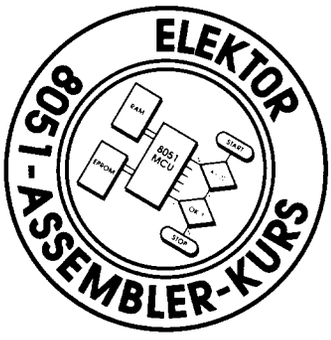
8051-es Mikrokontroller- és Assembler-tanfolyam

5. rész: Analóg jelek feldolgozása és veremtár kezelés



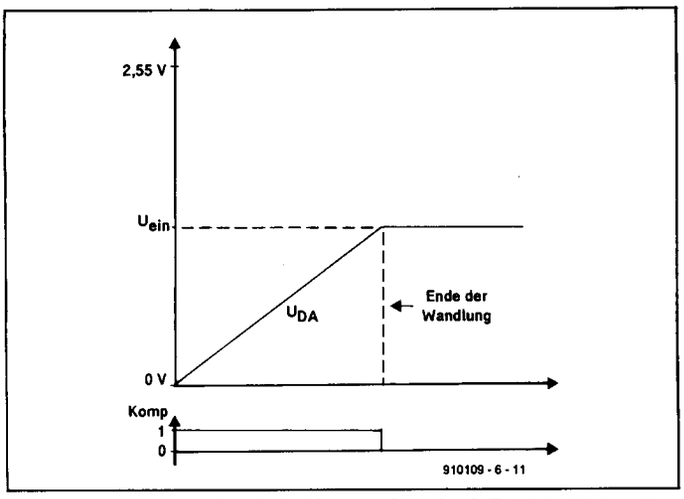
A tanfolyam e részében először azt beszéljük meg, hogy hogyan lehet analóg jeleket egyszerűen feldolgozni a bővítőkártyával. Ezután a 8051 veremtárára (stack memóriájára) térünk ki, mely a szubrutinkezelésben játszik fontos szerepet. Ezzel megteremtettük a következő téma, az interruptfeldolgozás alapjait. Mint mindig, a megbeszélt programok most is rendelkezésre állnak a tanfolyam diszkettjén és a többfunkciós kártya, illetve a Compuboard segítségével rögtön tesztelhetők.

D/A átalakítás

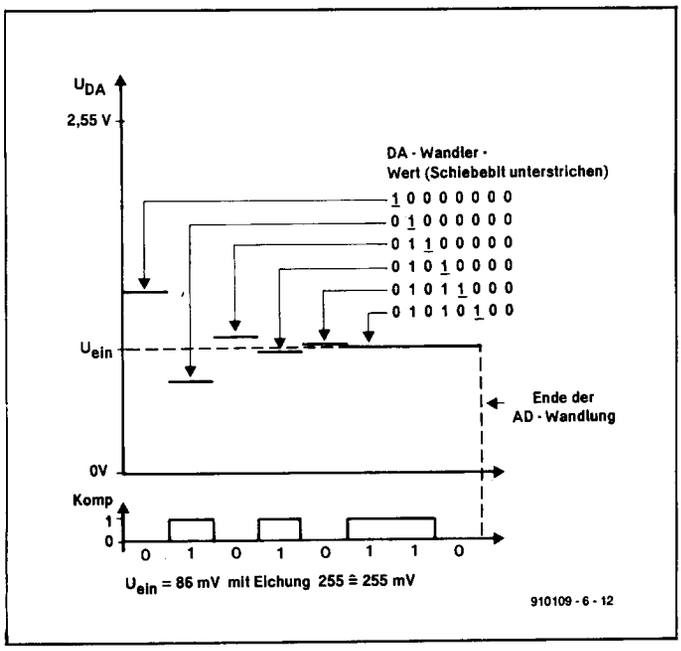
A mikrovezérlők számos alkalmazásában van szükség mért analóg értékek feldolgozására és analóg jelek outputjára. A többfunkciós kártyán ehhez egy D/A átalakítót helyeztünk el, melyet a tanfolyam 3. részében már megismerhettünk. Ebben a részben a többfunkciós kártya komparátorait fogjuk analóg mérési értékek felvételére céljából a D/A átalakítóval együtt használni.

Átalakítási eljárások

Az A/D átalakítás egyszerű módja a rámpaeljárás. Ezt vázlatosan az *1. ábra* mutatja be. Az eljárás során egy D/A átalakítóra lineárisan emelkedő rámpafeszültséget adunk. Ez a feszültség mindaddig nő, míg a komparátor nem jelenti, hogy értéke már meghaladta a bemeneti feszültséget. A D/A átalakítóra adott legutolsó érték felel meg ezután a bemeneti feszültség értékének. A rámpafeszültséget a szoftver úgy generálja, hogy egy regiszter értékét 0-tól kezdve fokozatosan növeli és egyúttal kiadja a D/A átalakítóra. Egy pontosan így működő program található BSP14.A51 jelöléssel a tanfolyamdiszketten. Mivel a program viszonylag egyszerű, közelebbről itt nem tárgyaljuk.



1. ábra. A feszültség alakulása a rámpás A/D átalakításnál



2. ábra. A feszültség alakulása a szukcesszív approximációval működő A/D átalakításnál

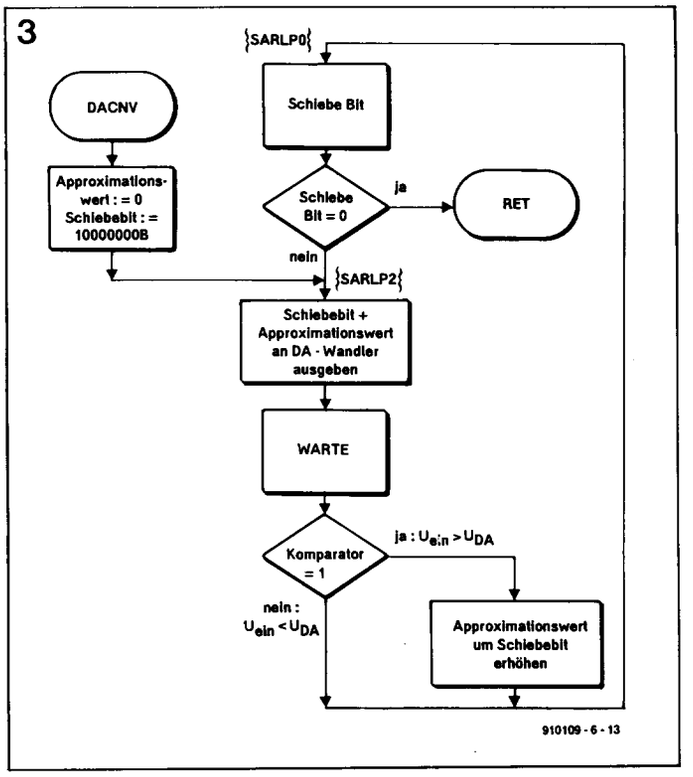
A rámpaeljárás legfőbb hátránya a kis sebesség. 8-bites felbontás esetén a rámpa értékét a legkedvezőtlenebb esetben akár 255-ször is meg kell növelni és a bemeneti feszültséggel mindennyiszer össze kell hasonlítani. 12 bitnél már 4096 lépés szükséges. Minden egyes lépésnél ki kell természetesen várni a D/A átalakító és a komparátorok tranziens (beállási) idejét. Ebből kifolyólag ez az eljárás számos alkalmazás szempontjából elfogadhatatlanul lassú. Gyorsabb eljárás a sok D/A átalakító chipben használatos **szukcesszív approximációs átalakítás.** Ez a második eljárás a Compuboardon szoftver úton valósítható meg.

Az eljárás minden egyes lépésében az átalakítandó analóg érték egy bitje kerül meghatározásra. 8 bites átalakításhoz tehát pontosan nyolc lépés szükséges. Az átalakítás a legnagyobb helyértékű bitnél kezdődik. Az éppen meghatározás alatt lévő bit egy regiszterben helyezkedik el. Ezt a bitet léptetőbitnek vagy tolóbitnek nevezik. Az analóg érték már ismert bitjeit ugyancsak egy regiszter gyűjti. Ezek a bitek együttesen képezik az úgynevezett approximációs értéket.

1. ábra.
Ende der Wandlung: az átalakítás vége

2. ábra.
D/A-Wandler-Wert (Schiebebit unterstrichen): D/A-átalakító-érték (léptetőbit aláhúzva)
Ende der A/D-Wandlung: A/D átalakítás vége
U_{ein}: U_{be}
U_{ein}=86 mV mit Eichung 255=255 mV:
U_{be}=86 mV, 255=255 mV kalibráció mellett

Az eljárás működési módját a *2. ábra*, a folyamatot a *3. ábra* mutatja be. Először az approximációs értéket 0-ra, a léptetőbit tároló regiszterét pedig 10000000_2 értékre állítjuk be. A 7-es bittel kezdünk. Egy átalakítási lépés a következőképpen zajlik le: először új D/A output érték meghatározására kerül sor. Ez a léptetőbit és az eddigi approximációs érték logikai VAGY művelet segítségével képzett összegéből adódik. Az így kapott értéket adjuk ki a D/A átalakítóra és hasonlítjuk össze a komparátor útján a bemeneti feszültséggel. Ha a bemeneti feszültség nagyobb a D/A átalakító feszültségénél, akkor az éppen kiadott összehasonlítási érték válik az új approximációs értékké. Benne tehát a tolóbit pozíciójában egy egyes áll. Ellenkező esetben a régi approximációs érték marad meg, melyben a léptetőbit helyén 0 szerepel. Ezáltal megtörtént az approximációs értékben az új bit meghatározása. Ezt követően a léptetőbit pozíciója jobbra tolódik és végbemegy a következő átalakítási lépés. Az eljárás mindaddig folytatódik, amíg az összes bit meghatározása meg nem történt. A módszer tehát a mindenkor meghatározandó bit próbaképpen történő beírásából és a kapott approximációs értéknek a bemeneti feszültséggel történő összehasonlításából áll. Ha az approximációs érték így túl naggyá válik, akkor az éppen vizsgált bit nullázásra kerül benne. Ha a kapott érték még mindig túl kicsi, akkor a bitet meghagyjuk. Még néhány megjegyzés következik az alábbiakban arról, hogy hogyan alakítottuk át az imént ismertetett elvet programmá.



3. ábra. A folyamatábra a lépésenkénti A/D átalakítás programjának magyarázatára szolgál

4. ábra. Olyan A/D átalakítóprogram listája, mellyel mind a három csatorna tesztelhető

4

```
****** LISTING of EASM51 (BSP15) ******
LINE LOC  OBJ          T       SOURCE
   1 0000                ; ****** DATEI BSP15.A51 ********************************
   2 0000                ;
   3 0000                P1      EQU 090H
   4 0000                ;
   5 0000                        ORG   4100H          ; Programm startadresse
   6 4100 7B 01     [1] START   MOV   R3,#00000001B  ; Maske Kanal 1
   7 4102 31 20     [2]         ACALL DACNV          ; DA Wandler aufrufen
   8 4104 31 44     [2]         ACALL BYTE           ; Byte und Leerzeichen ausgeben
   9 4106 31 4A     [2]         ACALL BLANK
  10 4108 7B 02     [1]         MOV   R3,#00000010B  ; Kanal 2
  11 410A 31 20     [2]         ACALL DACNV
  12 410C 31 44     [2]         ACALL BYTE
  13 410E 31 4A     [2]         ACALL BLANK
  14 4110 7B 04     [1]         MOV   R3,#00000100B  ; Kanal 3
  15 4112 31 20     [2]         ACALL DACNV
  16 4114 31 44     [2]         ACALL BYTE
  17 4116 74 0D     [1]         MOV   A,#13          ; Zeilenvorschub
  18 4118 31 4C     [2]         ACALL CHR
  19 411A 74 0A     [1]         MOV   A,#10
  20 411C 31 4C     [2]         ACALL CHR
  21 411E 80 E0     [2]         SJMP  START          ; Endlos-schleife
  22 4120               ;
  23 4120 D2 91     [1] DACNV   SETB  P1.1       ; fuer Oszilloskop
  24 4122 7E 00     [1]         MOV   R6,#0      ; Approximationswert
  25 4124 74 80     [1]         MOV   A,#080H    ; SCHIEBE-BIT
  26 4126 80 09     [2]         SJMP  SARLP2     ; springe in Schleife
  27 4128 EF        [1] SARLP0  MOV   A,R7       ; Hole SCHIEBE-BIT
  28 4129 C3        [1]         CLR   C
  29 412A 13        [1]         RRC   A          ; schiebe rechts ,fuelle mit 0 auf
  30 412B 70 04     [2]         JNZ   SARLP2     ; wenn SCHIEBE-BIT<>0 weitermachen
  31 412D EE        [1]         MOV   A,R6       ; Resultat nach A holen
  32 412E C2 91     [1]         CLR   P1.1       ; Ende der Wandlung am Oszilloskop zeigen
  33 4130 22        [2]         RET
  34 4131 FF        [1] SARLP2  MOV   R7,A       ; SCHIEBE-BIT in R7 retten
  35 4132 4E        [1]         ORL   A,R6       ; zum Approximationswert zufuegen
  36 4133 90 C0 00  [2]         MOV   DPTR,#0C000H
  37 4136 F0        [2]         MOVX  @DPTR,A    ; an DA Wandler ausgeben
  38 4137 FC        [1]         MOV   R4,A       ; Summe fuer spaeter merken
  39 4138 7A 64     [1]         MOV   R2,#100    ; Einschwingzeit
  40 413A DA FE     [2] SARWT   DJNZ  R2,SARWT   ; abwarten
  41 413C E0        [2]         MOVX  A,@DPTR    ; Komparatorausgaenge holen
  42 413D 5B        [1]         ANL   A,R3       ; gewuenschten KOMP. selektieren
  43 413E 60 E8     [2]         JZ    SARLP0     ; Bit=0 wenn Uin<Udac, kein Bit setzen
  44 4140 EC        [1]         MOV   A,R4       ; Bit=1 if Uin>Udac
  45 4141 FE        [1]         MOV   R6,A       ; Approximationswert:=Summe
  46 4142 80 E4     [2]         SJMP  SARLP0
  47 4144               ;
  48 4144               ; Monitor interface
  49 4144               COMMAND EQU  030H
  50 4144               MON     EQU  0200H
  51 4144               ccCHR   EQU  001H
  52 4144               ccBYTE  EQU  003H
  53 4144               ;
  54 4144 75 30 03 [2] BYTE    MOV   COMMAND,#ccBYTE
  55 4147 02 02 00 [2]         LJMP  MON
  56 414A 74 20    [1] BLANK   MOV   A,#' '
  57 414C 75 30 01 [2] CHR     MOV   COMMAND,#ccCHR
  58 414F 02 02 00 [2]         LJMP  MON
  59 4152                      END
********** SYMBOLTABLE (13 symbols) **********
      P1 :0090     START :4100     DACNV :4120    SARLP0 :4128
  SARLP2 :4131     SARWT :413A   COMMAND :0030       MON :0200
   ccCHR :0001   ccBYTE :0003      BYTE :4144     BLANK :414A
     CHR :414C                                              910109-6-14
```

3. ábra.
Schiebe Bit: lépj egy bitet
Schiebebit=0: léptetőbit=0
Approximationswert:=0: approximációs érték:=0
Schiebebit:=10000000B: léptetőbit:=10000000B
ja: igen
nein: nem
Schiebebit + Approximationswert an D/A-Wandler ausgeben: Léptetőbit és approximációs érték összegét D/A átalakítóra kiadni
Warte: várj
Approximationswert um Schiebebit erhöhen: Approximációs értéket a léptetőbittel növelni

4. ábra.
Program startadresse: program kezdőcím
Maske Kanal 1: 1-es csatorna maszkolása
DA Wandler aufrufen: DA átalakító behívása
Byte und Leerezeichen ausgeben: Byte és Space (szóközjel) outputja
Kanal 2: 2-es csatorna
Kanal 3: 3-as csatorna
Endlos -schleife: végtelen hurok
fuer Oszilliskop: oszcilloszkópra
Approximationswert: approximációs érték
SCHIEBE-BIT: LÉPTETŐBIT
springe in Schleife: ugrás a hurokba
Hole SCHIEBE-BIT: LÉPTETŐBIT behozása
schiebe rechts, fuelle mit 0 auf: léptetés jobbra, feltöltés 0-val
wenn SCHIEBE-BIT 0 weitermachen: folytatás, ha a léptetőbit 0
Resultat nach A holen: eredmény átvitele A-ba
Ende der Wandlung am Oszilloskop zeigen: átalakítás végének kijelzése oszcilloszkópon
SCHIEBE-BIT in R7 retten: léptetőbit mentése R7-be
zum Approximationswert zufuegen: hozzáadás az approximációs értékhez
an DA Wandler ausgeben: output a DA átalakítóra
Summe fuer spaeter merken: összeg megjegyzése a későbbiek számára
Einschwingzeit: berezgési idő
abwarten: kivárás
Komparatorausgaenge holen: Komparátorkimenetek behozása
gewuenschten KOMP. selektieren: kívánt komparátor kiválasztása
Bit=0 wenn Uin<Udac, kein Bit setzen: bit=0, ha Uin<Udax (bitet nem beírni)
Bit=1 if Uin<Udac: bit=1, ha Uin>Udac
Approximationswert: =
Summe: approximációs érték := összeg

Az A/D átalakító-program

A program a három bemeneti feszültséget hexadecimális formában adja ki. Ahhoz, hogy az A/D átalakítást mindhárom bemeneti csatornára egyszerűen el lehessen végezni, egy DACNV nevű szubrutint írtunk. A szubrutin számára az R3 regiszterben van megadva a kiértékelendő komparátor bitpozíciója. A szubrutin eredményeként az akkumulátorban szolgáltatja az átalakított értéket. Ezáltal a *4. ábra* listájában START-tal kezdődő főprogram igen egyszerűen alakul. A három csatornára vonatkozólag a megfelelő bitet R3-ba kell mindig beírni. Ezután be kell hívni a D/A átalakítást végző DACNV szubrutint és végül a kapott értéket a monitor útján V24-en át hexadecimálisan ki kell adni (BYTE szubrutin). Mindhárom csatorna outputja után még egy kocsivissza/soremelés (CR/LF) byte-kombináció is következik. Ezt követően az egész folyamat elölről kezdődik.

Nem ilyen egyszerű a D/A átalakítás programozása. Ehhez a következőket kell tudnunk. Az approximációs érték tárolása az R6 regiszterben történik. A léptetőbit az R7 regiszterben helyezkedik el (időnként az akkumulátorban is előfordul). Az időközönként a léptetőbitből és az approximációs bitből képzendő összeget a program az R4 regiszterben jegyzi meg magának. Maga az átalakítási lépés az SARLP2 címkénél indul. Ezen a ponton az akkumulátorban a pillanatnyi léptetőbit, R6-ban pedig a régi approximációs érték tartózkodik.

Először az R7-be kerül a léptetőbit (34-es sor). A 35-ös sorban összegződik R6 és az akku, majd az összeg a D/A átalakítóra kerül. Ez a pillanatnyi összehasonlítási érték az R4-ben is eltárolódik. Ezután az SRWT várakozóhurok kivárja a D/A átalakító beállási idejét. A 41-es sorban a komparátor kimenetek a többfunkciós kártyáról az akkuba kerülnek. Az R3-ban levő bittel történő maszkolás útján a 42-es sorban megtörténik a helyes komparátor-kimenet kiválasztása. Az R6-ban az új approximációs érték képzésére attól függően kerül sor, hogy a komparátor 1-et vagy 0-t szolgáltat-e. Az átalakítás következő lépését az SARLP0 címkére való ugrás vezeti be. A léptetőbit az akkumulátorba kerül és jobbra tolódik (28-as és 29-es sor). Ha ezután az akku tartalma 0, akkor az átalakítás befejeződött. Ellenkező esetben a program SARLP2-nél folytatódik.

Az átalakítási folyamat oszcilloszkóppal (és hangszóró útján füllel) való követhetősége érdekében az átalakítás ideje alatt a P1.1 portbit értékét 1-re állítjuk. A tárolócsöves oszcilloszkópon megjelenített feszültséggörbéket az *5. ábra* mutatja be. A D/A átalakítás végén az R6 értéke még átkerül az akkumulátorba.

Ennek az eljárásnak az előnye nyilvánvaló. Csupán 8 hurok lefutása szükséges. Gyors D/A átalakítóval és komparátorral a 8051-es a 8-bites átalakítást minden probléma nélkül elvégzi 250 mikroszekundum alatt, ami már elég gyorsnak tekinthető (4000 minta/szekundum letapogatási sebességnek felel meg).

Ez a program is a szubrutinok hasznosságát demonstrálja. A 8051-es szubrutin-kezelésének megértése céljából beszéljük most meg az úgynevezett veremtárat (STACK-et), melyben a 8051-es a szubrutinok kezeléséhez szükséges adatokat tárolja.

Veremtárak

A szubrutin ugrások és interruptok (programmegszakítások) visszaugrási címeit tároló veremtár kezelésére az SP (Stack Pointer) veremmutató (081 című SFR) szolgál. Ez a veremtár a belső RAM-ban helyezkedik el. Szubrutin hívása esetén először eggyel növeljük a veremmutatót, majd beírjuk a visszatérési cím első byte-ját arra a (belső RAM-ban levő) tárolóhelyre, amelyre az SP veremmutató mutat. Ezt követően megint az SP növelése, majd a visszatérési cím következő byte-jának betárolása következik. Álljon itt egy kis példaprogram a veremtár alkalmazásának illusztrálására. Az utána következő táblázat bemutatja, hogy a program futása a veremtárban milyen értékeket eredményez.

LINE	LOC	OBJ	T	SOURCE (FORRÁS)
43	4118	12 41 23	(2)	LCALL UPRO1
44	411B	00	(1)	NOP
47	4123	12 4A BC	(2)UPRO1	LCALL UPRO2
48	4126	00	(1)	NOP
50	4ABC	85 08 44	(2) UPRO2	MOV 68,8

SP a 43. sor előtt: 07H
Belső RAM címek: 08 09 0A 0B 0C 0D
Tartalom a 43. sor előtt: FF FF FF FF FF FF (korábbi beírás)
Tartalom az 50. sor után: 1B 41 26 41 FF FF
(2 cím a veremben)
SP az 50. sor után: OBH

Nullázás után az SP értéke 7 (vesd össze a tanfolyam 1. részének 5. ábrájával és a 2. rész 4. ábrájával), tehát a belső RAM-ban a veremtár a 8-as címmel kezdődik és a cím felfelé növekszik. Így a verem ugyanazokat a helyeket foglalja el, mint az 1-es regiszterbank (és a következő bankok). Ezzel legalábbis az 1-es regiszterbank kiesik a használatból, ha az SP-t nem írjuk át a program kezdetén egy új értékkel. Ha a programozás során több egymásba ágyazott szubrutint használunk, akkor szubrutinonként 2 byte-ra van a veremben szükség. Ennek megfelelő számú hely vész el a belső RAM-ban.

Még bonyolultabbá válik a helyzet interrupt-kezelést is tartalmazó programok futása során. A visszatérési cím tárolásához itt is mindig 2 byte szükséges. Ehhez jön még hozzá a kimentendő regiszterek (SFR-ek), valamint az interrupt-kezelő rutin által behívott szubrutinok visszatérési címeinek helyszükséglete. Ezért mindig gondoskodni kell arról, hogy elegendő veremtár álljon rendelkezésre. A 8052, illetve a 8032 alkalmazói itt jól felhasználhatják a belső RAM járulékos 128 byte-ját. Ehhez a

```
MOV SP,080H ; felső RAM
veremtárként használva
```

utasítást kell kiadni, ami azt fogja eredményezni, hogy a felső 128 byte veremtárként áll rendelkezésre. Ez a stack-méret már több célra elegendő lesz.

Adatok átmeneti tárolása a veremtárban

Gyakran kerülünk olyan helyzetbe, hogy egy bizonyos byte-ot (SFR-t vagy akkumulátort) rövid időre ki kell menteni, mert azokra nem sokkal később újra szükség lesz.
Ez a következő utasítással érhető el:

```
PUSH ACC ;
    mentsd az akkut
    a veremtárba
... ; bármilyen utasítások
POP ACC ;
    hozd vissza az akku
    tartalmát a veremtárból
```

A PUSH utasítás a megadott byte-ot (közvetlen címzés) a veremtárba tárolja be. A POP a legutoljára PUSH-olt byte-ot a veremtárból visszahozza. PUSH és POP során SP automatikusan 1-gyel megnő (beírás), ill. lecsökken (kiolvasás). Mivel a kimentéshez a veremtárban hely szükséges és a veremtárban hely csak korlátozottan áll rendelkezésre, PUSH és POP használata során mindig előrelátónak kell lennünk. A PUSH-, illetve POP utasításokat gyakran használjuk szubrutinokon belül SFR-ek mentésére, hogy azok a szubrutin befejeződése után változatlanul rendelkezésre álljanak.

Interruptok

Interruptok – tehát programmegszakítások – mindig akkor kerülnek alkalmazásra, amikor bizonyos (például külső) eseményekre gyorsan kell reagálni. Lehet ez az esemény például egy olyan mért érték beérkezése, melyet gyorsan fel kell dolgozni.

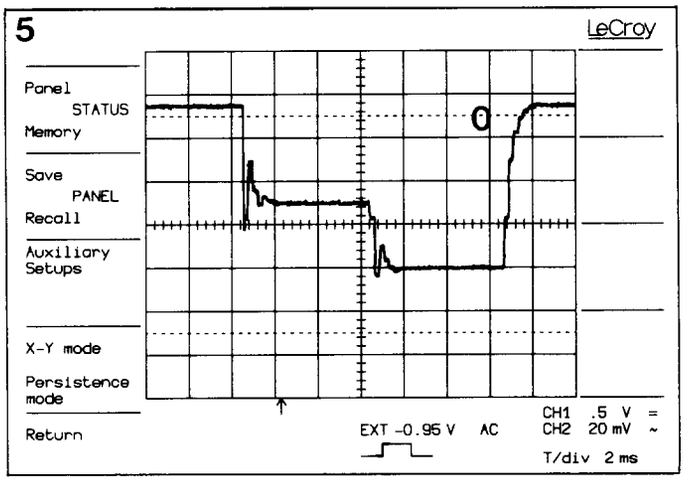
Éppen az interrupt előre nem láthatósága teszi problematikussá az interrupt-programozást és a hibakeresést az interrupt-vezérelt rendszereknél. Aki tehát saját maga kíván interruptokat programozni, azt már itt figyelmeztetnünk kell: talán sehol másutt nem adódik ilyen sok hibalehetőség!

Most pedig térjünk rá azokra az interruptokra, amelyeket a 8051-es tesz lehetővé. Külső interruptokként az INT0 és INT1 vonalakon levő jelek jönnek szóba. A Compuboard INT0 és INT1 jelei (a 64-pólusú csatlakozó c3 és c5 pontjain) a Compuboard IC12 jelű IC-je által invertálva jutnak a processzorra (12-es és 13-as kivezetés), mint $\overline{\text{INT0}}$ és $\overline{\text{INT1}}$ jelek. Ezeknél a külső interrupt-forrásoknál még az is előírható, hogy az interrupt csak INT0, ill. INT1 pozitív élénél, vagy az INT0 = 1, ill. INT1 = 1 állapotnál következzék-e be.

További interruptok válthatók ki egy Timer, például a TF0 (= Timer Flag 0), illetve TF1 (= Timer Flag 1) számláló-túlcsordulás vagy a soros interfész (RI = Receiver interrupt, TI = Transmitter interrupt) útján. A 8032-nek, illetve a 8052-nek van még egy további interrupt-forrása is, mégpedig a Timer 2, illetve az EXF2 külső bemenet. Amikor az interrupt kiváltása megtörténik, akkor a processzor LCALL utasítást ad ki az úgynevezett interrupt-címre. Az interrupt-forrásokat összefoglalva (interruptcímeikkel együtt) a *6. ábra* mutatja be.

Ezek a címek azonban mind az EPROM tartományban találhatók és az EPROM-ot nem programozhatjuk minden alkalommal újra csak azért, hogy az EPROM megfelelő címére egy megfelelő ugrási parancsot vagy szubrutin-hívást írjunk be. Kiutat az EMON51.DOC dokumentáció-

5. ábra. Ilyen az A/D átalakítási folyamat egy tárolócsöves oszcilloszkóp ernyőjén



6. ábra. Az interrupt-variánsok és a hozzájuk tartozó belépési címek táblázata

6

Interrupt	Interruptadresse	Link-index	LINK-Sprung
IE0	0003H	1	4003H
TF0	000BH	2	4006H
IE1	0013H	3	4009H
TF1	001BH	4	400CH
RI+TI	0023H	5	400FH
TF2+EXF2	002BH	6	4012H

ban LINK címszó alatt leírt módon a Monitor-EPROM kínál. Segítségével tetszőleges interrupt-rutinok használata válik lehetővé. Az IEO interrupt hatására a monitor a RAM-ban elhelyezkedő kódtároló 4003_{16} címére ugrat. Erre a RAM címre a monitor nullázás után egy belső interrupt-rutinra történő ugrási utasítást ír be. Aki egy interrupt fellépésekor saját interrupt-rutint kíván indítani, ezt az ugróutasítást a LINK monitor-szubrutin behívásával változtathatja meg. LINK hívásakor DPTR-ben a kívánt rutin címének, az akkumulátorban pedig az interrupt LINK-INDEX-ének kell állnia. Most már az is világos, hogy miért indítjuk programjainkat mindig a 4100_{16} (és nem a 4000_{16}) címtől. A RAM-ban a 4000_{16}...$40FF_{16}$ címek a monitor számára vannak fenntartva! Az interruptok szervezését az IE (Interrupt Enable, címe = $0A8_{16}$) és IP (Interrupt-Priority, címe = $0B8_{16}$) SFR-nek veszik át. Biteknek az IE-be történő beírásával vezérelhető, hogy mely interruptok legyenek ténylegesen engedélyezve. Részletes információkat a *7. ábra* tartalmaz. Az IT0 és IT1 bitek *(8. ábra)* azt határozzák meg, hogy a külső interruptok szintfüggően (bit = 0) vagy élfüggően (bit = 1) kerüljenek-e kiváltásra. Mindkét bit a TCON Timer-Controll-SFR-ben (címe 088_{16}) helyezkedik el (IT0 = TCON.0, IT1 = TCON.2). Egy kis tesztre nyújt módot a következő program (mely kivételesen nem található meg a tanfolyamdiszketten, de szerencsére egy nyúlfarknyi programocskáról van csak szó):

```
IE EQU 0A8H ;
      új:
      Interrupt Enable SFR
;
   ORG 4100H ;
      a program kezdőcíme
   MOV IE, OFFH ;
      összes interrupt
      engedélyezve
EWIG SJMP EWIG
   END
```

Interrupt a 64-pólusú csatlakozó c3 vagy c5 pontjára adott 5 V-os impulzussal váltható ki. Mi történik ekkor és miért?

A 8051 magas és alacsony interrupt-prioritásszintet ismer. A magasabb prioritású interrupt az alacsonyabb prioritású interrupt-rutinját megszakíthatja. Az interruptok prioritása a biteknek az interrupt-prioritás SFR-be történő beírásával rögzíthető.

Az IP–SFR egyes bitjeinek magyarázatára a *9. ábra* szolgál. A *10. ábra* az interrupt-ellenőrző rendszerről nyújt áttekintést. Az ábrán az a sorrend is felismerhető (Polling sequence), mellyel az interruptokat fel kell dolgozni, ha a kiváltó események egyidejűleg léptek fel.

Interrupt-rutinok

Interrupt kiváltása esetén a 8051 az előírt rutinra ugrik. Hogy a megszakított főprogram az interrupt kiszolgáló rutinjának lefutása után akadálytalanul folytatható legyen, az interrupt-rutinnak gondoskodnia kell arról, hogy a főprogram által használt regiszterek egyike se kerüljön megváltoztatásra. Ezt az SFR mentésével (PUSH) és a visszaugrás előtti POP-pal valósítjuk meg. Az RO...R7 regiszterek védelme céljából a regiszterbank átkapcsolásra kerül. Ez már a programtervezés fázisában megkívánja a regiszterbankoknak az egyes programsíkokhoz történő egyértelmű hozzárendelését. A kérdést nehezíti a regiszterbankoknak a veremtárral kapcsolatos, előzőekben vázolt komplikációja. Az előre nem látható és nehezen megtalálható hibák oka sokszor abban rejlik, hogy az említettek közül valamelyik veszélyforrást figyelmen kívül hagytuk.

Az interrupt-program befejezése – tehát a főprogramra való visszatérés kezdeményezése – a következő utasítással történik:

```
RETI; Return from Interrupt
```

A RET utasítástól eltérően ez az utasítás az interruptot kiváltó bitek szükség szerinti törlését is elvégzi. Az interrupt-programozásra gyakorlati példát a következőkben, a Timer megbeszélése során mutatunk be.

Számláló és óra

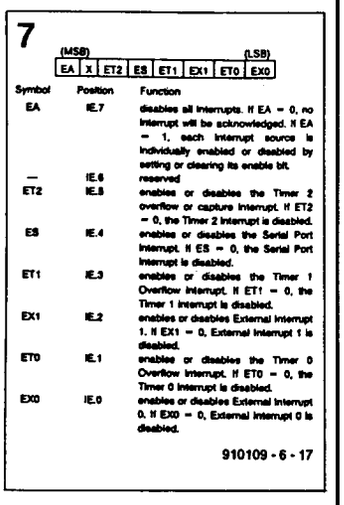
A 8051 két olyan számlálót tartalmaz, melyekkel események számlálhatók. Ha ezeket a számlálókat belső órajellel működtetjük, akkor visszaszámláló óraként (Timerként) is használhatók. Mindkét számláló különböző üzemmódokra alkalmas. A mindenkori üzemmódot itt is a megfelelő speciális funkcióregiszterben található bitminta határozza meg. A pillanatnyi számlálóértékek is az SFR-ekből olvashatók ki és azokba írhatók be.

Az órák üzemmódjának vezérlésére szolgáló SFR-ek szerepét a Time-Modus-Register (TMOD: 089_{16} című, nem bitcímezhető SFR) és a Timer-Control-Register (TCON: 088_{16} című, bitcímezhető SFR) töltik be . Az egyes bitek jelentését a *8. ábra* és *9. ábra* tartalmazza.

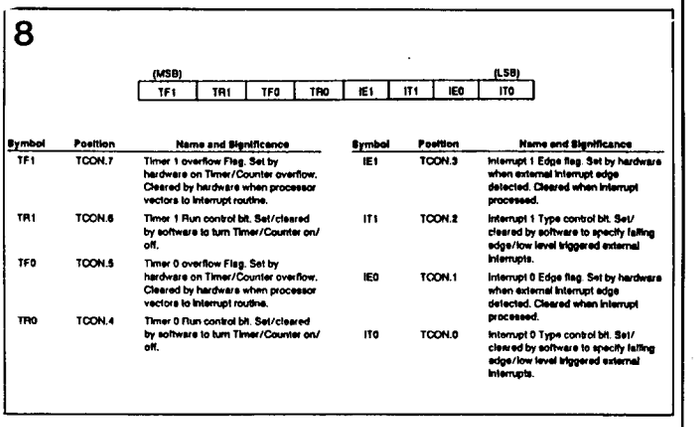
Nézzük először a TIMER/COUNTER1 üzemmódjait. Ennek a számlálónak az alacsony helyértékű byte-jára TL1 nevű SFR-ként hivatkozhatunk (címe $08b_{16}$) a magas helyértékű byte neve TH1 és címe $08D_{16}$. Az általános üzemmódot a TMOD.4 és TMOD.5 módusbitek határozzák meg. Ha mindkét bit 0, akkor a számláló 13-bites számlálóként működik. A különböző bitek és a külső jelek összjátékát a *12. ábra* mutatja be. A C/$\overline{T}$ bittel vezérelhető az, hogy a Timer belső óráról (C/$\overline{T}$ = 0) vagy külső jelről (C/$\overline{T}$ = 1) kerüljön-e triggerelésre.

A TR1, a GATE és az $\overline{INT1}$ kivezetés bitjei vezérlik, hogy mikor kell a számlálónak valóban működnie. TR1 segítségével például a számláló szoftver úton kapcsolható be és ki (amennyiben GATE = 0). Ha a számláló a maximális értékről 0-ra lép át – tehát túlcsordul – akkor a TF1 állapotjelzőbit mindig beírásra kerül. Ezt az

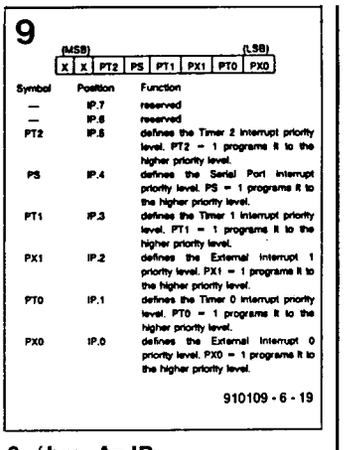
7. ábra. Az egyes bitek jelentése az IE Interrupt-Enable-Registerben



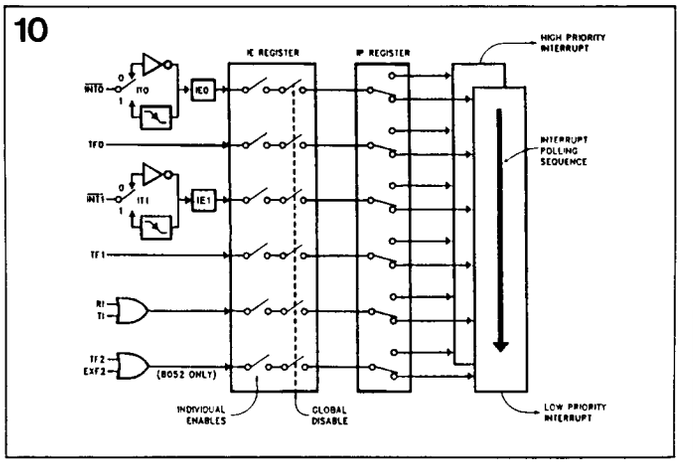
8. ábra. A TCON Time-Control Register bitjei



9. ábra. Az IP Interrupt-Priority Register bitjei



10. ábra. A 8051-es Interrupt-vezérlő rendszere



állapotjelzőt például szoftver úton le lehet kérdezni, illetve interrupt kiváltására lehet felhasználni. Ebből a célból az Interrupt-Enable-Regiszter a 3-as bitjét (IE.3) be kell írni.

C/$\overline{T}$ = 0 esetén a számláló a 12-szeresen leosztott kristályfrekvenciával, a mi esetünkben tehát 1 MHz-cel számol. A MODUS 1-ben (TCON.4 = 1, TCON.5 = 0) a számláló pontosan ugyanúgy számál, mint a MODUS 0-ban, de 16-bites számlálóként.

A 2-es üzemmódban a számláló olyan 8-bites számlálóként működik (a számérték a TL1 SFR-ben található), mely túlcsordulás esetén a TH1 SFR-ből kerül újratöltésre (8-bites Auto-Reload). Ezt az üzemmódot a *13. ábra* szemlélteti. A 3. üzemmódban Timer 1 számláló egyszerűen nem működik.

Az EMON51 monitorban a Timer 1 egyébként a soros interfész számára baudsebesség generátorként kerül felhasználásra, ezért csak akkor áll rendelkezésre, ha a soros interfészről lemondunk. A Timer 0 pontosan ugyanúgy működik, mint az 1-es Timer 1, de természetesen a hozzá tartozó biteket és SFR-eket kell használni. Lényeges különbség azonban, hogy a Timer 0 a 3-as módusban is üzemeltethető (TCON.0 = 1 és TCON.1 = 1). Ebben az esetben Timer 0 a *14. ábra* szerinti két független 8-bites számlálóként időzítőként működik. Ennek során a második 8-bites számláló néhány olyan vezérlőbitet is felhasznál, mely tulajdonképpen a Timer 1-hez tartozik. Ha tehát a Timer 0 a 3-as módusban működik, akkor a szóban forgó bitek a Timer 1-hez már nem használhatók.

A 8032 és a 8052 processzorok még egy további időzítővel, a Timer 2-vel is rendelkeznek. A későbbi modelleknél, például a Siemens 80537-esénél további számlálók és órák is rendelkezésre állnak.

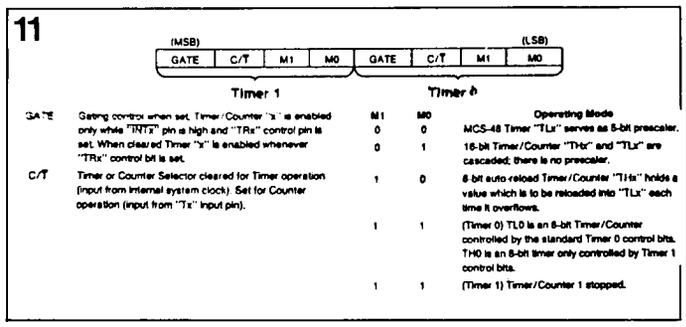
Az elméleti megközelítés után térjünk most át a gyakorlatra. Alkalmazásként programozzunk be egy olyan egyszerű órát, mely interrupt-vezérelten minden másodpercben egy-egy jelzést ad a főprogram számára. A főprogram ennek hatására másodpercenként egy csillagot küld az RS–232 interfészre. A program például olyan digitális óra programozásának alapjául is szolgálhat, amely tetszőleges vezérlési feladatokat is átvehet (vagy tulajdonosát születésnapján egy dallammal köszönti).

Interrupt-vezérelt óra

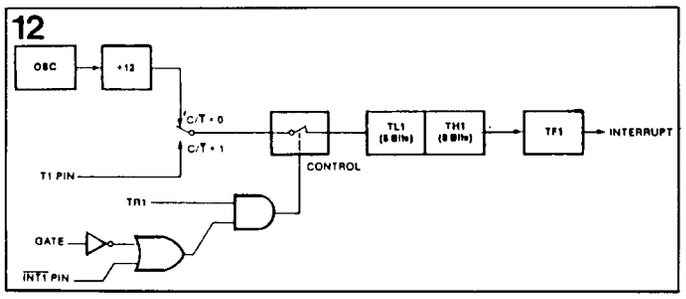
Beszéljük meg először magát az interrupt-rutint, mely az INTT0 címkénél kezdődik *(15. ábra)*. Egy másodperces ütemet egyetlen időzítővel előállítani sajnos közvetlenül nem lehet. Még akkor is maximálisan csak a két számláló-túlcsordulás közötti 65536 mikroszekundumot (mintegy 0,065 másodpercet) tudunk megvalósítani, ha a belső órajelet 16-bites számlálóval osztjuk. Ezért más utat kell járnunk.

4 kHz-es interrupt-gyakorisággal dolgozunk (azaz a másodpercenkénti Timer-Interruptok száma 4000). Két interrupt között tehát pontosan 250 mikroszekundumnak kell eltelnie. Ezt kényelmesen el tudjuk érni, ha a Timer 0-t Auto-Reload-Módusban (MODE2, azaz 2-es üzemmód) üzemeltetjük és 256–250=6-os Reload-értéket használunk. Az interrupt-rutinon belül két egymás után kapcsolt számlálóbytetal (ZAEHL1 és ZAEHL2) 40x100=4000-ig számálunk. A ZAEHL1 változó minden esetben 100-tól kezdődően számál vissza nulláig. Amikor a nullát eléri, akkor ZAEHL2 mindig egyet számál (40-től kezdődően). Amikor ZAEHL2 a 0 értéket elérte, akkor pontosan 250x4000 mikroszekundum telt el. Akkor üzenetet hagyunk a főprogramnak oly módon, hogy a POST „levélszekrény-változóba" 1-es értéket írunk be. Az egész folyamat jól követhető a *16. ábra* blokkdiagrammján.

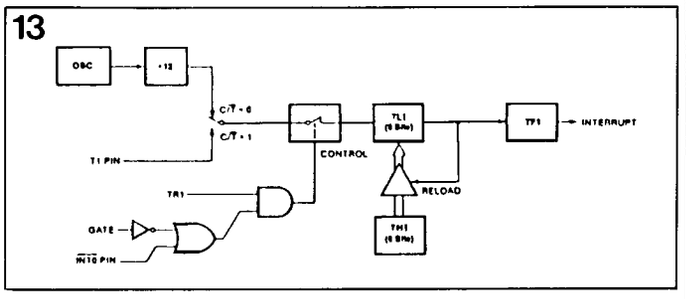
11. ábra. A TMOD Timer-Moder Register bitjei



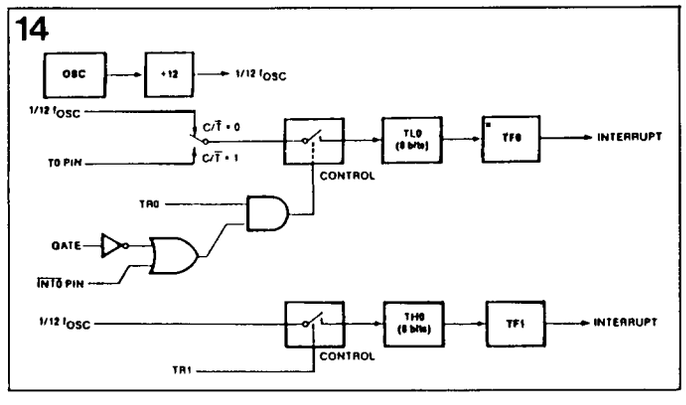
12. ábra. A Timer működése a 0-ás üzemmódban (MODE 0)



13. ábra. A Timer a 2-es üzemmódban (MODE 2)



14. ábra. A Timer a 3-as üzemmódban (MODE 3)



Főprogram

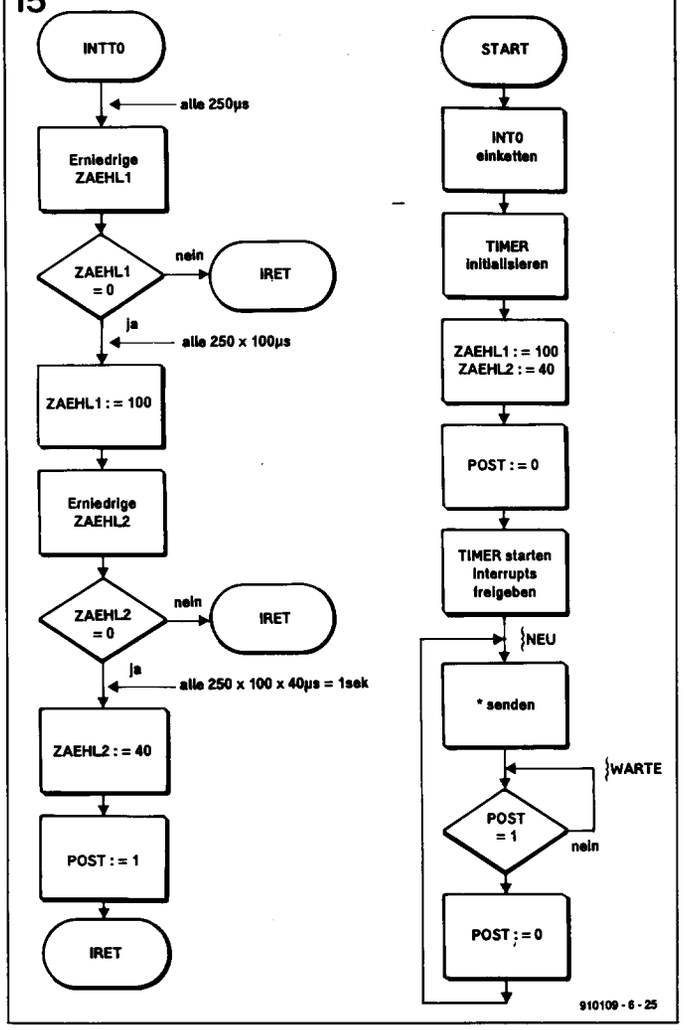
A főprogram legnehezebb feladata a számláló- és interruptvezérlő SFR-ek helyes inicializálása. Először a számláló üzemmódja kerül betöltésre és a RELOAD érték beírása történik meg (a listában a 25...27. sor). A számláló üzemmódjának beállítása során arra is ügyelni kell, hogy az 1-es számlálónak továbbra is a 2-es üzemmódban kell működnie, annak érdekében, hogy a soros interfész számára a baudsebesség generátor szerepét betölthesse. Ezután kerül sor a számláló-változók inicializálására, majd a számláló indítására és az interruptok engedélyezésére (31...33. sor). Ami ezután következik, az csupán egy egyszerű hurok. A NEU-nál kezdődően a program először egy csillagot (*) ad ki a soros interfészre. Ezután a WARTE várakozóhurokban vár a „postaládába" kerülő üzenetre. Az üzenet megérkezését az jelzi, hogy a POST változó értéke nem egyenlő nullával. Ha egy másodperc eltelte után ez a helyzet, akkor „a postaláda kiürítésre kerül" azáltal, hogy a POST-ba újra 0 értéket írunk be (39-es sor). A program futása ezután ismét a hurok elején folytatódik.

Feladatok

Az eddigiekben már sokat megtanultunk. Feladatként vállalkozhatunk arra, hogy egy olyan programot írjunk, amely három analóg bemeneten méri a bemeneti feszültséget és a digitalizált mérési értékeket az RS–232-n át a PC-re továbbítja. Akinek ez túl könnyű, az decimális mérési érték outputot valósíthat meg és beprogramozhatja az egyes értékeknek előírt korrekciós tényezőkkel való beszorzását is.

Alternatívaként el lehet készíteni egy komparátor szoftver programját is. Ez egy olyan program, amely a PC felé vezető illesztőegységre az OK jelzést adja, ha a többfunkciós kártya 2-es bemenetére adott feszültség az 1-es beme-

15



15. ábra. A másodperces óraként működő szoftver blokksémája egy interrupt-rutint tartalmaz

16

```
******* LISTING of EASM51 (BSP11) ******
LINE LOC  OBJ          T       SOURCE
   1 0000                 ; ****** DATEI BSP11.A51 ****************************************
   2 0000                 ;
   3 0000                 IE      EQU   0A8H
   4 0000                 ACC     EQU   0E0H
   5 0000                 PSW     EQU   0D0H
   6 0000                 TCON    EQU   088H     ; neu : Timer/Counter Kontrolle
   7 0000                 TMOD    EQU   089H
   8 0000                 TL0     EQU   08AH
   9 0000                 TH0     EQU   08CH
  10 0000                 ;                      ; 250*100*40 MikroSek = 1 Sek
  11 0000                 T0wert  EQU   250      ; 250 mikrosek. fuer TIMER 0
  12 0000                 ZEIT1   EQU   100      ; Schleifenanzahl fuer ZAEHL1
  13 0000                 ZEIT2   EQU   40       ; Schleifenanzahl fuer ZAEHL2
  14 0000                 ;
  15 0000                        ORG 050H        ; ueber MONITOR RAM
  16 0050                 ZAEHL1  DS  1          ; Software Zaehler 1
  17 0051                 ZAEHL2  DS  1          ; Zaehler 2
  18 0052                 POST    DS  1          ; 1 heisst eine Sekunde ist um , 0 sonst
  19 0053                 ;
  20 0053                        ORG   4100H
  21 4100 90 41 34 [2] START  MOV   DPTR,#INTT0     ; Interruptroutine einketten
  22 4103 74 02    [1]        MOV   A,#2
  23 4105 75 30 40 [2]        MOV   COMMAND,#ccLINK
  24 4108 12 02 00 [2]        LCALL MON
  25 410B 75 8C 06 [2]        MOV   TH0,#256-T0wert ; alle 250 Mikrosekunden
  26 410E 75 8A 06 [2]        MOV   TL0,#256-T0wert ; einen Interrupt von TIMER0
  27 4111 75 89 22 [2]        MOV   TMOD,#022H      ; Beide Zaehler MODUS2 2
  28 4114 75 51 28 [2]        MOV   ZAEHL2,#ZEIT2   ; Software Zaehler vorbesetzen
  29 4117 75 50 64 [2]        MOV   ZAEHL1,#ZEIT1
  30 411A 75 52 00 [2]        MOV   POST,#0         ; keine Post
  31 411D D2 8C    [1]        SETB  TCON.4          ; TIMER0 starten
  32 411F D2 A9    [1]        SETB  IE.1            ; Interrupt von TIMER0 an
  33 4121 D2 AF    [1]        SETB  IE.7            ; Interrupts an
  34 4123 74 2A    [1] NEU    MOV   A,#'*'          ; STERN ausgeben
  35 4125 75 30 01 [2]        MOV   COMMAND,#ccCHR
  36 4128 12 02 00 [2]        LCALL MON
  37 412B E5 52    [1] WARTE  MOV   A,POST          ; auf POST
  38 412D 60 FC    [2]        JZ    WARTE          ; warten
  39 412F 75 52 00 [2]        MOV   POST,#0        ; POST abholen
  40 4132 80 EF    [2]        SJMP  NEU            ; wiederholen
  41 4134                 ;
  42 4134 C0 D0    [2] INTT0  PUSH  PSW            ; sichern
  43 4136 C0 E0    [2]        PUSH  ACC
  44 4138 D5 50 0C [2]        DJNZ  ZAEHL1,IRET1   ; Softwarzaehler 1 erniedrigen
  45 413B 75 50 64 [2]        MOV   ZAEHL1,#ZEIT1  ; bei Erreichen von 0 nachladen
  46 413E D5 51 06 [2]        DJNZ  ZAEHL2,IRET1   ; und softwarezaehler 2 erniedrigen
  47 4141 75 51 28 [2]        MOV   ZAEHL2,#ZEIT2  ; bei Erreichen von 0 nachladen
  48 4144 75 52 01 [2]        MOV   POST,#1        ; und POST deponieren
  49 4147 D0 E0    [2] IRET1  POP   ACC            ; nach Rueckspeicherung
  50 4149 D0 D0    [2]        POP   PSW
  51 414B 32       [2]        RETI                 ; Interruptende
  52 414C                 ;                    MONITOR INTERFACE
  53 414C                 COMMAND EQU 030H   ; MONITOR Kommando Speicherstelle
  54 414C                 MON     EQU 0200H  ; MONITOR Einsprungadresse
  55 414C                 ccLINK  EQU 040H   ; Interrupt Einkettung
  56 414C                 ccCHR   EQU 001H   ; Zeichen ausgeben
  57 414C                         END
********** SYMBOLTABLE (22 symbols) **********
      IE :00A8      ACC :00E0      PSW :00D0      TCON :0088
    TMOD :0089      TL0 :008A      TH0 :008C    T0wert :00FA
   ZEIT1 :0064    ZEIT2 :0028   ZAEHL1 :0050   ZAEHL2 :0051
    POST :0052    START :4100      NEU :4123     WARTE :4128
   INTT0 :4134    IRET1 :4147  COMMAND :0030       MON :0200      910109-6-26
  ccLINK :0040    ccCHR :0001
```

16. ábra. Másodperces óránk listája

net és a 3-mas bemenet feszültsége közé esik.

Tanfolyamunk eddigi részeiben kapott információk alapján a kitűzött feladatok minden nehézség nélkül megoldhatók. ■

15. ábra.
alle 250 µs: 250 µs-onként
Erniedrige ZAEHL1: ZAEHL1 csökkentése
nein: nem
ja: igen
Erniedrige ZAEHL2: ZAEHL2 csökkentése
INT0 einketten: INTT0 befűzése
TIMER initialisieren: TIMER inicializálása
TIMER starten Interrupts freigeben: TIMER indítása, interruptok engedélyezése
NEU: ÚJ
* senden: * outputja
WARTE: VÁRJ (címke)

16. ábra.
neu: Timer/Countern Kontrolle: új: TIMER/számláló ellenőrzés
250 mikrosek. fuer TIMER 0: 250 mikroszek. TIMER 0-nak
Schleifenzahl fuer ZAEHL1: hurokszám ZAEHL1-nek
Schleifenzahl fuer ZAEHL2: hurokszám ZAEHL2-nek
ueber MONITOR RAM: MONITOR RAM-ra
Software Zaehler 1: 1-es szoftverszámláló
Zaehler 2: 2-es számláló
1 heisst eine Sekunde ist um, 0 sonst: 1 jelentése: egy másodperc elmúlt (egyébként 0)
Interruptroutine einketten: interrupt-rutin befűzése
alle 250 Mikrosekunden: 250 mikroszekundumonként
einen Interrupt von TIMER 0: egy interrupt TIMER 0-tól
Beide Zaehler MODUS2 2: mindkét számláló MODUS2 2
Software Zaehler vorbesetzen: szoftverszámlálók beállítása
keine Post: nincs üzenet
TIMER 0 starten: TIMER 0 indítása
Interrupt von TIMER 0 an: TIMER 0 interrupt be
Interrupts an: interruptok be
STERN ausgeben: CSILLAG outputja
auf POST: POST-ra
warten: várakozás
POST abholen: postaláda kiürítése
wiederholen: ismétlés
sichern: biztosítás
Softwarezaehler 1 erniedrigen: 1-es szoftverszámláló csökkentése
bei Erreichen von 0 nachladen: 0 elérésekor utántöltés
und Softwerzaehler 2 erniedrigen: és 2-es szoftverszámláló csökkentése
bei Erreichen von 0 nachladen: 0 elérésekor utántöltés
und POST deponieren: és üzenet elhelyezés
nach Rueckspeicherung: visszatárolás után
Interruptende: interrupt vége
MONITOR Kommando Speicherstelle: MONITOR parancs tárrekesz
MONITOR Einsprungadresse: MONITOR belépési cím
Interrupt Einkettung: interrupt befűzése
Zeichen ausgeben: karakter output