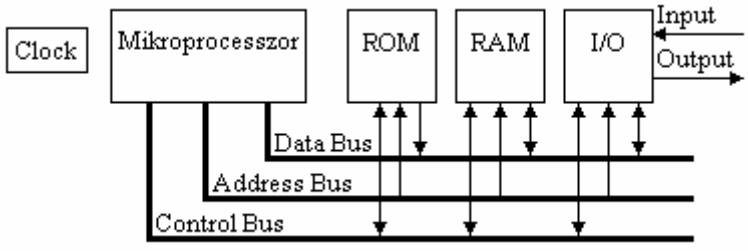
1. Bevezetés

A mikroelektronika és a számítástechnika története rövid. A 19. században terveztek számítógépeket, amelyek utasításkészlettel rendelkeztek (Charles Babbage). E gépeket mechanikus szerkezetként tervezték, így működőképes modellek nem készülhettek. A 20. század első felében az elektromechanikus relék jelentették azt az eszközt, amelyekkel ténylegesen működőképes berendezéseket lehetett létrehozni. Egy személyt említsünk meg e terület számtalan tudósa közül, Neumann Jánost, aki a mai értelemben is korszerű számítógépek működésének alapelveit lerögzítette. Tevékenysége alapvetően meghatározta a számítástechnika fejlődését. E korból a leghíresebb számítógép az ENIAC volt, melyet a II. világháború idején építettek tudományos célból a Pennszilvániai Egyetemen. A számítógépek kereskedelmi méretű elterjedését az 50.-es években megjelenő tranzisztor tette lehetővé. A 60-as években pedig az integrál áramkörös technológia rohamos árcsökkenést eredményezett. Újabb áttörést és újabb forradalmat jelentett a számítástechnikában a a mikroprocesszorok megjelenése. Ezek első képviselője az Intel cég Intel 4004-es 4 bites processzora volt. A következő évben készen volt az Intel8008, az első 8 bites mikroprocesszor. Az ezt követő fejlettebb változat, a z I8080, amelyből már az első „home computer”-ek készültek.

A 80-as évek elejére a mikroprocesszorok rohamosan elterjedtek. A 8 bites mikroprocesszorokat követték a 16 és rövidesen a 32 bites processzorok is. A szoftverek szintén fejlődtek, mialatt az árak relatívan gyorsan csökkentek. Megjelentek a személyi számítógépek, amelyek néhány száz dolláros áruk ellenére a 30 évvel korábbi szuperszámítógépek teljesítményét nyújtották, vagy meghaladták. A személyi számítógépek mellett a mikroprocesszor technika az élet minden területén elterjesztette a számítástechnika alkalmazását. Jelenleg a járműipartól a szórakoztató iparig, egészségügytől az űrkutatásig, telekommunikációig, mesterséges intelligenciáig mikroprocesszorok millióit alkalmazzák, forradalmasítva a fejlődést az élet minden területén.

2. A mikroszámítógépek felépítése

A mikroszámítógépek alapja a mikroprocesszor. Elemei a mikroprocesszor, memória, és input/output eszközök. A komponenseket valamilyen buszrendszer köti össze, amelyen az egységek közötti adatcsere zajlik. A mikroszámítógép blokkvázlata az 1. ábrán látható.



1. ábra: A mikroszámítógép felépítése.

3. Mikrovezérlők

A mikroszámítógép alacsony ára lehetővé tette, hogy ne csak a szoros értelembe vett számítástechnikai feladatokra használják. Az irányítástechnikával foglalkozó szakemberek hamar meglátták a mikroszámítógépekben rejlő lehetőségeket saját szakterületüket illetően is, azt, hogy vezérlési és szabályozási feladatokra is alkalmasak ezek az eszközök. Nagyszámú alkalmazásban jelentek meg a mikroprocesszorból, memóriából és megfelelően megválasztott input/output eszközökből álló mikroszámítógépek. A mikroprocesszor gyártók, amikor a technológiai fejlődés lehetővé tette, megkezdték az egy chipes mikroszámítógépek, az úgynevezett mikrokontrollerek kifejlesztését és gyártását. A mikrokontrollereket olyan alkalmazások céljára gyártották, amelyek nem kimondottan számítástechnikai alkalmazások, de komplikált vezérlő algoritmust és flexibilis megoldást igényelnek. Ilyenre példa az automata mosógép, vagy egy belsőégésű motor gyújtás vezérlése. Ma nagy választékban kaphatók mikrovezérlők

Az alkalmazások túlnyomó része két kategóriába sorolható: vezérlési, vagy nyílt hurkú irányítási feladatok, illetve szabályozási, vagy zárt hurkú feladatok. A nyílt hurkú feladatokat szekvenciális vezérléseknek is nevezik, mivel a folyamatot állapotok sorozata jellemzi. Az állapotok közti átmeneteket valamilyen diszkrét események eredményezik. Az ilyen alkalmazásokat esemény vezérelt alkalmazásoknak is nevezik. Példa erre az áruautomata. A berendezés különböző értékű pénzérméket fogad el, felismeri az áru kiválasztást, kiadja az árut, meghatározza a pontos értéket, és visszaadja a különbözetet. Ha a bedobott pénz nem elegendő, vagy az áru kifogyott, kijelzéseket ad.

A zárt hurkú alkalmazások esetén valós idejű jelleggel ellenőrizzük az irányított folyamatot, hogy folyamatosan hatékony legyen az irányítás. Automata szerszámgépeknél, vagy robottechnikában számtalan ilyen alkalmazás található. A mikrovezérlők alkalmazása rendkívül széles. Robot technikában, alakfelismerésben, vagy például telekommunikációs alkalmazásokban is számtalan mikrovezérlő megoldás található.

A mikrovezérlőket gyakran beágyazott vezérlőknek (embedded controller) is nevezik, mivel általában egy nagyobb rendszer részét képezik. A rendszer felhasználója nem kell, hogy feltétlenül tudjon arról, hogy a rendszer mikrovezérlőt tartalmaz.

5. A mikrovezérlők általános jellemzői

Napjainkban sokféle mikrovezérlőt gyártanak, és az egyes típusokon belül is nagy a változatok száma. Az összehasonlításokból azonban kiderül, hogy sok olyan jellegzetességük van, amelyek általánosan jellemzik őket. A közös jellemzőket az alábbi pontban foglaljuk össze.

1. Egy chipes áramkörök, amelyek valamilyen mikroszámítógép konfiguráció valamennyi alkotóelemét, CPU, RAM, ROM, I/O elemek, rendszer óragenerátor, tartalmazzák.

2.Túlnyomórészt Harvard felépítésűek, vagyis külön program és adat memóriával rendelkeznek. A program memória 4-8 Kbyte, az adat memória viszonylag kicsi, maximum

egy-két Kbyte nagyságú. Ez abból a tényből adódik, hogy egy általános vezérlési feladat sok vezérlő utasítással és kevés átmeneti adat tárolásávaloldható meg.

3. Utasításkészletük általában egyszerű, programozásuk könnyen elsajátítható.

4. Külső memóriával nem bővíthetők, vagy ha igen, az az I/O vonalak felhasználásával történik, és ez a szabadon maradó I/O vonalak számát csökkenti.

5. Jellegzetes erőforrásokkal rendelkeznek. Ilyenek a Watch Dog timer, általában több számláló/időzítő áramkör, nagyszámú digitális bemeneti és kimeneti vonal. Újabban vannak olyan típusok, amelyek analóg bemenő jelek feldolgozására is alkalmasak.

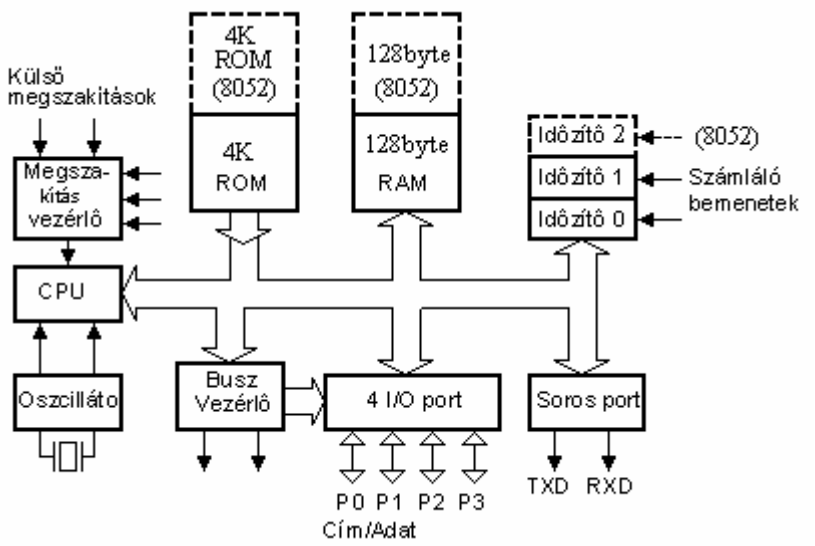
6. A kivezetések számának csökkentése céljából többcélú kivezetéseik vannak.

7. Számos külső megszakítás vonallal rendelkeznek, ezenkívül a belső eszközök nagy része is megszakítás vonallal van ellátva.

8. Általában tartalmaznak soros kommunikációs portot, amely a programozásukat megkönnyíti, valamint lehetővé teszi beillesztésüket egy számítógépes

Az MCS 51 család

Ez a mikroszámítógép-család az MCS 48-as család jelentős továbbfejlesztésének tekinthető, elvi felépítése az alábbi ábrán látható:



2. ábra: Az MCS 51 család blokkvázlata

Az MCS 51 család főbb jellemzői:

- 8 bites, vezérlési feladatokra optimalizált központi egység (CPU);
- 64 Kbyte memória címezhető;
- az órajel-generátor áramköre a lapkán van;
- 32 be- illetve kimeneti vonal;
- mind az adat, mind a programtároló 64 Kbyte-ra bővíthető;
- két, illetve egyes típusoknál három időzítő-számláló;
- soros duplex adatátviteli port;
- Bool-algebrai processzor.

A család elemei és jellemzőik:

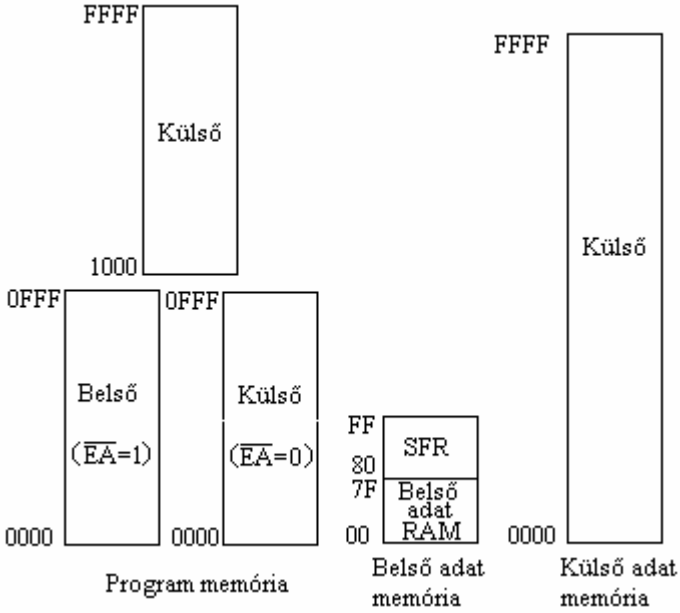
Típus	**ROM nélküli változat**	**EPROM-os változat**	**ROM méret (Byte)**	**RAM méret (byte)**	**8-bites I/O port**	**16-bites számláló**	**Megszakítás szám/vektor**
8051	8031	nincs	4K	128	4	2	6/5
8051AH	8031AH	8751H 8751BH	4K	128	4	2	6/5
8052AH	8032AH	8752BH	8K	256	4	3	8/6
80C51BH	80C31BH	87C51	4K	128	4	2	6/5
80C52	80C32	nincs	8K	256	4	3	8/5

Felépítés, jellemzők

A következőkben röviden összefoglaljuk az MCS 51 család azon tulajdonságait, amelyek az alkalmazásoknál a kiválasztást elősegíthetik.

Társzervezés

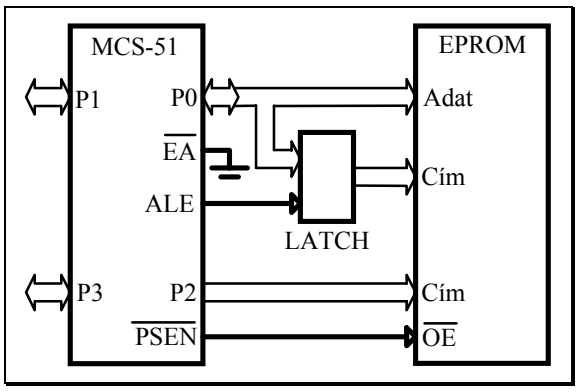
A CPU az adat- és a programmemóriát teljesen eltérő módon kezeli. A mikroszámítógépben különálló program- és adattároló van, az előbbi 4K, illetve 8K méret és az alsó címtartományt foglalja el. Az adatmemóriát a lapkán kialakított 128 vagy 256 byte-s RAM alkotja. A RAM 8 bittel címezhető, ami gyorsabb működést tesz lehetővé. A teljes 64 Kbyte a DTPR regiszteren keresztül érhető el. (A regiszterkészlet leírását lásd később.) Az áramkörbe beépített memórián kívül, vagy azzal együtt, külső memória használatára is van mód, a normál CPU-knál megszokott módon.



3. ábra: A 8051memóriatérképe

Programmemória

A belső programmemória használatát az $\overline{EA}$ jellel szabályozhatjuk. Ha erre a bementre 1-es szintet (tápfeszültség) kötünk, akkor a processzor a 0000H-0FFFH címek esetén az utasításokat a belső programmemóriából olvassa be. (8 Kbyte belső ROM esetén a címtartomány: 0000H-1FFFH.) Ha az $\overline{EA}$ bemenetre 0 szintet kötünk, akkor a processzor mindig a külső memóriából olvassa az utasításokat. Ebből következik, hogy a ROM nélküli változatoknál az $\overline{EA}$ bemenetet mindig 0 szintre kell kötni. Külső programmemóriát a következőképpen használhatunk:



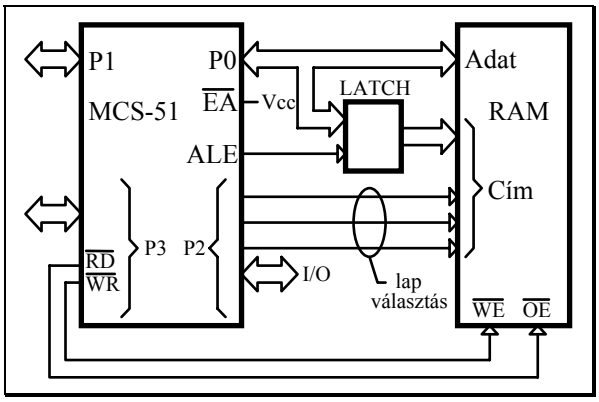
Ebben az esetben a P0 port multiplexelt cím/adatbuszként mûködik, a P2 pedig a címbusz felső fele lesz.

A multiplexelt cím és adat szétválasztását az ALE jel segíti, míg a $\overline{\text{PSEN}}$ jel használható olvasás jelként. A memóriacím mindig 16 bites, és a külső memória elérése lefoglalja a P0 és P2 portokat, így azok másra nem használhatók.

Amennyiben használjuk a belső memóriát, de a cím kívül esik a belső címtartományon, a CPU a külső memóriához fordul, akár van akár nincs. Ha tehát nincs külső memória és a portokat I/O-ként használjuk, akkor vigyáznunk kell, hogy a program ne tévedjen a belső programmemórián kívülre.

Adatmemória

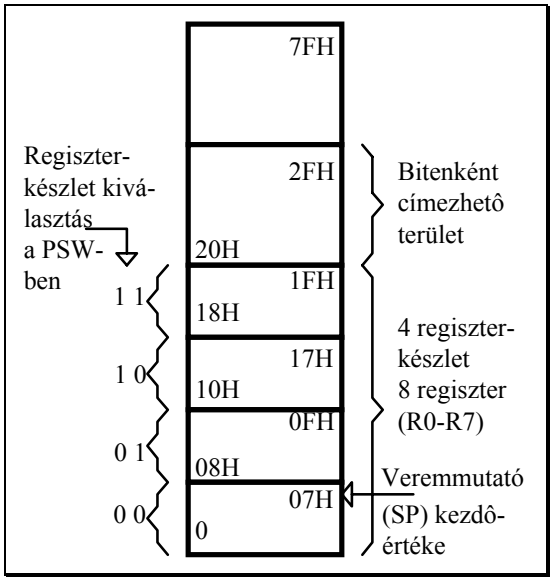
Amennyiben külső adatmemóriát kapcsolunk az áramkörhöz, úgy az csak a MOVX utasítással érhető el. Ezzel az utasítással 8 vagy 16 bites címzést használhatunk. 8 bites címzés esetén gyakori, hogy több 256 byte-os memória közül a P2 portra kiírt értékkel választjuk ki az aktív lapot.



Ebben a példában a belső ROM-ot használjuk programmemóriaként. A külső RAM mérete 2K (8x256 byte). A RAM-ot 8 bittel címezzük, a 256 byte-os lapokat előzőleg mindig a P2 portra kiírt értékkel választjuk ki. Ez azért előnyös, mert így a P2 port fennmaradó 5 vonala I/O-ként felhasználható, míg 16 bites címzés esetén a P2 buszként működik, így nem használható fel (külső programmemória használata esetén mindig ez a helyzet - lásd az előbb).

A belső memória 8 bittel címezhető, tehát maximum 256 byte lehet. Ebből típustól függően 128 byte, vagy 256 byte használható RAM-ként. Ez a terület két részre oszlik: az alsó 128 byte (00H-7FH) és a felső 128 byte (80H-0FFH) eltérően kezelendő. (A címzési módokat részletesebben majd a processzor utasításkészletének ismertetésénél mutatjuk be.)

Az alsó 128 byte direkt és indirekt módon is megcímezhető. Ezen a területen helyezkednek el a regiszterek (a 8 bites CPU-knál megszokott regiszterkészlet nincs a processzorban) és a verem. Az alsó 32 byte (00H-1FH) négy különálló regiszterkészletet tartalmaz. Minden regiszterkészlet 8 regiszterből (R0-R7) áll. A regiszterkészletek közül egyszerre természetesen csak egy lehet aktív, ezt a PSW (program-állapot szó) regiszter két bitje határozza, meg és programból állíthatjuk be. A processzor regiszterei tehát az áramkörbe beépített adatmemóriában vannak, így kétféleképpen is elérhetők, úgy is, mint regiszterek és úgy is, mint memória. A verem-mutató (SP) bekapcsolás utáni kezdőértéke is erre a területre mutat (a kezdőérték 7). A 20H-val kezdődő terület általános RAM-ként használható, melyen belül a 20H-2FH címterület bitenként is megcímezhető (bit cím: 0-7F).



A felső 128 byte csak a 8052 sorozatban használható RAM-ként és ott is csak indirekt címzéssel. Ha direkt címzést használunk akkor az MCS 51 sorozat minden típusánál az úgy-nevezett SFR-t érhetjük el (***S**pecial **F**unction **R**egisters*, azaz "különleges funkciójú regiszterek"). Az SFR tartalmazza a portok tárolóit, az időzítőket, a perifériák vezérlőregisztereit stb. Ezek a regiszterek tehát csak direkt címzéssel érhetők el. Az SFR területen tizenhat byte bitenként is megcímezhető. Ezek azok a byte-ok, amelyeknek a címe 8-cal osztható. Ezek a 80H-FFH bitcímen bitenként *is* elérhetők (elsősorban a portok és a vezérlőregiszterek).

A következő táblázat tartalmazza a regiszterek nevét, címét és a RESET utáni kezdőértéket. A csillaggal (*) jelölt regiszterek bitenként is címezhetők. A plusz jellel (+) jelölt regisztereket csak a 8052 sorozat tartalmazza. A kezdőértékekben előforduló X jel azt jelenti, hogy az a bit nem definiált.

Szimbólum	**Név**	**Cím (hexa)**	**Kezdőérték (binárisan)**
*ACC	Akkumulátor	E0	00000000
*B	B regiszter	F0	00000000
*PSW	Program állapot szó	D0	00000000
SP	Verem mutató	81	00000111
DPTR	Adat mutató, 2 byte		
DPL	Alsó byte	82	00000000
DPH	Felső byte	83	00000000
*P0	Port 0	80	11111111
*P1	Port 1	90	11111111
*P2	Port 2	A0	11111111
*P3	Port 3	B0	11111111
*IP	Megszakítás prioritás vezérlés	B8	8051 XXX00000 8052 XX000000

*IE	Megszakítás engedélyezés	A8	8051 0XX00000 8052 0X000000
TMOD	Számláló, üzemmód	89	00000000
*TCON	Számláló, vezérlés	88	00000000
*+T2CON	Számláló 2, vezérlés	C8	00000000
TH0	Számláló 0, felső byte	8C	00000000
TL0	Számláló 0, alsó byte	8A	00000000
TH1	Számláló 1, felső byte	8D	00000000
TL1	Számláló 1, alsó byte	8B	00000000
+TH2	Számláló 2, felső byte	CD	00000000
+TL2	Számláló 2, alsó byte	CC	00000000
+RCAP2H	Számláló 2, tároló regiszter felső byte	CB	00000000
+RCAP2L	Számláló 2, tároló regiszter alsó byte	CA	00000000
*SCON	Soros vonal vezérlés	98	00000000
SBUF	Soros vonal puffer	99	nem definiált
PCON	Fogyasztás vezérlés	87	HMOS 0XXXXXXX CHMOS 0XXX0000

Regiszterek

PSW (Program Status Word)

CY	AC	F0	RS1	RS0	OV	-	P

CY	PSW.7	Átvitel flag.
AC	PSW.6	Félátvitel (3. bitről a 4-ikre).
F0	PSW.5	Flag 0. Szabadon felhasználható.
RS1	PSW.4	Regiszterkészlet kiválasztás, 1. bit.
RS0	PSW.3	Regiszterkészlet kiválasztás, 0. bit.
OV	PSW.2	Túlcsordulás.
-	PSW.1	Felhasználó által definiálható.
P	PSW.0	Paritás. Hardware által beállított flag. Ha az akkumulátorban az egyesek száma páros, akkor 0, ha az egyesek száma páratlan, akkor 1.

A regiszterkészlet kiválasztása két bittel történik:

RS1	RS0	Regiszterkészlet	Cím
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

PCON (Power Control Register)

SMOD	-	-	-	GF1	GF0	PD	IDL

SMOD	Dupla baud sebesség bit. Ha a Timer 1 állítja elő a soros vonal órajelét és az SMOD=1, akkor a soros vonal sebessége duplázódik.
GF1	Általános célú bit.
GF0	Általános célú bit.
PD	Ennek a bitnek az 1-be állítása aktiválja a Power Down üzemmódot a 80C51BH típusnál. (Csak a CHMOS áramkörnél mûködik.)
IDL	Ennek a bitnek az 1-be állítása aktiválja az Idle Mode üzemmódot a 80C51BH típusnál. (Csak a CHMOS áramkörnél mûködik.)

Amennyiben a PD és az IDL bit egyidőben 1-es értékű, a PD-nek van elsőbbsége. A HMOS áramköröknél csak az SMOD bit van megvalósítva, a többi négy bit csak a CHMOS áramkörökben található meg. A 4-6 biteket (HMOS áramköröknél a 0-6 biteket) nem szabad 1-be állítani!

A CHMOS áramkörök fogyasztáscsökkentő üzemmódjai

Kétféle üzemmód van: az *Idle Mode* és a *Power Down*. Az Idle Mode bekapcsolása esetén az utolsó utasítás végrehajtása befejeződik, az oszcillátor tovább mûködik, a megszakításvezérlő, a soros vonal, a portok és az időzítők megkapják az órajelet, csak a CPU áll le. A teljes CPU állapot megőrződik, minden regiszter és port megtartja az értékét. Az ALE és a $\overline{\text{PSEN}}$ jelek logikai magas szinten vannak. Az üzemmódnak két dolog vethet véget. Ha egy engedélyezett megszakítás bejut, akkor az IDL bitet törli a processzor. A megszakító rutin után normál módon folytatódik a végrehajtás. A GF0 és GF1 biteket fel lehet használni a megszakítás jelzésére, pl. az utasítás, amely bekapcsolja az Idle Mode-ot, beállíthatja a GF flageket, a megszakító rutin pedig felhasználhatja az értéküket. A második módszer az Idle Mode megszüntetésére a hardware reset. Az RST bemenetre adott jel aszinkron módon törli az IDL bitet, és a program végrehajtás ott folytatódik, ahol abbamaradt. Jegyezzük meg, hogy az Idle Mode bekapcsolását követő utasítás nem írhat portra vagy külső RAM-ba!

Power Down esetében leáll az oszcillátor, minden működés megszűnik, de a RAM és az SFR megtartja az értékét. Az ALE és a $\overline{\text{PSEN}}$ jelek logikai alacsony szintűek lesznek. A Power Down módot csak hardware resettel lehet megszüntetni. Ilyenkor az SFR-ben minden regiszter a kezdőértéket veszi fel, de a RAM tartalma nem változik. Power Down üzemmódban (csak a bekapcsolása után) a tápfeszültséget 2V-ra lehet csökkenteni. Mielőtt a normál módot visszállítanánk, a tápfeszültséget vissza kell állítani a normál értékre. A reset jelet csak a tápfeszültség visszaállítása után szabad aktivizálni, és aktív állapotban kell tartani, amíg az oszcillátor újra beindul és stabilizálódik (ez rendesen kevesebb, mint 10 msec).

Megszakítások

A 8051 öt megszakítást kezel: két külső megszakítást, a két időzítő megszakításait és a soros vonal megszakítását. A 8052-ben három időzítő van, így ott hat forrása lehet a megszakításnak. A külső megszakítások lehetnek él- vagy szintvezéreltek. Minden megszakítást jelez egy bit valamelyik regiszterben (pl. az RI bit az SCON regiszterben a soros vonal vételi oldalának megszakításkérését jelzi). Ezek a bitek a programból is beírhatók, ami ugyanúgy kiváltja a megszakítást, mintha azt a hardware kezdeményezte volna.

IE (Interrupt Enable Register)

EA	-	ET2	ES	ET1	EX1	ET0	EX0

EA	IE.7	Minden megszakítást engedélyez, letilt. Ha ez a bit 0, akkor egyetlen megszakítás sem jut érvényre. Ha ez a bit 1, akkor a megszakítások egyenként tilthatók, engedélyezhetők.
ET2	IE.5	Engedélyezi, vagy tiltja a Timer 2 megszakításait (csak 8052-nél).
ES	IE.4	A soros port megszakításainak engedélyezése, tiltása.
ET1	IE.3	Timer 1 megszakítás engedélyező bit.
EX1	IE.2	Külső megszakítás 1 engedélyező bit.
ET0	IE.1	Timer 0 engedélyező bit.
EX0	IE.0	Külső megszakítás 0 engedélyező bit.

Ha az IE regiszterben az adott engedélyező bitet 1-be állítjuk, akkor az a megszakítás engedélyezve lesz, ha 0-ba, akkor tiltva. A külső megszakítások lehetnek él- vagy szintvezéreltek, ezt a TCON regiszterben állíthatjuk be (lásd ott). A megszakítások prioritása is programozható. Két prioritási szint van: magas, és alacsony. Bármely megszakítás prioritási szintje beállítható az IP regiszterben.

IP (Interrupt Priority Register)

-	-	PT2	PS	PT1	PX1	PT0	PX0

PT2	IP.5	Timer 2 megszakításainak prioritása (csak 8052-nél).
PS	IP.4	A soros port megszakításainak prioritása.
PT1	IP.3	Timer 1 megszakítás prioritás.
PX1	IP.2	Külső megszakítás 1 prioritás.
PT0	IP.1	Timer 0 megszakítás prioritás.
PX0	IP.0	Külső megszakítás 0 prioritás.

Ha a megfelelő bitet 1-be állítjuk, akkor az adott megszakítás magas szitntű lesz, ha 0-ba, akkor, pedig alacsony szintű. Az alacsony szintű megszakításokat a magas szintű megszakítások megszakíthatják, míg a magas szintűeket semmi sem. Ha egyszerre érkezik a CPU-hoz egy magas és egy alacsony szintű megszakítás-kérés, akkor a magas szintűnek van elsőbbsége. Ha két egyforma szintű megszakítás-kérés érkezik, akkor a prioritási sorrend a következő:

	Forrás	**Prioritási szint**
1.	IE0	magasabb
2.	TF0	
3.	IE1	
4.	TF1	
5.	RI + TI	
6.	TF2 + EXF2	alacsonyabb

A rövidítések jelentése: IE **=** külső megszakítás (External Interrupt); TF **=** időzítő megszakítás (Timer Interrupt); RI **=** soros vonali megszakítás vétel esetén (Receive Interrupt); TI **=** soros vonali megszakítás egy karakter kiküldése után (Transmit Interrupt); EXF2 **=** 2-es időzítő külső beíró jele (csak a 8052 sorozatnál).

A megszakítások kezelése

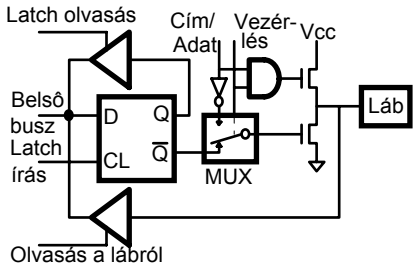
Ha egy külső megszakítást szintvezéreltre programoztunk, akkor a külső megszakításkérő jelet fenn kell tartani addig, amíg a CPU elkezdi a megszakítás végrehajtását, majd meg kell szüntetni, mielőtt a megszakító rutin befejeződik, mert különben újabb megszakítás keletkezik.

Amint egy megszakítás érvényre jut, a CPU befejezi az éppen végrehajtás alatt lévő utasítást, valamint ha ez az utasítás IRET, vagy olyan utasítás, ami befolyásolja az IE vagy IP regisztereket, akkor a *következő utasítást is*, majd automatikusan generál egy LCALL utasítást, vagyis a megszakítás forrásától függően meghív egy adott című szubrutint. Az LCALL utasítás elmenti a programszámlálót a verembe, de a PSW regisztert nem! Ezután a PC regiszter a következő értéket veszi fel:

Forrás	Cím
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI + TI	0023H
TF2 + EXF2	002BH

Ezt a módszert vektoros megszakításnak nevezik. Ezeken a címeken többnyire ugró utasításokat helyezünk el, amelyek a megfelelő szubrutinra adják a vezérlést. A megszakító szubrutinnak IRET utasítással kell végződnie, amely értesíti a processzort arról, hogy a megszakítás kezelése befejeződött. A CPU visszaállítja a programszámláló regiszter értékét, a program végrehajtása a megszakítás helyétől folytatódik. Ha ekkor is van érvényes megszakítás kérés, a processzor a főprogramból akkor is végrehajt legalább egy utasítást, mielőtt az újabb megszakítást elindítaná!

Be- és kimeneti portok



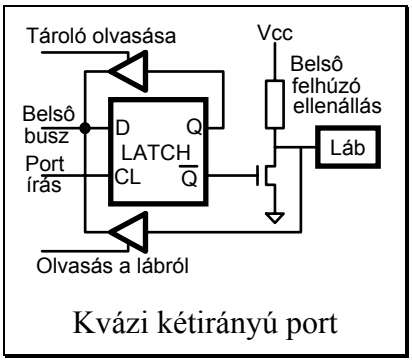
A Port 0 egy bitje

Mind a négy port kétirányú és output irányban tárolóval van ellátva. A tárolók az SFR területen érhetők el (P0, P1, P2, P3 regiszterek).

Ezen a rajzon a Port 0 egy bitjének belső felépítése látható. A Port 2 belső felépítése ettől csak annyiban tér el, hogy az ÉS kapu egy belső felhúzó ellenállást vezérel. A P0 és a P2 port a multiplexelt cím/adatbusz szerepét látja el, amennyiben külső memóriát használunk (bővebben lásd a Társzervezésnél).

A P3-as port, és a 8052 sorozatban a P1 port két bitje, szintén több funkcióval rendelkezik. Ezek a portok úgynevezett kvázi kétirányú portok. Ez azt jelenti, hogy ha a portra 0-t írunk ki, akkor az a kimenetet alacsony szintre kényszeríti. Ha azonban 1-et, akkor csupán a belső felhúzó ellenállás tartja a portot magas szinten, így az bemenetként is használható, hiszen kívülről alacsony szintre húzható.

Ebből következik, hogy ha egy portbitet bemenetként akarunk használni, akkor, mint kimenetet 1-be kell állítani. Ugyanez a helyzet az alternatív funkciójú bitekkel is. Amelyiket használjuk a programban, azt a bitet a porton állítsuk 1-be! A bitek és a funkciójuk:



Kvázi kétirányú port

Port, bit	**Másodlagos funkció**
P1.0	T2: Timer 2 külső bemenete.
P1.1	T2EX: Timer 2 mintavevő (trigger) bemenete.
P3.0	RXD: soros vonal bemenet.
P3.1	TXD: soros vonal kimenet.
P3.2	$\overline{\text{INT0}}$: külső megszakítás.
P3.3	$\overline{\text{INT1}}$: külső megszakítás.
P3.4	T0: Timer 0 bemenet.
P3.5	T1: Timer 1 bemenet.
P3.6	$\overline{\text{WR}}$: külső memória írás jel.
P3.7	$\overline{\text{RD}}$: külső memória olvasás jel.

Az 1, 2 és 3 portok 4 LS TTL bemenetet tudnak működtetni. A Port 0 külső busz üzemmódban 8 LS TTL bemenet meghajtására képes, portként használva külső felhúzó ellenállásokat igényel, ha bemeneteket akarunk vele működtetni.

Néhány utasítás az adatokat a tárolóról olvassa, néhány pedig a lábról. Azok az utasítások amelyek a portról leolvasott értéket visszaírják a portra, mindig a tárolót olvassák. Ezek az úgynevezett olvas-módosít-ír utasítások a következők: ANL; ORL; XRL; JBC; CPL; INC; DEC; DJNZ; MOV PX.Y, C; CLR PX.Y; SETB PX.Y.

Időzítők, számlálók

A 8052-es típusnak három, a többinek két 16 bites időzítő/számláló egysége van, a Timer 0 és a Timer 1. Mindegyikük időzítőként vagy eseményszámlálóként használható.

Időzítő üzemmódban a regiszterek tartalma minden gépi ciklus során eggyel nő. Mivel a gépi ciklus 12 órajel-periódusból áll, a számlálási sebesség az órajel-frekvencia 1/12-e.

Számláló üzemmódban a regiszterek tartalma a megfelelő bemeneten megjelenő lefutó él hatására eggyel nő. Mivel az átmenet érzékelése két gépi ciklus idejéig tart, ezért a maximális számlálási frekvencia az órajel frekvenciájának 1/24-e. A Timer 0-nak négy üzemmódja van, a Timer 1-nek és a Timer 2-nek (8052) három.

TMOD (Timer/Counter Mode Control Register)

GATE	C/$\overline{\text{T}}$	M1	M0	GATE	C/$\overline{\text{T}}$	M1	M0
Timer 1				Timer 0			

GATE Kapuzás. Ha ez a bit 1-es, akkor a számláló ki- illetve bekapcsolása az $\overline{\text{INTx}}$ bemenetre adott jellel végezhető (ha az $\overline{\text{INTx}}$ magas, akkor a számláló mûködik). Ha ez a bit 0, akkor a számláló a TCON regiszterben lévő TR bittel kapcsolható be, vagy ki.

C/$\overline{\text{T}}$ Számláló, vagy időzítő üzemmód választás: 1 **=** számláló, 0 **=** időzítő.

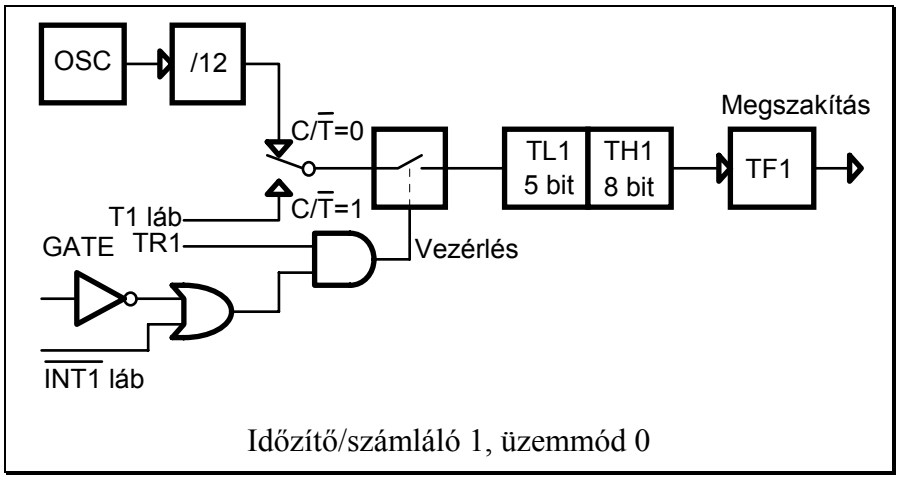
M1	**M0**	**Üzemmód**
0	0	A TH 8 bites időzítő/számláló, a TL 5 bites előosztóként mûködik.
0	1	A TH és a TL 16 bites számláló/időzítőt alkot, előosztó nincs.
1	0	A TL 8 bites időzítő/számlálóként mûködik, túlcsorduláskor automatikusan újratöltődik a TH-ból.
1	1	(Timer 0) TL0 8 bites időzítő/számláló a Timer 0 vezérlő bitjeivel normál módon vezérelve, a TH0 pedig 8 bites időzítő a Timer 1 vezérlő bitjeivel vezérelve.
1	1	(Timer 1) A számláló leáll, tartja az értékét.

TCON (Timer/Counter Control Register)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1	TCON.7	Timer 1 túlcsordulás jelző bit. A hardware 1-esbe állítja amikor a számláló túlcsordul. A hardware törli, amikor a megszakítás elkezdődik.
TR1	TCON.6	Timer 1 ki- ill. bekapcsolás. Beállításával illetve törlésével az időzítő/számláló elindítható illetve leállítható.
TF0	TCON.5	Timer 0 túlcsordulás jelző bit. A hardware 1-esbe állítja amikor a számláló túlcsordul. A hardware törli, amikor a megszakítás elkezdődik.
TR0	TCON.4	Timer 0 ki- ill. bekapcsolás. Beállításával illetve törlésével az időzítő/számláló elindítható illetve leállítható.
IE1	TCON.3	Megszakítás 1 él jelző. A hardware beállítja, amikor a külső megszakítás-kérés bemeneten detektált egy élt. A megszakítás végrehajtásakor törlődik.
IT1	TCON.2	Megszakítás 1 típus. Ha beállítjuk, akkor az $\overline{\text{INT1}}$ bemeneten megjelenő le-futó él fog megszakítást kiváltani. Ha töröljük, akkor az alacsony szint fog megszakítást generálni.
IE0	TCON.1	Megszakítás 0 él jelző. A hardware beállítja, amikor a külső megszakítás-kérés bemeneten detektált egy élt. A megszakítás végrehajtásakor törlődik.
IT0	TCON.0	Megszakítás 0 típus. Ha beállítjuk, akkor az $\overline{\text{INT0}}$ bemeneten megjelenő lefutó él fog megszakítást kiváltani. Ha töröljük, akkor az alacsony szint fog megszakítást generálni.

0-ás üzemmód



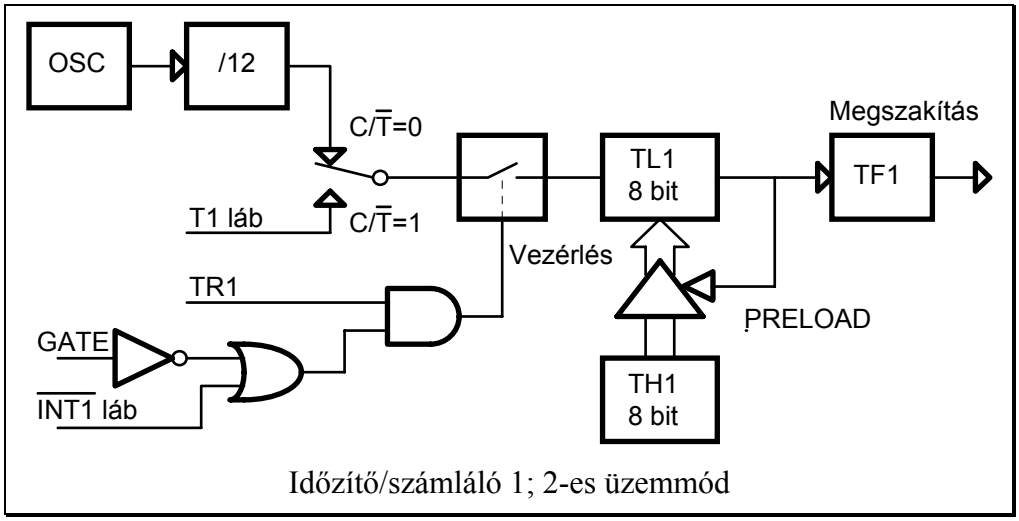
Időzítő/számláló 1, üzemmód 0

Ebben az üzemmódban a számláló 13 bites. Így kompatíbilis a 8048-as típussal. A TH regiszter alkotja a számláló alsó 8 bitjét, a TL regiszter alsó 5 bitje pedig előosztóként mûködik. A TL regiszter felső 3 bitje nem definiált érték, figyelmen kívül kell hagyni.

1-es üzemmód

Ez az üzemmód csupán annyiban különbözik a 0-ás üzemmódtól, hogy a TH, TL regiszterek 16 bites számlálót alkotnak.

2-es üzemmód

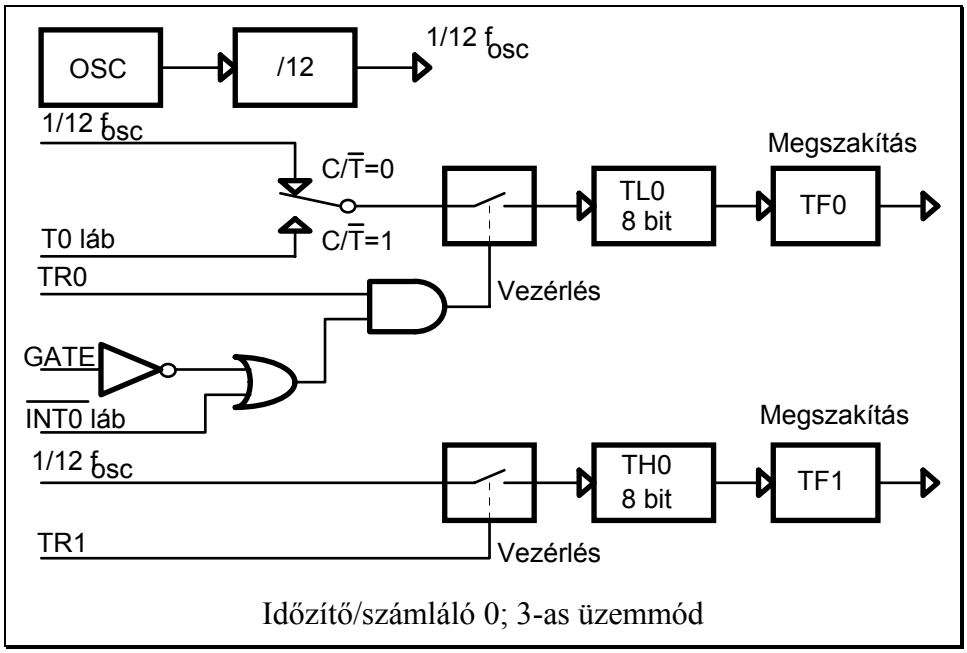


Időzítő/számláló 1; 2-es üzemmód

Ebben az üzemmódban a TL számláló 8 bites számlálót alkot, amely túlcsordulásakor automatikusan újratöltődik, méghozzá a TH regiszterből.

A számláló kezdőértékét, amelytől az előreszámol majd 255-ig, be kell tölteni a TH regiszterbe. Az érték áttöltése a TL regiszterbe nem változtatja meg a TH regisztert.

3-as üzemmód



Időzítő/számláló 0; 3-as üzemmód

Ebben az üzemmódban csak a Timer 0 használható. Ekkor a Timer 0 két egységre bomlik. A TL0 regiszter 8 bites időzítőként vagy számlálóként használható a megszokott módon, tehát a T0 és $\overline{\text{INT0}}$ bemenetek és a GATE, TR0, TF0 vezérlőbitek segítségével. A TH0 regiszter 8 bites időzítőként mûködik és a TR1, TF1 bitekkel (vagyis a Timer 1 vezérlőbitjeivel) működtethető. Mivel a Timer 0 lefoglalja a TR1-et, ezért a Timer 1-et a 3-as üzemmódba való átkapcsolással kapcsolhatjuk ki (lásd a TMOD regisztert). Mivel a TF1 bit is foglalt, ezért a Timer 1 csak olyan esetekben használható, amikor nincs szükség arra, hogy az megszakítást váltson ki (pl. ha baud rate generátorként használjuk a soros vonalhoz).

Timer 2 (8052)

A 8052 sorozatban 3 időzítő/számláló van. A Timer 2 lehet időzítő, vagy számláló és három üzemmódban működhet: "capture", "auto-load" és baud rate generátor üzemmódban. Ehhez az időzítőhöz egy újabb vezérlőregiszter tartozik.

T2CON (Timer/Counter 2 Control Register)

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{\text{T2}}$	CP/$\overline{\text{RL2}}$

TF2	T2CON.7	Timer 2 túlcsordulás jelző bit. 1-esbe állítódik amikor a számláló túlcsordul. A programnak kell törölnie. A TF2 nem állítódik be, ha RCLK=1 vagy TCLK=1.
EXF2	T2CON.6	A külső bemeneten (T2EX) megjelenő lefutó élt detektáló bit, ha EXEN2=1. Ha a Timer 2 megszakítás engedélyezve van, akkor az EXF2=1 megszakítást vált ki. Az EXF2-t a programnak kell törölnie.

RCLK	T2CON.5	Vevő órajel vezérlés. Ha ez a bit 1-es, akkor a soros vonal vételi oldala a Timer 2 túlcsordulásait használja órajelként az 1-es és a 3-as üzemmódokban. Ha ez a bit 0, akkor a soros vonal vételi oldala számára a Timer 1 állítja elő az órajelet.
TCLK	T2CON.4	Adó órajel vezérlés. Ha ez a bit 1-es, akkor a soros vonal adó oldala a Timer 2 túlcsordulásait használja órajelként az 1-es és a 3-as üzemmódokban. Ha ez a bit 0, akkor a soros vonal adó oldala számára a Timer 1 állítja elő az órajelet.
EXEN2	T2CON.3	Külső bemenet engedélyezés. Amennyiben a Timer 2 a soros vonalat működteti, vagy ez a bit 0, a T2EX bemeneten megjelenő lefutó élt a számláló figyelmen kívül hagyja.
TR2	T2CON.2	A Timer 2 ki- illetve bekapcsolását végző bit. 0-kikapcsolás; 1-bekapcsolás.
C/ $\overline{T2}$	T2CON.1	Időzítő vagy számláló üzemmód kiválasztása: 0-belső órajellel működtetett időzítő (OSC/12); 1-külső jelet (T2) számol (lefutó él).
CP/ $\overline{RL2}$	T2CON.0	"Capture" vagy "reload" mód kiválasztása. Ha ez a bit 1-es, akkor a T2EX bemeneten megjelenő lefutó él a számláló tartalmát átírja egy átmeneti regiszterbe (RCAP2L és RCAP2H). Ez a "Capture" mód. Ha ez a bit 0, akkor a számláló minden túlcsorduláskor vagy a T2EX bemeneten megjelenő lefutó él hatására (ha az EXEN2=1) az RCAP2L és RCAP2H regiszterekben lévő értéket veszi fel. Ez a "reload" mód. Ha az RCLK vagy TCLK bit 1-es, akkor ezt a bitet a CPU figyelmen kívül hagyja, és a Timer 2 "reload" módban fog működni.

Soros port

A duplex, azaz egyidőben adni és venni képes soros port nagymértékben megkönnyíti a mikroszámítógép és a környezet közötti kommunikációt. A portnak több üzemmódja van, amelyek a megfelelő regiszter programozásával lehetővé teszik az adatátviteli sebesség beállítását (az órajel-frekvencia 1/12, 1/32 vagy 1/64-e, változtatható) és az adatátviteli formátum (csak 8 adatbit; 1 start-, 8 adat- és 1 stopbit; 1 start-, 9 adat- és 1 stopbit) megválasztását.

SCON (Serial Port Control Register)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Az SM0 és az SM1 bitek határozzák meg az üzemmódot:

SM0	**SM1**	**Mód**	**Leírás**	**Baud Rate**
0	0	0	Shift regiszter	fosc/12
0	1	1	8 bit UART	változtatható
1	0	2	9 bit UART	fosc/32 vagy fosc/64
1	1	3	9 bit UART	változtatható

SM2 A 2-es és a 3-as üzemmódokban engedélyezi a multiprocesszor kommunikáció támogatását. A 0-ás üzemmódnál 0-ba kell állítani! Ha ez a bit 1-es, akkor vétel esetén csak akkor keletkezik megszakításkérés, ha a 9. vett bit (1-es módban a stopbit) 1-es értékű.

REN Vétel engedélyezés. Ha ez a bit 1-es, a vevő működése engedélyezve van.

TB8 Ez a 9-edik adatbit, amelyet az adó a 2-es és a 3-as üzemmódokban lead. Értéke szükség szerint állítható.

RB8 A 2-es és a 3-as üzemmódokban a 9-edik vett adatbit értékét tartalmazza. 1-es üzemmódban a stopbitet tartalmazza, ha az SM2 bit 0. 0-ás üzemmódban ez a bit nem használatos.

TI Adás megszakítás jelző bit. A hardware állítja be, de törölni programból kell.

RI Vétel megszakítás jelző bit. Vett adat esetén a hardware állítja be, de törölni a programból kell!

Sebesség

A sebesség mód0 esetén rögzített: az oszcillátor frekvencia 12-ed része.

Mód 2 esetén a sebességet az SMOD bit határozza meg (lásd a PCON regisztert). Ha az SMOD bit 1, akkor a baud rate az oszcillátor frekvencia 32-ed része, ha pedig 0, akkor az oszcillátor frekvencia 64-ed része.

Mód 1 és 3 esetén a 8051 sorozatnál a sebességet a Timer 1 határozza meg. A Timer 1 túlcsordulásaiból adódó frekvenciát a processzor elosztja 32-vel, így áll elő a soros vonal baud rate-je. Ekkor a Timer 1-et 8 bites időzítő üzemmódba állítjuk, automatikus újra feltöltéssel, a megszakítást pedig letiltjuk (a TMOD regiszter felső 4 bitje: 0010). Ekkor a sebességet a következő formulával számíthatjuk:

$$Mod1{,}3 Baud\,Rate = \frac{2^{SMOD}}{32} \times \frac{Oszcillator\ frekvencia}{12 \times \left[256 - (TH1)\right]}$$

Látható, hogy ezekben az üzemmódokban is megduplázhatjuk a soros vonal sebességét az SMOD bit (TCON regiszter) 1-be állításával. Ha nagyon kis sebességre van szükségünk, akkor programozzuk a Timer 1-et, hogy az 16 bites időzítőként működjön (TMOD regiszter felső négy bitje: 0001), engedélyezzük a megszakítást, és a számlálót a megszakító rutinban töltsük fel a kívánt értékkel.

A 8052 sorozat esetében a Timer 2-t is használhatjuk vagy az adó, vagy a vevő, vagy mindkettő sebességének beállítására (lásd a T2CON regisztert). Ha az RCLK vagy a TCLK bitet 1-be állítjuk, akkor a processzor a Timer 2 túlcsordulásainak frekvenciáját használja a soros vonalhoz (16-al elosztva!). Ekkor többnyire az időzítő üzemmódot programozzuk (a C/T2 bit 0). A Timer 2 baud rate generátor üzemmódban nem az oszcillátor frekvencia 1/12-éd, hanem 1/2-ét számolja! A sebességet tehát a következő képlet adja:

$$Mod1{,}3 Baud\,Rate = \frac{Oszcillator\ frekvencia}{32 \times \left[65536 - (RCAPH, RCAPL)\right]}$$

Itt a RCAPH és az RCAPL regiszterek tartalma 16 bites előjel nélküli egész számként értendő.

Mód 0

Ebben az üzemmódban a továbbított bitek száma 8. A vétel és az adás is az RXD kivezetésen keresztül zajlik, a TXD kivezetésen az órajel jelenik meg. Az adás az adatnak az SBUF regiszterbe történő beírásával indítható el. Az adatok bitről-bitre kisorjáznak a P3 port 0. bitjén (RXD), az 1. biten (TXD) a soros vonali órajel jelenik meg. Ha az egész byte kiment, akkor a processzor leállítja az adást és bebillenti a TI megszakításkérő bitet.

A vétel engedélyezésének a feltétele: REN=1 és RI=0. Vétel esetén csakúgy mint az adásnál, a P3.1 kivezetésen megjelenik a soros vonal órajele. Az adatbiteket a P3.0 kivezetésre kell kapcsolni. A CPU beolvas nyolc bitet, az adatot átírja az SBUF regiszterbe, leállítja a vétel üzemmódot és bebillenti az RI bitet.

Mód 1

Ebben az üzemmódban a CPU 10 bitet továbbít, illetve fogad. Egy startbitet (ez 0 értékű), nyolc adatbitet és egy stopbitet (ez 1-es értékű). Az adatok továbbítása a TXD kivezetésen keresztül történik, a vétel az RXD kivezetésen keresztül.

Az adás az SBUF regiszterbe való írással indítható el. Amint az adat elküldése befejeződött, a TI bit 1 lesz.

A vételt az RXD kivezetésen megjelenő startbit (lefutó él) indítja el. Az adatok vétele után a CPU beállítja az SBUF regisztert, az RB8 és az RI bitet. Mindezt azonban csak akkor teszi meg, ha

1. RI=0, és
2. Vagy SM2=0, vagy a vett stop bit=1.

Ha az előbbi feltételek közül valamelyik nem teljesül, akkor a vett adat elvész és a vételi áramkör az újabb adatra várakozik. Ha a feltételek teljesülnek, akkor az adat az SBUF regiszterbe kerül, a stopbit az RB8 bitbe és az RI bit 1-es lesz.

Mód 2, 3

Ezek az üzemmódok megegyeznek, csupán az órajel forrása eltérő. Mód 2 esetén az órajel az oszcillátor frekvencia 1/32-ed, vagy 1/64-ed része lehet (lásd az SCON regisztert). Mód 3 esetén az órajelet az időzítők segítségével határozhatjuk meg. A CPU az adatokat a TXD kivezetésen továbbítja, a vétel az RXD kivezetésen át zajlik. A továbbított bitek száma 11. Egy startbit, nyolc adatbit, egy 9. programozható adatbit és egy stopbit. Adás esetén a CPU a TB8 bitet továbbítja kilencedikként, vételkor a 9. bit az SCON regiszter RB8 bitjébe kerül.

Az adás és a vétel tulajdonképpen ugyanúgy zajlik, mint az 1-es módban, csak a bitek száma nagyobb eggyel. Ha egy adat vétele befejeződött és teljesülnek a következő feltételek:

1. RI=0 és
2. Vagy SM2=0, vagy a kilencedik adatbit=1,

akkor a vett adatbyte az SBUF regiszterbe kerül, a 9. vett adatbit az RB8 bitbe, az RI bit pedig 1-es lesz. Egyébként a vett adat elvész.

Ezeknek az üzemmódoknak tehát az a lényege, hogy egy plusz adatbit továbbítására van lehetőség. Ezt felhasználhatjuk paritásbitként, vagy megvalósíthatunk több készülék közötti kommunikációt is, ha kihasználjuk azt, hogy SM2=1 esetén csak akkor keletkezik vételi megszakítás, ha a 9. adatbit (ami az RB8 bitbe került) 1-es értékű.

A fő készüléknek az adatok továbbítása előtt ki kell jelölnie azt az alkészüléket, amelyhez az adatokat szeretné elküldeni, vagy amelyiktől adatokat vár. Ezt címzéssel lehet elérni. A fő készülék a közös vonalon először egy címet küld el. A címben a 9. bit 1-es értékű, így minden alkészülékben megszakítás keletkezik, vagyis a címet mindenki veszi. A megcímzett készülék felkészül az adattovábbításra, vagyis az SM2D bitet 0-ba állítja. Ezután a 9. bit értékétől függetlenül minden vett byte megszakítást fog kiváltani. Az adatokban a 9. bit 0, így csak a kiválasztott készülék veszi azokat.

ONCE™ mód

A ONCE™ jelentése "on-circuit emulation". Ez egy speciális üzemmód, amely lehetővé teszi az emulációt anélkül, hogy a CPU-t el kellene távolítani az áramkörből. Ezt az üzemmódot a következőképpen kapcsolhatjuk be:

1. Kapcsoljunk alacsony szintet az ALE kivezetésre amíg a CPU reset állapotban van és a $\overline{\text{PSEN}}$ kimenet magas szintű.
2. Tartsuk alacsony szinten az ALE kivezetést, amíg az RST bemenet inaktívvá válik.

Ebben az üzemmódban a Port 0 kivezetései lebegnek, az ALE és a $\overline{\text{PSEN}}$ kivezetéseket a CPU felhúzó ellenállásokkal magas szinten tartja. Az oszcillátor tovább mûködik. Amíg a processzor ebben az üzemmódban van, az emulátor, vagy a teszt CPU működtetheti az áramkört. A normál üzemmódot normál reset-tel lehet visszaállítani.

Címzési módok.

Azokban az utasításokban, amelyekben 16 bites adat, vagy cím szerepel, az utasítás kódját az adat magasabb helyiértékű byte-ja követi, majd az alacsonyabb helyiértékű byte következik.

Direkt címzés. Az utasításban egy 8 bites címet adunk meg. Ezzel a módszerrel csak a beépített RAM és az SFR címezhető meg.

Indirekt címzés. A címet egy regiszter tartalmazza. Ha a cím 8 bites, a regiszter az R0 vagy az R1 lehet. Ha a cím 16 bites, a regiszter csak a 16 bites DPTR lehet. Ez a címzési mód a belső és a külső RAM címzésére is használható, de a belső RAM esetében csak 8 bites indirekt címet használhatunk.

Regiszter címzés. Az aktív regiszterkészlet 8 regisztere közül (R0-R7) az egyik.

Közvetlen címzés. Az utasítás tartalmazza azt a konstanst amely a művelet egyik operandusa.

Bit címzés. Az SFR terület és a belső adatmemória-terület 20H-2FH-ig terjedő részének bitenkénti közvetlen címzése.

Bennfoglalt (implicit) címzés. Az utasítás címzés nélkül is azonosítja a művelet operandusát: pl. CLR C.

Indexelt címzés. A cím egy 16 bites regiszter és az akkumulátor tartalmának összege. A 16 bites regisztert bázisregiszternek nevezzük, ez a DPTR vagy a PC lehet. Ezzel a címzési móddal csak programmemória címezhető meg, és csak olvasni lehet. Ezzel a módszerrel kezelhetjük a programmemóriában tárolt táblázatokat.

Relatív címzés. Vezérlésátadó utasításokban használatos. Az utasításban egy 1 byte-os relatív (eltolási) értéket adunk meg. Ezt a számot a processzor előjeles kettes komplemens számnak tekinti és hozzáadja a programszámláló értékéhez (amely ilyenkor a következő utasítás címét tartalmazza).

Utasítások hatása a flag-bitekre

Utasítás	Jelzőbit			Utasítás	Jelzőbit		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,/bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

Az utasításkészlet leírásánál használt jelölések

Rn Az aktuális regiszterkészlet (R0-R7) egy regisztere, ahol n a regiszter száma.

direct A belső RAM (0-7FH) vagy az SFR (80H-0FFH) 8 bites címe.

@Ri A belső RAM egy byte-ja. A cím az R0 vagy az R1 regiszterben van.

#data 8 bites konstans.

#data16 16 bites konstans.

addr16 16 bites cím, amelyet az utasítás tartalmaz. Az LCALL és a LJMP utasításokban fordul elő.

addr11 11 bites cím. Az ACALL és az AJMP utasításokban használt.

rel Előjeles (kettes komplemens ábrázolású) 8 bites eltolási cím. Értéke -128 - +127 lehet. Ugrások célcímének megadáskor használt, értéke hozzáadódik a következő utasítás címéhez.

bit Közvetlen címzésű bit a belső RAM-ban vagy az SFR-ben.

byte Valamilyen címzési móddal megcímzett 8 bites érték.

src-byte Forráshely byte

dest-byte Célhely byte

Az MCS 51 család utasításkészlete

Aritmetikai utasítások

ADD A,<src-byte>

Az akkumulátor tartalmához hozzáadja a címzéssel kijelölt operandus tartalmát, az eredmény az akkumulátorban képződik. Az összeadáskor a CY jelzőbitet nem veszi figyelembe.

```
ADD A,Rn
ADD A,direct
ADD A,@Ri
ADD A,#data
```

ADDC A,<src-byte>

Az akkumulátor tartalmához hozzáadja a címzéssel kijelölt operandus tartalmát és a CY flag értékét. Az eredmény az akkumulátorban keletkezik.

```
ADDC A,Rn
ADDC A,direct
ADDC A,@Ri
ADDC A,#data
```

SUBB A,<src-byte>

Az akkumulátor tartalmából levonja a címzéssel kijelölt operandus tartalmát és a CY flag értékét. Az eredmény az akkumulátorban keletkezik.

```
SUBB A,Rn
```

SUBB A,direct
SUBB A,@Ri
SUBB A,#data

INC <byte>

A címben kijelölt operandus értékét eggyel megnöveli.

INC A
INC Rn
INC direct
INC @Ri

INC DPTR

A 16-bites adatterület-mutató értékét eggyel megnöveli.

DEC <byte>

A címben kijelölt operandus értékét eggyel csökkenti.

DEC A
DEC Rn
DEC direct
DEC @Ri

MUL AB

Előjel nélküli 8 bites egészek szorzása. A két operandust az A illetve a B regiszterben kell elhelyezni. Az eredmény 16 bites lesz, alsó byte-ja az akkumulátorban, felső byte-ja a B regiszterben keletkezik. Ha az eredmény nagyobb mint 255, akkor az OV flag 1 lesz.

DIV AB

Előjel nélküli 8 bites egészek osztása. Az osztandót az A, az osztót a B regiszter tartalmazza. A hányados az akkumulátorban, a maradék a B regiszterben keletkezik. Nullával való osztás esetén az OV flag 1 lesz.

DA A

Az akkumulátor tartalmának decimális korrekciója. A korrekció algoritmusa a CY és az AC flag-ek értékét figyelembe véve dolgozik. Használata csak összeadási műveletek után van értelmezve.

Logikai utasítások

ANL <dest-byte>,<src-byte>

Bitenkénti logikai ÉS kapcsolat a forrás és a cél operandusai között. A művelet eredménye a célhelyre kerül.

ANL A,Rn
ANL A,direct
ANL A,@Ri
ANL A,#data
ANL direct,A
ANL direct,#data

ORL <dest-byte>,<src-byte>

Bitenkénti logikai VAGY kapcsolat a forrás és a cél operandusai között. A művelet eredménye a cél-helyre kerül.

ORL A,Rn
ORL A,direct
ORL A,@Ri
ORL A,#data
ORL direct,A
ORL direct,#data

XRL <dest-byte>,<src-byte>

Bitenkénti logikai KIZARÓ-VAGY kapcsolat a forrás és a cél operandusai között. Az eredmény a cél-helyre kerül.

XRL A,Rn
XRL A,direct
XRL A,@Ri
XRL A,#data
XRL direct,A
XRL direct,#data

CLR A

Az akkumulátor tartalmának törlése (az összes bit "0"-ba állítása).

CPL A

Az akkumulátor tartalmának logikai komplementálása (egyes komplemens képzés, invertálás).

RL A

Az akkumulátor tartalmának balra forgatása. A kilépő 7. bit a 0. bit pozícióba kerül és minden bit egy helyi értékkel balra lép. A forgatás a CY-t nem érinti.

RLC A

Az akkumulátor tartalmának balra forgatása a CY bevonásával. Így tehát a műveletben kilenc bit vesz részt. A 7. bit bekerül a CY-be, a CY pedig belép a 0. bitpozícióba.

RR A

Az akkumulátor tartalmának jobbra forgatása a CY bit bevonása nélkül. A kilépő 0. bit a 7. bitpozícióba kerül és minden bit egy helyi értékkel jobbra lép. A forgatás a CY-t nem érinti.

RRC A

Az akkumulátor tartalmának jobbra forgatása a CY bit bevonásával. A műveletben kilenc bit vesz részt. A 0. bit kilép a CY-be, a CY belép a 7. bitbe, és minden bit egy helyi értékkel jobbra lép.

SWAP A

Az akkumulátor alsó és felső 4 bitjének (tetrád) felcserélése.

Adatmozgató utasítások:

MOV <dest-byte>,<src-byte>

A cím által kijelölt forrás byte tartalmának átmásolása a cím által kijelölt cél helyre. A forrás hely tartalma változatlan marad.

```
MOV A,Rn
MOV A,direct
MOV A,@Ri
MOV A,#data
MOV Rn,A
MOV Rn,direct
MOV Rn,#data
MOV direct,A
MOV direct,Rn
MOV direct,direct
MOV direct,@Ri
MOV direct,#data
MOV @Ri,A
MOV @Ri,direct
MOV @Ri,#data
```

MOV DPTR,#data16

Az adatterület-mutató feltöltése az utasításban lévő 16 bites értékkel. A 16 bites érték felső byte, alsó byte sorrendben következik az utasítás kódja után.

MOVC A,@A+<base-reg>

A bázisregiszter és az akkumulátor (indexregiszter) tartalmának összegéből képzett címmel kijelölt programmemória tartalmának beolvasása az akkumulátorba. Az operandus címének kiszámításakor a programszámláló (PC) a következő utasítás címét tartalmazza.

```
MOVC A,@A+DPTR
MOVC A,@A+PC
```

MOVX <dest-byte>,<src-byte>

Adatmozgatás a külső adatmemória és az akkumulátor között. Az egyik operandus mindig az akkumulátor. 8 és 16 bites címzést is használhatunk.

```
MOVX A,@Ri
MOVX A,@DPTR
MOVX @Ri,A
MOVX @DPTR,A
```

PUSH direct

Verembe írás. A verem-mutató értéke eggyel nő, majd a verem-mutató által megcímzett helyre íródik az operandus tartalma. A verem mindig a belső RAM-ban van.

POP direct

Olvasás a veremből. A verem-mutató által megcímzett belső RAM tartalma utasításban kijelölt helyre kerül, majd a verem-mutató értéke 1-el csökken.

XCH A,<byte>

A megcímzett memória és az akkumulátor tartalma kicserélődik. A művelet egyik operandusa mindig az akkumulátor.

XCH A,Rn
XCH A,direct
XCH A,@Ri

XCHD A,@Ri

Az akkumulátor alsó 4 bitjének a felcserélése a megcímzett belső RAM alsó 4 bitjével.

Boole algebrai műveletek

CLR C

A CY bit törlése ("0"-ba állítása).

CLR bit

A megcímzett bit törlése

SETB C

A CY bit "1"-be állítása.

SETB bit

A megcímzett bit "1"-be állítása.

CPL C

A CY bit komplementálása.

CPL bit

A megcímzett bit komplementálása.

ANL C,<src-bit>

A forrásbit és a CY logikai ÉS kapcsolatának eredménye a CY bitbe kerül. Az ÉS kapcsolat létrehozható az megadott bit ponált és negált értékével is.

ANL C,bit
ANL C,/bit

ORL C,<src-bit>

A forrásbit és a CY logikai VAGY kapcsolatának eredménye a CY bitbe kerül. A művelet a megadott bit ponált és negált értékével egyaránt elvégezhető.

ORL C,bit
ORL C,/bit

MOV <dest-bit>,<src-bit>

A forrásbit átmásolása a célbitbe. Az egyik bit mindig a CY.

MOV C,bit

MOV bit,C

Vezérlésátadó utasítások

JC rel

Feltételes ugrás. Ugrás, ha a CY bit 1-es, egyébként a program végrehajtása a következő utasítással folytatódik.

JNC rel

Feltételes ugrás. Ugrás, ha a CY bit 0, egyébként a program végrehajtása a következő utasítással folytatódik.

JZ rel

Feltételes ugrás. Ugrás, ha az akkumulátor 0, egyébként a program végrehajtása a következő utasítással folytatódik.

JNZ rel

Feltételes ugrás. Ugrás, ha az akkumulátor nem 0, egyébként a program végrehajtása a következő utasítással folytatódik.

JB bit,rel

Feltételes ugrás. Ugrás, ha a megadott bit 1-es, egyébként a program végrehajtása a következő utasítással folytatódik.

JNB bit,rel

Feltételes ugrás. Ugrás, ha a megadott bit 0, egyébként a program végrehajtása a következő utasítással folytatódik.

JBC bit,rel

Feltételes ugrás és bit törlés. Ugrás és a megadott bit törlése, ha az 1-es, egyébként a program végrehajtása a következő utasítással folytatódik.

CJNE <dest-byte>,<src-byte>,rel

Feltételes ugrás. Ugrás, ha az első két operandus tartalma nem egyforma, egyébként a végrehajtás a következő utasítással folytatódik.

CJNE A,direct,rel
CJNE A,#data,rel
CJNE Rn,#data,rel
CJNE @Ri,#data,rel

DJNZ <byte>,rel

Feltételes ugrás. Az utasítás először csökkenti eggyel a kijelölt byte-ot, és ha az nem nulla, akkor program futása a megadott relatív címen folytatódik. Ez az utasítás a jelzőbitekre nincs hatással.

DJNZ Rn,rel
DJNZ direct,rel

AJMP addr11

Feltétel nélküli, abszolút ugrás. Az utasításban megadott 11 bit bemásolódik a programszámláló alsó 11 bitjébe (a PC a következő utasítás címét tartalmazza), a többi bit nem változik. Ezzel az ugrással tehát csak azon a 2 Kbyte-os blokkon belülre lehet ugrani, amelyikben a következő utasítás van. Használata azért előnyös, mert csupán 2 byte helyet igényel a memóriában.

LJMP addr16

Feltétel nélküli, "hosszú" ugrás az utasításban megadott címre.

SJMP rel

Feltétel nélküli, "rövid" ugrás. A végrehajtás az utasításban megadott relatív címen folytatódik.

JMP @A+DPTR

Feltétel nélküli, indirekt ugrás. A következő utasítás címe a DPTR regiszter és az akkumulátor tartalmának (8 bites előjel nélküli szám) összege lesz.

ACALL addr11

Abszolút szubrutinhívás. A processzor a következő utasítás címét (a programszámlálót) elmenti a verembe, majd az utasításban megadott 11 bitet bemásolja a programszámláló alsó 11 bitjébe (a PC a következő utasítás címét tartalmazza), a többi bit nem változik. Ezzel a hívással tehát csak azon a 2 Kbyte-os blokkon belüli eljárás hívható, amelyikben a következő utasítás van. Használata azért előnyös, mert csupán 2 byte helyet igényel a memóriában.

LCALL addr16

"Hosszú" szubrutinhívás. A CPU elmenti a verembe a következő utasítás címét (a program-számlálót) majd elugrik a megadott címre.

RET

Visszatérés szubrutinból. A processzor a veremből beolvas két byte-ot, és a programszámlálóba tölti, vagyis a végrehajtás az elmentett címen folytatódik (szubrutinhívást követő utasítás).

RETI

Visszatérés megszakításból. Ugyanúgy mûködik, mint a RET utasítás, de a megszakítás-kezelő logikát is vezérli.

NOP

Üres utasítás, nem csinál semmit. Késleltetésre, vagy helykitöltésre használható.

Melléklet

Az utasítások kódja, mérete és végrehajtási ideje

Kód hex	Hossz byte	Uta-sítás	Ope-randus	Idő	Kód hex	Hossz byte	Uta-sítás	Ope-randus	Idő
00	1	NOP		12	30	3	JNB	bit addr,code addr	12
01	2	AJMP	code addr	24	31	2	ACALL	code addr	24
02	3	LJMP	code addr	24	32	1	RETI		24
03	1	RR	A	12	33	1	RLC	A	12
04	1	INC	A	12	34	2	ADDC	A,#data	12
05	2	INC	data addr	12	35	2	ADDC	A,data addr	12
06	1	INC	@R0	12	36	1	ADDC	A,@R0	12
07	1	INC	@R1	12	37	1	ADDC	A,@R1	12
08	1	INC	R0	12	38	1	ADDC	A,R0	12
09	1	INC	R1	12	39	1	ADDC	A,R1	12
0A	1	INC	R2	12	3A	1	ADDC	A,R2	12
0B	1	INC	R3	12	3B	1	ADDC	A,R3	12
0C	1	INC	R4	12	3C	1	ADDC	A,R4	12
0D	1	INC	R5	12	3D	1	ADDC	A,R5	12
0E	1	INC	R6	12	3E	1	ADDC	A,R6	12
0F	1	INC	R7	12	3F	1	ADDC	A,R7	12
10	3	JBC	bit addr,code addr	12	40	2	JC	code addr	24
11	2	ACALL	code addr	24	41	2	AJMP	code addr	24
12	3	LCALL	code addr	24	42	2	ORL	data addr,A	12
13	1	RRC	A	12	43	3	ORL	data addr,#data	24
14	1	DEC	A	12	44	2	ORL	A,#data	12
15	2	DEC	data addr	12	45	2	ORL	A,data addr	12
16	1	DEC	@R0	12	46	1	ORL	A,@R0	12
17	1	DEC	@R1	12	47	1	ORL	A,@R1	12
18	1	DEC	R0	12	48	1	ORL	A,R0	12
19	1	DEC	R1	12	49	1	ORL	A,R1	12
1A	1	DEC	R2	12	4A	1	ORL	A,R2	12
1B	1	DEC	R3	12	4B	1	ORL	A,R3	12
1C	1	DEC	R4	12	4C	1	ORL	A,R4	12
1D	1	DEC	R5	12	4D	1	ORL	A,R5	12
1E	1	DEC	R6	12	4E	1	ORL	A,R6	12
1F	1	DEC	R7	12	4F	1	ORL	A,R7	12
20	3	JB	bit addr,code addr	24	50	2	JNC	code addr	24
21	2	AJMP	code addr	24	51	2	ACALL	code addr	24
22	1	RET		24	52	2	ANL	data addr,A	12
23	1	RL	A	12	53	3	ANL	data addr,#data	24
24	2	ADD	A,#data	12	54	2	ANL	A,#data	12
25	2	ADD	A,data addr	12	55	2	ANL	A,data addr	12
26	1	ADD	A,@R0	12	56	1	ANL	A,@R0	12
27	1	ADD	A,@R1	12	57	1	ANL	A,@R1	12
28	1	ADD	A,R0	12	58	1	ANL	A,R0	12
29	1	ADD	A,R1	12	59	1	ANL	A,R1	12
2A	1	ADD	A,R2	12	5A	1	ANL	A,R2	12
2B	1	ADD	A,R3	12	5B	1	ANL	A,R3	12
2C	1	ADD	A,R4	12	5C	1	ANL	A,R4	12
2D	1	ADD	A,R5	12	5D	1	ANL	A,R5	12
2E	1	ADD	A,R6	12	5E	1	ANL	A,R6	12
2F	1	ADD	A,R7	12	5F	1	ANL	A,R7	12

Kód hex	Hossz byte	Uta-sítás	Ope-randus	Idő	Kód hex	Hossz byte	Uta-sítás	Ope-randus	Idő
60	2	JZ	code addr	24	99	1	SUBB	A,R1	12
61	2	AJMP	code addr	24	9A	1	SUBB	A,R2	12
62	2	XRL	data addr,A	12	9B	1	SUBB	A,R3	12
63	3	XRL	data addr,#data	24	9C	1	SUBB	A,R4	12
64	2	XRL	A,#data	12	9D	1	SUBB	A,R5	12
65	2	XRL	A,data addr	12	9E	1	SUBB	A,R6	12
66	1	XRL	A,@R0	12	9F	1	SUBB	A,R7	12
67	1	XRL	A,@R1	12	A0	2	ORL	C,/bit addr	24
68	1	XRL	A,R0	12	A1	2	AJMP	code addr	24
69	1	XRL	A,R1	12	A2	2	MOV	C,bit addr	12
6A	1	XRL	A,R2	12	A3	1	INC	DPTR	24
6B	1	XRL	A,R3	12	A4	1	MUL	AB	48
6C	1	XRL	A,R4	12	A5		-		
6D	1	XRL	A,R5	12	A6	2	MOV	@R0,data addr	24
6E	1	XRL	A,R6	12	A7	2	MOV	@R1,data addr	24
6F	1	XRL	A,R7	12	A8	2	MOV	R0,data addr	24
70	2	JNZ	code addr	24	A9	2	MOV	R1,data addr	24
71	2	ACALL	code addr	24	AA	2	MOV	R2,data addr	24
72	2	ORL	C,bit addr	24	AB	2	MOV	R3,data addr	24
73	1	JMP	@A+DPTR	24	AC	2	MOV	R4,data addr	24
74	2	MOV	A,#data	12	AD	2	MOV	R5,data addr	24
75	3	MOV	data addr,#data	24	AE	2	MOV	R6,data addr	24
76	2	MOV	@R0,#data	12	AF	2	MOV	R7,data addr	24
77	2	MOV	@R1,#data	12	B0	2	ANL	C,/bit addr	24
78	2	MOV	R0,#data	12	B1	2	ACALL	code addr	24
79	2	MOV	R1,#data	12	B2	2	CPL	bit addr	12
7A	2	MOV	R2,#data	12	B3	1	CPL	C	12
7B	2	MOV	R3,#data	12	B4	3	CJNE	A,#data,code addr	24
7C	2	MOV	R4,#data	12	B5	3	CJNE	A,data addr,code addr	24
7D	2	MOV	R5,#data	12	B6	3	CJNE	@R0,#data,code addr	24
7E	2	MOV	R6,#data	12	B7	3	CJNE	@R1,#data,code addr	24
7F	2	MOV	R7,#data	12	B8	3	CJNE	R0,#data,code addr	24
80	2	SJMP	code addr	24	B9	3	CJNE	R1,#data,code addr	24
81	2	AJMP	code addr	24	BA	3	CJNE	R2,#data,code addr	24
82	2	ANL	C,bit addr	24	BB	3	CJNE	R3,#data,code addr	24
83	1	MOVC	A,@A+PC	24	BC	3	CJNE	R4,#data,code addr	24
84	1	DIV	AB	48	BD	3	CJNE	R5,#data,code addr	24
85	3	MOV	data addr,data addr	24	BE	3	CJNE	R6,#data,code addr	24
86	2	MOV	data addr,@R0	24	BF	3	CJNE	R7,#data,code addr	24
87	2	MOV	data addr,@R1	24	C0	2	PUSH	data addr	24
88	2	MOV	data addr,R0	24	C1	2	AJMP	code addr	24
89	2	MOV	data addr,R1	24	C2	2	CLR	bit addr	12
8A	2	MOV	data addr,R2	24	C3	1	CLR	C	12
8B	2	MOV	data addr,R3	24	C4	1	SWAP	A	12
8C	2	MOV	data addr,R4	24	C5	2	XCH	A,data addr	12
8D	2	MOV	data addr,R5	24	C6	1	XCH	A,@R0	12
8E	2	MOV	data addr,R6	24	C7	1	XCH	A,@R1	12
8F	2	MOV	data addr,R7	24	C8	1	XCH	A,R0	12
90	3	MOV	DPTR,#data	24	C9	1	XCH	A,R1	12
91	2	ACALL	code addr	24	CA	1	XCH	A,R2	12
92	2	MOV	bit addr,C	24	CB	1	XCH	A,R3	12
93	1	MOVC	A,@A+DPTR	24	CC	1	XCH	A,R4	12
94	2	SUBB	A,#data	12	CD	1	XCH	A,R5	12
95	2	SUBB	A,data addr	12	CE	1	XCH	A,R6	12
96	1	SUBB	A,@R0	12	CF	1	XCH	A,R7	12
97	1	SUBB	A,@R1	12	D0	2	POP	data addr	24
98	1	SUBB	A,R0	12	D1	2	ACALL	code addr	24

Kód hex	Hossz byte	Uta-sítás	Ope-randus		Kód hex	Hossz byte	Uta-sítás	Ope-randus	Idő
D2	2	SETB	bit addr	12	EA	1	MOV	A,R2	12
D3	1	SETB	C	12	EB	1	MOV	A,R3	12
D4	1	DA	A	12	EC	1	MOV	A,R4	12
D5	3	DJNZ	data addr,code addr	24	ED	1	MOV	A,R5	12
D6	1	XCHD	A,@R0	12	EE	1	MOV	A,R6	12
D7	1	XCHD	A,@R1	12	EF	1	MOV	A,R7	12
D8	2	DJNZ	R0,code addr	24	F0	1	MOVX	@DPTR,A	24
D9	2	DJNZ	R1,code addr	24	F1	2	ACALL	code addr	24
DA	2	DJNZ	R2,code addr	24	F2	1	MOVX	@R0,A	24
DB	2	DJNZ	R3,code addr	24	F3	1	MOVX	@R1,A	24
DC	2	DJNZ	R4,code addr	24	F4	1	CPL	A	12
DD	2	DJNZ	R5,code addr	24	F5	2	MOV	data addr,A	12
DE	2	DJNZ	R6,code addr	24	F6	1	MOV	@R0,A	12
DF	2	DJNZ	R7,code addr	24	F7	1	MOV	@R1,A	12
E0	1	MOVX	A,@DPTR	24	F8	1	MOV	R0,A	12
E1	2	AJMP	code addr	24	F9	1	MOV	R1,A	12
E2	1	MOVX	A,@R0	24	FA	1	MOV	R2,A	12
E3	1	MOVX	A,@R1	24	FB	1	MOV	R3,A	12
E4	1	CLR	A	12	FC	1	MOV	R4,A	12
E5	2	MOV	A,data addr	12	FD	1	MOV	R5,A	12
E6	1	MOV	A,@R0	12	FE	1	MOV	R6,A	12
E7	1	MOV	A,@R1	12	FF	1	MOV	R7,A	12
E8	1	MOV	A,R0	12					
E9	1	MOV	A,R1	12					

A fenti táblázatban az első oszlop az utasítás kódját tartalmazza hexadecimális számrendszerben. A második oszlop az utasítás hossza byte-ban. A harmadik oszlop az utasítás mnemonikja, majd az operandusok következnek, végül a végrehajtáshoz szükséges idő az oszcillátor periódusainak számával megadva.

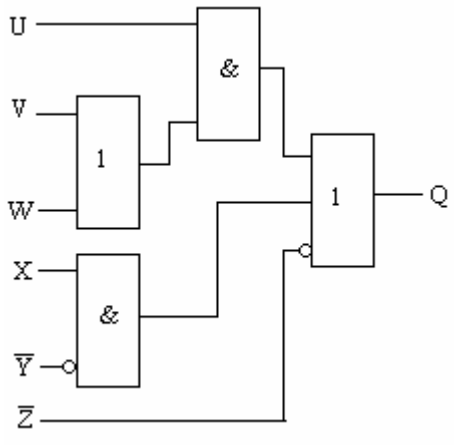
A fenti táblázatban az első oszlop az utasítás kódját tartalmazza hexadecimális számrendszerben. A második oszlop az utasítás hossza byte-ban. A harmadik oszlop az utasítás mnemonikja, majd az operandusok következnek, végül a végrehajtáshoz szükséges idő az oszcillátor periódusainak számával megadva.

Példák

1. Adott az alábbi kombinációs hálózat:

$$Q = \{U \cdot (V + W)\} + (\overline{Y}) + \overline{Z}$$

A megvalósítás többféle lehet, kapuáramkörökkel és relés áramkörökkel is.



A hálózat a 8051-es mikrovezérlővel néhány utasítással szintén megvalósítható. Jól kihasználhatók a bitekkel végzett logikai műveletek.

```
.
.
.
MOV   C,V          ; a V bemenet beolvasása a CY flag bitbe.
ORL   C,W          ; a VAGY kapcsolat előállítása
ANL   C,U          ; az ÉS kapcsolat megvalósítása
MOV   F0,C         ; közbenső érték tárolása az F0 szabadon felhasználható flag bitben
MOV   C,X          ; X bemenet beolvasása
ANL   C,Y          ; ÉS kapcsolat létrehozása
ORL   C,F0         ; VAGY kapcsolat a letárolt értékkel
ORL   C,Z          ; VAGY kapcsolat a Z bemenettel
MOV   Q,C          ; a logikai függvény kimenete
.
.
.
```

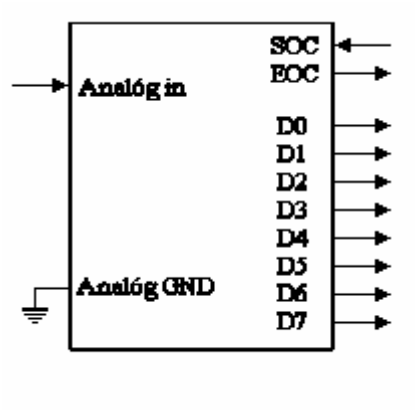
2. Érintkező szoftver prellmentesítése:
Ha nagy számú nyomógombot illesztünk a mikrovezérlőhöz, a szoftver prellmentesítés olcsóbb lehet. A prellmentesítést késleltetéssel érhetjük el. A késleltetési időt az adott kapcsolóhoz kell megválasztani. A PCNT prellmentesítő szubrutin a P1 port 0. bitjére kötött kapcsoló impulzusait számlálja. Lementi az akkumulátort a stack-be, a P0 port kivezetésekre logikai 1 szintet állít be és a LOOP1 hurokban vár a kapcsoló megnyomására. A kapcsoló a 0. bitet logikai 0 szintre húzza le.

A nyomógomb megnyomásakor növeli az R0 regiszter tartalmát 1-el, majd meghívja a DELAY késleltető szubrutint, hogy a prellezést hatástalanítsa. A késleltetés letelte után a LOOP2 hurokban várja meg, hogy a kapcsoló a 0. bitet felengedje. Ha ez megtörtént, a prellezést ugyancsak a DELAY szubrutin meghívásával küszöböli ki. A késleltetés letelte után visszaállítja az akkumulátor tartalmát, és visszatér a hívó programba.

```
.
.
.
            ORG     100H
SWITCH      DATA    P1              ; P0 input portra van csatlakoztatva a kapcsoló

PCNT:       PUSH    ACC             ; Az akkumulátor mentése
            MOVE    SWITCH, #0FFFH  ; P0 port bitjeit 1-be állítja

LOOP1:      JB      SWITCH.0, LOOP1 ; Várakozás  a nyomógombra

            INC     R0              ; R0 inkrementálása
            ACALL   DELAY           ; Késleltetés, a prellezés kiküszöbölésére

LOOP2:      JNB     SWITCH.0, LOOP2 ; Váralozás a nyomógomb felengedésére
            ACALL   DELAY           ; késleltetés az 1-0 átmenetkor keletkező

                                    ; prellezés kiküszöbölésére
```

3. példa. 8 bites ADC illesztése a 8051 mikrokontrollerhez.

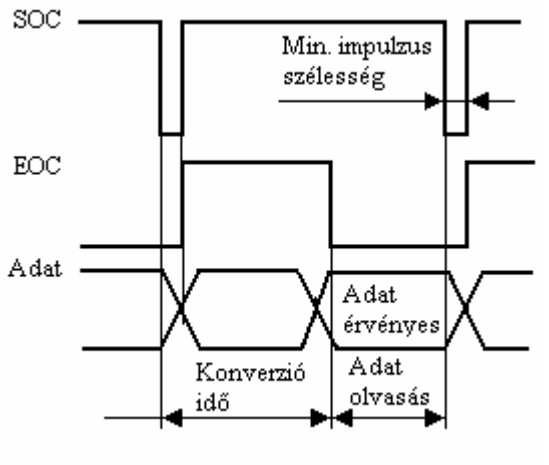
Az analód/digitál konverter kivezetései az ábrán láthatók.



Az A/D átalakító kivezetései.

Az illesztés a SOC és EOC jelekkel történik. A konverzió elindul, ha az SOC jel lemegy 0 szintre. A negatív impulzus szélességét a konverter adatlapjából lehet megválasztani. Ha a konverzió befejeződött, az EOC jel 1 szintről 0 szintre vált, amit megszakítás generálására lehet használni. Ha a jel lemegy 0-ra, az adat a konverterből kiolvasható. A konverzió idődiagramját az alábbi ábra

szemlélteti. A minimális SOC impulzus szélességet a konverter műszaki leírásából lehetmeghatározni. A példában feltételeztük, hogy egy NOP utasításnyi késleltetés elegendő.



Ezek figyelembevételével a program a következő lehet:

```
SOC     BIT   P2.0          ; A handshake jelek  definiálása.
EOC     BIT   P2.1          ;
BYTE    DATA  P1            ; adat port definiálása

        ORG   40H           ; a program kezdete
MAIN:   SETB  SOC           ; SOC beállítása 1-be.
        SETB  EOC           ; EOC beállítása 1-be.
        MOV   BYTE, #0FF0H  ; Adat bemenetek beállítása 1-be.
ACALL   A2D                 ; konverzió indítása.
NOP                         ; A program innen folytatódik a
                            ; szubrutin után


        ORG   200H          ; a szubrutin eleje.
A2D:    CLR   SOC           ; Konverzió start jel (1→átmenet)
        NOP                 ; Minimális impulzus szélesség.
        SETB  SOC           ; SOC visszaállítása 1-be.
LP:     JB    EOC, LP       ; várakozás a konverzió végére.
        MOV   A, BYTE       ; A konvertált adat beolvasása.
        RET                 ; Visszatérés a programba.
```

A konverzió vége jelet természetesen megszakítással is lehet kezelni

Tartalomjegyzék