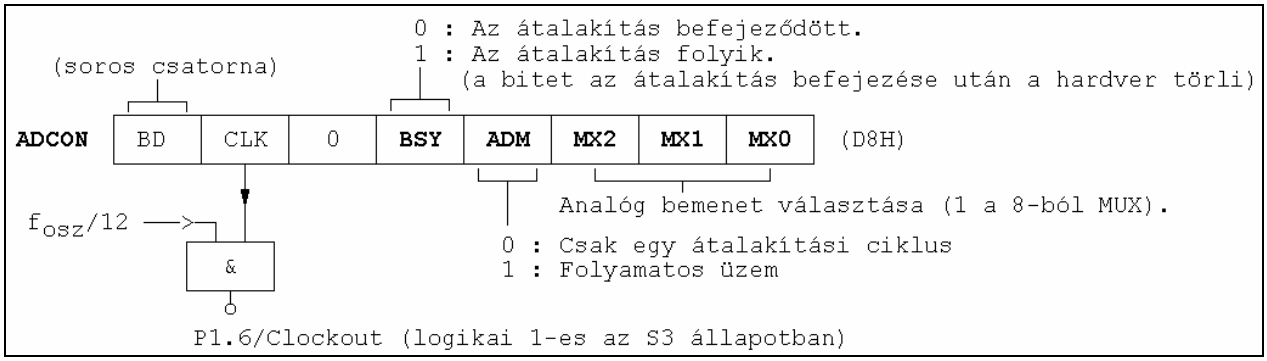
```
        ORG  200H               ; A program kezdőcíme.
        MOV  P5,#0FFH           ; P5 és
        MOV  P4,#0FFH           ; P4 bemenetek.
        MOV  T2CON,#00010001B   ; Reload túlcsorduláskor, belső ütem = f_OSZ/12.
        MOV  CCEN,#00001000B    ; 0-ás compare mód a CC1 számára.
ISM:    JNB  P1.2,IMP           ; Ugrik, ha P1.2 kikapcsolva.
        JNB  P1.3,FRQ           ; Ugrik, ha P1.3 kikapcsolva.
        LJMP KESZ
IMP:    MOV  CCH1,P5            ; Komparálási érték a CC1-be a P5-P4-ről.
        MOV  CCL1,P4            ; (impulzusidő)
        LJMP KESZ
FRQ:    MOV  CRCH,P5            ; Utántöltési érték a CRC/CC0-ba a P5-P4-ről.
        MOV  CRCL,P4            ; (frekvencia)
KESZ:   LJMP ISM
        END
```

3.10. Az analóg-digitális átalakító

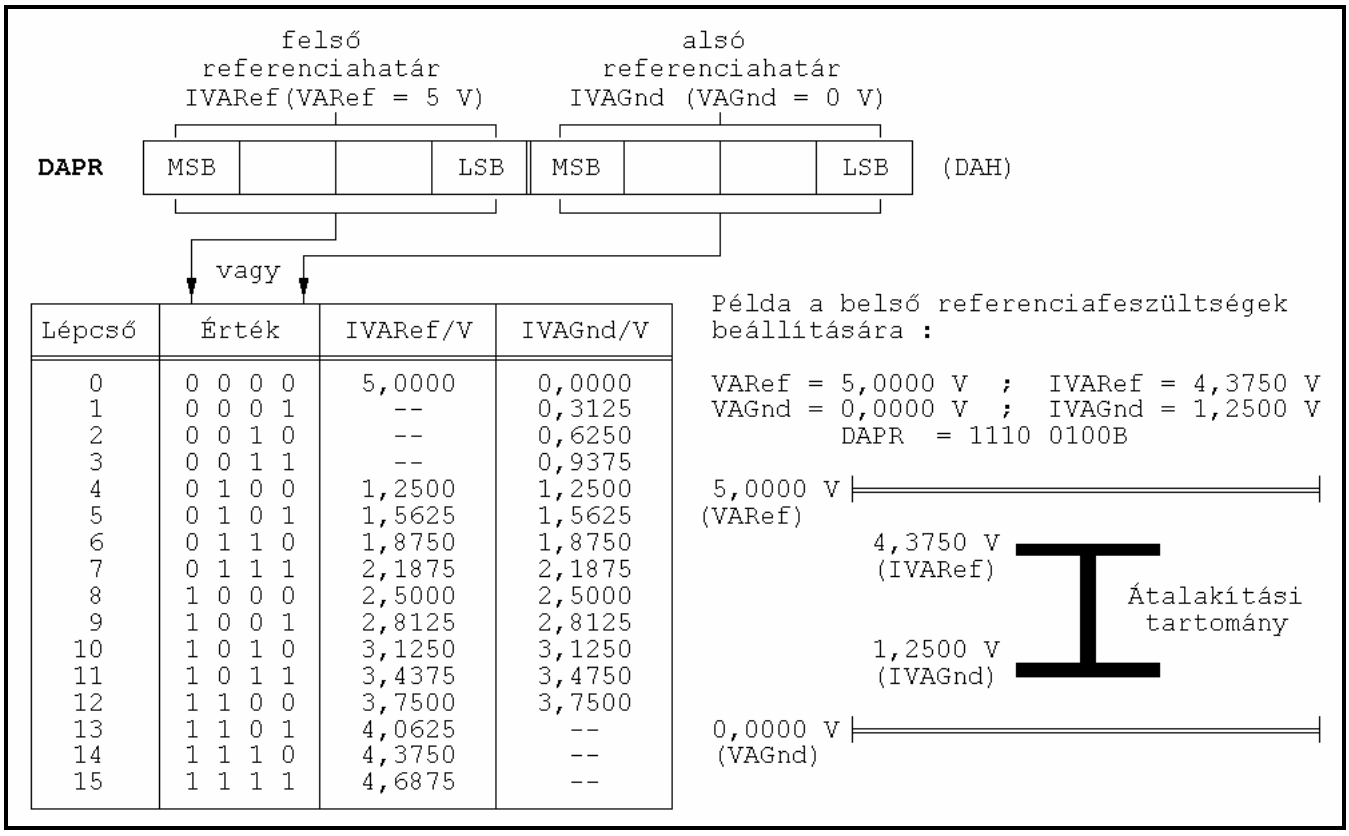
Az A/D átalakító 8 bites felbontású. A nyolc analóg bemenet (AN_0-$AN_{7)}$ multiplexelt, kiválasztásuk az **ADCON** (0D8H) regiszterben történik. Az analóg bemenőjelet az átalakítás időtartamára egy mintavevő-tartó tárolja. A mintavevő-tartó tároló-kondenzátorának töltési ideje (2 gépi ciklus) alatt az analóg bemenőjelet állandó szinten kell(ene) tartani. További követelmény, hogy az analóg jelforrás a töltési idő alatt a kondenzátor töltéséhez elegendő áramot tudjon szolgáltatni. Ez akkor teljesül, ha kimenő-impedanciája $\leq$(5 kΩ). Az átalakítás szukcesszív approximációval történik. Az átalakítási idő MYMOS mikrokontrollerben 15, ACMOS-ban 13 gépi ciklus. Az átalakítás hibája maximum 3, tipikusan ± 1 LSB. 5 gépi ciklussal az átalakítás vége előtt az **IRCON** (0C0H) regiszter **IADC** megszakításkérés jelzőbitje bebillen. Az átalakítás eredménye az utolsó gépi ciklusban íródik az **ADDAT** (0D9H) regiszterbe. Átalakítás alatt a regiszterek (ADCON, ADDAT, DAPR) tartalmát nem szabad megváltoztatni.
A mikrokontroller analóg bemenetei a hetedik portra (P6) vannak kötve amit az ACMOS változatban digitális portként is használhatunk, de csak olvasásra. A **VARef** (V_{CC} ± 0.5 V) és **VAGnd** (V_{SS} ± 0.2 V) külső referencia-feszültségekkel állíthatjuk be az átalakítási tartomány maximális alsó- és felső határát, amit a **DAPR** regiszterrel 16-16 fokozatban (4-4 bit) finomíthatunk.

Az A/D átalakító beállítása **:**

Az A/D átalakítót az **ADCON** (**AD CON**trol Register) és a **DAPR** (**DA** converter **P**rogram **R**egister) regiszterekkel kezelhetjük. Az eredmény az átalakítás utolsó gépi ciklusában az **ADDAT** (**AD** converter **DAT**a Register) regiszterbe íródik. Ha nincs szükségünk az A/D átalakítóra, az **ADDAT** regisztert általános célokra használhatjuk.
Az **MX2**, **MX1** és **MX0** (**M**ultiple**x$_{2/1/0}$**) bitekkel választhatunk a nyolc analóg bemenet közül. Az **ADM** (**AD** conversion **M**ode) bit határozza meg, hogy egyszeri vagy folyamatos konverziót végzünk. A **BSY** (**B**u**SY**) flag jelzi, hogy az átalakítás még tart, ezért az **ADDAT** tartalma még nem helyes. A flag az átalakítás vége előtt 2 gépi ciklussal visszabillen.



Az A/D átalakító a **DAPR** (**DA** converter **P**rogram **R**egister) regiszter írását követő gépi ciklusban indul. (A DAPR-ben állítjuk be a belső referenciahatárokat.)



Lépcső	Érték	IVARef/V	IVAGnd/V
0	0 0 0 0	5,0000	0,0000
1	0 0 0 1	--	0,3125
2	0 0 1 0	--	0,6250
3	0 0 1 1	--	0,9375
4	0 1 0 0	1,2500	1,2500
5	0 1 0 1	1,5625	1,5625
6	0 1 1 0	1,8750	1,8750
7	0 1 1 1	2,1875	2,1875
8	1 0 0 0	2,5000	2,5000
9	1 0 0 1	2,8125	2,8125
10	1 0 1 0	3,1250	3,1250
11	1 0 1 1	3,4375	3,4750
12	1 1 0 0	3,7500	3,7500
13	1 1 0 1	4,0625	--
14	1 1 1 0	4,3750	--
15	1 1 1 1	4,6875	--

Példa a belső referenciafeszültségek beállítására :

VARef = 5,0000 V ; IVARef = 4,3750 V
VAGnd = 0,0000 V ; IVAGnd = 1,2500 V
DAPR = 1110 0100B

Az átalakítási tartomány, amelyet a belső referenciafeszültségekkel (**IVARef** és **IVAGnd**) állítottunk be, 255 lépcsőre (8 bit) oszlik. 5V esetén a felbontás 19,6 mV. Ha a tartományt a **DAPR**-rel korlátozzuk, a felbontás nő (a fenti példában 12,25 mV). A legkisebb megengedett távolság az IVARef és IVAGnd referenciafeszültségek között 1 V.

Példa :

```
          ANL     ADCON,#0E0H   ; 1-es analóg csatorna, egyszeri átalakítás.
KONV:     MOV     DAPR,#0       ; Belső referencia azonos a külső referenciával (5V).
                                ; Átalakítás indul.
VAR:      JB      BSY,VAR       ; Vár az átalakítás végére.
          MOV     P4,ADDAT      ; Eredmény a P4 portra.
          LJMP    KONV          ; Újabb átalakítás.
          END
```

<u>Az analóg-digitális átalakító megszakításkérésének jelzése :</u>

Az **IRCON** (**I**nterrupt **R**equest **CON**trol) regiszter **IADC** (**I**nterrupt **AD** **C**onversion) bitje 5 gépi ciklussal az átalakítás vége előtt bebillen. **<u>A jelzőbitet programból kell törölni.</u>**

A megszakításkéréseket a processzor minden gépi ciklusban lekérdezi. Amikor az átalakítás vége előtt 5 gépi ciklussal az IADC bitet bebillenve találja, a megszakítás-kiszolgáláshoz először egy LCALL utasítást generál, ami 2 gépi ciklusig tart. A hívott megszakítás belépési címén talált LJMP (átirányító) ugróutasítás végrehajtása szintén 2 gépi ciklust vesz igénybe. A tényleges kiszolgálóprogram első

utasítását, amely az ADDAT regisztert olvassa már csak azután a gépi ciklus után tudja végrehajtani, amelyben az A/D átalakító az eredményt az ADDAT regiszterbe beírta.

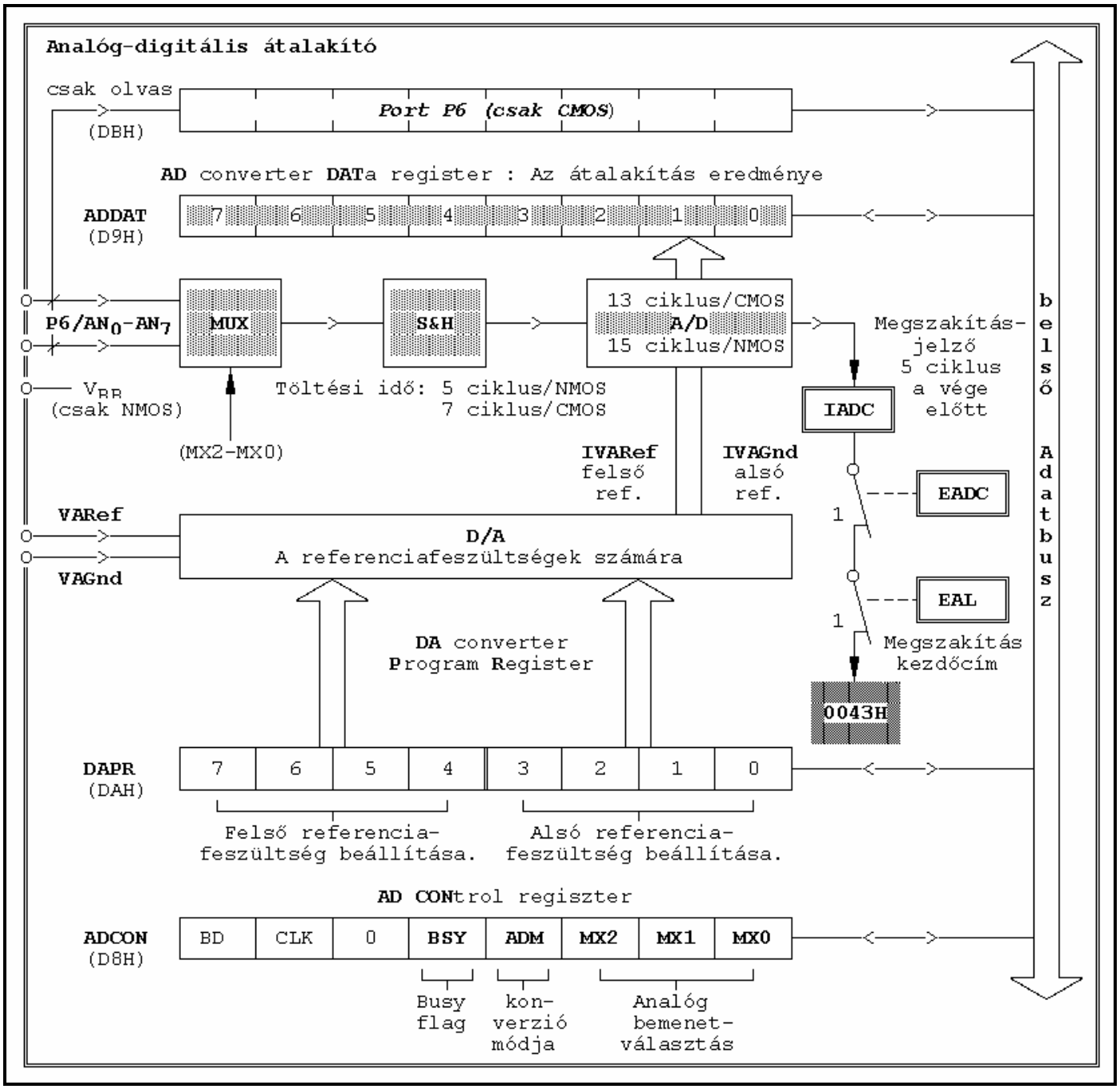
IRCON :	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	**IADC**	(C0H)

A megszakítás engedélyezése :

Az **IEN1** (**I**nterrupt **EN**able **1**) regiszter **EADC** (**E**nable **AD** **C**onverter) bitjével és az **IEN0** (**I**nterrupt **EN**able **0**) regiszter **EAL** (**E**nable **AL**l) globális engedélybitjével tudjuk az A/D átalakító megszakítását-kiváltását engedélyezni.

IEN1 :	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	**EADC**	(B8H)
IEN0 :	**EAL**	WDT	ET2	ES	ET1	EX1	ET0	EX0	(A8H)

Az analóg-digitális átalakító összefoglalása:

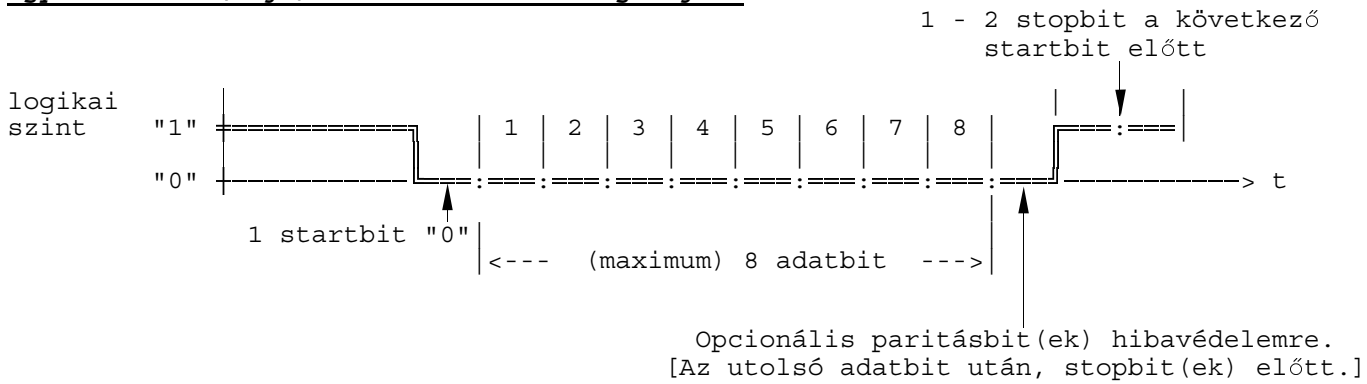


3.11. Soros adatátvitel

A számítógépek és az egyéb készülékek közötti adatátvitel **kisebb távolságokra *párhuzamosan***, **nagyobb távolságokra *sorosan*** történik. A soros adatátvitel lehet szinkron, amikor az adás ütemjelét is átvisszük, vagy aszinkron, amikor a vétel ütemezését a vett jelfolyam valamely jól felismerhető pontjához (start) kell "szinkronizálni".

3.11.1. Soros aszinkron adatátvitel

Egy karakter (bájt) átvitelének idődiagramja :



A soros aszinkron adatátvitel tulajdonságai :

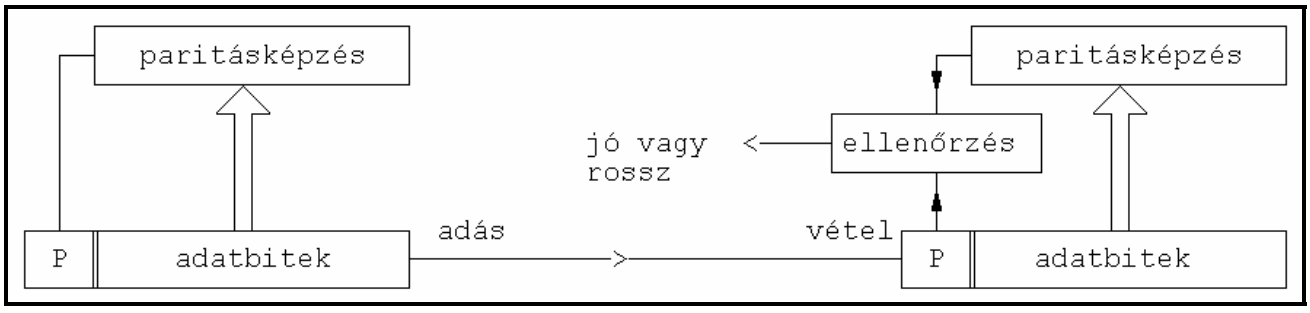
- → Az adatbitek száma 5-8 közé eshet, ezeket időben egymás után viszik át, általában a legkisebb helyiértékű bittel (LSB) kezdődően.
- → Az adó és a vevő saját, független ütemforrással rendelkezik (ezért "aszinkron").
- → Az átvitel szünetében a vezeték nyugalmi szinten (logikai "1") van.
- → Szinkronizálásra az átvitt karakter első lefutóéle (startbit) szolgál.
- → Az adatbitek az ütemezés által meghatározott időrésekben következnek.
- → Hibavédelemre a paritásbit(ek) szolgál(nak).
- → Végül 1-2 stopbit következik, ezalatt az adó és a vevő alapállapotba áll.

Az adatbitek számának, a bitsorrendnek, az átviteli sebességnek, a paritásnak és a stopbitek számának az adó- és vevőoldalon **azonosnak** kell lenni.

Az átvitel hibavédelme :

A hibavédelem legegyszerűbb lehetősége a paritásbit :

páros paritás : a paritásbit a karakter kódjának 1-eseit páros darabszámúra egészíti ki.
páratlan paritás: a paritásbit a karakter kódjának 1-eseit páratlan darabszámúra egészíti ki.



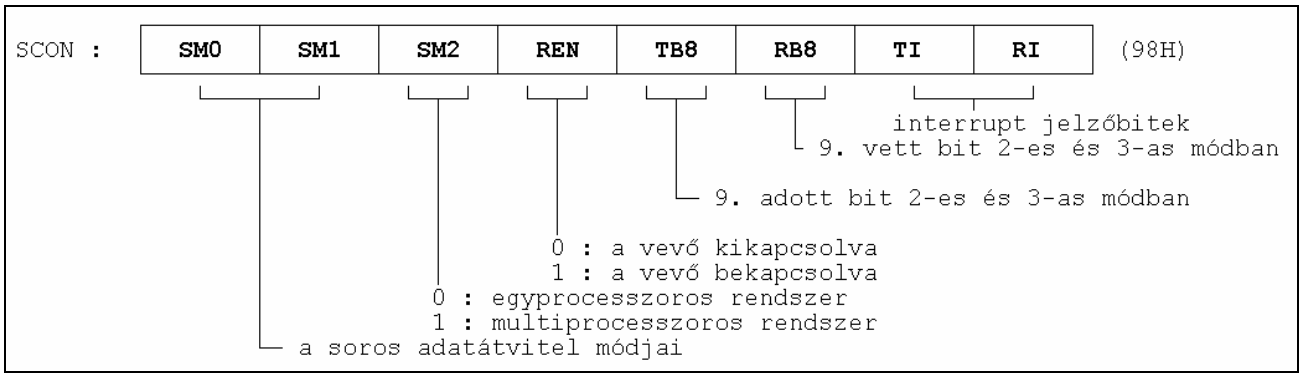
3.11.2. A mikrokontroller soros csatornájának felépítése és tulajdonságai

A soros adatátvitel csatlakozópontjai :

ADÁS : (TxD : **T**ransmitted **D**ata) : P3.1 (Az adás és a vétel az LSB-vel kezdődik)
VÉTEL : (RxD : **R**eceived **D**ata) : P3.0

A soros csatorna beállításai :

A 80515-ös mikrokontroller egy **UART-ot** (**U**niversal **A**synchronous **R**eceiver **T**ransmitter) tartalmaz, amely lényegében az adott ill. vett karaktert tároló **SBUF** (**S**erial **BUF**fer) regiszterből és a soros adatátvitelt vezérlő **SCON** (**S**erial port **CON**trol register) regiszterből áll. A duplex üzem azért lehetséges, mert az SBUF adási és vételi célra két külön regisztert tartalmaz, bár a programozó ugyanazzal a szimbolikus névvel érheti el írásra és olvasásra.



SM0	SM1	Jelentése
0	0	**0-ás mód :** szikron soros adatátvitel. Az UART shiftregiszter $f_{osz}/12$-vel. Az RxD az adat, a TxD az ütemjel be- és kimenet.
0	1	**1-es mód :** 8 bites UART. Az ütemjelforrás belső baud-rate generátor, vagy a Timer1 túlcsordulása, a **BD** bittel átkapcsolhatóan.
1	0	**2-es mód :** 9 bites UART rögzített ütemmel : $f_{osz}/64$: SMOD [PCON.7] = 0 $f_{osz}/32$: SMOD [PCON.7] = 1 A 9. bit adásnál a TB8, vételnél az RB8. Ez lehet paritásbit, második stopbit vagy kilencedik adatbit.
1	1	**3-as mód :** 9 bites UART. Az ütemjelforrás belső baud-rate generátor, vagy a Timer1 túlcsordulása, a **BD** bittel átkapcsolhatóan. A 9. bit adásnál a TB8, vételnél az RB8. Ez lehet paritásbit, második stopbit vagy kilencedik adatbit.

SM2 = 0 : egyprocesszoros rendszer.
Az RI jelzőbit minden bájt vétele után bebillen.

SM2 = 1 : master-slave rendszer.
Az RI jelzőbit akkor billen be, ha 2-es és 3-as módban a kilencedik bit 1.

Master-slave rendszerben a master processzor először a 256 lehetséges slave processzor egyikének 8 bites címét küldi (a kilencedik bit 1), majd egy adatsorozatot (a kilencedik bit 0). Valamennyi slave processzor (SM2 = 1) megvizsgálja, hogy a küldött cím neki szól-e, és ha igen átkapcsolja az SM2 bitet 0-ba az adatok vételéhez. Az adatsorozat végét speciális karakter jelzi.

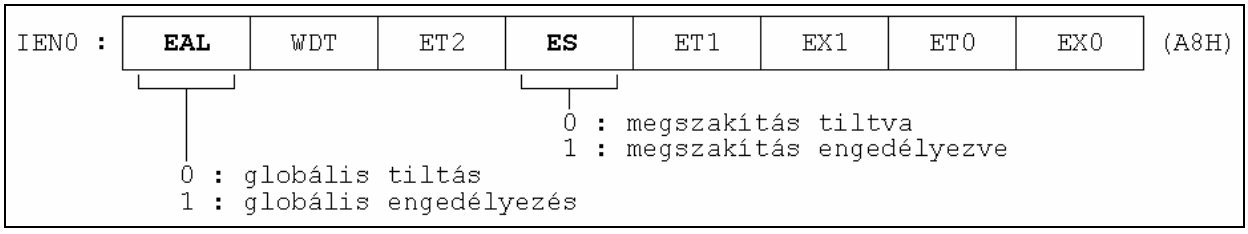
Az adási/vételi megszakításkérések kijelzése:

RI = 1 : az **SCON** regiszter vételi megszakításkérés jelzőbitje (**RI; R**eceive **I**nterrupt) bebillen a stopbit vételekor; ilyenkor a ***vételi puffer megtelt.*** **Az RI jelzőbitet szoftverből kell törölni.** Nagy bitsebesség esetén az RI-nek magas prioritást kell adni, hogy más megszakítások ne okozhassanak bitvesztést a vételben.

TI = 1 : az **SCON** regiszter adási megszakításkérés jelzőbitje (**TI; T**ransmit **I**nterrupt) bebillen a stopbit adásakor; ilyenkor az ***adási puffer üres.*** **A TI jelzőbitet szoftverből kell törölni.**

A megszakítás-kiszolgálások programból is elindíthatók a TI vagy RI bebillentésével.

Az adási/vételi megszakítások engedélyezése:
Az **IEN0** (**I**nterrupt **EN**able **0**) regiszter **ES** (**E**nable **S**erial) bitjével és az **EAL** (**E**nable **AL**l) regiszter globális engedélybitjével tudjuk a soros csatorna TI+RI megszakítását



Ütemjel kijelölése 1-es és 3-as módban :

Az ütemjelet ***választhatóan*** vagy a Timer1 túlcsordulásai (ld. 3.10.1), vagy a belső baud-rate generátor szolgáltathatja. Választásra az ADCON (A/**D** **CON**verter Control) regiszter **BD** (**B**au**D**) bitje szolgál. A **PCON** (**P**ower **CON**trol) regiszter **SMOD** (**S**erial **MOD**us) bitje megduplázza a bitsebességet.

ADCON :

BD	CLK	0	BSY	ADM	MX2	MX1	MX0

(D8H)

0 : az ütemjelet a Timer1 túlcsordulásai szolgáltatják
1 : az ütemjelet a belső baud-rate generátor szolgáltatja
f_{osz} = 12 MHz : 4800 baud, ha SMOD = 0
9600 baud, ha SMOD = 1

PCON :

SMOD	PDS	IDLS	SD	GF1	GF0	PDE	IDLE

(87H)

TILOS MEGVÁLTOZTATNI !!!
(ezek a bitek többek között a Power down módot vezérlik)
csak bájtcímezhető !!!

0 : 4800 baud bitsebesség/BD = 1 (ANL PCON, #01111111B)
1 : 9600 baud bitsebesség/BD = 1 (ORL PCON, #10000000B)

2-es módban :

0 : $f_{osz}/64$ baud bitsebesség
1 : $f_{osz}/32$ baud bitsebesség

A szükséges bitsebesség előállítása a Timer1 túlcsordulásával 1-es és 3-as módban:

Átviteli sebesség	f_{osz}	SMOD	Timer1 (C/$\overline{T}$ = 0)	
			üzemmód (reload)	reload érték
62500 baud (multiprocesszoros) rendszerekben)	12.0 MHz	1	2	FFH (255)
19200 baud (RS 232D)	11.059 MHz	1	2	FDH (253)
9600 baud	11.059 MHz	0	2	FDH (253)
4807 baud	12.0 MHz	1	2	F3H (243)
2403 baud	12.0 MHz	1	2	E6H (230)
1201.9 baud	12.0 MHz	1	2	CCH (204)
110 baud (géptávíró)	12.0 MHz	0	(nincs reload) 1	FEEBH (65259) megszakításki-szolgálásból min-dig újratölteni

$$\text{16-szoros átviteli sebesség} = f_{osz}/12 * \frac{1}{(256 - \text{reload érték})}$$

A soros adatátviteli csatorna összefoglalása :