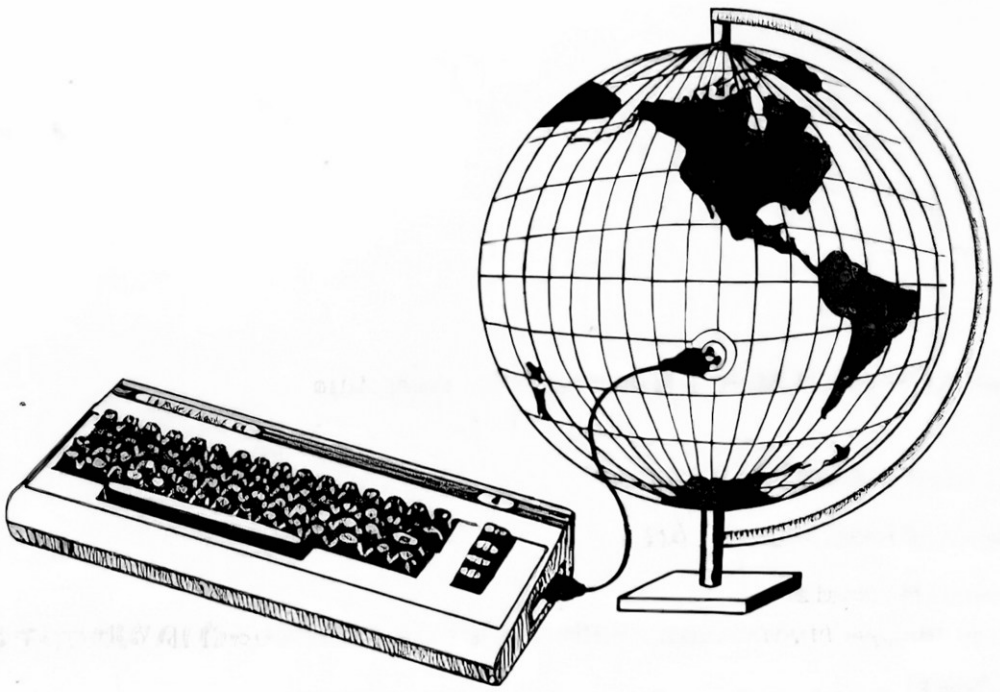
Brückmann

A COMMODORE 64-ES CSATLAKOZÁSI LEHETŐSÉGEI

DATA BECKER – NOVOTRADE

Brückmann

A COMMODORE 64-ES CSATLAKOZÁSI LEHETŐSÉGEI



DATA BECKER – NOVOTRADE

A könyv eredeti címe: Der Commodore 64 und der Rest der Welt (1984)

Fordította: APEX Szervező, Szolgáltató GMK

Lektorálta: DR. LENGYEL JÓZSEF, SZIDAROVSZKY FERENC

A kiadásért felel: RÉNYI GÁBOR, a Novotrade Rt. igazgatója
Budapest, 1988.

Szerkesztette: TARR KÁLMÁNNÉ

Műszaki szerkesztő: ERDŐSI ZOLTÁN

Szedte a Szekszárdi Nyomda
Készült a Somogy Megyei Nyomdaipari Vállalat kaposvári üzemében (10 A/5 ív)

ISBN 963 02 50667

FONTOS TUDNIVALÓ

A jelen könyv keretén belül ismeretetett kapcsolások, eljárások és programok nem tekinthetők szabadalmi oltalom alá eső ipari termékeknek. Ezek elsősorban amatőr és oktatási célokat szolgálnak. A szerzők rendkívül nagy gondot fordítottak a kapcsolások, műszaki adatok és programok helyességére, a részletek kidolgozása során többszöri ellenőrzést végeztek. Mindez azonban nem zárja ki az esetleges hibalehetőségeket.
Az előforduló hibákért és az ebből adódó következményekért a DATA BECKER cég sem szavatosságot, sem jogi felelősséget nem vállal. Az esetlegesen előforduló hibák közlését a szerzők hálásan fogadják.

TARTALOMJEGYZÉK

ELŐSZÓ

Aki a számítógépre gondol, az általában egyszerre gondol a programozásra, játékokra és a kereskedelmi alkalmazásra. Kevés embernek jut eszébe azonban, hogy pontosan a modern mikroszámítógépek alkalmasak – kiváló ár- és teljesítményviszonyuk, valamint sokoldalúságuk miatt – a műszaki alkalmazásra. Ez hobbi- és hivatásos szinten egyaránt érvényes. Ezen a területen a szakirodalom is igen szegényes. Ezt a hiányt szeretnénk ezzel a könyvvel némileg pótolni. Készítője Rudolf Brückmann, olyan műszaki szakember, akinek a számítógép egyszerre hobbi és hivatás is. 1980-ban tűnt fel a számítástechnika világában. A szokástól eltérően nem vásárolt, hanem épített magának egy 8080 alapú gépet. A könyvben szereplő összes kapcsolást is ő maga fejlesztette ki. Nemcsak a Commodore 64-est ismeri úgy, mint a tenyerét, hanem az ehhez kapcsolódó területeken is megalapozott tudása van. Ezt a tudást adja át most Önnek ezzel a könyvvel. Sok örömet kívánunk az olvasásához és sok sikert az alkalmazásához.

Dr. Achim Becker

BEVEZETÉS

A Commodore 64-es csatlakozási lehetőségei – a számítógép szemszögéből vizsgálja a „külvilágot".
Ez a külvilág a digitális és analóg technika széles területét jelenti.
A számítástechnika alapelemeivel, a számrendszerekkel fogjuk kezdeni, majd lépésről lépésre haladunk az egyszerű kísérletezéstől az összetett mérési- és vezérlési kapcsolások felé a szükséges programozási ismeretekkel együtt.
A könyvben olyan mérőkészülékekkel ismerkedhetünk meg, mint pl. a frekvenciamérő és a digitális voltmérő, vagy olyan kapcsolásokkal találkozunk, mint a hangképzést segítő be- és kiviteli áramkör, de hasznos ötleteket meríthetünk pl. hangkapcsoló készítéséhez is.
Természetesen a könyvnek is vannak bizonyos elvárásai velünk szemben. Meg kell tanulnunk kapcsolási rajzot olvasni, ismernünk kell a kereskedelemben kapható alkatrészek (pl. kondenzátorok, ellenállások, ...) jelöléseit, meg kell tanulnunk, hogy mi az az INVERTER, a NAND-kapu, vagy hogy a TTL IC 5V-os tápfeszültséget igényelnek.
Ha már tisztában vagyunk azzal, hogy a SYNTAX ERROR hibaüzenet nem a 64-es meghibásodását jelenti, akkor már az induláshoz szükséges BASIC-programozási tudásunk is megvan.
A programokat az 1. fejezetben részletesen tárgyaljuk. A további fejezetekhez újabb ismeretek szükségesek, különösen akkor, ha a programokat nem csak begépelni szeretnénk, hanem érteni is. Néhány fejezetben, a jóval kedvezőbb feldolgozási sebesség miatt, nem kerülhettük el a gépi kódú programozást.
Aki eddig azért nem használt gépi nyelvű programokat, mert úgy gondolta, hogy számára a BASIC is megfelelően gyors, valószínűleg kellő ösztönzést kap majd könyvünktől arra, hogy ezt a rendkívül hatékony programozási módot kipróbálja. Hamar belátja, hogy éppen a hardver és szoftver összefüggései miatt a gépi nyelvű programozás gyakran elengedhetetlen. A könnyebb kezelhetőség kedvéért a könyvben szereplő összes gépi kódú rutin BASIC-betöltő listáját is közöljük.

A kapcsolások megépítéséhez természetesen szükségünk van némi szerszámkészletre. Kezdetben elegendő egy finom hegyű forrasztópáka (30 watt), néhány fogó, egy csipesz és egy univerzális mérőműszer.
A C 64-esről többnyire biztosítható a külső kapcsolások áramellátása, ennek ellenére jó, ha van egy rövidzárbiztos hálózati tápegységünk is +5 és +12 V-os kimenettel.

A könyvben közölt összes kapcsolást és programot ellenőrizték, megépítésre és futtatásra alkalmasnak találták.* Ezért nem kell rögtön a könyvet szidni, ha egy kapcsolás vagy program nem működik azonnal tökéletesen. Először ellenőrizzük a rendszer felépítését, nézzük át a programot, majd próbáljuk meg maximális pontossággal behatárolni a hiba forrását. Ha ez sikerült, akkor a leírás segítségével már könnyen kijavíthatjuk a hibát. Jegyezzük meg: hibakereséskor fontosabb a következetesség és némi fantázia, mint egy drága mérőkészülék.
Végezetül néhány gyakorlati tanács: Ha a kapcsolások megépítéséhez nyomtatott áramköröket használunk (lyukrácsos vagy maratott), érdemes az IC-ket foglalatba helyezni. Ezek olcsón beszerezhetők és az így beépített alkatrészeket később sokkal egyszerűbb kiemelni. A gyors építéshez kísérleti feltűző tábla használatát javasoljuk. Ezen a kisebb kapcsolások igen gyorsan felépíthetők, a változtatások könnyen elvégezhetők és az alkatrészek újra felhasználhatók.

* (Ez természetesen csak akkor igaz, ha pontosan a könyv által említett alkatrészeket használjuk fel. Néhány kapcsolási rajzon olyan alkatrészek is szerepelnek, amelyek hazánkban nehezen beszerezhetők, vagy drágák. Ezek más, olcsóbb, vagy könnyebben hozzáférhető típusokkal helyettesíthetők ugyan, de ez esetben a kapcsolás, sőt esetenként a hozzá tartozó program is módosulhat. – A fordító megjegyzése.)

1. SZÁMOLJUNK ÚGY, MINT A SZÁMÍTÓGÉP

Attól, aki máris bekapcsolta a forrasztópákát és várja az első építési javaslatot, egy kis türelmet kérünk. Rövid ideig még a száraz elmélettel kell foglalkoznunk.

Először is meg kell ismerkednünk a különböző számrendszerekkel, pontosabban szólva hárommal: a kettessel (bináris), a tizenhatossal (hexadecimális) és a tízessel (decimális). Mire jó mindez? – gondolhatnánk, hiszen a megszokott tízes számrendszer tökéletesen megfelel számunkra. Végül is tíz ujjunk van és ebben a számrendszerben, sok fáradsággal, már az általános iskolában megtanultunk számolni.
A bináris és hexadecimális számok azonban pontosan a külső egységek programozásánál igen fontosak. E számrendszerek ismerete nélkül nehezen érthetők meg a könyvben szereplő kapcsolások.

Tízes számrendszerben szerzett tapasztalatainkat azonban felhasználhatjuk a többi számrendszernél is. Nézzük csak meg pontosan, hogyan is működik!
Ha egy számot az ujjainkon akarunk kiszámolni, a hüvelykujjal kezdjük és addig számolunk, még minden ujjunk el nem fogy. Ekkor éppen 10-nél tartunk. Megjegyezzük az „átvitelt" és a számolást ismét a hüvelykujjnál kezdjük.
Vizsgáljuk meg ezt az „átvitelt" egy példán keresztül. Képzeljünk magunk elé 247 Ft-ot, csupa egyforintosból. Képezzünk az érmékből 10-es (10 Ft) oszlopokat. Egy idő után számos érmeoszlop lesz előttünk és 7 db maradék egyforintos, amiket természetesen már nem rakhatunk tízes oszlopba. Ezt a 7 egyforintost tegyük félre és a továbbiakban foglalkozzunk a 10 forintot tartalmazó oszlopokkal. Rakjunk ezekből is tízet-tízet egymásra. Ha jól végeztük el, most két oszlopunknak kell lennie 100-100 forinttal és négynek 10-10 forinttal. Továbbá ott van még a 7 db egyforintosunk. Mivel a 100 érmét tartalmazó oszlopokból már csak kettő van, több tízes csoportot már nem képezhetünk. Tegyük tehát félre a két oszlopunkat, mint maradékot. Mivel most már minden pénzünk „maradék"-ként szerepel, befejeztük az eljárást.
Most nézzük meg magát a 247-es számot. Világosan felismerhető, hogy a számjegyek számon belüli helyzete miként határozza meg értéküket.

a számjegy helye:	2	1	0
az oszlopok száma:	2	4	7

	az oszlop helyiértéke		az oszlop értéke
szer 10^0 (= 1)		=	7
szer 10^1 (= 10)		=	40
szer 10^2 (= 100)		=	200
összesen:			247

Számítógépünk a tízes számrendszert nem tudja használni, mert képletesen szólva, nincs tíz ujja. Ehelyett csupán két elektromos állapot megkülönböztetésére képes: áram van, áram nincs. Kénytelen tehát ezekkel, vagyis a kettes számrendszerrel dolgozni.

A kettes számrendszerben tehát összesen két számjegyünk van a számok ábrázolásához: az 1 (áram van) és a 0 (áram nincs). Kérdés, hogyan kell ezekkel számolni. Nagyon egyszerűen. Pontosan ugyanúgy, mint az előbb tettük. Nézzük a példánkat: a 247 db egyforintost először kettes oszlopokra osztjuk. Amikor elfogynak az érmék, összesen 123 ilyen oszlopunk és 1 db maradék egyforintosunk lesz. Ezt félretesszük és most a kettes oszlopokat kezdjük (szintén kettesével) csoportosítani. Az eredmény 61 darab négyforintot tartalmazó és 1 db kétforintot tartalmazó oszlop. Ezt a maradékot megint félretesszük és folytatjuk a csoportosítást (mindig kettesével) mindaddig, amíg az összes pénzünk „maradék" nem lesz. Ha mindent jól csináltunk, végezetül 7 pénzoszlopunknak kell lennie. Rakjuk ezeket sorba úgy, hogy balra kerüljön a legtöbb érmét tartalmazó (tehát utolsóként kapott), jobbra pedig a legkevesebbet tartalmazó oszlop.

Az egész kb. így fog kinézni:

az oszlop helye:	7	6	5	4	3	2	1	0
Az oszlop darabszáma	1	1	1	1	0	1	1	1

	az oszlop helyiértéke		az oszlop értéke
szer	2^0(= 1)	=	1
szer	2^1(= 2)	=	2
szer	2^2(= 4)	=	4
szer	2^3(= 8)	=	0
szer	2^4(= 16)	=	16
szer	2^5(= 32)	=	32
szer	2^6(= 64)	=	64
szer	2^7(= 128)	=	128
		összesen:	247

A hármas helyen szereplő 0 azt jelenti, hogy a negyedik csoportosításkor nem volt maradék, tehát 8 forintot tartalmazó pénzoszlopunk nincs.
Ezzel a decimális 247-et binálissá alakítottuk. Az eljárást összefoglalóan így írhatjuk le: A decimális számot addig osztjuk folyamatosan 2-vel, míg 0-t nem kapunk, majd az egyes osztások maradékait (ami csak 1 vagy 0 lehet) fordított sorrendbe egymás mellé írjuk.

A fenti vázlat alapján az átszámítás fordítva is pillanatok alatt elvégezhető. Ezt meg is tesszük. Kiszámítjuk a létező legnagyobb nyolcjegyű bináris szám decimális értékét: Mivel a kettes számrendszerben csak a 0-t és az 1-est használjuk, a legnagyobb nyolcjegyű bináris szám: 11111111. Ebből a számból a fenti vázlat alapján a következő összeget kapjuk: 128+64+32+16+8+4+2+1 = 255

Hasonló módon a legkisebb szám: 00000000, azaz 0.

Példánkban nem véletlenül választottunk olyan számot, aminek bináris megfelelője 8 számjegyből áll. A 64-es mikroprocesszora ugyanis egy ún. nyolcbites mikroprocesszor. A bit a legkisebb információs egység, aminek két állapota lehet: 1 (igen), vagy 0 (nem).
Ennek megfelelően nyolc bittel legfeljebb nyolcjegyű bináris számokat ábrázolhatunk.
Eredeti példánknál maradva: ezekkel az „igen"-ekkel és „nem"-ekkel ábrázolhatjuk, hogy milyen értékű érmeoszlopokkal rendelkezünk és milyen értékűekkel nem.
Minden meglévő oszlopot 1-essel és minden hiányzót 0-val jelölünk.
A digitális technikában más jelölések is léteznek. Ilyenek pl. a magas (HIGH) és az

alacsony (LOW). Ezek különböző feszültségszintekre utalnak. Ha egy bit magas szinten van (1), feszültséget mérhetünk (C 64-es esetében ez +2,8... +5 V). Ezzel szemben alacsony szinten (bit = 0) feszültséget nem mérhetünk, és C 64-esnél ez a +0,8 V-nál kisebb feszültségszintet jelenti.
8 bittel összesen 256 különböző értéket ábrázolhatunk (0...255), ez tehát a mikroprocesszor számtartománya.

Ezek után ismerkedjünk meg a tizenhatos számrendszerrel. Ugyanúgy számolunk, mint eddig, de vegyük úgy, mintha 16 ujjunk lenne. Számjegyünk azonban csak 10 van, ezért a többi szám jelölésére kölcsönvesszük az ABC első 6 betűjét. A hexadecimális számjegyek tehát a következők:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, ahol A = 10, B = 11, ... F = 15. A decimális 16 hexadecimális megfelelője: 10.
Az átvitelt a többi számrendszerhez hasonlóan értelmezzük.

Vizsgáljunk meg közelebbről egy egyjegyű hexadecimális számot. Könnyű belátni, hogy ezzel 16 különböző értéket (állapotot) ábrázolhatunk. Egy ilyen hexaszám bináris megfelelője max. 4 számjegyből állhat. Ha a processzor 8 bitjét (amit adatbusznak is nevezünk) két négybites félre osztjuk, akkor minden egyes adatként használt bináris számot egy kétjegyű hexadecimális számmal fejezhetünk ki. (Az adatbusz mellett létezik ún. címbusz is, ami 16 bites és a tárolócímzést szolgálja, de erről majd később.)
Ennek az ábrázolási módnak még az az igen nagy előnye, hogy sokkal könnyebb megjegyezni pl. azt, hogy ØE, mint azt, hogy 00001110.
Maga a HEXADECIMÁLIS elnevezés nem pontos. Helyesen SEDECIMÁLIS-t kellene mondanunk, de a köztudatban az előző terjedt el, ezért, bár helytelennek tartjuk, mi is ennél maradunk. Több számrendszer egyidejű alkalmazása esetén problémát jelenthet egyértelmű meghatározásuk. Pl. a 10 mindhárom számrendszerben létezik, de értéke különböző. A kettes számrendszerben 2-t, a tízesben 10-et és a tizenhatosban 16-ot jelent. A számok biztonságos megkülönböztetésére ezért bevezették az alábbi jelöléseket:

% 10 – bináris
$ 10 – hexadecimális
10 – decimális

A továbbiakban ezeket a jelöléseket mi is használni fogjuk. Sajnos ez a megállapodás csak a 65xx típ. processzorokra érvényes. 8080, vagy Z80 esetében (pl. CP/M cartridge) a hexadecimális számokat a szám elé írt „H"-betűvel jelölik. Ebben a könyvben a 65xx-re vonatkozó jelöléseket használjuk.

A teljesség kedvéért még a nyolcas (OKTÁL) számrendszert is megemlítjük. Eredete az egészen korai mikroszámítógépek idejére vezethető vissza.
Ebben a rendszerben összesen 8 állapotot különböztetnek meg (0-7), amihez max. 3 bitre van szükség.

1.1. PEEK, POKE és hasonlók

Tekintettel arra, hogy e könyv célja elsősorban a C 64-es hardver jellegű kiegészítési lehetőségeinek bemutatása, nem foglalkozhatunk a BASIC-nyelv részletes tárgyalásával. Van azonban néhány utasítás, amit feltétlenül ismernünk kell, ezért ezekre itt külön kitérünk.

a) POKE

Alakja: POKE cím, adat

Ezzel az utasítással egy adott tárolórekeszbe adatot írhatunk.
A mikroprocesszor 16 bittel (címvezeték, ill. együttesen címbusz) rendelkezik ahhoz, hogy a számítógép tárolórekeszeit megcímezze. A bitek mindegyike 0 vagy 1 lehet, ezért a 16 bit együttesen 65536 különböző értéket vehet fel (0-65535).
Ez azt jelenti, hogy a processzor összesen 65536 tárolórekeszt képes megcímezni és ezáltal elérni. A tárolórekeszek „szélessége" 8 bit (= 1 byte), amivel, mint tudjuk, 256 különböző számot ábrázolhatunk. Ebből következik, hogy az adat értéke csak 0 és 255 közötti egész szám lehet.
A POKE 49152, 255 utasítással tehát a 49152 ($ C000) című tárolórekeszbe 255-öt ($ FF) írunk. A rekesz eredeti tartalma felülíródik, vagyis elvész.

b) PEEK

Alakja: PEEK (cím)

Ez az utasítás a POKE fordítottja, vagyis a megadott tárolócím tartalmának kiolvasására szolgál. Függvény típusú, ami azt jelenti, hogy kizárólag valamilyen összefüggésben vagy egyéb kifejezésben szerepelhet (pl. A = PEEK (cím) vagy PRINT PEEK (cím)).
Ha 0-nál kisebb, vagy 65535-nél nagyobb címet próbálunk használni, a gép hibaüzenettel válaszol, mert ezek az értékek kívül esnek a 16 bittel meghatározható tartományon.

c) AND, OR és NOT

Ezek az utasítások az ún. logikai utasítások csoportjába tartoznak. Ezekkel különbözőképpen módosíthatjuk a tároló byte-jainak tartalmát. Nézzünk erre egy egyszerű példát:
Képzeljünk el 8 db relét, amikkel lakásunkban ki- és bekapcsolhatjuk a kávéfőzőt, a világítást, a bojlert, stb. A reléket az 56577-es tároló címen keresztül, az adatbusz egy-egy vezetékével vezéreljük úgy, hogy ha a bit magas szintű (1), akkor a relé bekapcsol, ha alacsony szintű (0), akkor elenged, azaz kinyit. A reléket megkülönböztetésül 0 ... 7-ig megszámozzuk. Ezek a jelölések megegyeznek az adatbusz bitjeinek szokásos jelölésével.

Az adatbusz bitjei:	7	6	5	4	3	2	1	0
a relé száma:	7	6	5	4	3	2	1	0

Ha a 4-es relét akarjuk bekapcsolni, a következő bitmintát kapjuk:

az adatbusz bitjei:	7	6	5	4	3	2	1	0
az adatbusz állapota:	0	0	0	1	0	0	0	0

Ezt a bitmintát át kell alakítanunk decimális számmá. Mivel csak egy bitet kapcsoltunk be, ez nagyon egyszerű: $2^4 = 16$.

A 4-es relét tehát a

POKE 56577, 16

utasítással kapcsolhatjuk be.
Amennyiben a 6-os és 2-es relét együtt akarjuk bekapcsolni, az utasítás a következők szerint alakul:

POKE 56577, 2 ↑ 2 + 2 ↑ 6

vagy

POKE 56577,4+64

vagy

POKE 56577,68

Nézzük a logikai kapcsolatokat. A 6-os és 2-es relé után kapcsoljuk még be a 7-est is. Erre a logikai VAGY kapcsolat alkalmas, amit BASIC-ben az OR-utasítás valósít meg. A VAGY kapcsolatot úgy kell elképzelni, mint két, párhuzamosan kötött villanykapcsolót. Ahhoz, hogy a lámpa kigyulladjon, elég csak egy (tehát VAGY az egyik VAGY a másik) kapcsolót bekapcsolni.
Nézzük a bitmintát:

	7	6	5	4	3	2	1	0	
	0	1	0	0	0	1	0	0	= 68
VAGY	1	0	0	0	0	0	0	0	= 128
	1	1	0	0	0	1	0	0	= 196

Ha akár az egyik, akár a másik tényezőben egy bitet bekapcsolunk, ugyanaz a bit az eredményben is bekapcsolódik. Azt mondhatnánk erre, hogy ezt az eredményt egy sima összeadással is megkaphattuk volna. Ebben az esetben ez igaz, de itt az is lényeges, hogy a művelet megismétlésekor az eredmény ne változzon. Sima összeadásnál a második művelet eredménye már 324 lenne, ami a POKE-utasításban adatként felhasználva hibaüzenethez vezet.
128-nál kisebb értékek esetében az összeadás ismétlése az eredmény eltorzulásához vezet, vagyis egészen más bitek kapcsolódnak be, mint amiket szeretnénk.

VAGY kapcsolat alkalmazásakor mindez nem következik be. Akárhányszor ismételjük is meg, az eredmény nem változik. BASIC-ben a programsor helyesen a következő:

POKE 56577, PEEK (56577) OR 128

A PEEK-utasítással megvizsgáltuk a relék aktuális állapotát, OR 128-cal bekapcsoltuk a 7. bitet és a POKE-utasítással gondoskodtunk a tárolásról.
Ilyen módon bármelyik relét a többitől függetlenül bekapcsolhatjuk.

A relék egymástól független kikapcsolására az ÉS kapcsolat alkalmas, amit BASIC-ben az AND-utasítás valósít meg.
Előző példánknál maradva az ÉS kapcsolat két, sorosan kapcsolt villanykapcsolóhoz hasonlítható.
A lámpa csak akkor világít, ha az egyik ÉS a másik kapcsoló is be van kapcsolva.
Az előbb bekapcsolt három relé közül most kapcsoljuk ki a 2-est, úgy hogy a többi ne változzon!

	7	6	5	4	3	2	1	0	
	1	1	0	0	0	1	0	0	= 196
ÉS	1	1	1	1	1	0	1	1	= 251
	1	1	0	0	0	0	0	0	= 192

Tehát a kikapcsolandó bit helyére 0-át, a többi helyre pedig 1-est írunk.

Ha egy bit az egyik tényezőben is ÉS a másik tényezőben is bekapcsolt, akkor az eredmény megfelelő bitje is bekapcsolódik. Az összes többi helyen, ahol csak legfeljebb az egyik bit értéke 1, az eredmény 0.

BASIC-ben mindez a következőképpen alakul:

POKE 56577, PEEK (56577) AND 251

Itt is a PEEK-utasítással vizsgáljuk meg a relék aktuális állapotát, majd az AND 251 utasítással töröljük a 2-es bitet. Az eredményt a POKE-utasítással tároljuk.

Felvetődhet a kérdés, hogy a második tényezőben miért kapcsoltuk be az 6...3 és 1...0 biteket is. Nos, ennek az a magyarázata, hogy nem ismerhetjük pontosan a byte-on belüli bitek állapotát, vagyis, hogy melyik 0, és melyik 1. Ezért azoknak a biteknek a helyére, amelyeket meg akarunk tartani, a kikapcsolási maszkban (második tényező) 1-est kell írnunk.

Ennyi elmélet után jöhet az első gyakorlat!
Az 5.1. fejezetben találunk egy olyan kapcsolást, ami 8 LED (világító dióda) userporton keresztüli csatlakoztatására mutat egy megoldást. Ezzel kipróbálhatjuk fenti példáinkat és elmélyíthetjük a tanultakat.

2. A SZÁMÍTÓGÉP NYÚJTOTTA LEHETŐSÉGEK

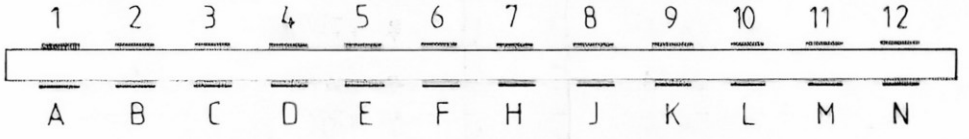
C 64-esünket nagyszámú, univerzális csatlakozási lehetőséggel látták el, fantasztikus illesztési lehetőségeket kínálva ezzel saját építésű készülékeink számára. Ezek közül legsokoldalúbb az ún. bővítő, vagy idegen szóval expansion port. Sajnos viszonylag nehezen kezelhető. Szerényebb, de mégis bőséges illesztési lehetőséget a felhasználói vagy közismertebb nevén user-port. A két vezérlő (vagy control) -portot általában a botkormány és a paddle-k csatlakoztatási lehetőségeként ismerik. Majd látni fogjuk, hogy egyéb célokra is felhasználhatók. Végezetül a magnetofon csatlakozási helyét, a kazetta (casetten) -portot említjük meg. Ez is érdekes, új alkalmazási lehetőségeket kínál.

Nézzük először részletesen a felhasználói- és bővítő-portot.

2.1. A felhasználói (vagy user)-port

Ez a csatlakozási hely a Commodore cég valamennyi számítógépén különleges jelentőségű. Ez éppúgy érvényes a legendás PET 2001-re, mint a VC 20-ra, vagy a 600-as és 700 sorozatú új készülékekre.
A felhasználói-port a C 64-es legegyszerűbben kezelhető csatlakozási lehetősége.
Az illesztéshez egy 24 pólusú, 3,96 mm osztású csatlakozó dugót használhatunk. Az érintkezők bekötése a következő:

1	FÖLD	2	+5 V	3	$\overline{\text{RESET}}$	4	$\overline{\text{CNT 1}}$
5	SP1	6	CNT 2	7	SP2	8	$\overline{\text{PC 2}}$
9	ATN IN	10	9VAC	11	9VAC	12	FÖLD



A	FÖLD	B	$\overline{\text{FLAG 2}}$	C	PBO	D	PB1
E	PB2	F	PB3	H	PB4	J	PB5
K	PB6	L	PB7	M	PA2	N	FÖLD

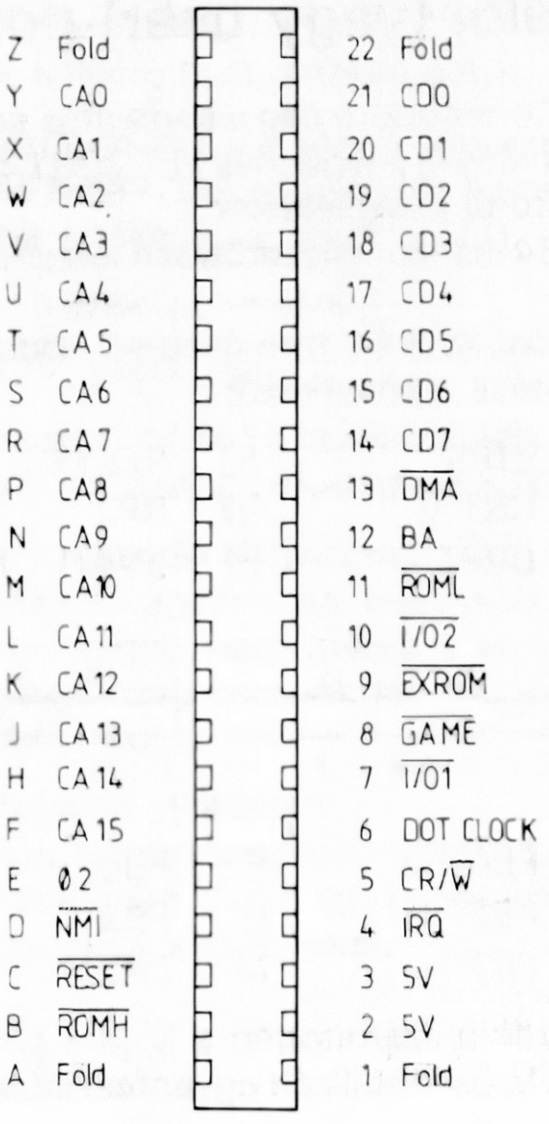
Az 1, 12, A és N érintkezők a számítógép ∅ V-jára csatlakoznak.
A 2-es érintkezőn egy +5 V-os stabilizált egyenfeszültség áll rendelkezésünkre,

ami max. 100 mA-rel terhelhető. A 10-es és 11-es érintkezőkről 9 V-os váltófeszültséget vehetünk le, szintén max. 100 mA-ig.
A 3-as vezeték a processzor RESET-vezetékével áll összeköttetésben. Ez teszi lehetővé azt, hogy a számítógép bekapcsolásakor keletkező RESET-jelet a külső egységek is használhassák. (Az 1/3 érintkezőkre külső RESET-gombot csatlakoztathatunk.)

2.2 A bővítő (vagy expansion)-port

A bővítő-port általában a játék-modulok, BASIC-bővítők vagy be/kimeneti egységek (pl. IEC-busz, 80 jel/sor – kártya, stb.) csatlakoztatási lehetősége. Modul csatlakozónak is nevezik.
Itt állnak rendelkezésünkre a legfontosabb processzor- és rendszerjelek. A C 64-es speciális felépítése lehetővé teszi, hogy az I/O (be/kimeneti) – egységen keresztül a 6510-es helyett más mikroprocesszort is alkalmazzunk (pl. CP/M).

A 44 érintkezővel ellátott csatlakozó lábkiosztása a következő:
(Megjegyzés: az érintkezők általánosan használt elnevezése a „láb".)



A felhasználói porthoz hasonlóan a négy szélső érintkező a földelésre csatlakozik.
A 2-es és 3-as érintkezőkről +5 V-os stabilizált egyenfeszültséget vehetünk le.
Az F-Y érintkezőkön található a processzor cimbuszának (CA15–CA0) kivezetése. A processzor a címbuszon keresztül címezi meg a RAM- és ROM-terület byte-jait, valamint a külső egységeket. DMA (Direkt Memory Acces = közvetlen tárelérés) üzemmódban ezeken a vezetékeken keresztül egy külső processzor, ill. egy másik számítógép is közvetlenül hozzáférhet a tárolóhoz.
Az adatbusz (CD7–CD0) kivezetése a 14-21-es érintkezőkön van. Ezeken a vezetékeken keresztül zajlik az adatforgalom. A címbuszhoz hasonlóan ezt is használhatják külső egységek, mint pl. DMA, játék-modul, stb.

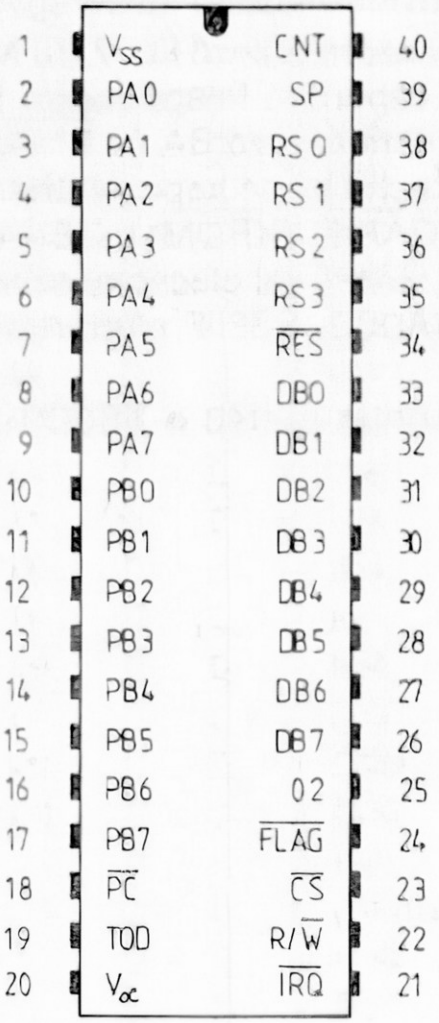
A RESET-jel a C kivezetésre érkezik. Feladata ugyanaz, mint a felhasználói port esetében.
Az E érintkező a processzor órajelének (Ø2) kimenete. A jel frekvenciája kb. 985 kHz.
A 4-es érintkezőn az $\overline{\text{IRQ}}$, a D-n pedig az $\overline{\text{NMI}}$ a megszakításokat vezérli.
Az 5-ös érintkezőre érkező R/W-jel az írási és olvasási műveleteket vezérli. A két buszhoz hasonlóan DMA üzemmódban külső egység is használhatja.
A DOT CLOCK a 6-os érintkezőre érkező kb. 7,88 MHz frekvenciájú jel. Ez az ún, video-órajel, amit a VIC a képernyő felépítésekor használ.
A 12-es és 13-as érintkezőkre érkező BA és DMA jeleket egy másik proceszor működésekor, ill. a DMA technikával kapcsolatban használjuk.
A 8-as és 9-es érintkezők (GAME, $\overline{\text{EXROM}}$) a C 64-es tárfelosztásának átszervezésére használhatók. Ha a GAME-jel alacsony szinten van, kikapcsolódik a BASIC-ROM és helyette a $A000–$BFFF címtartományban egy külső ROM-ot használhatunk.
Az $\overline{\text{EXROM}}$-jel alacsony szintje esetén a $8000-$9FFF címtartományú RAM kapcsolódik ki.

3. A CIA 6526

A CIA (Complex Interface Adapter) 6526-os az ismert VIA 6522-es továbbfejlesztett változata. Legfontosabb jellemzői a következők:

- 2-szer 8 bites párhuzamos I/O kapu
- 2 handshake (=kézfogás) bemenet a párhuzamos adatátvitelhez
- 2 16 bites, kaszkádolható időzíthető
- 1 24 órás (AM/PM) óra, programozható riasztójelzéssel (ALARM)
- 8 bites léptető (shift)-regiszter a soros I/O kapuhoz

A C 64-esben a CIA-k több feladatot is ellátnak, többek között a billentyűzet lekérdezését, az IEC-busz vezérlését, stb.



A 6526-os érintkezőinek kiosztása:

PA0-PA7 és PB0-PB7: Az A és B a 8 bites, párhuzamos I/O kapu kivezetései. Tetszés szerint használhatók ki- és bemenetként.

$\overline{\text{PC}}$ és $\overline{\text{FLAG}}$: Ezek az ún. handshake (= kézfogás) vezetékek. A $\overline{\text{FLAG}}$-re érkező alacsony szintű jel a B kapura vonatkozóan az adatok érvényességét jelzi. Egyidejűleg az ICR (Interrupt-Control-Register = megszakítás-vezérlő-regiszter) 4. bitje is bekapcsolódik. Ezzel a B kapurá érkező adatok bevihetők a tárolóba. Ha a processzor elérte a B regisztert, a $\overline{\text{PC}}$ érintkezőre egy rövid, alacsonyszintű impulzus érkezik.

$\overline{\text{CNT}}$: (Count) Tetszés szerint bemenetként vagy kimenetként használható. Az időzítővel összekapcsolva ún. triggerbemenetként szolgál, amire a számlálandó impulzusok érkeznek. A soros léptetőregiszterrel összekapcsolva a CNT-n megjelenik az időzítő által kialakított léptetési ütem.

TOD: (Time Of Day = valós idejű óra) Erre az érintkezőre érkezik a valós idejű órát működtető ütemjel, amit az 50 Hz-es hálózati frekvencia vezérel. Ezáltal az óra „hosszú idejű pontossága" rendkívül jó.

$\overline{\text{SP}}$: A léptetőregiszter be- és kimenete.

DB7–DB0: Ezek az érintkezők a processzor adatbuszával vannak összeköttetésben. Rajtuk keresztül zajlik a processzor és a CIA-k közti adatforgalom.

A többi érintkező számunkra pillanatnyilag érdektelen, ezért ezekkel itt nem foglalkozunk.

A CIA 16 saját regiszterrel rendelkezik, amelyek az RSØ-RS3 érintkezőkön keresztül vezérelhetők. Ez a négy érintkező összeköttetésben van az alsó négy címvezetékkel, ezáltal a CIA regiszterei úgy kezelhetők, mint a normál tárolórekeszek.

A CIA 6526 belső regiszterei:

0. regiszter — PRA (az A kapu regisztere)
Ez a regiszter az A kapu 8 vezetékét képviseli. Bitjei külön-külön kimenetként és bemenetként is programozhatók.

1. regiszter — PRB (a B kapu regisztere)
Ugyanaz, mint a PRA, de a B kapura vonatkoztatva.

2. regiszter — DDRA (az A kapu adatirány regisztere)
Ennek a regiszternek a tartalmától függ, hogy az A kapu bitjei kimenetként, vagy bemenetként működnek-e. Amelyik bit értéke ebben a regiszterben 1, az a PRA-ban kimenetként, amelyik pedig 0, az bemenetként működik.

3. regiszter — DDRB (a B kapu adatirány regisztere)
Ugyanaz, mint a DDRA, de a B kapura vonatkoztatva.

4. regiszter TA LO (az A időzítő (timer) alsó byte-ja)
Kiolvasáskor az időzítő pillanatnyi állapotának alsó byte-ját adja, valamint ide írjuk be a visszaszámlálás kezdőértékének alsó byte-ját is.

5. regiszter TA HI (az A időzítő felső byte-ja)
Ugyanaz, mint a 5. regiszter, de a felső byte-ra vonatkozóan.

6. regiszter TB LO (a B időzítő alsó byte-ja)
Ugyanaz, mint a 4. regiszter, de a B időzítőre vonatkoztatva.

7. regiszter TB HI (a B időzítő felső byte-ja)
Ugyanaz, mint az 5. regiszter, de a B időzítőre vonatkoztatva.

8. regiszter TODT (a valós idejű óra tizedmásodpercei)
A 0...3 bitek a valós idejű óra tizedmásodperceit tárolják, BCD kódban. Ennek megfelelően csak 0 és 9 közti értékeket vehet fel.
A 4...7 bitek értéke mindig 0.
A regiszter írásakor a 15. regiszter 7. bitjétől függően vagy riasztóidőt (bit = 1) vagy óraidőt (bit = 0) írunk.

9. regiszter TODS (a valós idejű óra másodpercei)
Ebben a regiszterben a 0...6 biteket használjuk.
A valós idejű óra másodperceit rögzíti BCD-kódban oly módon, hogy a 0...3 bitekbe az egyes, a 4...6 bitekbe pedig a tízes helyiértéknek megfelelő szám kerül.
(Pl. 23 másodperc = 0...3 bitekben 3, 4...6 bitekben 2)

A regiszter írásakor szintén figyelembe kell venni a 15. regiszter 7. bitjét.

10. regiszter TODM (a valós idejű óra percei)
Ugyanúgy működik, mint a 9. regiszter, de a percekre vonatkoztatva.

11. regiszter TODH (a valós idejű óra óraértékei)
A regiszter a valós idő óráit tartalmazza BCD kódban, úgy hogy a 0...3 bitekbe az egyes, a 4. bitbe a tízes helyiértékű szám kerül (ld. 9. regiszter)
Az 5. és a 6. bit mindig 0.
A 7. bit jelentése 1 = délelőtt, 0 = délután.

A regiszter írására ugyanaz vonatkozik, mint a 9. regiszter esetében.

12. regiszter SDR (soros adatregiszter)
Ez tulajdonképpen egy léptető-regiszter, amibe az adatok sorosan írhatók be és ugyanúgy olvashatók ki.

13. regiszter ICR (megszakítást vezérlő regiszter)

Ez a regiszter jegyzi a CIA különböző állapotait és meghatározott esetekben megszakítást kér a processzortól. A megszakítás engedélyezésekor a megszakítási cím az INT DATA regiszterben található.

A regiszter bitjeinek jelentése:

0. bit = 1 – az A időzítő alulcsordulása
1. bit = 1 – a B időzítő alulcsordulása
2. bit = 1 – az óraidő és a riasztóidő azonos
3. bit = 1 – üzemmódtól függően a soros adatregiszter (SDR) vagy megtelt vagy üres
4. bit = 1 – lefutó él a FLAG-en
5. bit – mindig 0
6. bit – mindig 0
7. bit = 1 – az INT MASK és az INT DATA regiszterek legalább egy bitje azonos. Az ICR írásakor az adatok az INT MASK regiszterbe kerülnek. Az INT MASK regiszter nem olvasható.

Az egyes bitek jelentése ugyanaz, mint az INT DATA regiszterben, a 7. bit kivételével. Ha a 7. bit = 1, az 1-es bitek bekapcsolják a megfelelő maszk-bitet. A többi bitet nem befolyásolja.
Ha a 7. bit = 0, az 1-es bitek törlik a megfelelő maszk-bitet. A többi bit nem változik.
Ha az INT DATA regiszter egyik bitje bekapcsolódik és ugyanaz az INT MASK-ban is bekapcsolt bit, az IRQ-érintkezőre egy alacsony szintű jel érkezik.

14. regiszter CRA (az A időzítő vezérlő regisztere)

Az egyes bitek jelentése a következő:

0. bit = 1 – az A időzítő indul
= 0 – az A időzítő leáll
1. bit = 1 – az A időzítő alulcsordulását a PB6 érintkező jelzi
2. bit = 1 – az A időzítő alulcsordulása a PB6 érintkezőt ellenkező polaritásra kapcsolja;
= 0 – az A időzítő alulcsordulása esetén a PB6 érintkezőn egy rendszerütemnyi időtartamú (kb. 1 mikrosecundum) magas jelet állít elő.
3. bit = 1 – az A időzítő a bevitt kezdőértéktől csak egyszer számol vissza 0-ig.
= 0 – az A időzítő a 0 elérésekor automatikusan újra kezdi a visszaszámlálást.

4. bit = 1 – egy új kezdőérték feltétlen betöltése;
Ez a bit lehetővé teszi a számláló betöltését bármely tetszés szerinti időpontban.

5. bit = 1 – Az időzítő felfutó éleket számol a CNT bemeneten, ami lehetővé teszi külső impulzusok számlálását.
= 0 – Az időzítő a rendszeróra impulzusait számolja.

6. bit = 1 – az SP soros bemenetként működik
= 0 – az SP soros kimenetként működik

7. bit = 1 – a valós idejű óra 50 Hz hálózati frekvenciával dolgozik (pl. Európa)
= 0 – a valós idejű óra 60 Hz hálózati frekvenciával dolgozik (pl. USA)

15. regiszter CRB (a B időzítő vezérlőregisztere)

A bitek jelentése a következő:

0...4 bitek: ugyanaz, mint a CRA esetében, de a B időzítőre és a PB7 érintkezőre vonatkoztatva.

5/6 bitek: A B időzítő léptetésének forrását határozzák meg a következők szerint:
00 = rendszerütem számlálás
01 = felfutó (pozitív) CNT-élek számlálása
10 = az A időzítő lefutó éleinek számlálása
11 = az A időzítő lefutó éleinek számlálása, ha a CNT magas szintű.

7. bit = 1 – TOD riasztó regiszter beírása
= 0 – TOD óraidő beírása

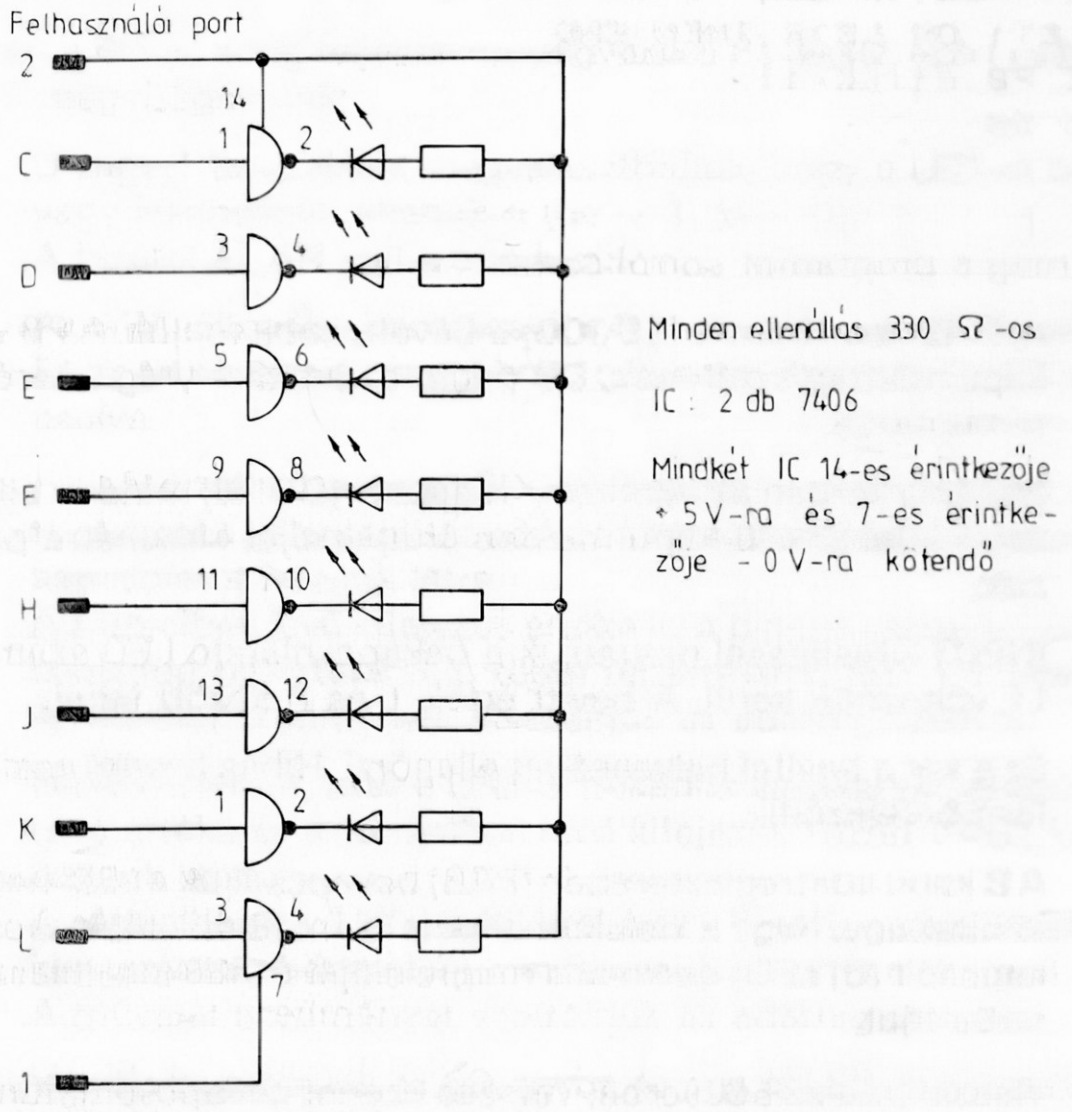
A C 64-esbe összesen 2 db CIA 6526-ost építettek be. Regisztereik pontosan ugyanúgy érhetők el, mint a 64-es egyéb tárolórekeszei. Alapcímei a következők:

CIA 1 = $DC00 (56320); CIA 2 = $DD00 (56576)

A felhasználói porton rendelkezésünkre áll a CIA 2 teljes B portja és a CIA 1 és CIA 2 néhány más kivezetése. Első kísérletünkben közelebbről is megismerkedünk velük.

3.1 LED-ek a felhasználói porton

A 4. ábrán található kapcsolás áramellátását a C 64-esről biztositjuk az 1-es és 2-es csatlakozókon keresztül. A terhelés az összes LED bekapcsolásakor is 100 mA alatt marad.



Az SN7406 típusú IC 6 db open-kollektoros invertert tartalmaz. Emiatt a 8 LED-hez 2 darab IC szükséges. Az adatlap szerint egy kimenet max. 40 mA-es áramot képes kapcsolni. A LED-ek előtétellenállása az áramot 10 mA alatti értékre korlátozza. Egy tetszőleges LED bekapcsolása az alábbi kis programmal valósítható meg:

```
1 REM *** P 1 ***
2 REM
10 CI=56576
20 PB=CI+1
30 DR=CI+3
40 POKE DR,255
50 INPUT"MELYIK LED (1-8)";LE
60 IF LE<1 OR LE>8 THEN END
70 POKE PB,2↑(LE-1)
80 GOTO 50
READY.
```

Vizsgáljuk meg a programot sorról sorra:

10–30: A CIA2 kezdőcímét ($D000) a CI változóba töltjük. A PB változó a B kapu adatregiszterének, DR pedig az adatirányregiszterének címét tartalmazza.

40: Ebben a sorban az adatirány-regiszter (DDRB) minden bitjét bekapcsoljuk, azaz a B kapu minden érintkezőjét kimenetre programozzuk.

50: INPUT-utasítással megadjuk a bekapcsolandó LED számát, ami az LE változóba kerül. A bevitt érték 1 és 8 között lehet.

60: Ez a sor a bevitel helyességét ellenőrzi. Hibás bevitel esetén a program befejeződik.

70: A B kapu adatregiszterében (PRB) bekapcsoljuk a LED-nek megfelelő bitet úgy, hogy a kialakult bináris szám (8 db bekapcsolt (1) és kikapcsolt (0) LED) decimális megfelelőjét POKE-utasítással a regiszterbe írjuk.

80: Visszatérés az 50. sorba. Tetszés szerint bekapcsolhatunk egy másik bitet vagy befejezhetjük a programot.

Ennek a kis programnak van egy szépséghibája. Nevezetesen az, hogy egyszerre, sőt egymás után is, csak egy LED-et tud bekapcsolni. Amint egy újat bekapcsolunk, a régi azonnal kikapcsolódik. A következő programban ezt a hibát már kiküszöbölték.

```
1 REM *** P 2 ***
2 REM
10 CI=56576
20 PB=CI+1
30 DR=CI+3
40 POKE DR,255
50 INPUT "MELYIK LED (1-8)";LE
```

```
60 IF LE < 1 OR LE > 8 THEN END
70 INPUT "BE (1) VAGY KI (0)";AN
80 AW=PEEK (PB)
90 IF AN=0 THEN POKE PB,AW AND (255-(2↑(LE-1)))
100 IF AN=1 THEN POKE PB,AW OR (2↑(LE-1))
110 GOTO 50
READY.
```

Ez a program a 60-as sorig teljesen megegyezik a P1-essel, ezért csak a további sorokat kell megvizsgálnunk:

70: 0 vagy 1 bevitelével megválaszthatjuk, hogy a LED-et bekapcsolni vagy kikapcsolni akarjuk-e (be = 1, ki = 0).
A bevitel az AN változóba kerül.

80: Az AW változóba beolvassuk a B kapu adatregiszterének tartalmát. Ez az érték azt adja meg, hogy pillanatnyilag melyik LED van bekapcsolva.

90: Ha AN értéke 0, azaz a LED-et kikapcsolni akarjuk, akkor az AW változó értéke (jelenlegi állapot) és a zárójelben lévő kifejezés között ÉS kapcsolatot hozunk létre.
A zárójelben lévő kifejezés értéke az a bináris szám, amiben a kikapcsolandó bit értéke 0, a többi bit értéke 1. (ld.: 1.1 fejezet)
A művelet eredményét visszaírjuk az adatregiszterbe.

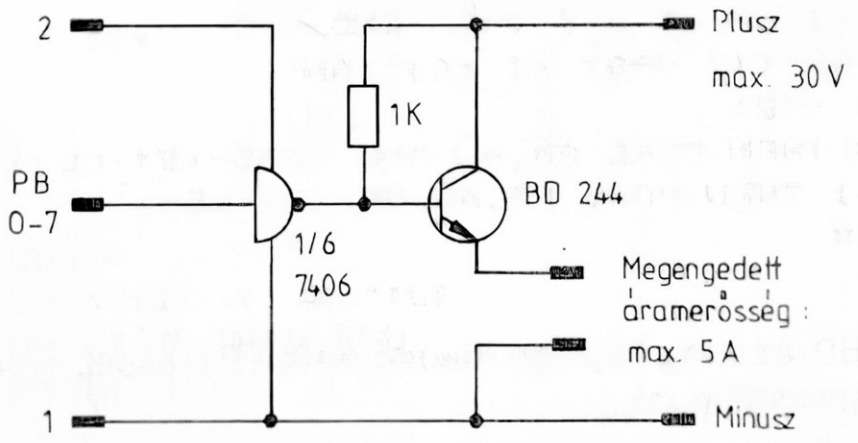
100: Ha AN értéke 1, azaz a LED-et bekapcsolni akarjuk, akkor az AW változó értéke és a zárójelben lévő kifejezés között VAGY kapcsolatot hozunk létre.
A zárójelben lévő kifejezés értéke az a bináris szám, amelyben a bekapcsolandó biten kívül minden más bit értéke 0. (ld: 1.1 fejezet).
A művelet eredményét visszaírjuk az adatregiszterbe.

110: Végül visszatérünk az 50. sorba és egy új LED-et kapcsolhatunk be vagy ki.

3.2 A kapcsolási lehetőségek

3.2.1 Tranzisztoros kapcsolás

A CIA kimenetei csak nagyon csekély áramerősséggel terhelhetők, még egy világító diódát sem bírnak el. Emiatt használtunk a 3.1 alfejezetben ismertetett kapcsolásban is erősítési célra open-kollektoros kapcsolású invertereket. A 40 mA-es terhelhetőség azonban számos más kapcsolásban nem elegendő, ezért bemutatunk egy olyan megoldást, ami maximálisan 5 A terhelhetőséget tesz lehetővé. Ilyen áramerősség mellett a tranzisztoron keletkező hő elvezetésére már célszerű kisebb hűtőbordát alkalmazni.



Természetesen a C 64-es áramellátását nem méretezték ilyen nagy pótlólagos teljesítményre, ezért a szükséges feszültségszintet és áramerősséget külső, hálózati tápegységről kell biztosítani. A tranzisztor kollektorfeszültsége nem haladhatja meg a 30 V-ot.

FONTOS FIGYELMEZTETÉS!

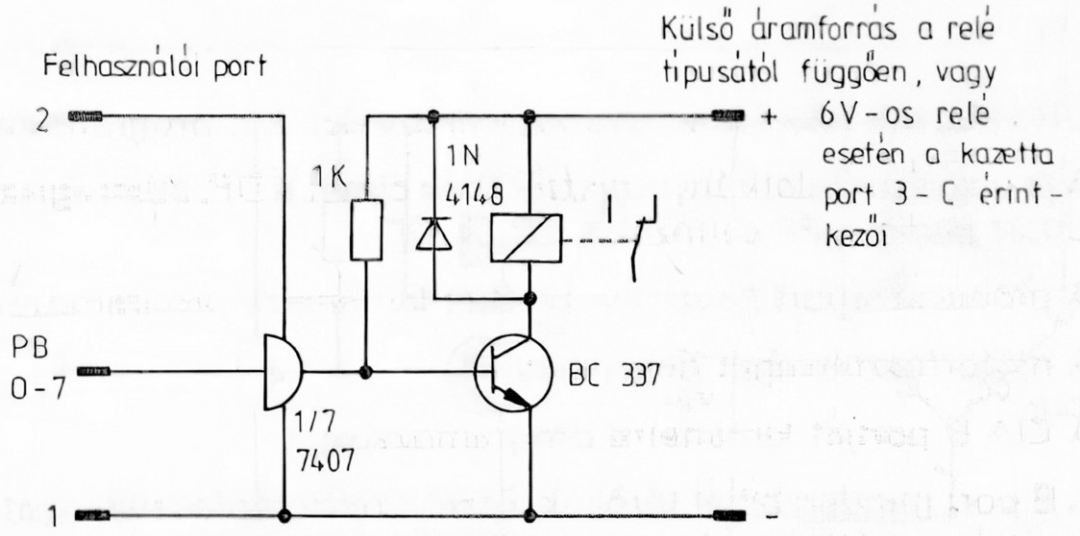
Ha közvetlenül a felhasználói portra tévedésből 5 V-nál nagyobb feszültséget kapcsolunk, a CIA azonnal és végérvényesen tönkremegy! Mivel nehezen beszerezhető és viszonylag drága alkatrészről van szó, mielőtt nagyobb feszültséggel dolgoznánk, különös gondossággal ellenőrizzük kapcsolásunk helyességét. A felesleges kockázatot egy relé vagy optocsatoló beépítésével könnyen elkerülhetjük.
A következő két fejezetben ezekre mutatunk gyakorlati példát.

3.2.2 Relés kapcsolás

Annak ellenére, hogy a számítástechnikában csak ritkán alkalmazzák, a relék még ma is igen hasznos elemei a különböző áramköröknek. Az érintkezők ellenállása kicsi és rajtuk keresztül a váltófeszültség problémamentesen kapcsolható. A modern relék tömege meglepően csekély, sőt időközben olyan változataik is születtek, amelyek megfelelnek egy 14 pólusú IC-nek.
Sajnos 5 V-os relékhez nehéz hozzájutni, a 6 V-os típusokat pedig nem célszerű 5 V-tal üzemeltetni. Szerencsére a C 64-es rendelkezik egy igazán nagy teljesítményű 6 V-os kimenettel, ami ráadásul még stabilizált és kapcsolható is. Ez a kazetta port 3-as és C-jelű érintkezője, ezen, amely alapesetben a kazettamotor vezérlését „közvetíti". Ezt a kimenetet a processzor párhuzamos portjának 5. bitje vezérli. A bit bekapcsolása a motorkimenet kikapcsolását jelenti, és fordítva. Ahhoz tehát, hogy a kapcsolásunk működjön, a processzorport 5. bitjét törölni kell. A processzorport a nullaslap 0 és 1 címeit használja. A CIA a portokhoz hasonlóan adatirányregiszterből és adatregiszterből áll. Az adatirányregiszter címe 0, az adatregiszteré 1. Mivel az 1-es regiszter 0...2 bitjei a tárolófelosztást is vezérlik, nagyon óvatosan kell eljárnunk, amikor e két byte tartalmához nyúlunk. A tá-

rolófelosztás akaratlan megváltoztatásával a gép kezelhetetlenné válhat és csak a ki- és bekapcsolással segíthetünk magunkon.

A kereskedelemben kapható szokványos 6 V-os relék kb. 60-100 mA névleges áramot igényelnek. Mivel a kazettásegység motorkimenete max. 500 mA árammal terhelhető, legfeljebb 8 relét használhatunk külső áramforrás igénybevétele nélkül.



Ha a kapcsolás bemenetét összekötjük a párhuzamos port kimenetével, a relé a portbit bekapcsolásakor behúz. Mindenekelőtt természetesen be kell kapcsolni a kazettásegység motorkimenetét.
A következő program a POB-n keresztül vezérelt relét ki- és bekapcsolja:

```
1 REM **** P 3 ***
2 REM
10 CI=56576
20 PB=CI+1
30 DR=CI+3
40 DP=0
50 PP=1
60 POKE DP,63
70 POKE PP,7
80 POKE DR,255
90 POKE PB,0
100 PRINT "RELE KIKAPCSOLASA"
110 PRINT "PROGRAM VEGE= NYOMJON LE EGY BILLENTYUT!"
120 FOR I=1 TO 1000
130 GET A$:IF A$ <> "" THEN GOTO 220
140 NEXT I
150 POKE PB,1
160 PRINT "RELE BEKAPCSOLASA"
170 PRINT "PROGRAM VEGE= NYOMJON LE EGY BILLENTYUT!"
180 FOR I=1 TO 1000
```

```
190 GET A$:IF A$ <> "" THEN GOTO 220
200 NEXT I
210 GOTO 90
220 POKE DP,47
230 POKE PP,55
240 POKE PB,0
250 PRINT "MINDEN KIKAPCSOLVA"
READY.
```

A CI, PB és DR változók jelentése ugyanaz, mint előző két programunkban.

40–50: A processzor adatirányregiszterének címét a DP, adatregiszterének címét pedig a PP változóba töltjük.

60: A processzorport összes vezetékét kimenetre programozzuk.

70: A motorfeszültséget bekapcsoljuk.

80: A CIA B portját kimenetre programozzuk.

90: A B port minden bitjét töröljük. Ezzel a relék tápfeszültségét bekapcsoltuk, a relék azonban még alapállapotban vannak.

100–110: A képernyőn kijelzésre kerül a relék állapota.

120–140: Ez a három sor egy billentyű lenyomására vár. Ha ez megtörtént, a program elágazik a 220-as sorba.

150: A B port 0. bitjét bekapcsoljuk, a relé zár.

160–200: Ugyanaz, mint a 100–140. sor.

210: A program visszaugrik a 90. sorba. A reléket kikapcsoljuk és a folyamat kezdődik elölről.

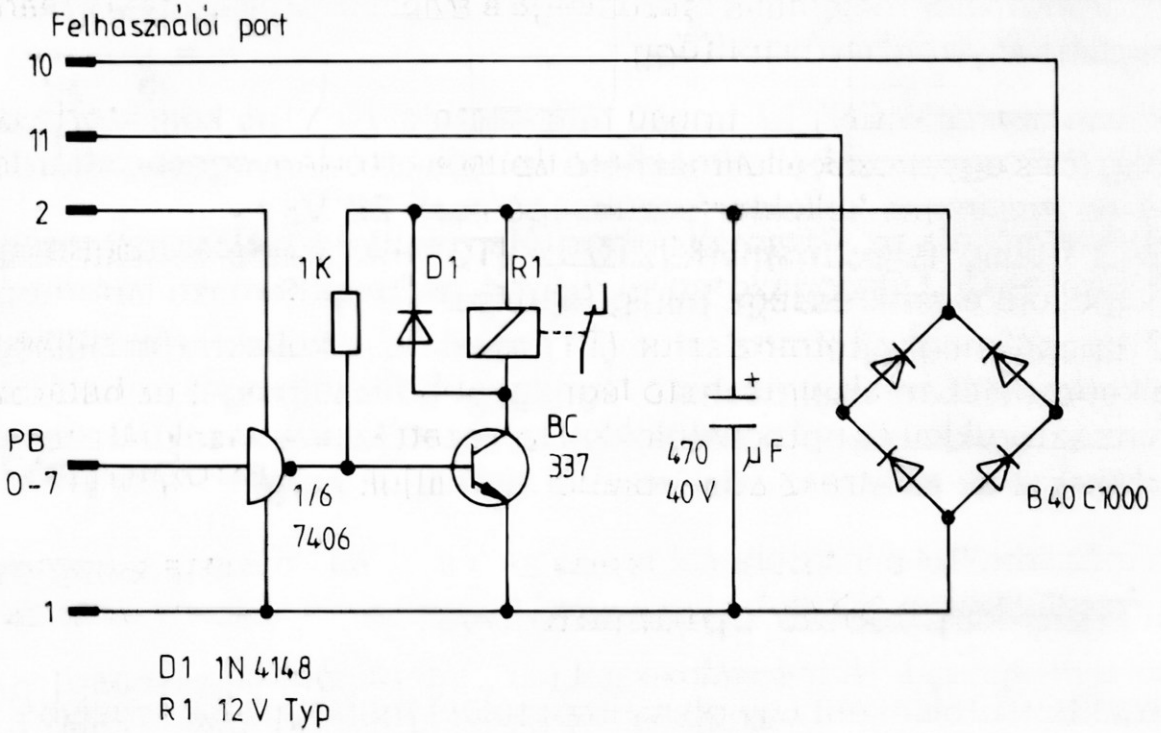
220–250: Ha valamelyik billentyűt lenyomtuk, a program itt folytatódik. Visszaállítjuk az adatirányregiszter és az adatregiszter eredeti értékét, kikapcsoljuk a B portot és a program befejeződik.

Ha csak egy relére van szükségünk, azt természetesen a kazetta porton keresztül is vezérelhetjük. Ebben az esetben nincs szükség a kiegészítő áramkörökre.

Függetlenül a relék vezérlésének módjától, a berajzolt diódát is be kell építenünk. A relé kikapcsolásakor ugyanis magas önindukciós feszültségek keletkeznek (típustól függően több 100 V!), amit a diódával zárunk rövidre.

A C 64-esről 12 V üzemi feszültségű reléket is működtethetünk. A felhasználói port 10-11-es csatlakozóin 9 V-os váltófeszültség áll rendelkezésünkre, amiből egyenirányítás és szűrés után kb. 12 V-os egyenfeszültséget kapunk. Terhelhetősége a C 64-es kézikönyve szerint, 100 mA. Ez azonban csak a kazettásegység egyidejű üzemeltetésekor érvényes, anélkül akár 300–400 mA is megengedett.

Ez max. 8 darab relé egyidejű üzemeltetéséhez elegendő. A 9 V-os váltakozófeszültségű kapcsok túlterhelése esetén (pl. rövidzárlat) kiolvad a C 64-esbe épített biztosító. Ez az olvadóbetét 1,25 A-es. Már csak néhány régebbi gépben található 1 A-es biztosító, amit nyugodtan kicserélhetünk 1,25 A-esre.

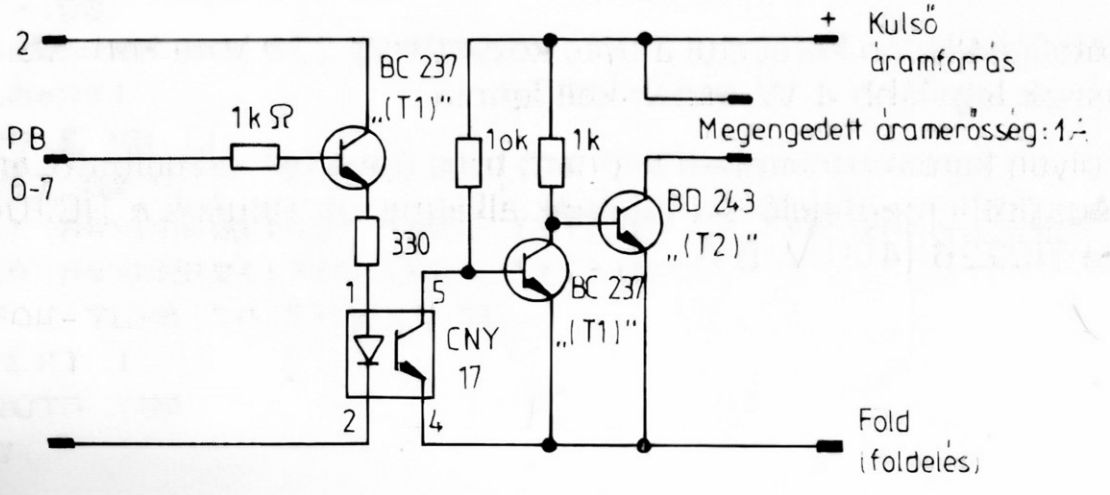


3.2.3 Optocsatolós kapcsolás

Az előző fejezetben leírt relés vezérlés sok esetben nem alkalmazható. Egyrészt a relék működtetéséhez szükséges áram nem áll minden korlátozás nélkül rendelkezésünkre, másrészt az elérhető kapcsolási frekvencia rendkívül alacsony.

Minden olyan esetben, amikor 5 V-nál nagyobb feszültséggel akarunk dolgozni, ajánlatos ún. optocsatolókat beépíteni. Ezek a számítógép és a külső egységek között teljes galvanikus leválasztást jelentenek és a lehető legjobb védelmet nyújtják a számítógépnek.

A következő kapcsolás az előzőekben ismertetett tranzisztoros kapcsolás módosítása.



Az optocsatoló egyik oldala egy világító dióda, a másik egy egyszerű fototranzisztor. Amikor a B port bitjét bekapcsoljuk, akkor az optocsatoló diódáján áram folyik keresztül, ami fénykibocsátást eredményez. A fény hatására a fototranzisztor kinyit és zárja a T1 tranzisztort. Ezzel a T2 tranzisztort átkapcsolja.

Az „erősáramú"-kör maximális feszültsége a tranzisztorok átütési szilárdságától és a teljesítménytranzisztortól függ.

A példában szereplő CNY17 típusú optocsatoló 70 V-os kollektorfeszültséget bír el, míg más ugyancsak alkalmazható típusok ettől lényegesen eltérhetnek (pl. az IL74-es maximális kollektorfeszültsége csak 20 V).
A BD243 típusú teljesítménytranzisztor (T2) maximális kollektorfeszültsége 60 V, kapcsoló áramerőssége pedig legfeljebb 6 A.
BC237 típusú meghajtótranzisztor (T1) maximális kollektorfeszültsége 45 V, ezért a kapcsolásban alkalmazható legnagyobb feszültséget ez határozza meg. Más tranzisztorokkal és optocsatolókkal tervezett kapcsolásoknál ezeket a maximumértékeket az alkatrész adatlapjából tudhatjuk meg.

3.2.4 Triac-kapcsolás optocsatolóval

Fejezetünkben utolsóként egy olyan kapcsolást mutatunk be, amelyben egy optocsatoló és egy triac alkalmazásával 220 V-os fogyasztót vezérelhetünk.

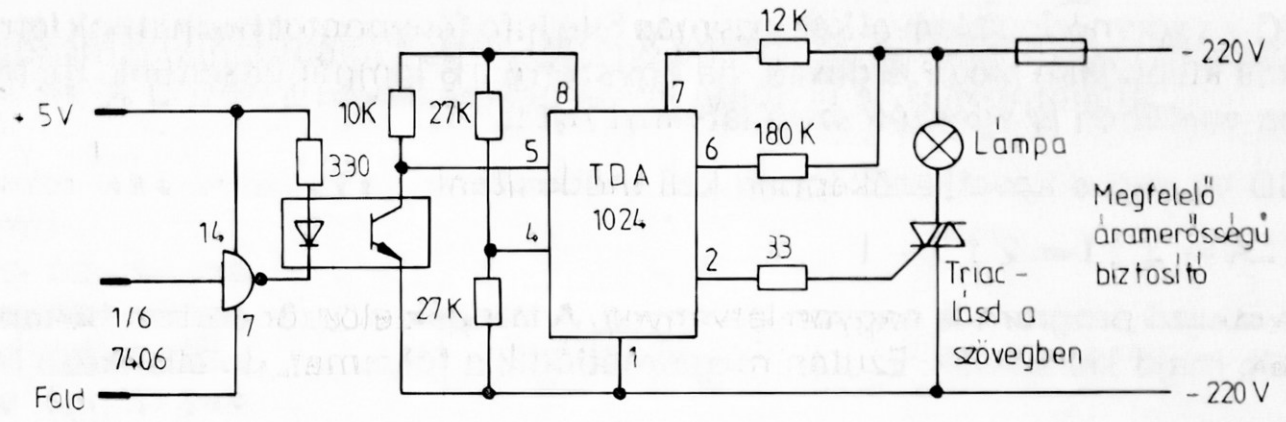
A 220 V-os kapcsolásokat csak azoknak ajánljuk, akik ebben a tekintetben már nem számítanak kezdőnek. Feltétlenül tartsuk be a biztonsági előírásokat! Ebben az esetben a kapcsolásokon csak feszültségmentes állapotban szabad dolgozni, legjobb, ha a hálózati csatlakozót is kihúzzuk. Kizárólag szigetelt vezetékeket használhatunk, a kész kapcsolást pedig zárt műanyag házban kell elhelyezni.

FIGYELEM! 220 V HALÁLOS ÁRAMÜTÉST OKOZHAT!

Az egész kapcsolás egy optocsatolóból, egy triacból, néhány ellenállásból és egy TDA1024-esből áll. Ez az IC minden olyan áramköri elemet tartalmaz, amellyel ún. nullafeszültség-átmenet kapcsolót építhetünk. Ezzel a triacot úgy vezérelhetjük, hogy a kapcsolás a hálózati váltakozó feszültség nullaátmenetének közelében következzék be. Így hatásosan csökkenthetjük a triac üzemeltetésénél egyébként fellépő zavarfeszültségeket.

Egy előtétellenálláson keresztül a triac közvetlenül 220 V-ra köthető. Az előtétellenállásnak legalább 4 W-osnak kell lennie.

Minden olyan kereskedelemben kapható triac típus felhasználható, amelynek a feszültségszintje megfelelő. Az általunk alkalmazott típusok a TIC206 (400 V, 3 A) és a TIC226 (400 V, 8 A).



A megépített kapcsolást a felhasználói porton keresztül, az előzőekben ismertetett programmal üzemeltethetjük. Amikor az optokapcsoló LED-je fényt emittál, a triac bekapcsol.

3.2.5 Fényorgona

Aki fényorgonát szeretne készíteni és szeret kísérletezni a különböző fényhatásokkal, az több triackal fantasztikus lámpavarázslatokat produkálhat.

Mivel egy triac meglehetősen drága, a kapcsolásokat és a programot érdemes először a felhasználói porton LED-ekkel kipróbálni.

Minden programnál 8 lámpából indulunk ki, de kevesebb és – kiegészítő külső IC-k beiktatásával (6522 vagy 8255) – több is elképzelhető. Ilyenkor a programot a megváltozott körülményekhez kell igazítani.

Az első program (P.4) egy futó fényt állít elő, aminek sebessége az F1 és F3 funkcióbillentyűkkel állíthatók be.

```
1 REM *** P 4 ***
2 REM
100 C2=56576
110 PB=C2+1
120 DR=C2+3
130 POKE DR,255
140 LA=0
150 T=100
160 FOR I=0 TO 7
170 LA=2↑I
180 POKE PB,LA
190 GET A$
200 IF A$=CHR$(133) THEN T=T-10:IF T<0 THEN T=0
210 IF A$=CHR$(134) THEN T=T+10
220 FOR TL=0 TO T:NEXT TL
230 NEXT I
240 GOTO 160
READY.
```

A 160-as sor módosításával két, egymás felé futó fénypontot hozhatunk létre. Ez a hatás különösen akkor érdekes, ha egyszerre 16 lámpát vezérlünk, de már 8 lámpa esetében is egészen szép látványt nyújt.

A 160-as sort a következőképpen kell módosítani:

160 LA = 2 ↑ I = 2 ↑ 7 – I

A következő program is nagyon látványos. A lámpák először sorban bekapcsolódnak, majd kialszanak. Ezután megismétlődik a folyamat, de ellenkező irányban.

```
1 REM *** P 5 ***
2 REM
100 C2=56576
110 PB=C2+1
120 DR=C2+3
130 POKE DR,255
140 LA=0
150 T=100
160 FOR I=0 TO 7
170 LA=LA+2↑I
180 POKE PB,LA
190 FOR TL=0 TO T:NEXT TL,I
200 FOR I=0 TO 7
210 LA=LA-2↑I
220 POKE PB,LA
230 FOR TL=0 TO T:NEXT TL,I
240 FOR I=7 TO 0 STEP-1
250 LA=LA+2↑I
260 POKE PB,LA
270 FOR TL=0 TO T:NEXT TL,I
280 FOR I=7 TO 0 STEP-1
290 LA=LA-2↑I
300 POKE PB,LA
310 FOR TL=0 TO T:NEXT TL,I
320 GET A$
330 IF A$=CHR$(133) THEN T=T-10:IF T<0 THEN T=0
340 IF A$=CHR$(134) THEN T=T+10
350 GOTO 160
READY.
```

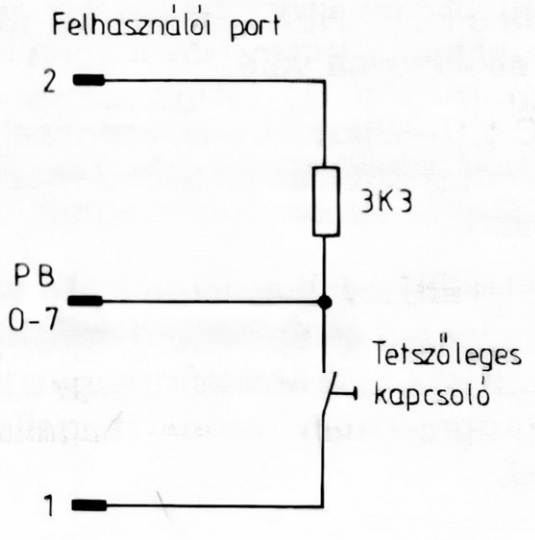
Utolsóként a felhasználói porton történő adatkivitel egy különleges módját mutatjuk be. A C 64-es rendszerprogramját adjuk ki a „fényorgonára".

```
1 REM *** P 6 ***
2 REM
100 C2=56576
110 PB=C2+1
120 DR=C2+3
130 RA=57344
140 RE=65535
150 POKE DR,255
160 T=100
170 FOR I=RA TO RE
180 LA=PEEK(I)
190 POKE PB,LA
200 GET A$
210 IF A$=CHR$(133) THEN T=T-10:IF T<0 THEN T=0
220 IF A$=CHR$(134) THEN T=T+10
230 FOR TL=0 TO T:NEXT TL
240 NEXT I
250 GOTO 170
READY.
```

3.3 Adatbevitel a felhasználói porton keresztül

Eddig csak a felhasználói porton történő adatkivitelről volt szó. Az adatirányregiszter megfelelő programozásával azonban az adatáramlás iránya meg is fordítható.

A legegyszerűbb esetben az adatbevitel egy nyomógomb állapotának lekérdezéséből áll. Ha a gombot nem működtetjük, a felhasználói port bemenete magas szintű, a gomb lenyomásakor pedig alacsony szintre kerül.



A következő programot 8 darab, felhasználói portra csatlakoztatott nyomógomb állapotának kijelzésére használhatjuk.

```
1 REM *** P 7 ***
2 REM
10 CI=56576
20 PB=CI+1
30 DR=CI+3
40 POKE DR,0
50 W=PEEK(PB)
60 FOR I=7 TO 0 STEP-1
70 PRINT CHR$(49+(X=W AND 2↑I));
80 NEXT I
90 PRINT
100 GOTO 50
READY.
```

Nézzük röviden a programsorok jelentését:

10–30: A CI, PB és DR változókhoz hozzárendeljük a CIA2 báziscímét és a B port adat- és adatirányregiszterének címét.

40: Az adatirányregiszter minden bitjét töröljük, ezáltal a B port minden érintkezőjét bemenetre programozzuk.

50: A B port állapotát PEEK-kel leolvassuk és hozzárendeljük a W változóhoz.

60–80: A W változó értékét bináris számmá alakítjuk és kivisszük a képernyőre.

90–100: Soremeléssel előkészítjük egy új érték megjelenítését és a program visszatér az 50. sorba.

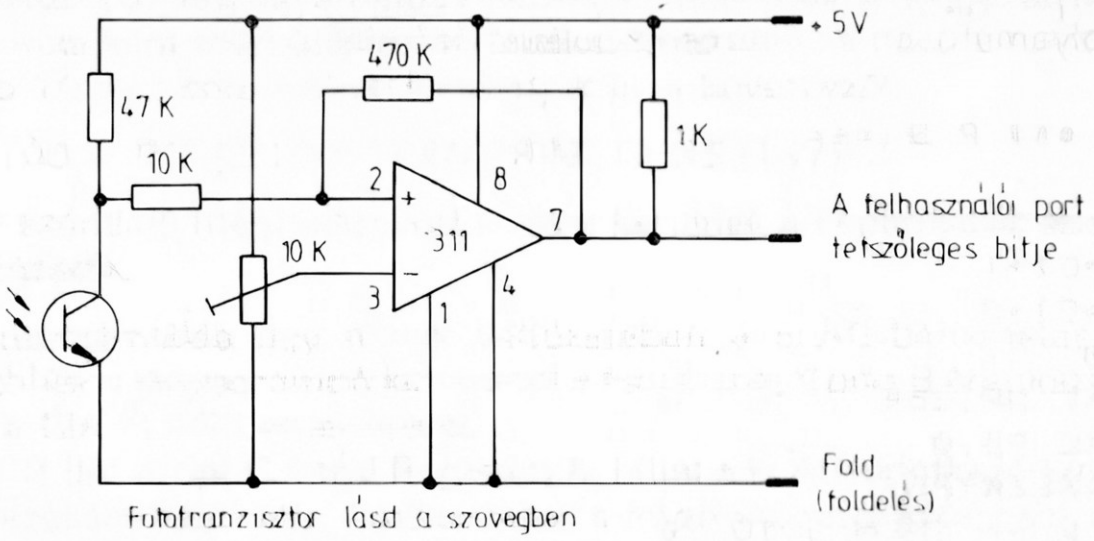
A 60–80-as sorokat más programokban is felhasználhatjuk egy szám bitmintájának megjelenítésére. Ha a FOR...NEXT ciklust 15-től 0-ig kibővítjük, 16-bites számok átalakítására is alkalmassá válik.

- A kapcsoló helyére egy IC-t, tranzisztort vagy optocsatolót is beépíthetünk. Ezzel a CIA-ról közvetlenül lekérdezhetjük az alacsony-magas szinteket. Ezekről a következő fejezetben lesz szó.

A kapcsolók állapotának lekérdezése és képernyőn való megjelenítése nem túl izgalmas feladat. Ennek ellenére elképzelhető néhány alkalmazási lehetősége. A nyomógombokat például lakásunk ablakaira vagy a bejárati ajtóra megfelelően felszerelve (egy bővebb programmal) azonnal megállapíthatnánk, ha valamelyik nincs rendesen becsukva.

3.4 A fénysorompók

A következő ábrán egy egyszerű fényérzékeny kapcsolást mutatunk be. Ez lényegében egy fototranzisztorból és egy 311-es komparátorból áll. Ha a fototranzisztort megvilágítjuk, akkor vezetni kezd, pontosan úgy, mintha a bázison keresztül vezérelnénk. Ezáltal a komparátor nem invertáló bemenetén csökken a feszültség. A referenciafeszültséget az invertáló bemeneten állítjuk be.



Amennyiben a fototranzisztor miatt a nem invertáló bemenet feszültsége a referenciaszint alá esik, a kimeneten alacsony szint mérhető. A fototranzisztor megfelelő megvilágítása esetén a kimenet alacsony szintű, gyenge megvilágításkor magas szintű.

Ha a „+"-bemenet és a kimenet közé egy ellenállást iktatunk, a kapcsolásban ún. hiszterézis jön létre. Erre akkor van szükség, ha a kapcsolást pl. fénycsővilágítású helyiségben üzemeltetjük. Köztudott, hogy a fénycső fénye „vibrál", azaz a világosság a hálózati frekvencia ütemében nő ill. csökken. A fototranzisztor mindezt érzékeli, ami kis hiszterézisű áramkörben a kimenet állandó ki- és bekapcsolását okozhatja. Ha azonban a leírt jelenség (hiszterézis) miatt a kapcsolás nem működik megfelelően, csökkenthetjük az ellenállás értékét. Ekkor viszont a tranzisztor átkapcsolásához nagyobb fényingadozásra van szükség. Ha ezt nem tudjuk megvalósítani, mert pl. túl nagy a környezeti világítás, akkor rendszerint segít a fototranzisztorral párhuzamosan kötött, kb. 470 nF-os kondenzátor.

A kapcsolás kimenetét a B port tetszőleges bitjére köthetjük, így megfelelő szereléssel megoldhatjuk a különböző helyiségek ellenőrzését.

A másik fontos lehetőség bizonyos dolgok számlálása. Az ehhez szükséges szoftvert a következő fejezetben ismertetjük.

3.4.1. Impulzusszámlálás fénysorompóval

Fejezetünk témája, az impulzusszámlálás, természetesen nemcsak fénysorompóval valósítható meg. Erre egy egyszerű kapcsoló vagy más áramkör is megfelel, csupán arra kell ügyelnünk, hogy a felhasználói portra 5 V-nál magasabb feszültséget nem szabad kapcsolni.

Most azonban foglalkozzunk a szoftverrel. Tételezzük fel, hogy az előző fejezetben bemutatott fénysorompót a felhasználói port 0. bitjére csatlakoztattuk.

Először folyamatosan az összes impulzust számláljuk:

```
1 REM *** P 8 ***
2 REM
10 CI=56576
20 PB=CI+1
30 DR=CI+3
40 Z=0
50 POKE DR,254
60 POKE PB,0
70 WE=PEEK(PB)
80 IF WE=0 THEN GOTO 70
90 Z=Z+1
100 PRINT CHR$(147);Z
110 WE=PEEK(PB)
120 IF WE=1 THEN GOTO 110
130 GOTO 60
READY.
```

Mint mindig, most is a szükséges változók meghatározásával kezdjük:

10–40: CI, PB, DR értékei ugyanazok, mint eddigi programjainkban. Z-t az impulzusok számlálójaként használjuk, így kezdőértéke természetesen 0.

50: A 0. bit kivételével a B port minden bitjét kimenetre programozzuk.

60: A B port minden bitjét nullázzuk. Ennek az az előnye, hogy mivel impulzus úgyis csak a 0. biten érkezhet, az 1–7 biteket nem kell figyelembe vennünk, mert értékük mindig 0. Így a fénysorompó állapotától függően mindig csak 0 vagy 1 értéket kapunk.

70–80: Amíg a 0. bit alacsony szintű (impulzus nem érkezik), a program felváltva ezt a két sort hajtja végre. A 70. sorban leolvassa a 0. bit állapotát, a 80-asban pedig megvizsgálja, hogy értéke 0 vagy nem. Ha 0, új lekérdezés következik, ha nem, rátér a következő sorra.

90: A számláló (Z) értéke 1-gyel nő.

100: A számláló pillanatnyi értékének kijelzése.
A CHR$ (147) – utasítás előzőleg törli a képernyőt, így az érték mindig a bal felső sarokban jelenik meg.

110–130: Az első két sorban a program egy alacsony szintű jelre vár. Amikor ez bekövetkezik, a program a harmadik sorra tér, ahonnan visszaugrik a 70. sorba.

Ez a program csak akkor alkalmazható, ha az impulzusok száma másodpercenként legfeljebb 10. Ha a folyamatos kijelzéstől eltekintünk, lényegesen növelhető a másodpercenként mérhető impulzusok száma.
A program 100-as sora helyett beírhatjuk pl. a következőt:

100 IF Z/100 = INT (Z/100) THEN PRINT CHR$ (147); Z

Így csak a számláló meghatározott értékei kerülnek a képernyőre, esetünkben minden századik.

Az impulzusszámlálás egy másik változatában a $\overline{\text{FLAG}}$-bemenetet vesszük igénybe. Ehhez a fénysorompó kimenetét a felhasználói port B érintkezőjén összekötjük a CIA $\overline{\text{FLAG}}$-bemenetével.
A CIA az ICR (Interrupt Control Register) 5. bitjét a $\overline{\text{FLAG}}$-érintkezőn megjelenő lefutó él jelzésére használja. Amikor tehát a fototranzisztor kikapcsol, az ICR 5. bitje bekapcsolódik. Mint már említettük, ez a regiszter olvasáskor automatikusan törlődik.

```
1 REM *** P 9 ***
2 REM
10 CI=56576
20 IC=CI+13
30 WE=PEEK(IC)
40 IF WE<>16 THEN GOTO 30
50 Z=Z+1
60 PRINT CHR$(147);Z
70 GOTO 30
READY.
```

Ez egy rövidebb program, mint az előző, mégsem gyorsabb annál. A képernyő törlése és a kijelzés sok időt vesz igénybe és ezalatt a számlálás szünetel.

3.4.2. Időmérés fénysorompóval

Sportrendezvényeken és még sok más esetben, pl. sebességméréshez, pontos stopperórára van szükség. Egy ilyen óra megvalósítására több lehetőségünk is van. Ezek közül ebben a fejezetben kettővel foglalkozunk. Mindkét esetben a 3.4 fejezetben bemutatott fényérzékeny kapcsolásból két-két darabra van szükségünk. Kimenetüket közösen a FLAG-bemenetre kapcsoljuk.

Az egyik fénysorompót a rajtnál, a másikat a célnál szereljük fel. Körpályánál, ahol a rajt és a cél ugyanabban a pontban van, természetesen egy is elegendő. Amikor a FLAG-bemeneten egy lefutó él jelenik meg, a mérés elindul és a következő lefutó él megjelenéséig tart. Az eredmény a képernyőre kerül.

Első példánkban a számítógép belső óráját használjuk. Az idő a TI$ rendszerváltozóban áll rendelkezésünkre.

```
1 REM *** P 10 ***
2 REM
10 CI=56576
20 IC=CI+13
30 WE=PEEK(IC)
40 PRINT CHR$(147):PRINT"                    *** STOPPER ***"
50 PRINT:PRINT"          A RENDSZER MERESRE KESZ"
60 PRINT:PRINT"            A VERSENY INDITHATO"
70 WE=PEEK(IC)
80 IF WE<>16 THEN GOTO 70
90 TI$="000000"
100 PRINT CHR$(147)
110 PRINT"                    MERES"
120 PRINT:PRINT:PRINT
130 WE=PEEK(IC)
140 IF WE<>16 THEN GOTO 130
150 A$=TI$
160 PRINT"                 A MERT IDO:"
170 PRINT:PRINT
180 HR=VAL(LEFT$(A$,2))
190 MI=VAL(MID$(A$,3,2))
200 SE=VAL(RIGHT$(A$,2))
210 PRINT"                 ";HR;"ORA"
220 PRINT"                 ";MI;"PERC"
230 PRINT"                 ";SE;"MASODPERC"
240 PRINT"HA UJRA KIVAN MERNI, NYOMJA MEG AZ F1"
250 PRINT:PRINT"                    BILLENTYUT"
260 GET A$
270 IF A$=CHR$(133) THEN RUN
280 GOTO 260
READY.
```

Ez az óra sajnos nem túl pontos. Az eltérés elérheti a napi fél órát is.
A C 64-es azonban rendelkezik két ún. valós idejű órával is, amit a hálózati frekvencia vezérel. Ezek nagyon jól megfelelnek a céljainknak. Hosszú idejű pontosságuk viszonylag jó és a tizedmásodperceket is mérik.
Az órákat a CIA 8, 9, 10 és 11-es regiszterein keresztül érhetjük el, figyelembe véve a CRA és CRB regiszterek legmagasabb helyiértékű bitjeit is.

A CRA 7. bitje azt határozza meg, hogy az órát 50 Hz-cel, vagy az USA-ban használatos 60 Hz-cel vezéreljük-e. Ha a bitet bekapcsoljuk, az 50 Hz van érvényben. Ezenkívül beprogramozhatjuk arra is, hogy előre meghatározott időben riasztójelet adjon. Ahhoz, hogy az óraidőhöz és a riasztóidőhöz ne kelljen külön-külön 4–4 regisztert igénybe venni, a CRB regiszter 7. bitjével választhatunk a két lehetőség között (bit = 1 riasztóidő, bit = 0 óraidő).

A regiszter BCD alakban tárolja az óraidőt, ezért a kijelzéséhez elegendő egy viszonylag egyszerű számítás.

Különleges jelentőségük van az óra (TODH) és a tizedmásodperc (TODT) regisztereknek. Az óraregiszter olvasásakor az aktuális óraidő egy közbenső tárolóba kerül, ahol bármikor rendelkezésünkre áll. Az óra ugyan tovább fut, de a változások nem zavarják a közbenső tárolót, mivel az nem tartalmazza a tizedmásodperceket. Így az óraregiszter leolvasása után tetszőleges idő telhet el a következő leolvasásig. Ezalatt az előző kijelzés marad a képernyőn.

Ha az óraidőt akarjuk beállítani, először az óraregisztert kell feltölteni. Ezalatt az óra áll. Ezután a perc (TODM) és másodperc (TODS) regiszterek következnek. Utoljára marad a TODT regiszter, aminek betöltésével az óra a beállított időtől elindul.

Ahhoz, hogy az órák pontosságát kihasználhassuk, gépi nyelvű programra van szükségünk. A következőkben ismertetésre kerülő P 11-es assemblerprogram a két CIA kényelmes kezelését biztosítja. A CIA 1-et az aktuális óraidő kijelzésére, a CIA 2-t pedig stopperként használjuk. Ezt a stoppert is a FLAG-bemeneten keresztül vezéreljük. Ebben az esetben azonban az impulzus egy nem maszkolható megszakítást (NMI) vált ki, ami rögtön megállítja a stoppert. Az óraidő és a stopper idejének kijelzése is a megszakítás ideje alatt történik. Ezt a megszakítást azonban nem a mi programunk hozza létre. Mi a szabályos rendszermegszakítást használjuk, amit megszakításidőzítésnek is neveznek, és egy maszkolható megszakítás (IRQ). Ez kb. 16 milliszekundumonként automatikusan lezajlik. Ez hajtja végre a billentyűzet lekérdezést, a rendszeróra (TI$) léptetését, a kurzor villogtatását és még néhány egyéb feladatot.
Mivel az IRQ- és NMI-rutinok mutatói a RAM-ban találhatók, könnyen irányíthatók saját rutinjainkra, amelyek végrehajtása után a program visszaküldhető a rendszerrutinra.

Nézzük magát a programot:

```
23 ;*** P 11 ***
24 ;
25 .OPT P8,O1
30 *=$CEB5
35 SZIN =$0286; PILLANATNYI KURZORSZIN
40 BUSY1 =$FB;FLAG AZ ORAIDO KIJELZESHEZ
45 BUSY2 = $FC; FLAG A STOPPER KIJELZESHEZ
```

```
 50 COLRAM =$F9
 55 VIDEO =$F7
 60 CIA1 =$DC00
 65 CIA2 =$DD00
 70 NMIREGI =$FE47
 75 NMIVEC =$0318
 80 IRQREGI =$EA31
 85 IRQVEC =$0314
 90 TODT1 =CIA1+8; TIZEDMASODPERC
 95 TODS1 =CIA1+9; MASODPERC
100 TODM1 =CIA1+10; PERC
105 TODH1 =CIA1+11; ORA
110 TODT2 =CIA2+8
115 TODS2 =CIA2+9
120 TODM2 =CIA2+10
125 TODH2 =CIA2+11
130 ICR1 =CIA1+13; MEGSZAKITASREGISZTER
135 ICR2 =CIA2+13
140 CRA1 =CIA1+14
145 CRB1 =CIA1+15; VEZERLOREGISZTER
150 CRA2 =CIA2+14; CIA 1 ES 2
155 CRB2 =CIA2+15
160 DISP =$04DA; VIDEORAM
165 COL =$D8DA; SZINRAM
166 ;
167 ;==== MEGSZAKITASBEALLITAS ====
168 ;
170 SEI
175 LDA #0; STOPPIDO KIJELZES
180 STA BUSY2; KIKAPCSOLASA
185 LDA #<IRQUJ; A PROGRAMUNK
190 STA IRQVEC; INTERRUPT VEKTORANAK
195 LDA #>IRQUJ; BEALLITASA
200 STA IRQVEC+1
205 LDA #<COL
210 STA COLRAM; SZINRAM ES
215 LDA #>COL; VIDEORAM CIMEK
220 STA COLRAM+1; BEALLITASA
225 LDA #<DISP
230 STA VIDEO
235 LDA #>DISP
240 STA VIDEO+1
245 LDA ICR2
250 LDA #<NMIUJ
255 STA NMIVEC; NMI VEKTOR
260 LDA #>NMIUJ; BEALLITASA
```

```
265 STA NMIVEC+1
270 LDA CRA1; 50 HZ TRIGGER
275 ORA #$80; BEKAPCSOLASA
280 STA CRA1
285 LDA CRB1; ORAIDO  VALASZTAS
290 AND #$7F; RIADOIDO KIKAPCSOLAS
295 STA CRB1
300 LDA CRA2; UGYANEZ
305 ORA #$80; CIA2-NEL
310 STA CRA2
315 LDA CRB2
320 AND #$7F
325 STA CRB2
330 LDA #0
335 STA TODT2; AZ ORA REGISZTEREINEK
340 STA TODH2; 0-RA ALLITASA
345 STA TODM2; ES ORA INDITAS
350 STA TODS2;
355 STA TODT1; 1-ES ORAT INDITANI
360 CLI
365 RTS
366 ;
367 ;==== UJ IRQ ====
368 ;
370 IRQUJ SEI
375 LDY #9
380 LDA SZIN; SZINRAM TOLTESE
385 LOOP1 STA (COLRAM),Y
390 DEY
395 BPL LOOP1
400 LDY #90
405 LOOP2 STA (COLRAM),Y
410 DEY
415 CPY #79
420 BNE LOOP2
425 LDA #$7F; NMI-T A FLAG-BEMENETROL
430 STA ICR2; INDITANI
435 LDA #%10010000
440 STA ICR2
442 ;
443 ;==== CIA 1 ORAIDO ====
444 ;
445 LDA BUSY1; ORAIDO KIIRATAS VIZSGALAT
450 BEQ VEGE1
455 LDY #0; POZICIO SZAMLALO
```

```
460 LDA TODH1; ORAREGISZTER
465 BPL WEI; AM VAGY PM VIZSGALAT
470 AND #$7F
475 CLC; PM-NEL
480 SED; 12-T HOZZAADNI
485 ADC #$12; 0-23-IG
490 CLD
495 WEI JSR CRUNCH; EREDMENYT KIADNI
500 LDA #":"
505 STA (VIDEO),Y; ":" KIJELEZNI
510 INY; A POZICIOT NOVELNI
515 LDA TODM1; PERCREGISZTERT
520 JSR CRUNCH; KIADNI
525 LDA #":"; ES TOVABB
530 STA (VIDEO),Y; ":"
535 INY
540 LDA TODS1; MASODPERCET KIJELEZNI
545 JSR CRUNCH
550 LDA TODT1; REGISZTERT ISMET SZABADDA TENNI
552 ;
553 ; ==== CIA 2 STOPIDO ====
554 ;
555 VEGE1 LDA BUSY2; STOPPERIDO KIJELZES ELLENORZESE
560 BEQ VEGE2
565 LDY #80; POZICIO STOPPERIDOHOZ
570 LDA TODH2
575 BPL WEI1; MINT CIA1-NEL
580 AND #$7F
585 CLC
590 SED
595 ADC #$12
600 CLD
605 WEI1 JSR CRUNCH
610 LDA #":"
615 STA (VIDEO),Y
620 INY
625 LDA TODM2
630 JSR CRUNCH
635 LDA #":"
640 STA (VIDEO),Y
645 INY
650 LDA TODS2
655 JSR CRUNCH
660 LDA #":"
665 STA (VIDEO),Y
670 INY
```

```
675 LDA TODT2; TIZEDMASODPERCET OLVASNI
680 JSR H1; ES KIJELEZNI
685 VEGE2 JMP IRQREGI; A REGI IRQ RUTINHOZ
687 ;
688 ;==== KIERTEKELES ====
689 ;
690 CRUNCH PHA; MEG SZUKSEGES
695 LSR
700 LSR; 10-ES HELYIERTEKET AZ ALSO
705 LSR; BYTE-FELBE ATVINNI
710 LSR
715 ORA #"0"; SZAMOT KEPEZNI
720 STA (VIDEO),Y; ES KIJELEZNI
725 PLA; REGI ERTEK VISSZA
730 H1 INY; POZICIOT NOVELNI
735 AND #$0F; 10-ES HELYIERTEKET NULLAZNI
740 ORA #"0";SZAMOT KEPEZNI
745 STA (VIDEO),Y; KIJELEZNI
750 INY; POZICIOT NOVELNI ES
755 RTS; VISSZA
757 ;
758 ;==== UJ NMI ====
759 ;
760 NMIUJ PHA
765 TXA; OSSZES REGISZTERT MENTENI
770 PHA
775 TYA
780 PHA
785 LDA BUSY2; STOPPER K1, ELLENORZES
790 BNE CEL; HA IGEN, AKKOR A CELHOZ
795 START LDA #$0; STOPPER 0-NAL
800 STA TODT2; INDITANI
805 LDY #0; TOVABBI NMI-KET
810 PRELL LDA ICR2; MEGAKADALYOZNI ES
815 INY; ESETLEGES PRELLT
820 BNE PRELL; KIKUSZOBOLNI
825 LDA #$1; KIJELZEST
830 STA BUSY2; BEKAPCSOLNI
835 BNE RET
840 CEL LDA TODH2; ORAT LEALLITANI
845 LDY #0; TOVABBI NMI-KET
850 PREL1 LDA ICR2; MEGAKADALYOZNI ES
855 INY; ESETLEGES PRELLT
860 BNE PREL1; MEGSZUNTETNI
865 LDA #0
870 STA BUSY2
```

```
875 STA TODT2; ORAT ALAPALLAPOTBA
880 STA TODH2; HOZZUK
885 STA TODM2
890 STA TODS2
895 RET PLA
900 TAY; REGISZTEREKET VISSZATULTJUK
905 PLA
910 TAX
915 PLA
920 RTI
922 ;
923 ;==== VEGE ====
924 ;
925 SEI;MEGSZAKITASJEL BEAALITAS
930 LDA #<IRQREGI
935 STA IRQVEC; NMI-T ES IRQ-T
940 LDA #>IRQREGI; EREDETI
945 STA IRQVEC+1; ERTEKERE VISSZA-
950 LDA #<NMIREGI; IRJUK
955 STA NMIVEC
960 LDA #>NMIREGI
965 STA NMIVEC+1
970 CLI
975 RTS;VISSZATERES

READY.
```

Aki járatos a gépi kódú programozásban, annak ez a program semmiféle nehézséget nem okoz. A „csak BASIC-ben" programozóknak kihívást jelenthet, hogy megismerkedjenek és megbarátkozzanak e rendkívül hatékony és gyors módszerrel.

A programot nem fogjuk részleteiben tárgyalni, csak néhány lényeges pontjára és a használatára térünk ki.
A program pontossága kb. 100 milliszekundum. Ez a hiba a CIA felépítéséből adódik. A hálózati frekvencia ugyanis aszinkron fut a számítógép ütemfrekvenciájával, ezért az első tizedmásodperc hosszúságát nem tudjuk pontosan meghatározni. Ez az időtartam általában rövidebb, mint 1/10 másodperc.

A következő lista a program, BASIC betöltője, amit a „csak BASIC-ben" programozók számára közlünk.

```
 1 REM *** P 12 ***
 2 REM
 100 FOR I= 52917 TO 53247
 110 READ X:POKE I,X:S=S+X:NEXT
 120 DATA 120,169,  0,133,252,169, 20,141, 20,  3,169,207
```

```
130 DATA 141, 21,  3,169,218,133,249,169,216,133,250,169
140 DATA 218,133,247,169,  4,133,248,173, 13,221,169,172
150 DATA 141, 24,  3,169,207,141, 25,  3,173, 14,220,  9
160 DATA 128,141, 14,220,173, 15,220, 41,127,141, 15,220
170 DATA 173, 14,221,  9,128,141, 14,221,173, 15,221, 41
180 DATA 127,141, 15,221,169,  0,141,  8,221,141, 11,221
190 DATA 141, 10,221,141,  9,221,141,  8,220, 88, 96,120
200 DATA 160,  9,173,134,  2,145,249,136, 16,251,160, 90
210 DATA 145,249,136,192, 79,208,249,169,127,141, 13,221
220 DATA 169,144,141, 13,221,165,251,240, 42,160,  0,173
230 DATA  11,220, 16,  7, 41,127, 24,248,105, 18,216, 32
240 DATA 153,207,169, 58,145,247,200,173, 10,220, 32,153
250 DATA 207,169, 58,145,247,200,173,  9,220, 32,153,207
260 DATA 173,  8,220,165,252,240, 50,160, 80,173, 11,221
270 DATA  16,  7, 41,127, 24,248,105, 18,216, 32,153,207
280 DATA 169, 58,145,247,200,173, 10,221, 32,153,207,169
290 DATA  58,145,247,200,173,  9,221, 32,153,207,169, 58
300 DATA 145,247,200,173,  8,221, 32,163,207, 76, 49,234
310 DATA  72, 74, 74, 74, 74,  9, 48,145,247,104,200, 41
320 DATA  15,  9, 48,145,247,200, 96, 72,138, 72,152, 72
330 DATA 165,252,208, 19,169,  0,141,  8,221,160,  0,173
340 DATA  13,221,200,208,250,169,  1,133,252,208, 27,173
350 DATA  11,221,160,  0,173, 13,221,200,208,250,169,  0
360 DATA 133,252,141,  8,221,141, 11,221,141, 10,221,141
370 DATA   9,221,104,168,104,170,104, 64,120,169, 49,141
380 DATA  20,  3,169,234,141, 21,  3,169, 71,141, 24,  3
390 DATA 169,254,141, 25,  3, 88, 96
400 IF S <>42208 THEN PRINT"ADATHIBA!!":END
410 PRINT"OK"
READY.
```

A betöltőprogramot SYS 52917 utasítással indíthatjuk. Mivel teljesen megszakításvezérelt, bármilyen BASIC-programból indítható, de a SYS-utasítás közvetlen üzemmódban is akalmazható.

Különleges szerepe van a nulláslap 251-es és 252-es tárolórekeszeinek. Az előző az óraidő, az utóbbi a stopperidő kijelzését vezérli. A SYS-utasítás bevitele után mindkét idő rögtön kijelzésre kerül. Kikapcsolásához 0-t, visszakapcsolásához 1-et kell a rekeszekbe írni.
A stopper indítása és leállítása, mint már említettük, egy NMI-vel történik. Mivel nem kell a megszakítás okát lekérdezni, a stopper a RESTORE-billentyűvel is ki- és bekapcsolható. Amikor azonban ezzel kísérletezünk, a RUN/STOP-RESTORE billentyűkombinációval nem léphetünk ki a programból. A programot csak a SYS 53225 utasítással hatástalaníthatjuk. Ezzel visszatérünk a régi megszakító rutinhoz és a RUN/STOP-RESTORE billentyűk is visszakapják eredeti működésüket.

A következő BASIC nyelvű program feltételezi, hogy a gépi kódú rutin már a tárolóban van. Ehhez használhatjuk a BASIC betöltőprogramot.
A program bekéri a kiinduló időt és különböző lehetőségeket kínál.
Ne tekintsük véglegesnek ezt a programot. Csak néhány lehetőséget akartunk bemutatni és kedvet csinálni a további kísérletezéshez.

```
1 REM *** P 13 ***
2 REM
10 C=56328
20 SYS 52917
30 SYS 53225
40 PRINT CHR$(147)
50 PRINT"KEREM AZ INDULOIDOT"
60 PRINT
70 PRINT"   HHMMSS FORMABAN"
80 PRINT
90 PRINT"              ";: INPUT A$
100 IF LEN(A$) <> 6 THEN 50
110 H=VAL(LEFT$(A$,2))
120 M=VAL(MID$(A$,3,2))
130 S=VAL(RIGHT$(A$,2))
140 IF H>23 THEN 50
150 IF H>11 THEN H=H+68
160 POKE C+3,16*INT(H/10)+H-INT(H/10)*10
170 IF M>59 THEN 50
180 POKE C+2,16*INT(M/10)+M-INT(M/10)*10
190 IF S>59 THEN 50
200 POKE C+1,16*INT(S/10)+S-INT(S/10)*10
210 POKE C,0
220 PRINT CHR$(147)
230 SYS 52917:POKE 252,1:POKE 251,1
240 PRINT"       STOPPER"
250 PRINT:PRINT:PRINT"       ORAIDO"
260 PRINT:PRINT"     STOP IDO"
270 FOR I=1 TO 5
280 PRINT CHR$(17)
290 NEXT
300 PRINT"     F1=STOPPER NULLAZAS"
310 PRINT
320 PRINT"     F3=KOZBENSO IDO KIJELZES"
330 PRINT
340 PRINT"     F7=PROGRAM VEGE"
350 GET AN$:IF AN$="" THEN 350
360 IF AN$=CHR$(133) THEN 400
370 IF AN$=CHR$(134) THEN 420
380 IF AN$=CHR$(136) THEN 440
```

```
390 GOTO 350
400 POKE 252,1:FOR I=1 TO 100:NEXT:POKE 252,0
410 GOTO 350
420 POKE 252,0:FOR I=1 TO 2000:NEXT:POKE 252,1
430 GOTO 350
440 PRINTCHR$(19):FOR I=1 TO 22:PRINTCHR$(17);:NEXT
450 PRINT"          VALOBAN VEGE";
460 INPUT EN$
470 IF EN$="I" THEN 510
480 PRINTCHR$(19):FOR I=1 TO 22:PRINTCHR$(17);:NEXT
490 PRINT"                                          "
500 GOTO 350
510 SYS 53225
520 PRINT CHR$(147)
530 PRINT:PRINT:PRINT
540 PRINT"             STOPPER VEGE"
READY.
```

A kiindulási idő bevitele után a képernyőn megjelenik az óraidő és a stopper ideje. A stopper állása: 00:00:00:0.
Egy NMI-vel, tehát a FLAG-bemenetre vitt jellel vagy a RESTORE -billentyű lenyomásával a stopper elindítható. Ha most lenyomjuk az F3-as billentyűt, akkor kb. 2 másodpercig láthatjuk a stopper közbenső állását. Ezután a mérés tovább fut.
Egy második NMI hatására a stopper teljesen leáll és a mért idő a képernyőre kerül.
Az F1-es billentyűvel a stoppert nullázzuk és ezzel egy újabb mérést készítünk elő. Ha ezt elmulasztjuk, a stopper a következő mérést ettől függetlenül is 0-nál kezdi.

3.5. Az időzítő alkalmazása

Minden CIA-ban két 16-bites időzítő található. Ezek többféle üzemmódban dolgozhatnak. Alkalmasak időtartam beállítására, jelsorozatok és négyszögjelek előállítására, vagy CNT érintkezőn keresztül külső jelek számlálására. Az üzemmód a CRA és a CRB vezérlőregiszterekkel választható ki. Ezzel a 3. fejezetben már foglalkoztunk.

Lássunk munkához és készítsünk a CIA-k felhasználásával egy univerzális négyszögjel generátort, ami kb. 20 kHz-ig használható.

```
1 REM *** P 14 ***
2 REM
100 C2 =56576
110 AL =C2+4
120 AH =C2+5
130 CA =C2+14
140 TF =985248
150 TH =TF/2
160 POKE CA,23
170 PRINTCHR$(147)
180 PRINT:PRINT:PRINT
190 PRINT"    KEREM A FREKVENCIAT HZ-BEN   "
200 PRINT:INPUT"          ";FR
210 FW =TH/FR
220 HB =FW/256
230 HB =INT(HB)
240 LB =FW-HB*256
250 LB =INT(LB+.5)
260 IF LB =256 THEN LB=0:HB=HB+1
270 POKE AH,HB:POKE AL,LB-1
280 PRINT:PRINT:PRINT:PRINT
290 PRINT"   AZ ELOALLITOTT FREKVENCIA ERTEKE "
300 PRINT:PRINT"                                "
310 PRINT CHR$(145); :PRINT"          ";
320 PRINT (INT(TH/(HB*256+LB)*10))/10;"HERZ"
330 PRINT:PRINT:PRINT:PRINT
340 PRINT"     F1 = A FREKVENCIA VALTOZTATASA"
350 PRINT"     F7 = A PROGRAM VEGE"
360 GET A$:IF A$="" THEN 360
370 IF A$=CHR$(133) THEN PRINT CHR$(19):GOTO 180
380 IF A$=CHR$(136) THEN 400
390 GOTO 360
400 PRINT CHR$(147)
410 PRINT:PRINT:PRINT"          A PROGRAM VEGE"
420 PRINT:PRINT"    A FREKVENCIA MOST A PB6-ON MARAD."
430 PRINT:PRINT"    AZ ATPROGRAMOZASIG "
440 PRINT:PRINT"    AZ IDOZITO MEGTARTJA AZ ERTEKET ! "
450 PRINT:PRINT:PRINT"          FIGYELEM!"
460 PRINT:PRINT"   A RUN-STOP/RESTORE TORLI AZ "
470 PRINT:PRINT"    IDOZITOT !          "
READY.
```

Ez a program a CIA2 A időzítőjét használja és négyszögjeleket állít elő a felhasználói port PB6-os érintkezőjén. A frekvencia beállításához nincs szükségünk potenciométerre, vagy egyéb beállító alkatrészre, mert erre a célra egyszerűen a billentyűzetet használhatjuk.

A frekvencia 8 Hz-től 246312 Hz-ig állítható. Ezt a frekvenciát a C 64-es ütemfrekvenciájával (órajelével) állítjuk elő. Ez kvarcstabilizált, tehát a négyszögjelünk is az.
Frekvenciaosztással a kívánt frekvenciákat nem lehet pontosan beállítani. Ez minél magasabb, annál nagyobbak lesznek a frekvencialépések. Így a legmagasabb frekvencia 246 kHz, a következő kisebb azonban már csak 146 kHz. A 10 Hz – 16 kHz közötti tartományban a hiba mindig kisebb 5%-nál, sőt a legtöbb esetben még a 0,1%-ot sem éri el. Ilyen beállítási pontosságot hagyományos négyszögjel-generátorral igen nehéz elérni.
Ezenkívül a hagyományos oszcillátoroknál szinte mindig gondot okoz a hőmérséklet módosító hatása. Ez a rögzített frekvenciájú oszcillátornál már eredetileg is kicsi és a frekvenciaosztás az ebből eredő hibát még tovább csökkenti.

Nézzük a program működését:

100–150: A felhasznált változók meghatározása.
A TF a C 64-es ütemfrekvenciáját, a TH pedig a fél ütemfrekvenciáját tartalmazza Hz-ben.

160: A CIA2 A időzítőjének előkészítése.
Részletesen: Az időzítő állapotát a PB6 érintkezőn jelezzük ki. Minden egyes lefutó él a portbitet ellenkező polaritásúra kapcsolja. Folyamatos üzemmódban a számláló a nullára futás után automatikusan újra indul. A számlálót bármikor feltölthetjük az új értékkel. Órajelként a rendszerórajelet használjuk.

170–200: A kívánt frekvenciaérték bevitele INPUT-utasítással.

210: A TH értéket a bevitt frekvenciaértékkel elosztva megkapjuk a számlálandó impulzusok számát, amit az időzítő regisztereibe kell írnunk.

220–250: Az előző sorban kiszámított értéket a beíráshoz alsó (LB) és felső (HB) byte-ra bontjuk. A 250. sorban az alsó byte-ot kerekítjük, ezáltal mindig a legközelebb eső frekvenciaértéket választjuk.

260: Ha a kerekítéssel LB értéke 255-nél nagyobbra adódna, az átvitel a HB-be kerül és LB-t töröljük.

270: A kiszámított értékeket az időzítő regisztereibe töltjük. Fontos, hogy a CIA-k szerkezetéből adódóan LB értékét 1-gyel csökkenteni kell.

280–320: Ellenőrzésként a 320. sorban a kiszámított értékből visszaszámoljuk a tényleges frekvenciát, kerekítjük és kijelezzük a képernyőn.

330–470: A 340–380-as sorokban választhatunk a frekvencia értékének módosítása vagy a program befejezése között. Az utóbbi esetben a 400–470-es sorok egy tájékoztató szöveget jelenítenek meg a képernyőn.

Még egy kis kiegészítés a 270. sorhoz:

Ha az időzítőt az előzőleg megbeszélt üzemmódban használjuk, a legkisebb beállítható osztóarány 1/4. Ezt úgy kapjuk, hogy a számláló felső byte-jába (HB) 0-t, az alsó byte-ba (LB) pedig 1-et töltünk. Ha mindkét regisztert nullázzuk, akkor a PB6 (B időzítőnél PB7) kivezetés magas szintű marad.

Ha a 248 kHz-es frekvenciához a programban megadott összefüggések szerint kiszámítjuk az értékeket, akkor a 210-es sor után FW értéke 1986 lesz. Ebből a felső byte értéke HB = 0. Az LB értéke a kerekítés után 2. Ha ezt töltenénk az időzítő regiszterébe, kb. 164 kHz-es frekvenciát állítana elő. LB értékét 1-gyel csökkentve a kívánt eredményt kapjuk.

Ha a négyszögjel-generátorunkkal más TTL kapcsolásokat akarunk meghajtani, kiegészítő teljesítményfokozatot kell közbeiktatnunk. A terhelés a kimeneten ugyanis legfeljebb 1 mA lehet, azonkívül a CIA a túlfeszültségekre is rendkívül érzékeny. Emiatt az illesztő áramkörünket egy 74128 típusú IC-vel célszerű megtervezni, ami a fenti problémák ellen hatásos védelmet jelent.
Ez az IC négy darab kétbemenetű NOR-kaput tartalmaz. Az a különlegessége, hogy minden kapu 48 mA árammal terhelhető, ami a legtöbb digitális kapcsolásban elegendő. Egy 47 Ω ellenállással a kimenő fokozat zárlatvédelméről is gondoskodhatunk.
Mivel a 74128-as IC négy azonos áramkört tartalmaz, mindkét időzítőt felhasználhatjuk és az előző program bővítésével két, egymástól függetlenül beállítható négyszögjel-generátort készíthetünk. A bővítés a program leírása alapján könnyen elvégezhető.

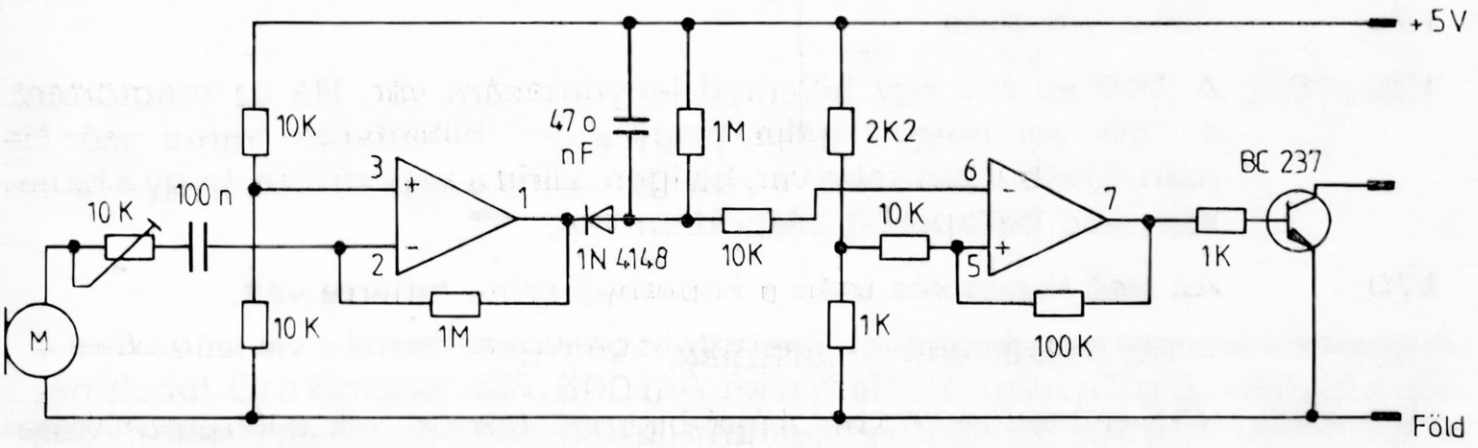
3.6 A hangkapcsoló

Ebben a fejezetben egy egyszerű hangkapcsolót fogunk építeni. A hangkapcsoló a távirányítás egyik módja. A hangot vagy zajt egy mikrofon érzékeli, amit felerősítve a kapcsoló-fokozatra vezetünk. Egy bizonyos hangerősségnél a kimeneten feszültségugrás lép fel. Ezt a kimeneti jelet használhatjuk fel a kapcsoló működtetésére.

Példánkban az erősítő és a komparátor egy LM 324 típusú integrált áramkör. A benne lévő négy műveleti erősítőből kettőt használunk.
Mikrofonként bármilyen, kb. 200 Ω impedanciájú dinamikus mikrofon megfelel, aminek frekvenciakimenete tetszőleges lehet.

Az első áramkört egyszerű váltófeszültségű erősítőként kapcsoljuk. Az erősítést egy potenciométerrel állítjuk be, értéke 200–1000-szeresig változtatható. Ezáltal a kapcsolás különböző típusú mikrofonokhoz illeszthető. Az erősítő után egy egyenirányító következik, ami a felerősített mikrofonjelet egyenirányítja és egy ideig tárolja. Ezt az egyenfeszültségű jelet egy küszöbérték kapcsolóra (komparátor) vezetjük, amit a 2-es erősítőből építettünk fel. Ezzel a kimeneten egy határozott négyszögjelet kapunk, amivel meghajtjuk a kimeneti tranzisztort.

Most már csak az a kérdés, hogy tudjuk a hangkapcsoló állapotát a legegyszerűbben lekérdezni. Erre a célra kiválóan alkalmas az 1-es vezérlőport. (Controlport 1) Aki már használt botkormányt, az bizonyára tapasztalta, hogy közvetlen üzemmódban működtetve különböző jeleket visz a képernyőre.



Ezeket a jeleket egy kis BASIC programmal, a GET-utasítással lekérdezhetjük. A botkormány helyett tranzisztorokat is alkalmazhatunk. Ezt a lehetőséget használjuk ki a hangkapcsolónál. Ebben az esetben az 1-es vezérlőport 2-es érintkezőjét használjuk bemenetként. Ha ez alacsony szintű, a billentyűzet pufferba egy CHR$ (95) íródik. Ez a „←" – billentyűt jelenti.

A következő példaprogram a hangkapcsoló alkalmazásának nem túl szerencsés esetét mutatja be. Arra mindenesetre jó, hogy a lekérdezés egyik lehetséges módját szemléltessük. Egy kis fantáziával ennél sokkal használhatóbb programok is készíthetők:

```
1 REM *** P 15 ***
2 REM
100 BC=53280
110 BG=53281
120 PRINT CHR$(147)
130 PRINT"                HANG-KAPCSOLO"
135 PRINT
140 GET A$:IF A$="" THEN 140
150 IF A$ <> CHR$(95) THEN 140
160 PRINT"                BEKAPCSOLVA !"
165 PRINT:PRINT
170 POKE BG,1
180 TI$="000000"
190 GET A$
200 IF A$=CHR$(95) THEN TI$="000000":Z=Z+1
205 POKE BC,Z
210 PRINT CHR$(145);"                    "TI$
220 IF TI$ <> "000005" THEN 190
230 POKE BG,6
240 GOTO 120
READY.
```

Nézzük a programsorok jelentését:

100–110: Ebben a két sorban a képernyőkeret és a háttér színét tartalmazó tárolórekeszek címét egy-egy változóba töltjük (BG = háttér, BG = keret).

120: Képernyőtörlés

130–165: A 140-es sor egy billentyű lenyomására vár. Ha ez megtörtént, a 150. sor megvizsgálja, hogy a „←" billentyűről van-e szó. Ha nem, további bevitelre vár, ha igen, kiírja a képernyőre, hogy a hangkapcsoló bekapcsolt állapotban van.

170: Az első kapcsolás után a képernyő színe fehérre vált.

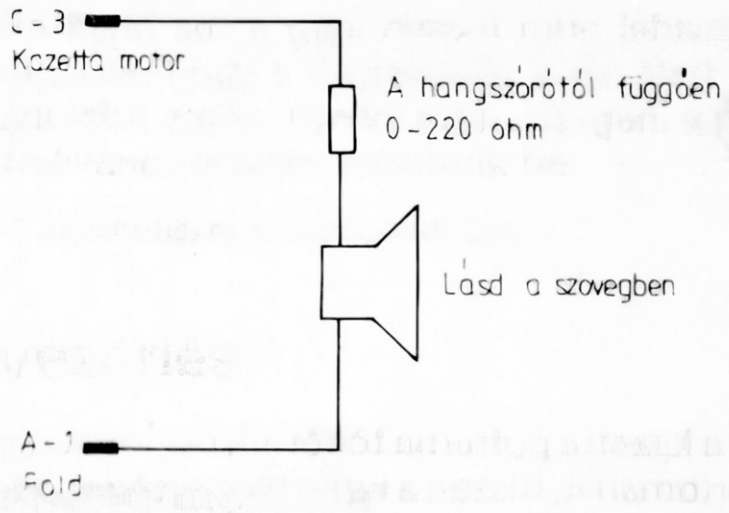
180: A TI$ (rendszeróra) nullázása.

190–240: Ha 5 másodpercen belüi újabb kapcsolás történik, a keretszín változik és az óra ismét nullára áll. Ha a kapcsolás éppen az 5. másodpercben történik, a háttér színe ismét kékre vált és a program, visszatérve a 120-as sorba egy új kapcsolásra vár.

3.7 Hangképzés a kazetta porton

Első pillantásra feleslegesnek tűnhet a hangképzés következőkben ismertetésre kerülő módja, hiszen a C 64-esbe beépített SID regiszter számtalan lehetőséget kínál hangok, zajok, dallamok... stb. előállítására. Ne felejtsük el azonban azt, hogy ehhez tévére vagy erősítőre is szükségünk van.

A számítógéppel számos olyan feladatot oldhatunk meg, pl. hőmérsékletváltozás figyelése, vagy gyártási folyamat ellenőrzése, amihez a tévékészülék üzemeltetése egyébként nem lenne szükséges. Ilyenkor nagyon jól használható egy kiegészítő akusztikus áramkör, amit a kazetta porton keresztül csatlakoztathatunk a számítógépre.
Ezzel az éjszakai órákban pl. tényleges riasztóberendezésként használhatjuk számítógépünket. Ellenőriztethetjük vele, hogy lakásunk ablakai és a bejárati ajtó megfelelően zárva van-e. Ha rendellenességet „észlel", az akusztikus illesztőt működésbe hozza és hangjelzéssel figyelmeztet. Ráadásul nyugodtan nézhetjük ezalatt a tv éjszakai műsorát is, hiszen a riasztóberendezésünk nem foglalja le a készüléket.
Egy kazetta-port csatlakozóból, egy ellenállásból és egy hangszóróból olyan riasztócsengőt készíthetünk, ami egy kis program segítségével legmélyebb álmunkból is képes felébreszteni, ugyanakkor kikapcsolt állapotban nem fogyaszt áramot.



A működési elv a kazettásegység motorfeszültségének gyors be- és kikapcsolásán alapul. Ez a kimenet max. 500 mA-rel terhelhető, ezért a hangszóróval, melynek impedanciája 4–8 Ω, sorba kell kötni egy ellenállást, ami az áramerősséget korlátozza. Ezzel ugyan csökken a hangerő, de 200 mA-nél még mindig elég erős lesz. A kérdéses ellenállás legkisebb értéke 8 Ω hangszóró esetén 10 Ω lehet.
Hangszóró helyett telefonhallgatót is alkalmazhatunk. Mivel ennek 200 Ω az ellenállása, nem kell előtétellenállást közbeiktatni.

Sajnos BASIC-ből 50 Hz-nél magasabb frekvenciát nem tudunk előállítani. A

10 POKE 1,23 : POKE 1, 55 : GOTO 10

utasítássorral csak egy brummogó hangot tudunk előcsalni a hangszóróból. Emiatt kénytelenek vagyunk gépi kódú programot írni a hangszóró működtetéséhez. Ez most csupán néhány byte hosszúságú és a „csak" BASIC-ben programozók is könnyen megértik.

```
22 ; *** P 16 ***
23 ;
25 .OPT P8,01
100 *=$033C
110 PORT =1
120 BE =%010111
130 KI =%110111
140 DELAY =$F7
150 SEI
160 LDY #0
170 BEEP LDX DELAY
180 LDA #BE
190 STA PORT
200 LOOP1 DEX
210 BNE LOOP1
220 LDX DELAY
```

```
230 LDA #KI
240 STA PORT
250 LOOP2 DEX
260 BNE LOOP2
270 DEY
280 BNE BEEP
290 CLI
300 RTS
READY.
```

Ezt a kis programot a kazetta pufferba töltöttük be. Most nyugodtan igénybe vehetjük ezt a tárolótartományt, hiszen a kazettásegységet, a hangszóró csatlakoztatása miatt, úgysem használhatjuk.
Néhány szó a program működéséről:
Először kikapcsoljuk a megszakításokat. Erre nincs feltétlenül szükség, de ha elmulasztjuk, az előállított hang nagyon szabálytalan lesz. A Y-regisztert periódusszámlálóként használjuk. Az X-regiszter értéke a 247-es tárolórekeszből származik. Ide írhatjuk be POKE – utasítással a hangmagasságra jellemző értéket. Minél kisebb ez az érték, annál magasabb lesz az előállított hang.

A programot a következő BASIC nyelvű lista segítségével könnyen betölthetjük:

```
1 REM *** P 17 ***
2 REM
100 FOR I=828 TO 853
110 READ X:POKE I,X:S=S+X:NEXT
120 DATA 120,160,  0,166,247,169, 23,133,  1,202,208,253
130 DATA 166,247,169, 55,133,  1,202,208,253,136,208,235
140 DATA  88, 96
150 IF S<>3879 THEN PRINT"HIBA AZ ADATOKBAN!!":END
160 PRINT"OK"
READY.
```

Miután a fenti programot elindítottuk, a következő BASIC programmal kipróbálhatjuk, hogyan befolyásolja a 247-es tárolórekesz tartalma a hangmagasságot:

```
1 REM *** P 18 ***
2 REM
100 PRINT CHR$(147)
110 PRINT"KEREM AZ ERTEKET 1 ES 255 KOZOTT BEIRNI"
120 INPUT A
130 IF A=0 THEN END
140 POKE 247,A
150 SYS 828
160 GET A$:IF A$ <> "" THEN GOTO 100
170 FOR I=1 TO 200:NEXT I
180 GOTO 150
READY.
```

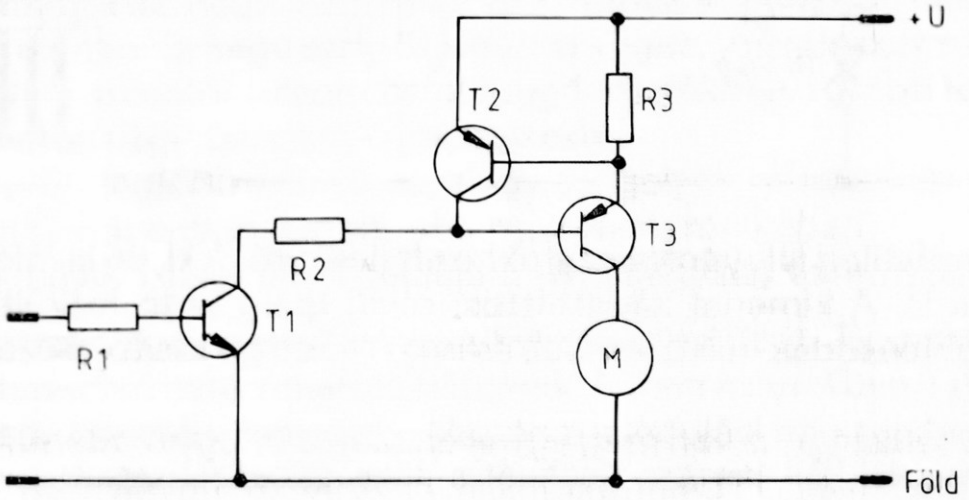
Az érték bevitele után kerül sor a gépi nyelvű rutin lehívására (SYS 828). A 160-as sorban ellenőrizzük, hogy a végrehajtás ideje alatt történt-e billentyűlenyomás. Ha nem újra lefut a gépi nyelvű rutin, ha igen, a program visszatér a 100-as sorba és új frekvenciaértéket adhatunk be.

A programot egy „∅" bevitelével fejezhetjük be.

3.8 A motorvezérlés

3.8.1. Az egyenáramú motorok

Az egyenáramú motorok vezérlése legegyszerűbben a 3.2 alfejezetben ismertetett tranzisztorkapcsolással valósítható meg. Fennáll azonban az a veszély, hogy a motor csak nagyon nehezen, vagy egyáltalán meg sem indul. Ilyenkor a motoráram gyorsan elérheti a tranzisztor kritikus áramát, ezért ebben az esetben ez az egyszerű kapcsolás már nem elegendő. Sokkal alkalmasabb a következő ábrán bemutatott áramkorlátozó kapcsolás:



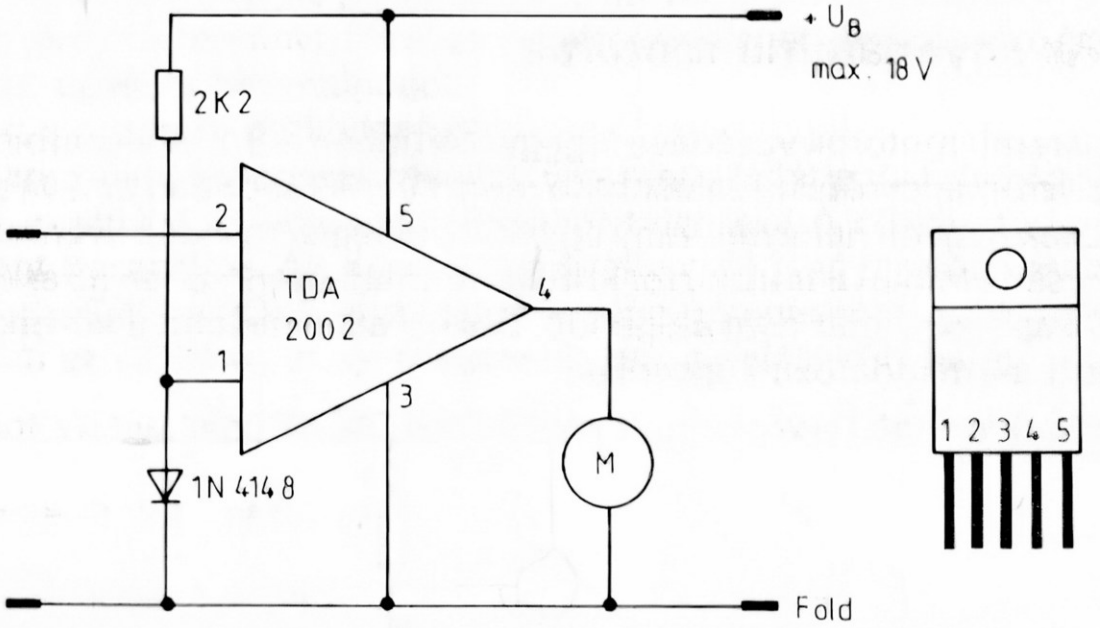
	15 V	30 V
R1	22 K	22 K
R2	4K7	10 K
R3	lásd a szövegben	
T1	BC 237	
T2	BC 337	
T3	a max. áramerősségtől függően hűtőborda !	

Amint a T1 bázisfeszültséget magas szintre vittük (R1-en keresztül), a motor elindul. Az R3 ellenálláson a motor áramfelvételével arányos feszültségesés jön létre. Ha az áramfelvétel túl nagy, a T2 bázisán csökken a feszültség. Amikor az így kialakult feszültségszint 0,7 V-tal kisebb, mint az üzemi feszültség, a T2 vezetni kezd és a T3 bázisa pozitívabb lesz. Ezzel a T3 zár és a motor kisebb áramot

kap. A maximális áramerősség az R3 ellenállás értékétől függ, amit a következő összefüggés alapján számítunk ki:

$R3 = 0{,}7/I_{max}$

Ha például a megengedett legnagyobb áramerősség 300 mA, akkor az ellenállás számított értéke 2.3333333 Ω, gyakorlatban 2.2 Ω.
Max 3.5 A áramfelvételű és 18 V-nál kisebb feszültségű motorokhoz igen egyszerű és megbízható motorvezérlést építhetünk egy TDA2002 típusú IC felhasználásával.

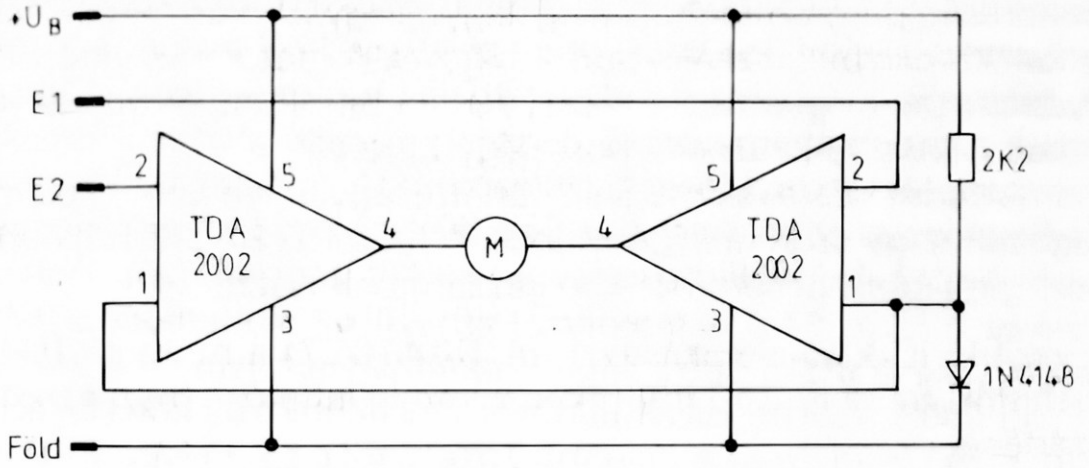


Ezt az IC-t eredetileg NF-végfokozatokhoz fejlesztették ki, de kiválóan alkalmas a mi céljainkra is. A kimenet zárlatbiztos, rövid ideig tartó feszültségcsúcsokat 40 V-ig meghibásodás nélkül elvisel, és egy olvadó biztosító védi az IC-t a túlterhelések ellen.
Az IC-n kívül szükségünk van még egy ellenállásra és egy diódára. A leírás szerint elkészített kapcsolás TTL-kompatibilis. Feltétlenül figyelembe kell vennünk, hogy invertáló, tehát az erősítő bemenetére érkező magas szintű jel hatására a motor leáll, alacsony szintű jel esetén viszont tovább forog.

Előfordulhat, hogy a motor kikapcsolt állapotban is, egészen lassan tovább forog. Ennek az az oka, hogy az erősítő kimenetét nem pontosan OV-ra szabályozták le. A legkisebb kimenő feszültség, ami kb. 0.15 V, már elegendő arra, hogy néhány motortípust mozgásban tartson. Ebben az esetben a motorhoz vezető ágba egy diódát kell beépíteni, aminek a típusát a motor áramfelvétele határozza meg. 1 A-ig elég egy 1N4001 típusú, ennél nagyobb áramerősség esetén pedig a 3 A-es 1N5401 típust, vagy ehhez hasonlót kell alkalmazni.

Eddigi példáinkkal a motort csak egyirányba forgathatjuk. Gyakran szükség van azonban arra, hogy egy motor mindkét irányban üzemeljen. Ez egy hídkapcsolással valósítható meg, amihez 2 darab TDA 2002 típusú IC-re van szükségünk. Az ábrán egy kipróbált kapcsolást mutatunk be, amivel a motor tetszés szerint balra

is és jobbra is forgatható. A vezérlés két portbiten (E1, E2) keresztül történik. A motor forgásiránya attól függ, hogy melyik bitet kapcsoljuk be. Ha mindkét bitet kikapcsoljuk, a motor leáll.



3.8.2 Léptetőmotorok

A léptetőmotorok alapvetően különböznek a közismert egyenáramú motoroktól. Működésük lényege az, hogy a betáplált elektromos impulzusok hatására tengelyük meghatározott nagyságú szögelfordulást végez. Vezérléskor a megfelelő jel a motor tengelyét, a motor felépítésétől függő mértékben, tovább forgatja. A vezérléshez a motor négy tekercccsel rendelkezik.
A léptetőmotorok legfőbb előnye az, hogy tengelyük mindenkori helyzetét ismerjük. Használatuk emiatt terjedt el a mikroelektronikában.
Alkalmazásuk egyik kézenfekvő példája a lemezegység és a nyomtató.

A lemezegységben ilyen motor végzi a fej pozicionálását. Ez a magyarázata a működéskor tapasztalható zakatoló hangnak. Ebben az esetben a motor a fejtovábbítót a 0. sáv irányába mozgatja. Miután a kiindulási helyzetét elérte, pontos helyzetének meghatározásához csak a lépések számát kell ismernünk.

A nyomtatónál a papírtovábbításhoz és a nyomtatófej mozgatásához használnak léptetőmotorokat. A helyzetmeghatározás ebben az esetben is a lépések számával történik.

Hogy történik maga a vezérlés?

Mindenekelőtt 3 üzemmódot különböztetünk meg.

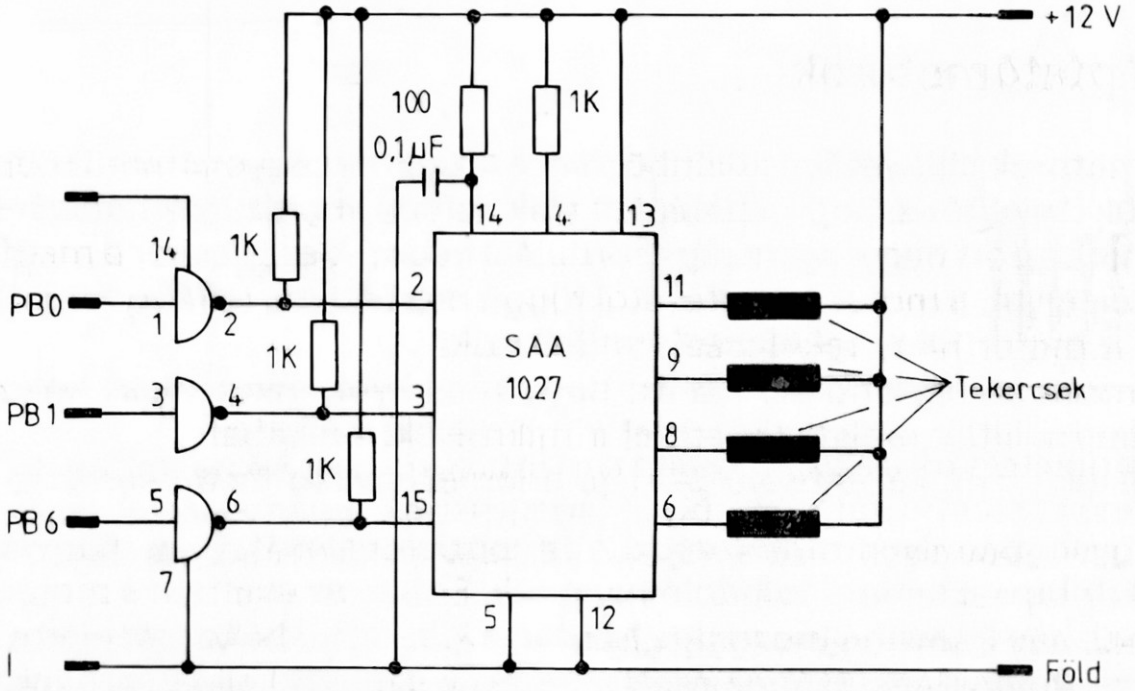
Az első esetben a tekercseket egymás után egy-egy áramimpulzussal gerjesztjük. A motor minden egyes impulzus hatására egy lépéssel továbbfordul.

A második üzemmódban mindig egyidejűleg gerjesztünk két-két egymás mellett lévő tekercset. Először tehát az 1-est és a 2-est, aztán a 2-est és a 3-ast, a 3-ast és a 4-est és végül a 4-est és az 1-est. Ezután az egész újból az 1-es és 2-es tekercccsel folytatódik.
Ez az eljárás kétszer annyi energiát követel, de a motor egyenletesebben forog.

A harmadik lehetőség az első kettő kombinációja. Először az 1-es, aztán az 1-es és 2-es, majd a 2-es tekercset gerjesztik, ... stb.

Közelebbről megvizsgálva a különböző eljárásokat, megállapíthatjuk, hogy a második lehetőség programtechnikailag nagyon egyszerűen megoldható. A motort össze kell kötni a felhasználói port 4 bitjével. A motor mozgatásához az akkuban folyamatosan forgatni kell a %00110011 bitmintát, és minden így kapott értéket ki kell vinni a felhasználói port négy bitjére.
Az eddig ismertetett eljárások megvalósíthatók ugyan, de nem igazán praktikusak. Túlságosan nagy szoftvert igényelnek. A motorok BASIC-ből nem vezérelhetők, ezért gépi kódú programot kell készítenünk hozzá.

Speciális vezérlő IC-k alkalmazásával (pl. SAA1027) a motorok BASIC-ből is programozhatók. Ez az IC 350 mA tekercsáramig lehetővé teszi a motorok közvetlen vezérlését.



Vizsgáljuk meg a kapcsolást:

Az IC R, S és T érintkezőjét (az ábrán 2, 3, 15) a felhasználói porttal kötjük össze. Jelentésük a következő:

R: Iránybemenet. A motor forgásirányát határozza meg. (alacsony szint = egyik irány; magas szint = másik irány)

S: Start–stop bemenet. A motor a bemenet magas szintjénél indul.

T: Ütem- vagy triggerbemenet. Ha az S bemenet magas szintű, akkor az erre a bemenetre érkező minden egyes ütemjel a tengely elfordulását váltja ki. Irányát az R bemenet állapota határozza meg.

A T bemenet a PB6 érintkezővel van összeköttetésben, amire, mint már arról korábbi fejezeteinkben beszéltünk, kiadható az A időzítő által előállított jelsorozat. Az időzítőt úgy programozzuk, hogy a jelsorozat folyamatos legyen.

A PBO érintkező az IC S bemenetét, a PB1 pedig az R bemenetét vezérli. Így a forgás be- és kikapcsolását a PBO, a forgásirányt pedig a PB1 bitekkel programozhatjuk.

Figyelmeztetés! Az SAA1027-es felépítéséből adódóan az S bemenetre adott alacsony szint a motort leállítja, de ez nem jelenti azt, hogy a motor feszültségmentes. Az IC kimenetei sokkal inkább tükrözik a megfelelő állapotot. A 6-os és 9-es érintkezők alacsony szintűek lesznek. Ezzel a megoldással a megfelelő tekercsen keresztül legfeljebb a tekercsellenállással korlátozott áram folyik. Alacsony tekercsellenállású motoroknál a viszonylag magas nyugalmi áram idővel káros felmelegedést okozhat. Ezt egy egyszerű, nyitott kollektoros kimenetű inverterrel kiküszöbölhetjük. Az inverter kimenetét az SAA1027 4-es érintkezőjére kötjük. Ezáltal az inverter bemenet magas szintjénél a 4-es bemenetre alacsony szintű jel kerül és minden kimenet magas szintű lesz. Ekkor semmiféle tekercsáram nem folyhat. Ez a kiegészítő áramkör a kapcsolási rajzon nem szerepel.

A motor vezérlésének teljes BASIC programja a következő:

```
1 REM *** P 13 ***
2 REM
100 S=20
110 C2=56576
120 PB=C2+1
130 BR=C2+3
140 AL=C2+4
150 AH=C2+5
160 CA=C2+14
170 POKE 650,255
180 POKE BR,255
190 POKE AH,20
200 POKE AL,0
210 POKE CA,23
220 PRINT CHR$(147):PRINT:PRINT:PRINT
230 PRINT"       LEPTETO MOTOR"
240 PRINT:PRINT"   F1/F2 MOTORT BE-/KIKAPCSOLASA"
250 PRINT:PRINT"   F3/F4 MOTOR BALRA/JOBBRA"
260 PRINT:PRINT"   F5/F6 MOTOR GYORSABBAN/LASSABBAN"
270 PRINT:PRINT"   F7 PROGRAM VEGE"
280 GET A$:IF A$=""THEN 280
281 IF ASC(A$)<133 OR ASC(A$)>139THEN GOTO 280
282 A=ASC(A$)-132
283 ON A GOSUB 290, 310, 330, 350, 300, 320, 340
284 GOTO 280
290 POKE PB, PEEK(PB) AND 254:RETURN
300 POKE PB, PEEK(PB) OR 1: RETURN
310 POKE PB, PEEK(PB) AND 253:RETURN
```

```
320 POKE PB, PEEK(PB) OR2:RETURN
330 S=S-1:GOSUB 370:POKEAH,S:RETURN
340 S=S+1:GOSUB 370:POKE AH,S:RETURN
350 POKE PB,1:END
370 IF S<4 THENS=4
380 IF S>255 THEN S=255
390 RETURN
READY.
```

Ezzel a programmal azonban nem használtuk ki a léptető motor legfontosabb tulajdonságát. Nem számoltuk a lépéseit, így a helyzetéről semmit sem tudunk. A következő programban már pótoltuk ezt a hiányosságot. Az alkalmazott elvet könnyen átültethetjük gépi kódba.

```
1 REM *** P 20 ***
2 REM
110 C2=56576
120 PB=C2+1
130 BR=C2+3
140 CA=C2+14
150 POKE 650,255
160 POKE BR,255
170 POKE CA,0
180 FOR I=1 TO 6:LS$=LS$+CHR$(157):NEXT
190 PRINT CHR$(147)
200 PRINT:PRINT:PRINT
210 PRINT"       LEPTETOMOTOR"
220 PRINT:PRINT" FORGASIRANY (B/J)   ";
230 INPUT RI$:GOSUB 280
240 PRINT:PRINT" A LEPESEK SZAMA     ";
250 INPUT SC:GOSUB 310
260 PRINT:PRINT" OSSZES ELMOZDULAS                    ";LS$;AZ
270 PRINT CHR$(19):GOTO 200
280 IF RI$="B"THEN POKE PB,0:FL=-1
290 IF RI$="J"THEN POKE PB,2:FL=1
300 RETURN
310 FOR I=1 TO SC
320 POKE PB,PEEK(PB) OR 64:POKE PB,PEEK(PB) AND 191
330 NEXT: AZ=AZ+(FL*SC):RETURN
READY.
```

4. TOVÁBBI IC-K CSATLAKOZTATÁSA

A felhasználói porton rendelkezésünkre álló be- és kimenő vezetékek egyidejűleg nem sok vezérlési feladat megoldását teszik lehetővé. Gondoljunk csak egy komolyabb modellvasútra. A számos kitérő, sorompó és jelző szabályozásához rengeteg vezetékre van szükség. Ugyanez elmondható a háztartásban vagy a hobbi elektronikában adódó vezérlési feladatokról is.
Ez a hiányosság szerencsére elég könnyen pótolható. A 64-eshez viszonylag egyszerűen csatlakoztathatunk külső IC-ket, mint pl. a CIA 6526-ost, a 6522-est vagy a 8255-öst. A csatlakoztatás a bővítő porton keresztül történik. Ezt nyomtatott áramkörök számára alakították ki, ezért sajnos a felhasználói porttól eltérően a csatlakoztatás nem oldható meg egy egyszerű dugósávval. Némi kísérletezés után azonban sikerült egy egyszerű és célszerű megoldást találnunk.

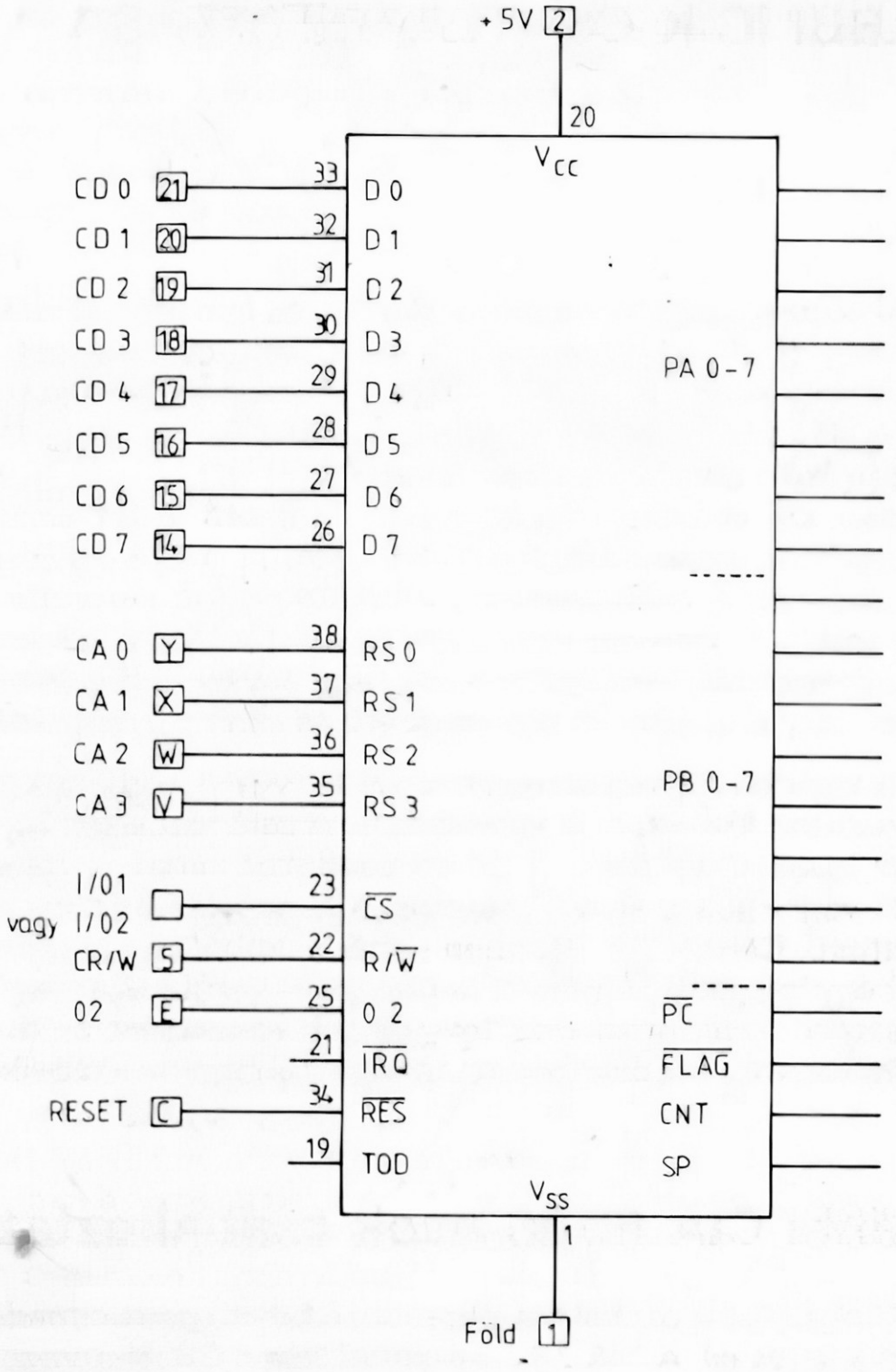
A közismert és kísérleti célokra szívesen használt lyukrácsos lapok mindkét oldalon csíkban maratott kivitelben is kaphatók. A vezető rézcsíkok közötti távolság 2,54 mm, pontosan megfelel a C 64-es csatlakozójának osztásával. A lapot lombfűrésszel vághatjuk a kívánt méretre. A kapcsolás ezután ezen kívánság szerint felépíthető. Célszerűbb azonban ebből a lapból egy adapterdarabot készíteni. Ehhez egy legalább 40 pólusú dugós csatlakozót kell a megfelelő méretű nyákra forrasztani. Ezzel most már tetszőleges kapcsolást építhetünk fel. Az adapter-kártyához való csatlakozás ezután egy többpólusú kábellel valósítható meg.

4.1 További CIA 6526-osok csatlakoztatása

A C 64-esre több CIA-t is csatlakoztathatunk. A bővítő porton rendelkezésünkre áll minden szükséges jel. A CIA 23-as érintkezőjén a CS jelez vagy az I/01 vagy az I/02 jellel kell összekötni. Ez szabadon megválasztható, de ügyelni kell arra, hogy báziscímük különböző: az I/01 báziscíme: $DE00 = 56832, az I/02 báziscíme: $DF00 = 57088. Az így csatlakoztatott 6526-osok PEEK és POKE utasításokkal ugyanúgy kezelhetők, mint a beépített CIA-k.

Ha több ilyen IC-t akarunk csatlakoztatni, akkor a $\overline{CS}$-jelet címdekóderekkel kell előállítani. Egy darab 74LS138 típusú IC legfeljebb 8, kettő már 16 CIA címzését tudja ellátni. A megfelelő kapcsolás a következő fejezetben, a VIA6522-nél található.

Abban az esetben, ha valóban több be- és kimenő vezetékre van szükségünk, a költségek miatt más, olcsóbb lehetőségeket kell keresnünk. Megfelelő lehet a már említett 6522-es, vagy a 8255-ös.



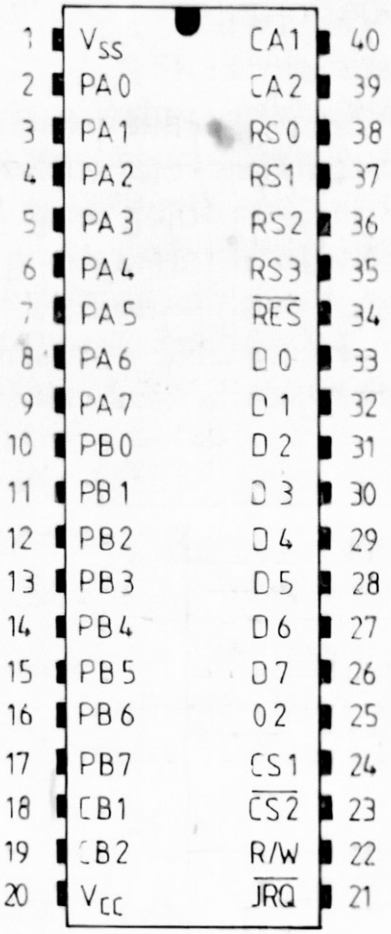
4.2 A VIA6522-es

A VIA6522-es IC (Versatile Interface Adapter) a 6526-os idősebb testvérének tekinthető. Tizenhat belső regisztere van, amelyekkel két 8 bites adat-portot, két 16 bites időzítőt, egy léptetőregisztert és különböző vezérlő vezetékeket kezelhetünk.

A 6522-esben még a valósidejű óra (TOD) is megtalálható. Kezelése valamivel bonyolultabb, mint a 6526-os esetében, de az árkülönbség, és az, hogy a VIA sokkal könnyebben beszerezhető, messzemenően kárpótol ezért a hiányosságért.

Vegyük szemügyre az IC csatlakozási lehetőségeit.

Az A port a programozástól függően bemenetként vagy kimenetként dolgozhat.



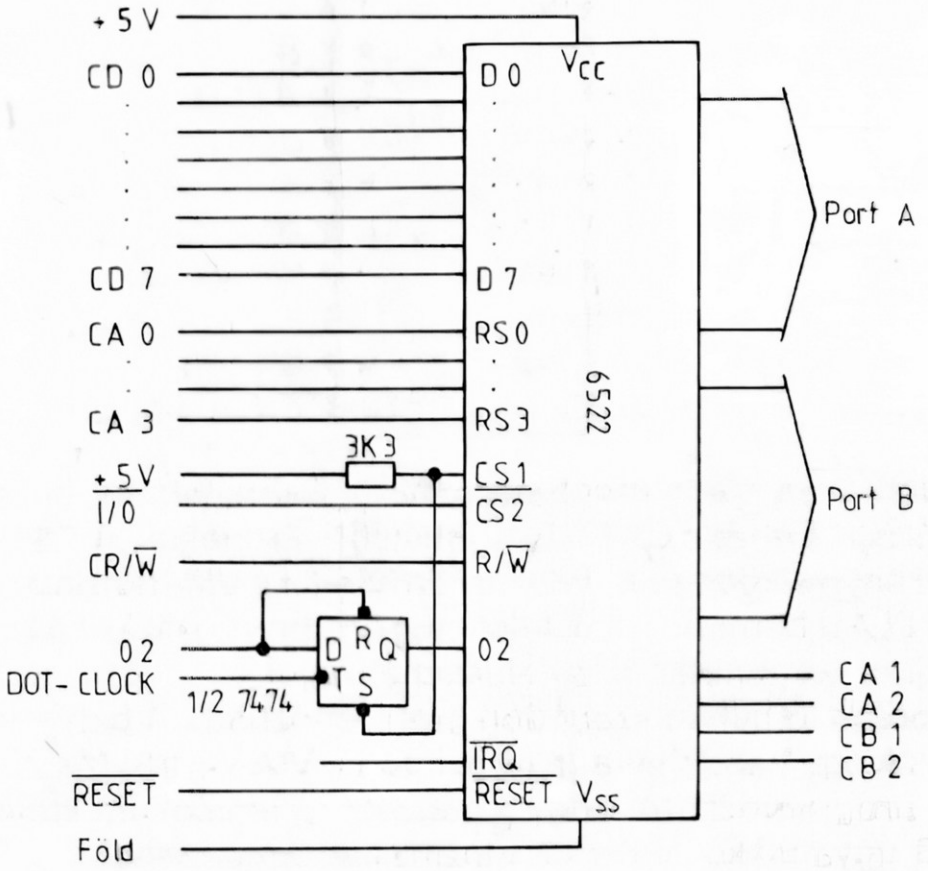
Minden bit adatiránya külön meghatározható. Bemenetként használva a bevitt adatot a VIA tárolja. Ehhez a CA1 jelű érintkezőt használjuk. A CA1 minden lefutó éle a bemenetként működő port-bitek információit a VIA-ba írja. Ez a csatlakozás így a 6526-os FLAG bemenetének felel meg. Programozáskor a két CA csatlakozást megszakítás bemenetként is felhasználhatjuk.
A B port is mindkét irányban szabadon programozható. A bemeneteken megjelenő információk egy handshake-jellel szintén a VIA-ba írhatók. Kiegészítésként a PB7 bit úgy programozható, hogy a két időzítő egyébként kimeneteként működjön. A PB6 ugyanakkor a másik időzítő bemenete lehet.

Megfelelő programozással a CB1 handshake-bemenetként működik, de a CB1 és CB2 más üzemmódokban is dolgozhat. A CA vezetékekhez hasonlóan ezek is használhatók megszakítás bemenetként, továbbá a léptetőregiszter be- és kimeneteiként. Ez utóbbi esetben ezekkel a vezetékekkel soros adatátvitel valósítható meg.

Az $\overline{\text{IRQ}}$ kimenetet általában egy megszakítás előállításához használjuk. Ez mindig akkor alacsony szintű, ha valamilyen előre meghatározott esemény lép fel. Ennek például az időzítővel, a léptetőregiszterrel vagy a vezérlő porttal kapcsolatban lehet jelentősége. Példánkban az $\overline{\text{IRQ}}$ érintkező tetszés szerint a bővítő port $\overline{\text{NMI}}$ vagy $\overline{\text{IRQ}}$ kivezetésével köthető össze.

A processzor az R/$\overline{W}$ érintkezőn keresztül jelzi, hogy az adatokat 6522-es 16 regisztere egyikébe beírni, vagy abból kiolvasni akarja-e. Az R/$\overline{W}$-nek egy különlegessége is van, amire rögtön kitérünk.
A processzor és a VIA közti adatforgalom a D0-D7 adatvezetékeken (adatbusz) zajlik.

A VIA 16 belső regiszterének címzése valamivel bonyolultabb. Erre a célra összesen hat érintkezője van. A tényleges regiszterszámot a négy RS (Register Select = regiszterkiválasztó) vezetékkel adjuk meg. Ezek rendszerint a processzor alsó négy címbitjével vannak összeköttetésben, így az összes regiszter címzése megvalósítható. Kiegészítésképpen minden 6522-esnek két CS (Chip-Select)-bemenete van. A CS1 magas-aktív, a $\overline{CS2}$ alacsony-aktív. A VIA kiválasztásához tehát a CS1-jelnek magas szintűnek, a $\overline{CS2}$-jelnek pedig egyidejűleg alacsony szintűnek kell lennie.



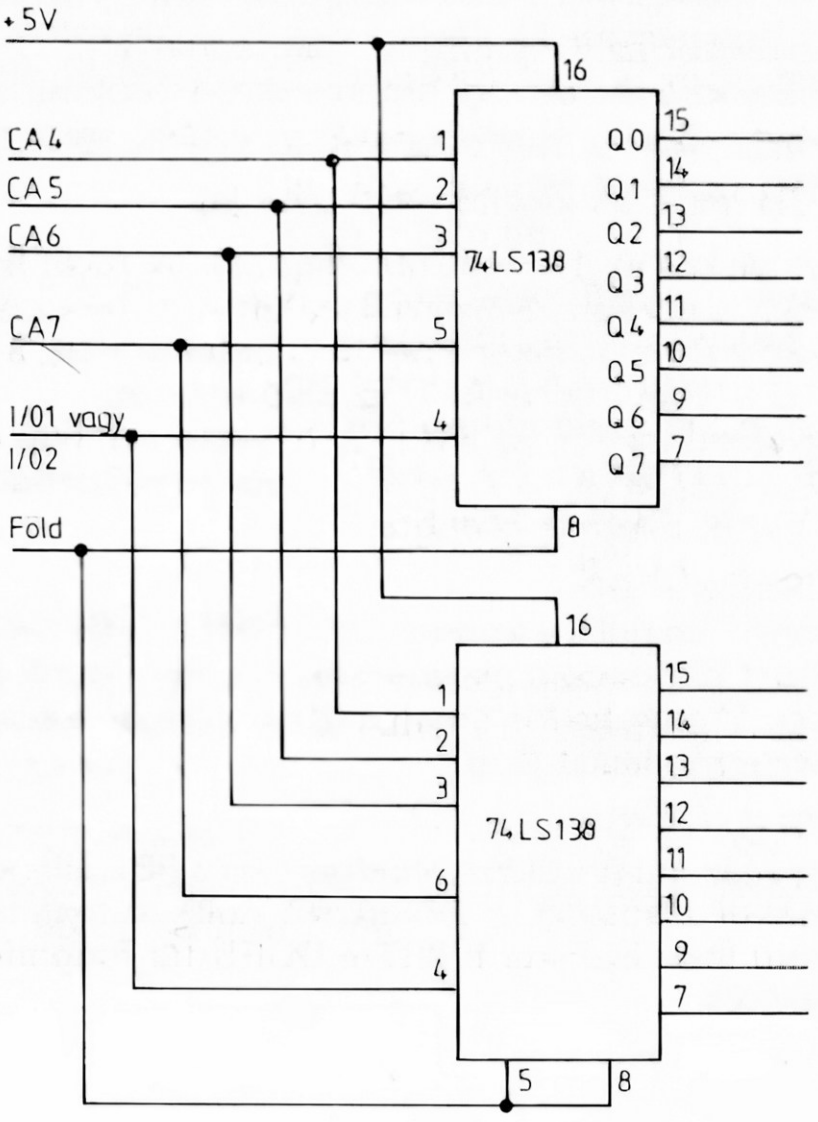
Habár ez az IC is a 65xx típusú építőelemek családjába tartozik, nem egyszerű a C 64-eshez való illesztése. Ennek oka a 64-es időzítőjében keresendő. A szabályos működéshez ugyanis a Ø2 ütemjelet az R/W-jelhez képest kis mértékben késleltetni kell.
A következőkben a 6522-es illesztésére egy kipróbált kapcsolást mutatunk be. A késleltetés egy 74LS74 típusú flip-floppal és a számítógép DOT CLOCK-jelével történik. A DOT CLOCK-jel frekvenciája a Ø2 nyolcszorosa. Amint a 64-es Ø2-jele magas szintű lesz, a DOT CLOCK következő felfutó élénél a flip-flop kimenete is magas szintű lesz. Így kapjuk meg a Ø2 csekély, de kielégítő késleltetését.

Sajnos azonban ezzel az eljárással a ∅2 lefutó élét is késleltetjük. Ezt elkerülhetjük, ha a számítógép ∅2 jelét a flip-flop CLEAR-bemenetére kötjük. Így, amikor a ∅2 alacsony szintű lesz, a flip-flop alapállapotba kerül.

A kapcsolási rajzon látható, hogy a $\overline{CS}$ bemenetet az $\overline{I/O1}$ jellel kötöttük össze. Ezzel a VIA számára a $DE00–$DEFF tárolótartományt foglaltuk le. Ebben a címtartományban a 16 belső regiszter mindegyike 16-szor fordul elő. Ha a VIA által rendelkezésünkre álló vezetékek száma elegendő, akkor ez így rendben is van. Ha azonban egy VIA kevés, akkor a következő kapcsolást alkalmazhatjuk, amivel több (de legfeljebb 16) VIA-t csatlakoztathatunk. Ebben az esetben egyszerre 256 db I/O vezetékkel rendelkezhetünk.
A kapcsolást 2 db 74LS138 típusú dekóderből építettük fel. A kimeneteken a mindenkori 16 byte-os címterületek számára alacsony-aktív jelet kapunk. Így pl. a felső címdekóder Q0 kimenete a $DE00–$DE0F tartományban, az alsó címdekóder Q3 kimenete a $DEB0–$DEBF tartományban alacsony.

Mivel a C 64-es a $DF00–DFFF címtartományban is használ I/O-jelet, nevezetesen az I/O2-t, a port-vezetékek száma megduplázható.

A legtöbb esetben nincs szükség ilyen nagyszámú külső IC-re, ezért az alsó címdekódertől eltekinthetünk. Így a csatlakoztatható 6526-os és 6522-es IC-k számát 8-ra korlátozzuk, de más megoldás is elképzelhető.



Ezek után nézzük meg közelebbről a VIA regisztereit:

0. regiszter ORB (a B port adatregisztere)

Ez a regiszter a B port állapotát rögzíti. Írható és olvasható. Figyelem! Az IFR 3. és 4. bitjét törli, ezenkívül a PCR 5...7 bitjének állapotától függően befolyásolja a CA2 vezetéket.

1. regiszter ORA (az A port adatregisztere)

Az A port állapotát rögzíti. Írható és olvasható.
Figyelem! Az IFR 0. és 1. bitjét törli, valamint a PCR 1...3 bitjének állapotától függően befolyásolja a CA2 vezetéket.

2. regiszter DDRB (a B port adatirány regisztere)

Ezzel a regiszterrel a B port vezetékeit külön-külön bemenetként vagy kimenetként programozhatjuk. A regiszter 0-ás bitjei a megfelelő adatvezetéket bemenetre, 1-es bitjei pedig kimenetre kapcsolják.

3. regiszter DDRA (az A port adatirány regisztere)

Ugyanaz, mint 2. regiszter, de az A portra vonatkoztatva.

4. regiszter T1CL (az 1-es időzítő alsó byte-ja)

Olvasáskor az 1-es időzítő állapotának alsó byte-ját adja.
Íráskor az érték először egy közbenső tárolóba kerül és csak a felső byte (5. regiszter) beírását követően kerül a számlálóba.

5. regiszter T1CH (az 1-es időzítő felső byte-ja)

Olvasáskor az 1-es időzítő állapotának felső byte-ját adja.
Íráskor a közbenső tárolóval együtt veszi fel a számláló felső bytejának értékét. Amikor a betöltés befejeződött, a közbenső tárolóból a 4. regiszter átveszi az alsó byte-ot.
Figyelem! Íráskor az IFR (13. regiszter) 6. bitje törlődik. Az ACR (11. regiszter) 6. és 7. bitjétől függően változhat a PB7 vezeték és az IFR 6. bitjének állapota:

ACR 6/7 = 00:
A számláló nulla-átmeneténél az IFR 6. bitje bekapcsolódik. Ez az állapot mindaddig megmarad, míg nem írunk az 5. regiszterbe. Még akkor is, ha folyamatos üzemmódban a számláló újra kezdi a visszaszámlálást 0-ig.

ACR 6/7 = 01:
Ugyanaz, mint az előző esetben, de a PB7 állapota az 5. regiszter írásakor alacsony, a következő nulla-átmenetnél pedig magas szintű lesz. Ilyenkor a PB7 a DDRB-től függetlenül kimenetként működik.

ACR 6/7 = 10:
Az időzítő az egyszer betöltött értéktől ciklikusan számlál nulláig, az IFR 6. bitjét azonban csak a T1CH betöltését követő első nulla-átmenetkor állítja be. Az IFR6 visszaállítása a T1CL olvasásával, a T1CH írásával, vagy közvetlen módon (ld.: a 13. regiszternél) történhet.

ACR 6/7 = 11:
Ugyanaz, mint az előző esetben, de a PB7 állapota minden egyes nulla-átmenetkor változik.

6. és
7. regiszter T1LL és T1LH (1-es időzítő alsó és felső byte-ja)

Ezeket a regisztereket írással is és olvasással is csak a közbenső tárolón keresztül tudjuk elérni. Ennek akkor van jelentősége, ha az időzítő ún. szabadonfutó üzemmódban (FREE RUNNING MODE) működik. (A 11. regiszter 6. bitje = 1.) Ebben az üzemmódban a számláló minden egyes nulla-átmenetkor ezekbe a regiszterekbe automatikusan beíródik a közbenső tároló tartalma.

8. és
9. regiszter T2CL és T2CH (a 2-es időzítő alsó és felső byte-ja)

A regiszterek írására és olvasására értelemszerűen a 4. és 5. regiszternél leírtak érvényesek. Itt azonban, a 11. regiszter 5. bitjének állapotától függően kétféle üzemmód létezik.

ACR 5. bit = 0: Ugyanaz, mint az 1-es időzítő 00-ás üzemmódja, de a nulla-átmenet az IFR (13. regiszter) 5. bitjét állítja be.

ACR 5. bit = 1: A 2-es időzítő a PB6-ra érkező külső jel hatására a betöltött értéktől nulláig számol. Nulla-átmenetnél a 13. regiszter 5. bitjét beállítja. Az időzítő ezután újra kezdi a visszaszámlálást, de az IFR 5. bitje mindaddig 1-es marad, míg nem írunk a 9. regiszterbe. A regiszter írása visszaállítja az IFR 5. bitjét.

10. regiszter SR (léptetőregiszter)

Ez a regiszter minden további nélkül olvasható és írható. A lejátszódó folyamatok az ACR (11. regiszter) 2-4 bitjének állapotától függnek:

ACR 2-4 = 000:
A léptetőregiszter írható és olvasható, anélkül, hogy ez bármilyen következménnyel járna.

ACR 2-4 = 100:
A CB2-re érkező adatok a T2CL (8. regiszter) által meghatározott sebességgel a léptetőregiszterbe kerülnek.
A léptetőütemet a

CLK/2/n

képletből számíthatjuk ki, ahol

CLK = a rendszer ütemfrekvenciája
n = a 8. regiszterbe töltött érték

Ez a léptetőütem a CB1-en is rendelkezésünkre áll, az adatforrás esetleges szinkronizálásához.
8 impulzus után a léptetőfolyamat megszakad és csak az SR kiolvasása után folytatódik. Ezenkívül az IFR (13. regiszter) 2. bitje bekapcsolódik.

ACR 2-4 = 010:
A léptetőütemet a rendszerütem állítja elő. Nagysága: CLK/2.
A továbbiakban megegyezik az előző esettel.

ACR 2-4 = 110:
Ebben az üzemmódban a léptetőütemet kívülről kell a CB1-re adni. 8 ütem után az IFR 2. bitje bekapcsolódik, de a léptetőfolyamat nem szakad meg.

ACR 2-4 = 001:
A léptetőregiszter tartalma a rendszerütem és a 8. regiszter által meghatározott sebességgel a CB2-re kerül. Ez a folyamat ciklikus, azaz a kiléptetett bitek folyamatosan visszakerülnek a regiszterbe. A léptetőütem a CB1-ben ismét rendelkezésünkre áll.

ACR 2-4 = 101:
A léptetőfolyamat 8 ütem után megszakad és csak a léptetőregiszter töltése után folytatódik. Ezenkívül az IFR 2. bitje bekapcsolódik. A továbbiakban ugyanaz érvényes, mint az előzőekben.

ACR 2-4 = 011:
A léptetőütem nagysága: CLK/2, egyébként ugyanaz, mint előzőleg.

ACR 2-4 = 111:
A léptetőimpulzust kívülről kell a CB1-re adni. 8 ütem után bekapcsolódik az IFR 2. bitje, de a léptetőfolyamat nem szakad meg.

Figyelem! Minden üzemmódra érvényes, hogy 8 ütem után akár kívülről, akár belülről származik, az IFR 2. bitje bekapcsolódik (1-re vált). Törölni, üzemmódtól függően, az SR írásával vagy olvasásával lehet.

11. regiszter ACR (segéd-vezérlőregiszter)

Itt csak azoknak a biteknek a szerepét ismertetjük, amelyek az időzítők és a léptetőregiszter kapcsán nem kerültek szóba.

0. bit = 0: Az ORA-ból (1. regiszter) kiolvasott adatok a PA0-7 vezetékek pillanatnyi állapotát adják.

= 1: Az IFR 1. bitjének CA1 által történő beállításakor a PA0-7 vezetékek állapotát az ORA mindaddig tárolja, amíg nem töröljük az IFR 1. bitjét.

1. bit: Ugyanaz, mint a 0. bit, de az ORB-re, a PB0-7 vezetékekre és az IFR 4. bitjére vonatkoztatva.

2-4. bit: A léptetőregiszter (10. regiszter) vezérlése

5. bit: A 2-es időzítő (8/9 regiszter) vezérlése

6/7 bit: Az 1-es időzítő (4/5 regiszter) vezérlése.

12. regiszter PCR (port vezérlőregiszter)

0. bit = 0: A CA1 megjelenő lefutó él beállítja az IFR 1. bitjét.
= 1: Az IFR 1. bitjét a CA1-en megjelenő felfutó él állítja be.

1-3 bit = 000: A CA2-n megjelenő lefutó él beállítja az IFR 0. bitjét. Visszaállítása az ORA írásával vagy olvasásával ill. közvetlenül történhet.

= 100: Ugyanaz, mint az előző esetben, de az IFR 0. bitje csak közvetlenül törölhető.

= 010: Ugyanaz, mint a bitek = 000 állapotánál, de felfutó élre vonatkoztatva.

= 110: Ugyanaz, mint a bitek = 100 állapotánál, de felfutó élre vonatkoztatva.

= 001: Az ORA írásakor és olvasásakor a CA2 alacsony szintre kerül. A CA1-en megjelenő aktív él hatására lesz ismét magas szintű.

= 101: Az ORA írása vagy olvasása után a CA2 egy rendszerütemnyi ideig alacsony szintű lesz.

= 011: CA2 alacsony szintű

= 111: CA2 magas szintű

4. bit = 0: A CB2-n megjelenő lefutó él beállítja az IFR 4. bitjét.
= 1: Ugyanez, de felfutó él esetén.

5-7 bit = 000: A CB2-n megjelenő lefutó él beállítja az IFR 3. bitjét, amit az ORB olvasásával, írásával vagy közvetlen módon állíthatunk vissza.

= 100: Ugyanaz, mint az előző esetben, de az IFR 3. bitje csak közvetlenül törölhető.

= 010: Ugyanaz, mint a bitek = 000 állapotánál, de felfutó élre vonatkoztatva.

= 110: Ugyanaz, mint a bitek = 100 állapotánál, de felfutó élre vonatkoztatva.

= 001: ORB írásakor a CB2 alacsony szintű lesz. A CB1-en megjelenő aktív él hatására ismét magas szintre kerül.

= 101: ORB írása után a CB2 egy rendszerütemnyi ideig alacsony szintű lesz.

= 011: CB2 alacsony szintű

= 111: CB2 magas szintű

13. regiszter IFR (megszakításjelző regiszter)

A regiszter minden bitje valamilyen esemény bekövetkezését jelzi. Ezeket az előzőekben már ismertettük, ezért itt csak megemlítjük őket:

0. bit = 1: aktív él a CA2-n. Visszaállítása a 1. regiszter írásával vagy olvasásával történik.

1. bit = 1: aktív él a CA1-en. Visszaállítása az 1. regiszter írásával vagy olvasásával történik.

2. bit = 1: minden 8 léptetőütem után. Visszaállítása a léptetőregiszter (10. regiszter) írásával vagy olvasásával történik.

3. bit = 1: aktív él a CB2-n. Visszaállítása a 0. regiszter írásával vagy olvasásával.

4. bit = 1: aktív él a CB1-en. Visszaállítása a 0. regiszter írásával vagy olvasásával.

5. bit = 1: a 2-es időzítő nulla-átmenete. Visszaállítása a T2CL (8. regiszter) olvasásával vagy a T2CH (9. regiszter) írásával történik.

6. bit = 1: az 1-es időzítő nulla-átmenete. Visszaállítása a T1CL (4. regiszter) olvasásával vagy a T1CH (5. regiszter) írásával történik.

7. bit = 1: ha legalább egy bit értéke ebben a regiszterben is és a 14. regiszterben is egyidejűleg 1. Ez a bit adja meg az IRQ-vezeték állapotát.

Figyelem! A 0-6. bitek közvetlenül is törölhetők (visszaállíthatók) úgy, hogy a regiszterbe beírjuk azt a byte-ot, amelyben a törlendő bitek értéke 0.

14. regiszter IER (megszakítás engedélyező regiszter)

A bitek rendeltetésre ugyanaz, mint a 13. regiszterben. Amennyiben valamelyik bit értéke mindkét regiszterben 1, megszakítás jön létre.
A bitek beállítása úgy történik, hogy a regiszterbe beírjuk azt a byte-ot, amelyben a megfelelő bitek értéke, valamint a 7. bit = 1. A 0 értékű bitek a regiszter megfelelő bitjeit nem módosítják.

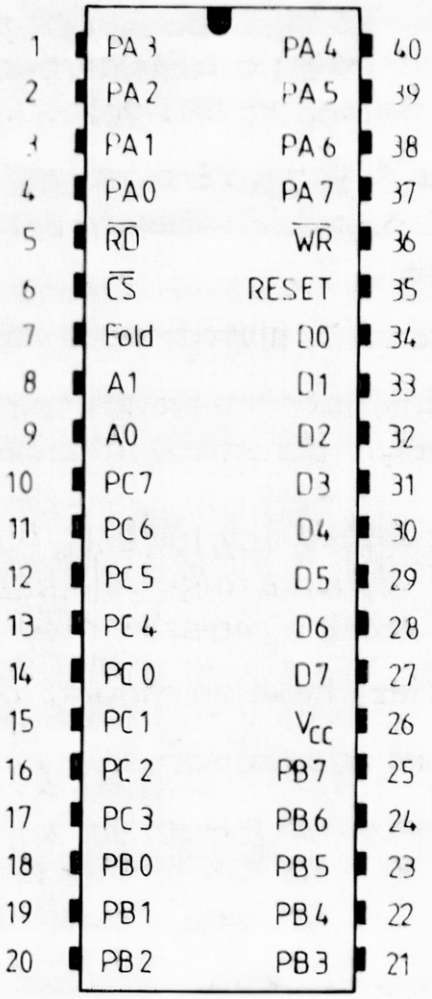
A bitek törlése hasonló módon történik, de a 7. bit = 0.

15 regiszter ORA (A port adatregisztere)

Az 1. regiszterhez hasonlóan a PA0-7 adatvezetékek állapotát tükrözi, de PCR és ICR bitekre nincs hatása.

4.3 A 8255 típusú IC – a párhuzamos csatlakozások építőeleme

Egy olyan rendszerben, ahol sok I/O-vezetékre van szükség, előnyösen alkalmazható a 8255 típusú IC. Ez tulajdonképpen a 8080 és Z80 mikroprocesszor-család periféria IC-je. Nem okoz gondot azonban a 64-eshez való csatlakoztatása sem.
Alkalmazásával 3 db 8 bites port áll rendelkezésünkre.
Mivel párhuzamos csatlakozási helyként szolgál, nincs benne sem időzítő, sem pedig léptető regiszter. Az adatáramlás iránya nem programozható bitenként, ezért az IC ehhez csupán 4 címet igényel. Ezek a címek gondoskodnak az A0 és A1 érintkezőkön a regiszterek kiválasztásáról.



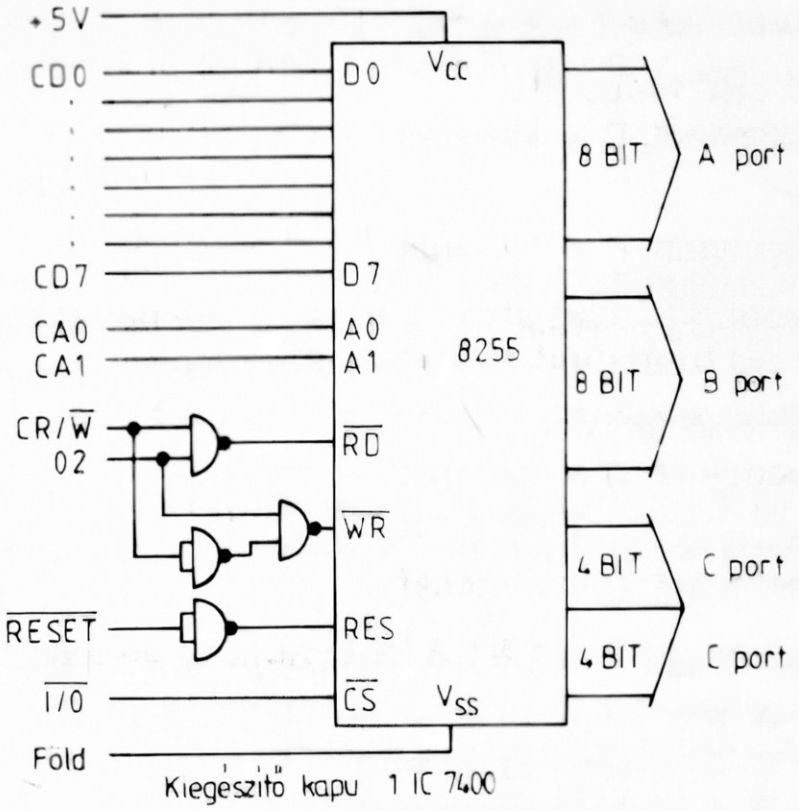
A $\overline{CS}$ csatlakozóval a már ismert módon végezhető az IC aktivizálása.
Ha a bemenet alacsony szintű, adatátvitel jön létre a processzor és a 8255-ös között.
Mivel ez az IC a 80-as típusú processzorok családjába tartozik, elkülönített READ és WRITE bemenetekkel rendelkezik. Három kapu alkalmazásával a kívánt jelek egyszerűen a processzor R/$\overline{W}$ jeléből állíthatók elő.

A RESET-bemenet feladata a szokásos. Magas szint esetén minden regisztert töröl és a portbiteket bemenetre állítja. Ügyelnünk kell arra, hogy a C 64-es által keltett RESET-jel fordított polaritású.

A processzor és az IC közti adatáramlás a D0-D7 adatvezetékeken (adatbusz) zajlik.

A maradék 24 vezeték az A, B és C porthoz tartozik.

Egy 8255-ös IC-vel felépített kapcsolás a következőképpen néz ki:



Az A port tartalmaz egy kiviteli közbenső tárolót és puffert, valamint egy beviteli közbenső tárolót. A B port egy be-/kiviteli és egy kiviteli közbenső tárolót, a C kapu pedig egy kiviteli közbenső tárolót és egy beviteli puffert. Ezenkívül a C port két négybites portként programozható és az A/B portok handshake (kézfogás) vezetékeként használható. Az A porthoz a felső fél byte (nibble), a B porthoz az alsó fél byte tartozik, emiatt A és B csoportnak is nevezzük őket.

Az IC alapvetően három üzemmódban működhet:

0-ás üzemmód: Ez az üzemmód a portbiteken történő egyszerű be-/kivitelt jelenti. Legjobban a 6526-os portbitjeinek működéséhez hasonlítható. A portbiteken keresztül kapcsolók állapotát kérdezhetjük le, vagy jeleket adhatunk ki.

1-es üzemmód: Ez az üzemmód kényelmes megszakításvezérlést jelent.

2-es üzemmód: Ez egy kétirányú be-/kiviteli üzemmód az A port és az adatbusz között.

Kiegészítésként a kimenetre programozott C port bitjei beállíthatók és törölhetők.

Ha az összes lehetőséget számba vesszük, önkéntelenül is adódik a kérdés, hogy miként lehet ennyi működési variációt egy vezérlőregiszterrel programozni.

A vezérlőregiszter minden bitjének külön feladata van. Meghatározható szerepet játszik a 7. bit.

A 0-6 bitek jelentése, ha a 7. bit = 1:

0. bit: C port alsó fél byte
1 = bemenet, 0 = kimenet

1. bit: B port
1 = bemenet, 0 = kimenet

2. bit: a b csoport (C port alsó fél byte-ja) üzemmódjának kiválasztása.
1 = 1-es üzemmód, 0 = 0-ás üzemmód

3. bit: C port felső fél byte
1 = bemenet, 0 = kimenet

4. bit: A port
1 = bemenet, 0 = kimenet

5/6. bit: az A csoport (C port felső fél byte-ja) üzemmódjának kiválasztása

bit	**6**	**5**	
	0	0	= 0-ás üzemmód
	0	1	= 1-es üzemmód
	1		= 2-es üzemmód

A 0-6 bitek jelentése, ha a 7. bit = 0:

0. bit: 0 = bitek visszaállítása, 1 = bitek beállítása

1-3 bitek: A bitek kiválasztása (binárisan)

bit	**3**	**2**	**1**	
	0	0	0	= 0. portbit
	0	0	1	= 1. portbit
	.		.	
	.		.	
	.		.	
	1	1	1	= 7. portbit

4–6. bit: nincs jelentésük.

Az 1-es és 2-es üzemmódot csak ritkán használjuk, ezért itt csak a 0-ás üzemmóddal foglalkozunk.

A port = kimenet, B port = bemenet, C port = kimenet
A vezérlőbyte: % $10000010_{bin} = 130_{dec}$

A port = kimenet
B port = kimenet
C port felső fél byte = kimenet
C port alsó fél byte = bemenet

A vezérlőbyte: % $1001\ 1000_{bin} = 152_{dec}$

Még egy fontos dolgot meg kell említenünk. Ha a vezérlőbyte-ot újra írjuk, a kimenetként tárolt adatok törlődnek. A vezérlőregiszter csak írható.

4.4 Egyéb IC-k

Az eddig ismertetett IC-k elsősorban a C 64-es alkalmazói számára lehetnek érdekesek. A kereskedelemben azonban még nagyon sokféle, érdekes teljesítményjellemzőkkel rendelkező IC kapható.
A párhuzamos csatlakozási lehetőségekkel a 64-eshez csatlakoztatható IC-k skálája is bővült.
Ilyenek a 6520, a Z80 PIO (párhuzamos I/O), az időzítő és számláló (mint a 8253-as), vagy a Z80 CTC, amelyek a leírt módszerekkel problémamentesen csatlakoztathatók.

Bármilyen kapcsolást akarunk is megépíteni, érdemes tanulmányozni a különböző félvezető gyártó cégek katalógusait. Ezek az IC-k minden adatáról tájékoztatást adnak, így könnyen kiválaszthatjuk a célnak legjobban megfelelőket.

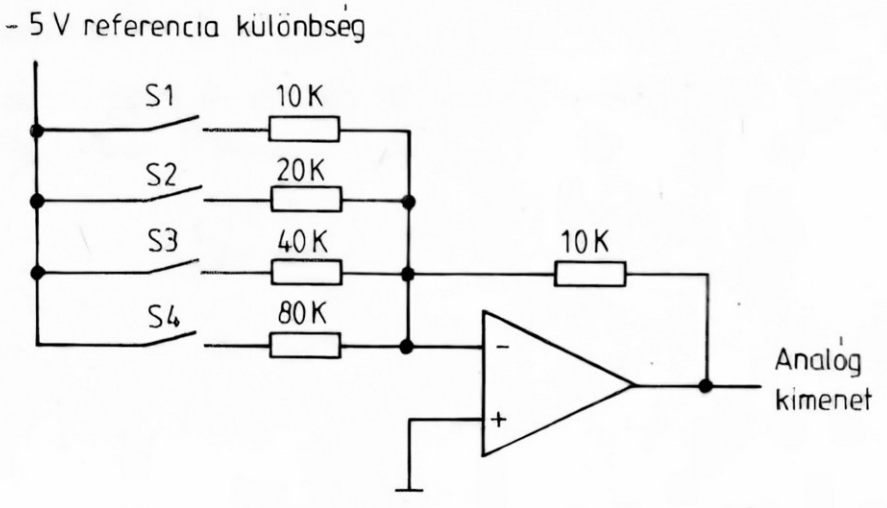
5. MI AZ A DIGITÁLIS/ANALÓG ÁTALAKÍTÁS?

Mint eddig láttuk, a számítógépünk csak azt tudja megállapítani, hogy a bemeneteken van-e 5 V körüli feszültség vagy nincs. Közbenső értékek nem lehetségesek, pedig sokszor ezekre is szükség lenne.
Gondoljunk csak a 3.8 fejezetben közölt motorvezérlő kapcsolásra. Ezzel csak a motor ki- és bekapcsolása valósítható meg, de a fordulatszám szabályozása nem. Ehhez olyan kapcsolásra van szükség, ami a számítógép által előállított bináris értéktől függő feszültségeket hoz létre. Az ilyen kapcsolás digitál/analóg átalakítóként, vagy más néven DAC (digital-analóg konverter)-ként ismert. A DAC-ok IC-k formájában kaphatók, melyek vagy kimenő feszültséget, vagy kimenő áramot hoznak létre.

A továbbiakban röviden foglalkozunk az elmélettel és ismertetjük a digitál/analóg átalakítók típusait.

5.1 A D/A-átalakítás módszerei

Nem köztudott, de a C 64-esben több DAC is található. Többek között a felhasználói port minden bitje is egy-egy DAC-ként fogható fel. Mivel azonban felbontásuk egy bitre korlátozódik, csak a 0 V-os és 5 V-os állapotok megkülönböztetésére alkalmasak. Ez nem jó, mert nekünk éppen a kettő közti feszültségértékekre van szükségünk. A kimenő feszültségszint finomabb felbontásához egynél több bitre van szükség. Így például négy bittel már $2 \uparrow 4 = 16$ különböző állapot, azaz 16 különböző feszültségszint állítható elő. Egy ilyen (elméleti) négybites átalakító egy műveleti erősítőből, öt ellenállásból, négy kapcsolóból és egy referenciafeszültségből áll.

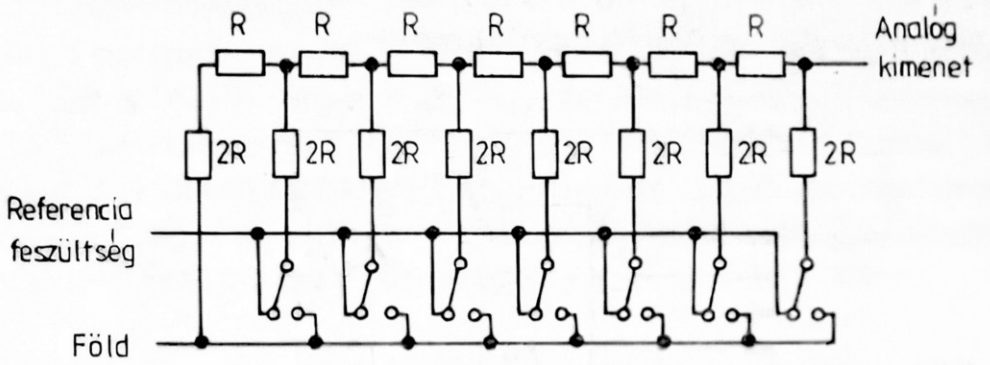


A kapcsolás lényege a következő. A műveleti erősítő összegző erősítőként működik. A kapcsolókat egy port működteti és ha minden kapcsoló nyitva van, a kimeneten 0 V mérhető. Az egyes kapcsolók zárásakor a kimeneten a következő feszültségszintek jelennek meg:

1-es kapcsoló (S1) zárva: 5 V
2-es kapcsoló (S2) zárva: 2,5 V
3-as kapcsoló (S3) zárva: 1,25 V
4-es kapcsoló (S4) zárva: 0,625 V

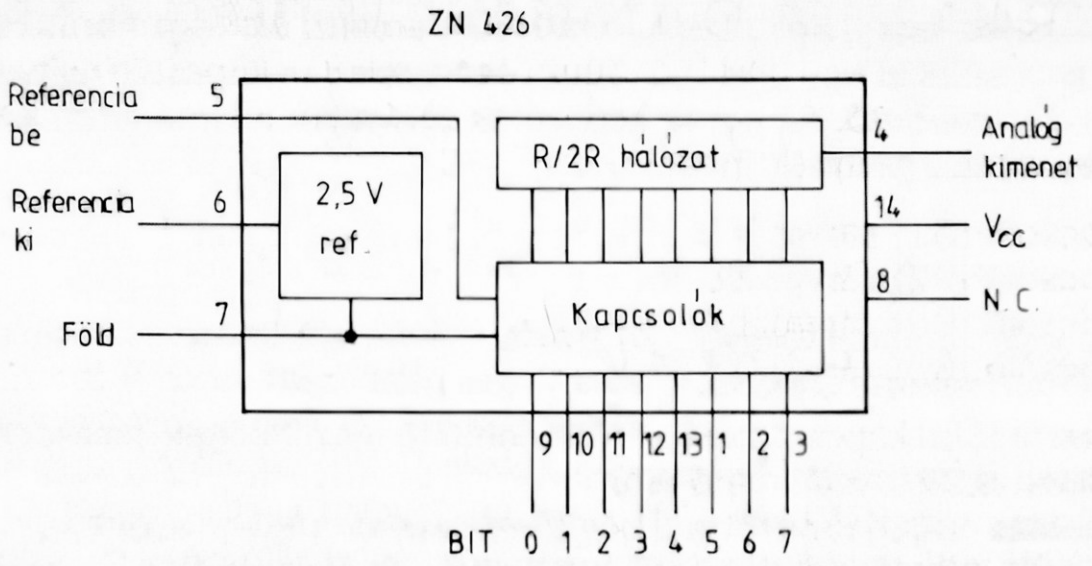
Ha egyszerre több kapcsoló van zárva, a kimenő feszültségek összeadódnak. Így maximálisan 9,375 V állítható elő.
Ez a megoldás elvileg több bitre is bővíthető, a gyakorlatban azonban ez mégsem alkalmazható. Minél nagyobb felbontást akarunk ugyanis elérni, annál nagyobb ellenállásértékekre van szükségünk és így gyorsan elérjük a gyártási folyamatban előírt határértékeket.
A megoldást az ún. R/2R elv jelenti, amely szerint a felbontástól függetlenül, mindössze két ellenállásértékre van szükségünk.

A kapcsolást a következő ábra szemlélteti:



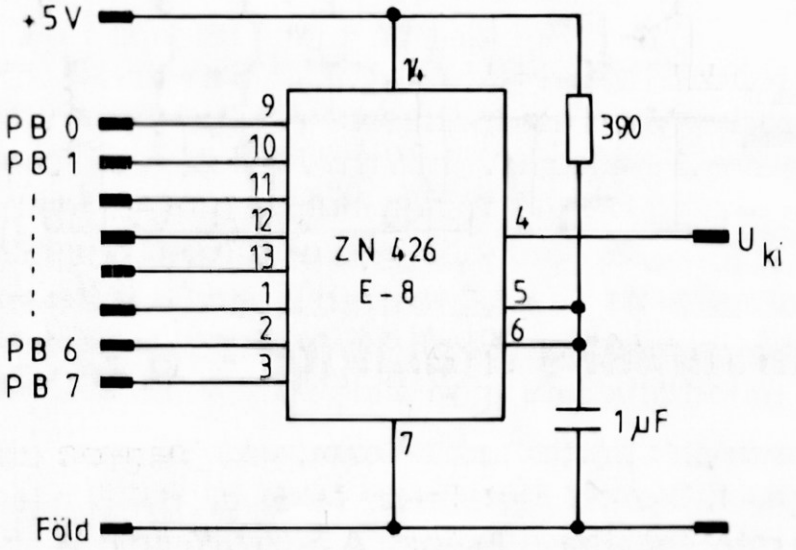
5.2 Egy univerzális átalakító – a ZN426

A ZN426 a felhasználói portra csatlakoztatható, nagyon praktikus építőelem. Minden szükséges fokozatot tartalmaz, tehát az R/2R ellenállás-hálózatot, a kapcsolókat és a referenciafeszültséget. A műszaki adatok az ábráról leolvashatók. A referenciafeszültség 2,5 V, az áramkör hőmérséklet-stabilitása nagyon jó. Az átalakítás sebessége kb. 1 µs, szintén jónak mondható. A felbontás 8 bites, ezért nagyon könnyen csatlakoztatható 8 bites számítógépekhez. Kiegészítésként csupán egy ellenállásra és egy kondenzátorra van szükségünk a referenciafeszültséghez.



A maximális kimenő feszültség a referenciafeszültség közelében van, tehát kb. 2,5 V. Mivel 8 bittel 256 fokozat valósítható meg, a kimeneten előállítható legkisebb feszültségkülönbség kb. 10 mV. Ha minden bit alacsony szinten van, a kimeneten egy 5 mV-nál kisebb (általában 3 mV) feszültség mérhető. Ebből adódik az 1/2 LSB hiba.

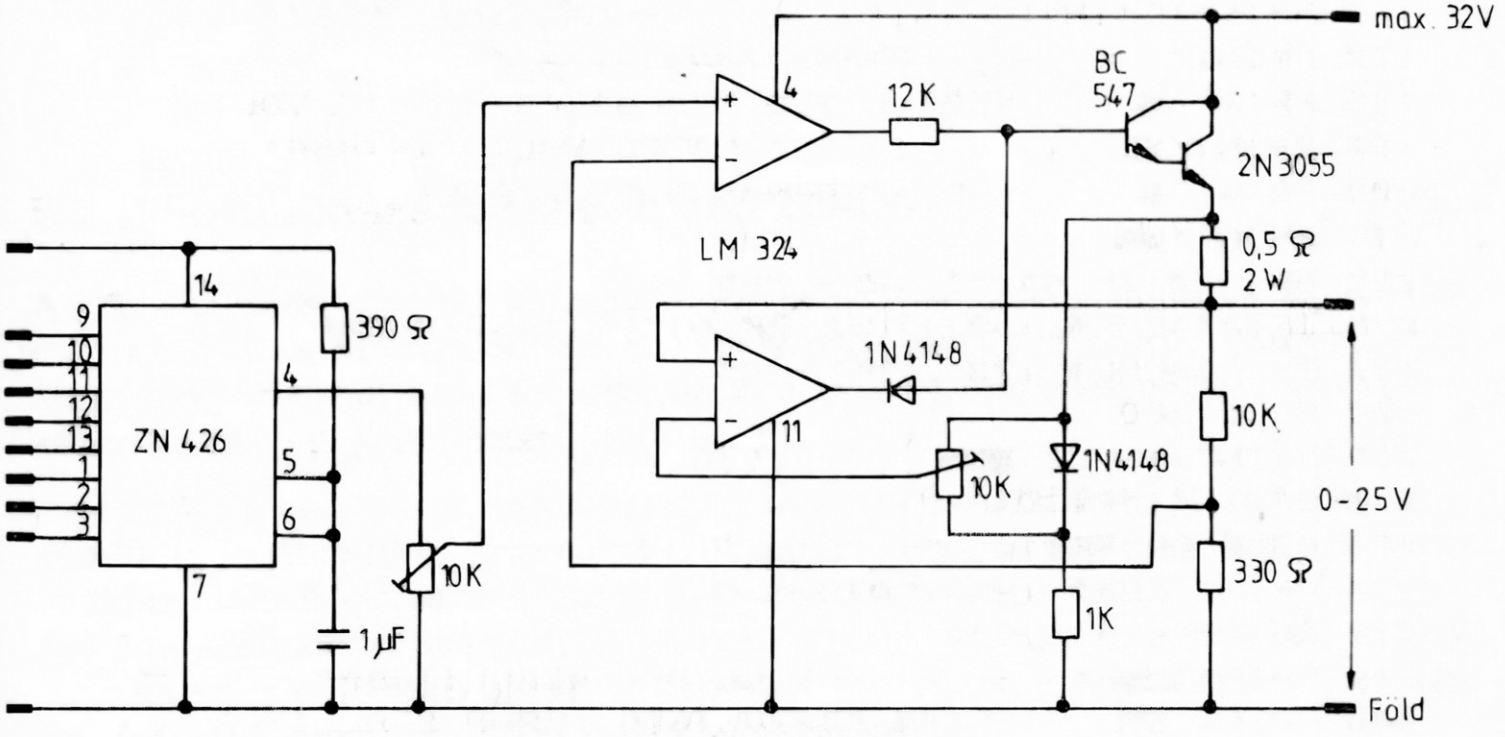
Mivel a ZN426 kimenete nagy (kb. 10 kΩ) ellenállású, a kimenő fokozatra szükség esetén egy műveleti erősítőt csatlakoztatunk.



Ennyi elmélet után térjünk rá a gyakorlati megvalósításra. Első példánkban a DAC alkalmazását egy programozható tápegység elkészítésén mutatjuk be.

5.2.1 Programozható tápegység

Az itt bemutatásra kerülő kapcsolással megközelítőleg 0 V és 25,5 V közötti kimenő feszültségeket állíthatunk elő. Ennek megfelelően az átalakító minden fokozata 100 mV kimenő feszültséget eredményez. A kimenő fokozat ezenkívül egy állítható áramkorlátozót is tartalmaz. Tervünkben a kimenő áramot trimmerpotenciométerrel állítjuk be. Ezzel tetszőleges kimenő áramot hozhatunk létre úgy, hogy a kimenetet egy ampermérővel rövidre zárjuk és ellenőrizzük a beállított határértéket.
Erre a helyre egy potmétert is beépíthetünk, amivel szabályozható áramkorlátozást valósíthatunk meg.
A DAC kimenő feszültségét a trimmeren 10 kΩ-mal adjuk meg. Mivel a DAC kimenő ellenállása is 10 kΩ, az átalakító kimeneti feszültségtartománya csupán 0...1, 25 V. Ennek a kapcsolásnak az az előnye, hogy az átalakító ofszet-feszültsége, így az elérhető legkisebb kimeneti feszültség a felére csökken. Így a tápegység legkisebb kimeneti feszültsége 0 V közelében lesz, kisebb, mintha a DAC kimeneti feszültségét közvetlenül a műveleti erősítőre adnánk.
A potméterrel egyidejűleg beállítjuk az áramforrás maximális kimenő feszültségét. Megfelelő kiegyenlítés esetén a kimenő feszültség minden fokozata pontosan 100 mV-nak felel meg.
A feszültséget a potméter csúszkájáról az első műveleti erősítő pozitív bemenetére vezetjük. Ezt a feszültséget a tápegység kimenetén összehasonlítjuk a feszültségosztó feszültségével (10 kΩ/330 Ω). A műveleti erősítő kimeneti feszültsége ekkor egy olyan értékre áll be, amelynél a tápegység kimenetén olyan feszültség jelenik meg, ami arányos az átalakító feszültségével és a műveleti erősítő bemeneteinek feszültségkülönbsége 0 V.



Ha a DAC kimenetén nő a feszültség, a műveleti erősítő kimenő feszültsége is nő. Ezáltal a tranzisztorok tovább nyitnak és nő a leadott feszültség is. A feszültségosztó visszacsatolásával a műveleti erősítő negatív bemenetére is magsabb feszültség kerül. Ez a folyamat akkor fejeződik be, amikor a két bemenő feszültség közti különbség 0 V.
A második műveleti erősítő feladata az áramellátás ellenőrzése. Ehhez a 0,5 Ω-os ellenálláson létrejövő feszültségesést mérjük. Ha nincs áram, vagy csak nagyon kicsi, a pozitív bemenet feszültsége nagyobb, mint a negatív bemeneté. Ezzel az erősítő kimenete a műveleti erősítő pozitív kimenetére csatlakozik. Nagyobb áramerősség esetén azonban a mérőellenálláson olyan nagy feszültségesés jön létre, hogy a bemeneteken megváltoznak az arányok, a kimenet alacsony szintű lesz. A dióda ekkor az alsó erősítő kimenő körében van. A diódán keresztül az első tranzisztor bázisfeszültsége csökken, így a kimenő feszültségszint is leesik.

A kapcsolás működésének ismertetése után nézzük az ehhez szükséges szoftvert. A következő programmal a billentyűzetről tetszőleges feszültségértékeket állíthatunk be. A kiegyenlítés után a billentyűzetről bevitt 10,3 V feszültségérték a kimeneten is biztosan 10,3 V-ot eredményez.

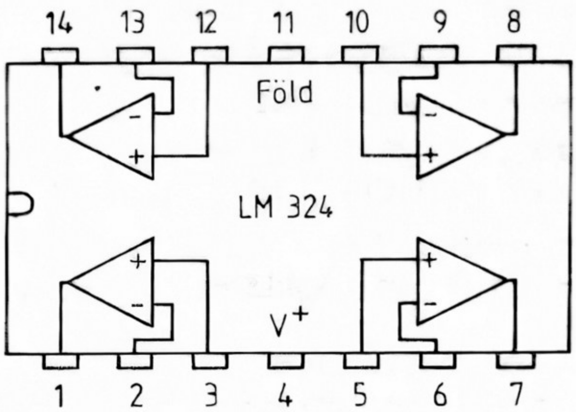
```
1 REM *** P 21 ***
2 REM
100 PRINT CHR$(147)
110 C2=56576
120 PB=C2+1
130 DR=C2+3
140 POKE DR,255
150 POKE PB,0
160 PRINT:PRINT:PRINT"            HALOZATI RESZ"
170 PRINT"            ================ "
180 PRINT"■     FESZULTSEG BEALLITASA";SP;" VOLT"
190 PRINT"■■       F1=FESZULTSEG VALTOZTATASA"
200 PRINT"■      F7=PROGRAM BEFEJEZESE"
210 PRINT"■■                ?";
220 GET A$:IF A$="" THEN 220
230 IF A$=CHR$(133) THEN PRINT" F1":GOTO 260
240 IF A$=CHR$(136) THEN 300
250 GOTO 220
260 PRINT:INPUT"■■■  UJ FESZULTSEG      ";SP
270 SP=INT(SP*10)/10
280 POKE PB,SP*10
290 PRINT CHR$(147):GOTO 160
300 PRINT CHR$(147)
310 PRINT"■■          FESZULTSEG BEALLITASA"
320 PRINT"■■  ";SP;"V FESZULTSEG MARAD BEALLITVA"
```

```
330 PRINT"     A USER-PORTON"
350 PRINT"              F I G Y E L E M ! "
360 PRINT"         A RUN/STOP-RESTORE KITORLI"
370 PRINT"        A KIMENETI FESZULTSEGET"
READY.
```

Ha a programot módosítjuk egy kicsit, ettől eltérő alkalmazási lehetőségek is adódnak. Egy FOR...NEXT ciklussal például 0-tól a maximális értékig folyamatosan növelhetjük a kimeneti feszültséget és így alkalmassá válik diódák vagy tranzisztorok jelleggörbéjének felvételére.
Pazarlásnak tűnhet az LM324-es alkalmazása, mivel a benne lévő négy műveleti erősítőből csak kettőt használunk. Ezek azonban olyan kedvező áron kaphatók, hogy nyugodtan felhasználhatjuk őket erre a célra is. Ráadásul más típusokkal, pl. az UA741-essel a kimeneti feszültség nem csökkenthető megközelítően 0 V-ra, ezért emellett még egy költséges kapcsolásra és negatív segédfeszültségre is szükség van, ami jelentősen emeli a kiadásainkat.

Feltétlenül gondoskodnunk kell a 2N3055-ös tranzisztor hűtéréséről, különben a kapcsolás nem lesz rövidzárbiztos.

A transzformátor szekunderfeszültsége nem haladhatja meg a 24 V-ot, mert tönkremennek a műveleti erősítők.



5.2.2 Fordulatszám-szabályozó ZN426-tal

A 3.8 fejezetben ismertettünk egy motorvezérlő kapcsolást, amivel az egyenáramú motor ki- és bekapcsolható. Sokkal érdekesebb és célszerűbb azonban egy olyan megoldás, amellyel a motor fordulatszámát szabályozhatjuk. A következőkben egy ilyen kapcsolást mutatunk be.

Ha a D/A-átalakítót és a motorvezérlő fokozatot (TDA2002 típ. IC-vel) már megépítettük, további hardverre nincs szükségünk. Egyszerűen csak össze kell kötnünk az átalakító kimenetét az IC 2-es érintkezőjével.
A 4.8 fejezetben már említettük, hogy az erősítő invertáló üzemmódban működik, ami azt jelenti, hogy alacsony bemeneti feszültség esetén a motor gyorsabban fut, mint egy magasabb feszültségérték esetén.

A kapcsolást a következő programmal próbálhatjuk ki. A sebességet az F1–F7 funkcióbillentyűkkel szabályozhatjuk.

```
1 REM *** P 22 ***
2 REM
100 PRINT CHR$(147)
110 C2=56576
120 PB=C2+1
130 DR=C2+3
140 POKE DR,255
150 POKE PB,240
160 POKE 650,255
170 FOR I=1 TO 40:B$=B$+" ":NEXT
180 PRINT"                MOTORFORDULAT"
190 PRINT"              ============="
205 PRINT"     F1                                    F7"
210 PRINT"LASSAN                                GYORSAN"
220 FOR I=1 TO 40:PRINT CHR$(162);:NEXT
230 W=240
240 D=(240-W)
250 POKE PB,W
260 PRINT LEFT$(B$,INT(D/6.1));"↑";
265 PRINT RIGHT$(B$,41-(INT(D/6.1)))
270 PRINT CHR$(145);CHR$(145);
280 GET A$:IF A$="" THEN 280
290 IF A$=CHR$(133) THEN 310
300 IF A$=CHR$(136) THEN 330
305 GOTO 280
310 W=W+6:IF W > 240 THEN W=240
320 GOTO 240
330 W=W-6:IF W < 0 THEN W=0
340 GOTO 240
READY.
```

A programban semmi különleges nincs, nagyon egyszerűen megérthető:
A 160-as sorban a POKE utasítással minden billentyűt ismétlőre kapcsolunk. Így csak lenyomva kell tartani a funkcióbillentyűket és a sebesség folyamatosan változik.

A kísérleti kapcsolást egy kis modellmotoron próbáltuk ki. Más (különösen nagy áramfelvételű) motoroknál előfordulhat, hogy csak viszonylag magas kimeneti feszültségnél indulnak el. Ilyenkor a TDA2002-es IC elé egy feszültségosztót kell beépíteni. Az ellenállások értékei kísérletezéssel állapíthatók meg.

Ha a motor átlagos áramfelvétele 100 mA felett van, gondoskodni kell az erősítő hűtéséről, különben működésbe léphet az IC belső védőkapcsolása, és bár semmi nem ment tönkre, a hőhatás megszűnéséig nem fog működni a kapcsolásunk.

6. ANALÓG/DIGITÁL ÁTALAKÍTÁS

A mindennapi életben analóg jeleket használunk. A szobahőmérséklet, az autónk sebessége, a rádió hangereje, a légnyomás és a világosság mind-mind analóg mennyiségek. Számítógépes tárolásuk és feldolgozásuk a számítástechnika egyik legérdekesebb problémája.

Az A/D-átalakítók alkalmazási területe rendkívül változatos, pl. fűtőberendezések szabályozása, fordulatszámmérés vagy egyéb méréstechnikai megoldások.

Az analóg mennyiségek sok közbenső értéket vehetnek fel, ezért számítógépes feldolgozásukkor mindig számítani kell a kerekítésből adódó hibákra. A számítógépben tárolt adat mindig csak megközelíti a ténylegesen mért analóg értéket, de az eltérés a különböző műszaki megoldásoknak köszönhetően megfelelően kicsi.

6.1 Az A/D-átalakítás elmélete

Hogyan alakíthatunk át egy analóg mennyiséget, pl. egy egyenfeszültség értéket, digitális jellé?
Erre a legkülönfélébb módszerek léteznek. A legelterjedtebbeket a következőkben ismertetjük, majd bemutatjuk két különböző A/D-átalakító gyakorlati megvalósítását is.

A feladatot legegyszerűbben ún. VCO-kal oldhatjuk meg. A VCO egy ún. feszültség vezérlésű oszcillátor (Voltage Controlled Oscillator), ami úgy működik, hogy a bemenetre érkező alacsony feszültség kis frekvenciát, a magasabb feszültség nagyobb frekvenciát hoz létre a kimeneten. VCO-ként használhatók pl. az ICL8038, vagy az XR2280 típusú IC-k. Ha a számítógépünk egy megfelelő programmal képes arra, hogy adott időintervallumban impulzusokat számláljon, az impulzusok számából következtethetünk az egyenfeszültség nagyságára.

Ez a módszer egyszerűnek tűnik, de sajnos komoly hátrányai is vannak. Mindenekelőtt a viszonylag hosszú mérési idő. Minél nagyobb pontosságot akarunk elérni, annál hosszabb mérési időt kell választanunk, de mindenképpen legalább 0,1 s.
Alacsony feszültségtartományban a VCO-k linearitása sem megfelelő, ezért szükség van egy bonyolult kalibrálási folyamatra is, ami további alkatrészek beépítésével és ezzel együtt költségnövekedéssel jár.

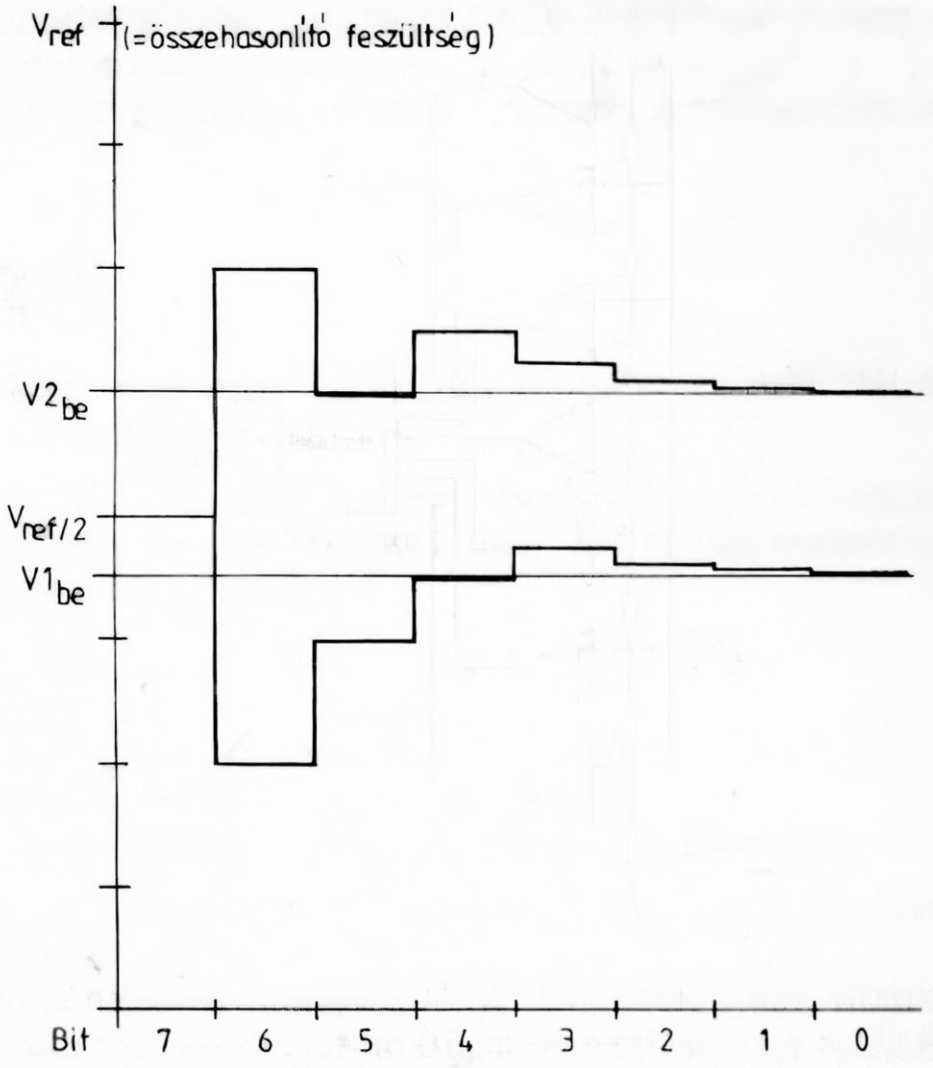
A másik ún. DUAL-SLOPE eljárás, amit a digitális voltmérőkben is alkalmaznak. Működési elve az előzőnél bonyolultabb, de ez is számláláson alapul.
Állandó árammal és a meghatározott bemenő feszültséggel feltöltünk egy kondenzátort. A töltés időtartama mindig állandó, amit egy oszcillátorral biztosítunk. Az oszcillátor frekvenciája viszonylag magas és a jeleket flip-flopokkal osztjuk le. Az állandó töltésidő miatt a kondenzátor feltöltése a bemenő feszültségek arányában változik.
A feltöltés után a kondenzátort a referenciafeszültség által létrehozott állandó áramerősséggel kisütjük. A kisütés időtartama arányos a bemenő feszültséggel. Ha ez alacsony, a kondenzátor hamar kisül, ha magas, akkor ez hosszabb időt vesz igénybe.

Az eljárás előnye: Az oszcillátor frekvenciájának csak a mérés ideje alatt kell állandónak lennie. A mérési pontosságot nem befolyásolja, ha a frekvencia hosszú idő alatt lassan változik. A mérőkondenzátorral szemben nincs különösebb követelmény.
Erre a módszerre is az a jellemző, hogy a mérési gyakoriság kicsi, vagyis a másodpercenkénti mérések száma alacsony. A kereskedelemben kapható IC-k esetében ez általában 3...100 mérés/s. Könyvünkben egy ilyen digitális feszültségmérő kapcsolást is bemutatunk majd, a hozzá tartozó programmal együtt.

Az A/D-átalakítás eddig ismertetett módszerei nem alkalmasak igazán az adatok számítógépre vitelére. A fejlesztő mérnökök a gyakorlatban sokkal jobban használható módszereket dolgoztak ki.
Ezek egyike az ún. szukcesszív approximáció. Ez egy hasonlóan gyors, pontos és egyszerű átalakítási módszer.

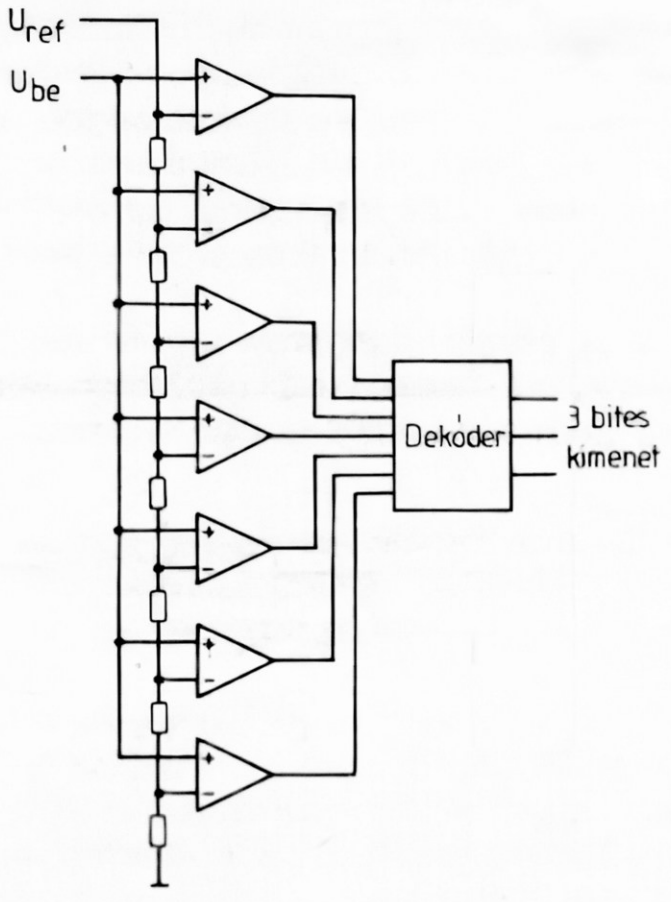
A módszer lényege a következő:

A bemenő jel egy komparátorra, azaz feszültség összehasonlítóra kerül. A komparátor másik bemenetére egy, a maximális bemeneti feszültség felének megfelelő nagyságú feszültséget kapcsolunk. A komparátor kimeneti jelének változása azt jelenti, hogy a bemenő feszültség kisebb, mint a maximális érték fele, egyébként ennél nagyobb. Ha kisebb a bementi feszültség, akkor az összehasonlítási feszültség a felére csökken, ellenkező esetben a felével nő. Ezután újabb összehasonlítás következik és az összehasonlító feszültség ismét nő vagy csökken, most már az előbbi összehasonlításkor beállított értéktől függő mértékben. A sorozatos összehasonlítások következtében az összehasonlító feszültség egyre inkább megközelíti a bementi feszültség értékét.
Az alábbi ábrával megpróbáljuk szemléletessé tenni a folyamatot. Két különböző bemeneti feszültségértéket ($V1_{be}$ és $V2_{be}$) választottunk.



Az ezen az elven működő átalakítók sebessége kb. 10 µs. Ez azt jelenti, hogy egy analóg értéket (pl. feszültség) kb. 10 µs alatt alakítják át egy 8 bites digitális értékké. Másodpercenként tehát mintegy 100000 mérés végezhető. A digitalizált érték párhuzamosan, pl. a felhasználói porton adható át a számítógépnek. A fenti elven működő monolitikus integrált átalakítók már viszonylag kedvező áron kaphatók. Ilyen pl. a ZN427.

A leggyorsabb, de egyben legdrágább eljárás az ún. párhuzamos átalakítás. Lényege az, hogy a bemenő jel komparátorok egész láncolatára kerül. A komparátorok másik bemenete feszültségosztó láncra van kötve, ez adja a referenciafeszültséget. Az ábrán látható 3 bites átalakítóhoz 7 db komparátor szükséges, de egy ilyen elven működő 8 bites átalakítóhoz már $2 \uparrow 8 - 1 = 255$ komparátor kell. Ez a magyarázata a magas áraknak.



Mivel az eredmény semmilyen számlálótól vagy más időmérő egységtől nem függ, némelyik típus sebessége eléri az 50 ns-t. Ezekkel már a tv-képek is digitalizálhatók és így tárolhatók a számítógépben. Áruk ennek megfelelően sajnos igen magas.

6.2 A/D-átalakítók a C 64-esben

A C 64-esbe beépített két A/D-átalakító működési elve eltér az eddig ismertetettektől.
A kondenzátor a csatlakoztatott potméteren (paddle) keresztül töltődik fel, 0,25 ms-onként. A kondenzátor feszültségét egy komparátor ellenőrzi. Amikor a feszültség meghaladja a SID-ben létrehozott referenciafeszültséget, a számláló leáll. Ezt a számlálót a rendszeróra vezérli.
Ha a potméter (paddle) ellenállása kicsi, a kondenzátor gyorsan feltöltődik és a számláló hamar leáll.
Minél nagyobb az ellenállás, annál tovább tart a kondenzátor feltöltése és természetesen annál magasabb lesz a számláló állása.
Ha az ellenállás túl nagy (kb. 200 kΩ felett), a feltöltési idő meghaladja a számláló maximális értékét, így az a felső határértékre (255) áll be. Nagyon kicsi ellenállás esetén ennek éppen a fordítottja történik, vagyis a számláló már a számlálás megkezdése előtt leáll, az A/D-regiszter értéke tehát 0 marad.
A mérési idő letelte után a kondenzátor az A/D-bemeneten keresztül kisül. További 0,25 ms után új mérési ciklus következik. A maximálisan elérhető mérési sűrűség 2000 mérés/s.

Sajnos az A/D-átalakítókkal nem lehet közvetlenül feszültséget mérni. Ehhez nem megfelelő a mérési pontosságuk.
Az ellenállás más kapcsolással, az átalakító nonlinearitása miatt, nem helyettesíthető.

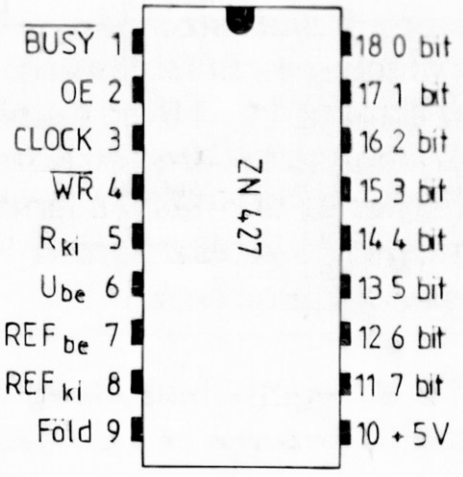
6.3 ZN 427E

Ebben a fejezetben megismerkedünk egy monolitikus A/D-átalakítóval és néhány kísérletet is elvégzünk.

A Ferranti cég ZN 427E típusú IC-je jó minőségével, kedvező árával és nem utolsó sorban könnyű beszerezhetőségével tűnik ki. A 8 bites átalakító legfontosabb műszaki jellemzői a következők:

- egyszerű működés a mikroszámítógép-rendszerekben
- teljes TTL és CMOS kompatibilitás
- szukcesszív approximáció, rövid átalakítási idő
- integrált hőmérsékletstabil referenciafeszültség
- néhány ellenállást kivéve nem igényel kiegészítő elemeket.

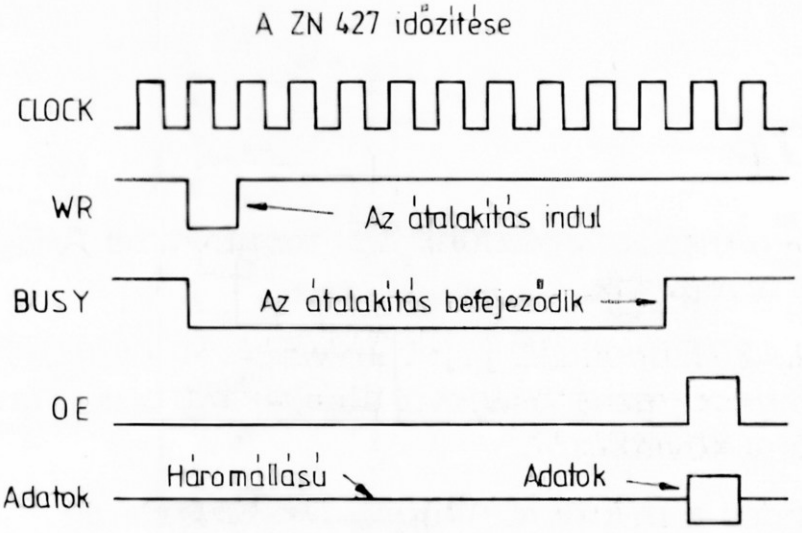
Némi hátránya, hogy negatív tápfeszültséget igényel, amit egy érdekes kapcsolással valósíthatunk meg.



A 18 pólusú IC csatlakozóinak kiosztása a következő:

A 6-os érintkező az analóg jel bemenete, a digitalizált érték pedig a 11–18-as érintkezőkről vehető le.
A szükséges referenciafeszültség a 8-as érintkezőn áll rendelkezésre. Értéke 2,56 V. Ha ezt a csatlakozást összekötjük a 7-es érintkezővel, az átalakító méréstartománya automatikusan 2,56 V lesz. Ez a legegyszerűbb módja a szükséges referenciafeszültség előállításának.
Az előállított feszültség nagyon stabil és a hőmérsékleti hatásokat szinte egyáltalán nem érzékeli. A katalógus adatai szerint hőmérsékleti együtthatója 50

ppm/°C. A ppm part per million rész per milliomodot jelent. Esetünkben a referenciafeszültség 1 °C hőmérsékletváltozás hatására 50 milliomoddal változik.



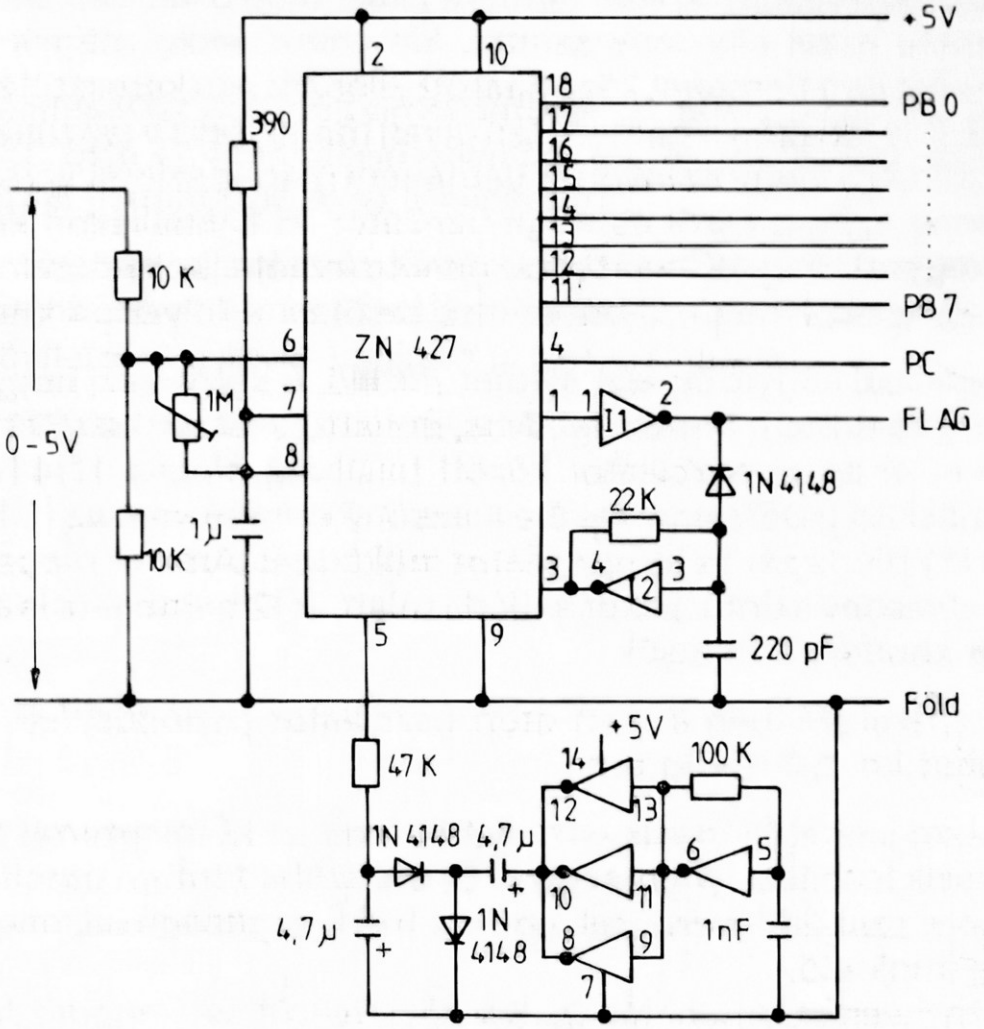
Az 1-es, 2-es, 3-as és 4-es kapcsok a mikroprocesszortól érkező vezérlőjeleket fogadják.
A 4-es (WR) érintkezőre érkező jel (negatív impulzus felfutó éle) indítja az átalakítást. Amikor a 4-es érintkező alacsony szintre kerül, azonnal alacsony lesz az 1-es (BUSY) érintkező is. Ezzel jelzi a processzornak, hogy a 11–18-as érintkezőkön pillanatnyilag nincsenek érvényes adatok. Az átalakítás 9 ütemimpulzust igényel, amik az IC 3-as (CLOCK) érintkezőjére érkeznek. Lényegtelen, hogy itt valójában 9 impulzus, vagy egy folyamatos négyszögjel jelenik-e meg.
9 impulzus után a BUSY ismét magas szintre kerül és jelzi a processzornak, hogy az átalakítás befejeződött, az érték a 11–18-as éritnkezőkről átvehető. A leolvasáshoz a 2-es érintkezőt (RD) magas szintre kell hozni. A felhasználástól és a kiegészítő kapcsolástól függően ez az érintkező állandó magas szintre is köthető, de egy dekóderrel a feszültségszint választható is lehet.
Példánkban az első megoldást alkalmazzuk.

Az 5-ös érintkezőre –3...–15 V negatív feszültség csatlakoztatható. Miért van szükség negatív feszültségre?

A ZN427 analóg bemenő fokozata két tranzisztorral differenciált bemenetként működik. A két tranzisztor emittere egymással összekötve az 5-ös érintkezőre csatlakozik.
A pozitív bemenet, mint az analóg jelek bemenete, a 6-os érintkezőre van kötve, míg az invertáló bemenet az IC-ben lévő D/A-átalakítóhoz csatlakozik.
Ahhoz, hogy 0 V-ot mérhessünk, az emittereket az alapokhoz viszonyítva legalább 0,7 V negatív potenciálra kell helyezni. Az emitterek negatív előfeszültségét a kimeneti ellenállás (R_{ki}) és a negatív tápfeszültség hozza létre.

De elég az elméletből! Melegítsük fel a forrasztópákát és próbáljuk ki mindezt a gyakorlatban is!

Az ábrán látható kapcsolás nagyon egyszerű, mégis kiváló teljesítménnyel rendelkezik. Azokban az esetekben, amikor az analóg mennyiségeket C 64-essel akarjuk feldolgozni, és a 8 bites pontosság elegendő, ez a kapcsolás felülmúlhatatlan. Egy teljes átalakítót építettünk fel (mindössze 2 db IC-vel), ami tisztán hardver úton működik. Üzemeltetéséhez nincs szükség speciális szoftverre. Az adatok egyetlen PEEK-utasítással beolvashatók a számítógépbe.



A két felhasznált IC egy ZN427-es és egy 4584-es. Az utóbbi 6 db Schmitt-triggeres CMOS invertert tartalmaz.
A TTL IC-ktől eltérően a CMOS IC-k jelölésében a betűknek nincs jelentésük. Emiatt a kapcsolásban a HEF4584 típus ugyanúgy használható, mint az MC4584.
Más betűkombinációk is léteznek, a gyártó cégek egyéni elképzelésétől függően.
Nézzük magát a kapcsolást:

A számítógéphez a felhasználói porton keresztül illesztjük mégpedig úgy, hogy az A/D-átalakító 11–18-as érintkezői (digitális kimenet) a PB0-PB7 portvezetékekre csatlakoznak. Mivel az átalakító 2-es érintkezőjét állandó +5 V-ra kötöttük, az átalakító adatai mindig a CIA2 B portjára érkeznek.
Amikor ezt a portot olvassuk, a PC csatlakozón automatikusan egy kb. 1 ms-os

negatív impulzus keletkezik. Mivel a PC az átalakító 4-es érintkezőjével van összekötve, a B port olvasásakor rögtön új mérési ciklust indít.
Mint már említettük, a $\overline{WR}$-impulzus után az átalakító BUSY-csatlakozója alacsony szintre kerül. Ez a jel az I1 inverteren keresztül a FLAG-bemenetre kerül, valamint elindítja az I2 inverterrel felépített oszcillátort.

Az oszcillátor (a kapcsolásunkban 2 db van belőle) a következő elven működik:

Bekapcsolás után a test (0 V) és a bemenet között lévő kondenzátor kisül. Az inverter bemenete ezzel alacsony szintre, kimenete pedig magas szintre kerül. Ezután a kimenet és a bemenet közé iktatott ellenálláson keresztül a kondenzátor töltődni kezd. Egy idő után a bemeneten olyan magas lesz a feszültségszint (kondenzátorfeszültség), hogy az inverter bemenete magas szintet érzékel. Kimenete ezáltal alacsony szintre kerül és a kondenzátor az ellenálláson keresztül kisül. A kisülés addig tart, míg az inverter bemenete ismét alacsony szintet nem érzékel. Ekkor a kimenet ismét magas szintre kerül és a folyamat kezdődik elölről.

A leírt kapcsolással négyszögjelet állíthatunk elő. A kérdés az, hogyan hozhatjuk létre a szükséges 9 ütem impulzust. Nos, emiatt építettük be a D1 diódát. A rajzon az I1 inverter és az oszcillátor között található, típusa 1N4148.
Amikor az átalakító BUSY érintkezője alacsony szinten van, az I1 kimenete magas szintű, a D1 dióda zár és az oszcillátor működhet. Amikor magas szinten van, I1 kimenete alacsony szintű. Ekkor a dióda miatt az I2 bemenete is alacsony szinten lesz és a kondenzátor kisül.

A megadott méretezésben a start-stop-oszcillátor periódusideje kb. 5 ms. Az átalakítás tehát kb. 50 ms-ig tart.

A kapcsolásban szereplő másik oszcillátor, amit az I3 inverterrel építettünk fel, nem rendelkezik leállítási lehetőséggel. Ez ún. szabadonfutó oszcillátorként működik. A három szabad inverterrel, egy kis trükk segítségével, negatív segédfeszültséget állítunk elő.
Amíg a három inverter kimenetén 0...5 V közti feszültség ingadozik, a kondenzátor után +2,5...–2,5 V közti tényleges váltófeszültséget mérünk. Ezt a D2 és D3 diódákkal felépített feszültségkétszerezőn keresztül egyenirányítjuk. Így kb. 3,5...4 V-os negatív egyenfeszültséget kapunk.

A kapcsolás részletes ismertetése meghaladná könyvünk kereteit. Akit mindez bővebben is érdekel, tanulmányozza át az alapvető félvezető kapcsolásokkal foglalkozó szakirodalmat.

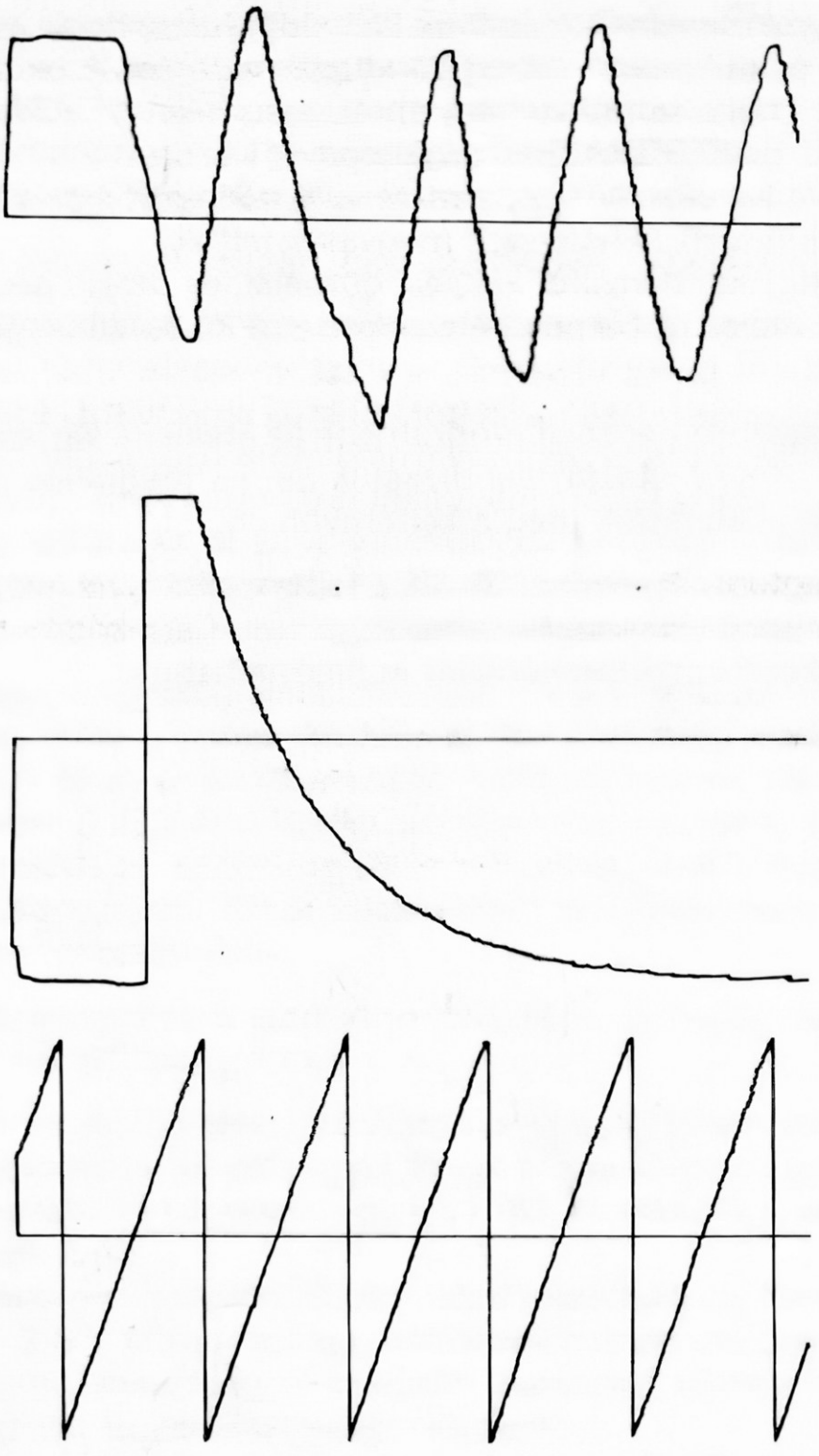
A hardver leírásból egy érdekes és megjegyzendő következtetést vonhatunk le. Azt mondtuk, hogy a B port kiolvasásakor rögtön új átalakítás kezdődik. Ez azt jelenti, hogy az éppen olvasott érték nem a pillanatnyi, hanem a B porton az előző PEEK utasítás pillanatában meglévő állapotnak felel meg. A legelső kiolvasás adata éppen ezért nem használható. A pillanatnyi tényleges érték kiolvasásához tehát két PEEK utasításra van szükség. Az első értékét nem használjuk és a második adja az előző PEEK utasítás pillanatában érvényes értéket.

A gyakorlatban mindenkinek magának kell eldöntenie, hogy mikor használhat minden értéket és mikor van szükség a dupla mérés elvének alkalmazására.
Fejezetünk végén bemutatunk néhány ábrát, amelyeket a DATA BECKER kiadásában megjelent SUPERGRAPHIK segítségével készítettünk.
Fentről lefelé haladva először egy potméter ellenállásgörbéjét láthatjuk. Az ellenállást a mérési idő alatt néhányszor megváltoztattuk.
A második rajz egy kondenzátor kisülési görbéjét, az utolsó pedig egy oszcillográfnak az A/D-átalakító bemenetére adott eltérítő feszültségének változását ábrázolja.

A grafika előállításához használt programban az átalakító kimenetén megjelenő értékeket PEEK 56577 utasítással olvastuk be, és megfelelő átszámítás után PLOT utasítással jelenítettük meg a képernyőn.

Sok mindent megtudtunk erről az IC-ről, a felhasználásával megépített kapcsolásról és a különböző alkalmazási lehetőségekről. Gépi kódú programmal akár tárolóval rendelkező oszcilloszkópként is használhatjuk.

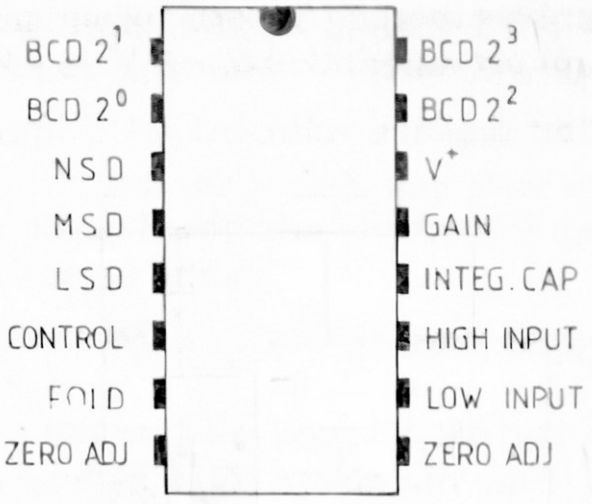
Lássunk munkához, mert még sok feladatunk van.



6.4 A CA 3162

Az RCA cég CA3162 típusú IC-je minden szükséges fokozattal rendelkezik, ami egy digitális voltmérőhöz kell. Működése a már említett dual-slope eljáráson alapszik.

Ezt az IC-t az ún. DVM-voltmérőkhöz fejlesztették ki. Mérési tartománya –99 mV ...+999 mV és mindössze 5 V tápfeszültséget igényel. Kiválasztásánál ez volt a döntő szempont, mert a többi DVM IC-knél negatív tápfeszültségre is szükség van.



Üzemeltetéséhez a bementi feszültségosztón kívül szükség van két potméterre és egy kondenzátorra, továbbá egy „jó" DVM-ben egy dekóderre is, ami a hétszegmenses kijelzőket vezérli. Esetünkben ez utóbbit szoftverrel helyettesítjük.

A ZN427-estől eltérően ennél az IC-nél az adatok nem bináris formában kerülnek átvitelre. A CA3162 kimeneti fokozata egy multiplexelt BCD kimenet. Ez azt jelenti, hogy az információk időben egymás után jelennek meg a kimeneten.

Az 1-es, 2-es, 15-ös és 16-os kimeneteken először a legmagasabb helyiértékű információk jelennek meg. Ezt a 4-es érintkező alacsony szintje jelzi. Kb. 5 ms után a 4-es érintkező ismét magas szintű lesz, az 5-ös pedig alacsony. Ezzel egyidejűleg a BCD kimeneteken a magas helyiértékű információkat az alacsonyabb helyiértékűek váltják fel. Újabb 5 ms múlva a középső helyiértékű információk jelennek meg. Ezeknek az adatoknak az érvényességét a 3-as érintkező alacsony szintje jelzi.

Az adatok kijelzésére alkalmas program az előzőnél bonyolultabb lesz. Ráadásul hiányzik az a kimenet, amelyiken minden mérés vége kijelezhető lenne.

Mivel ez az IC nem kifejezetten mikroprocesszoros rendszerek számára készült, csak megfelelő programmal használható. Ezt sajnos részben gépi nyelven kellett megírnunk, de azért nem kell megijedni. Viszonylag rövid és a „csak" BASIC-ben programozóknak is részletesen elmagyarázzuk.

Ennyi előkészítés után térjünk rá a gyakorlatra.

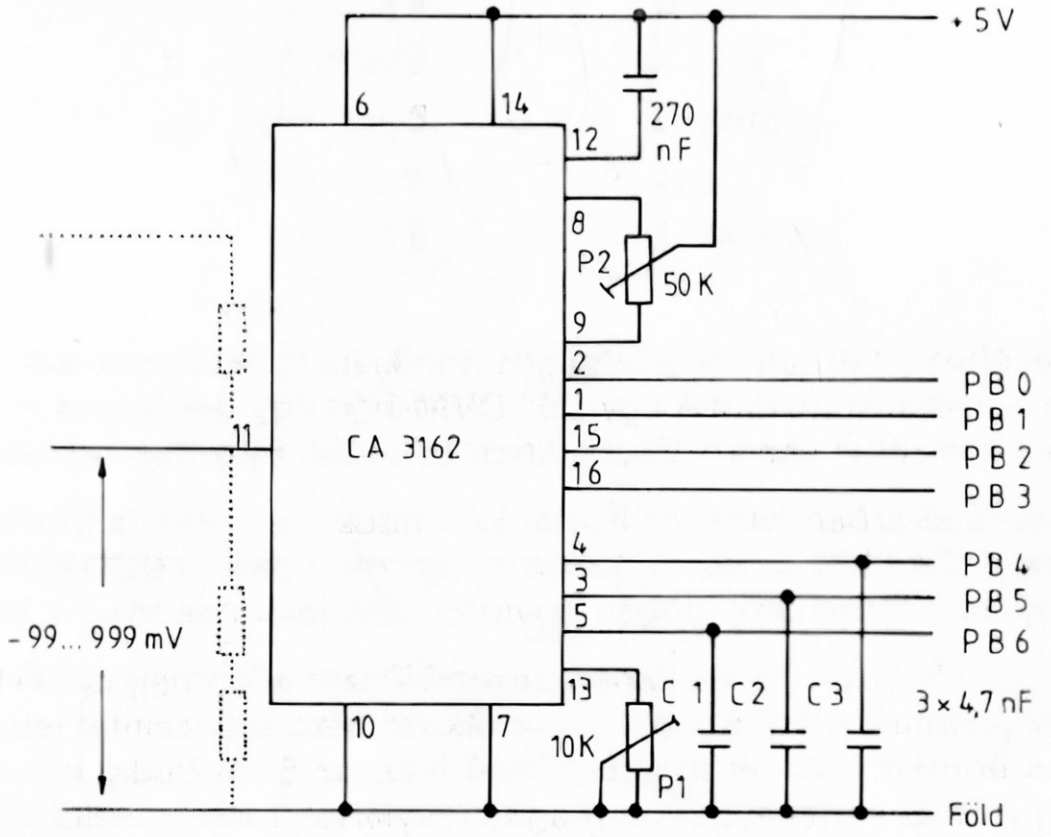
A dual-slope eljárásban szükséges integrációs kondenzátort a +5 V és a 12-es érintkező közé kell bekötni. Az egyik potmétert, amellyel rövidrezárt bemenetek mellett a mérési értéket 0 V-ra állítjuk, a 8-as és 9-es érintkezőre csatlakoztatjuk. A másikat a 12-es érintkező és a test közé kötjük. Ez a méréshatár beállítását végzi.

A mérendő feszültség a 11-es lábra kerül.

A 6-os érintkező a mérési sebességet határozza meg. Ha 0 V-on van, a mérési frekvencia 4 Hz, +5 V esetén 96 Hz. 1,2...2,8 V közti feszültség hatására a mérés

„befagy", vagyis az utolsóként mért érték kerül folyamatosan kijelzésre. A mérés gyorsasága érdekében mi ezt az érintkezőt +5 V-ra kötjük.

A CA3162-essel felépített digitális voltmérő két elemből áll.



A C1, C2 és C3 jelű kondenzátorokra akkor van szükség, ha ki akarjuk küszöbölni a multiplexelés okozta zavaró impulzusokat. Elhagyásuk esetén a kijelzés erősen váltakozó lesz.

Ha megépítettük a kapcsolást, ellenőrizzük még egyszer a helyességét. Ha minden rendben van, a kész modult csatlakoztassuk a felhasználói portra. Most kapcsoljuk be a számítógépet és töltsük be a programot. RUN után a képernyő bal felső sarkában egy háromjegyű számnak kell megjelennie. Ez a szám a DVM által mért érték. És most valami csodálatos dolog történik. Anélkül, hogy digitális voltmérőnk bemenetére feszültséget kapcsoltunk volna, az jelez valamilyen értéket. A jelenségnek nagyon egyszerű magyarázata van. Először is a voltmérőt még nem nulláztuk. Másodszor az IC bemeneti ellenállása rendkívül nagy (100 MΩ!) és emiatt itt statikai feltöltődések következhetnek be, amelyek mért értékként kerülnek kivitelre.
Első dolgunk tehát a nullázás, ami egyszerűen úgy történik, hogy az IC bemenetét összekötjük a testtel. Ezután a kijelzést a P1 potméterrel 0 V-ra állítjuk.
A másik potméter a mérési pontosságért felel. 500 mV bemenő feszültségnél a kijelzést is 500-ra állítjuk. Most már csak az a kérdés, honnan vegyünk pontos és ismert feszültséget. A legegyszerűbb az lenne, ha volna még egy digitális voltmérőnk. Ha ez nincs, akkor beszerezhetünk valamilyen speciális IC-t, pl.

LM0070-1 típusút 10 V, vagy LM336 típusút 2,5 V referenciafeszültséggel. Ezekkel és 2 db legalább 1%-os pontosságú ellenállással nagyon jó beállítást érhetünk el.

Van még egy ennél egyszerűbb megoldás, ami jóval olcsóbb is.
A zsebszámológépekben és a karórákban használt higanyelemek kb. 3% pontossággal 1,37 V feszültséget adnak.

Ezután nézzük a programot. Mint már említettük, a gépi nyelvű programozást ezúttal nem kerülhettük el.
BASIC-ben nem tudjuk a felhasználói portról kellő biztonsággal átvenni az adatokat. Ez egészen odáig fajulhat, hogy a program futási ideje miatt vagy egyáltalán nem történik mérés, vagy csak minden 30 másodpercben. Emiatt kellett az adatok beolvasására egy gépi kódú rutint írni, amit a kazetta pufferbe töltünk és SYS utasítással hívunk le. A rutin a felhasználói porton átvett értékeket három tárolórekeszbe tölti, ahonnan a BASIC-főprogram PEEK utasítással veheti át.

A program listája:

```
30 ;*** P 23 ***
35 ;
100 .OPT P8,02
110 *=$033C
120 CIA2 =$DD00
130 PB =CIA2+1
140 DR =CIA2+3
150 MERK =$FF
160 SEI
170 LDA #%00010000
180 STA MERK
190 LDX #$F9
200 LOOP LDA PB
210 TAY
220 EOR #$FF
230 AND #%11110000
240 CMP MERK
250 BNE LOOP
260 TYA
270 AND #%00001111
280 STA 0,X
290 INX
300 LDA MERK
310 ASL
320 STA MERK
330 BPL LOOP
340 CLI
350 RTS
READY.
```

A program a nulláslapon 4 tárolórekeszt vesz igénybe. Ezek címei 249...251 és 255. Ezeket a számítógép operációs rendszere csak az RS232 használatakor veszi igénybe. Mivel az LCD és az RS232 egyidejűleg úgysem csatlakoztatható a felhasználói portra, ezek a címek most szabadon rendelkezésünkre állnak.

Hogy döntjük el azt, hogy éppen melyik helyiértékű információ beolvasása történik?
Erre vonatkozóan a PB4...PB6 bitek adnak felvilágosítást. Ha a PB4 alacsony szintű, a PB0...PB3 bemeneteken a legnagyobb helyiértékű információ van. Ugyanúgy PB5 alacsony szintje esetén a középső helyiértékű és PB6 alacsony szintje esetén a legalsó helyiértékű információ olvasható.
A B port tartalma az akkumulátorba, majd az Y-regiszterbe kerül, ahol tárolódik. Végül az akkumulátor és a $FF tárolórekesz tartalma között egy EOR kapcsolatot, majd a %11110000 bináris értékkel egy AND kapcsolatot hozunk létre. A folyamat végeredményeként az akkumulátorban lévő értékben (a helyiértéknek megfelelően) a 4., 5. vagy 6. bit 1-es a többi 0. Ezután az akkumulátor értékét összehasonlítjuk a $FF (MERK) tárolórekesz tartalmával. Ha a két érték nem egyezik, a felhasználói portról újabb értéket olvas be. Ha az akkumulátor és a $FF rekesz tartalma megegyezik, az Y-regiszterben tárolt érték az akkumulátorba kerül. AND %00001111 utasítással töröljük a helyiértékre vonatkozó információkat, és megmarad a mérési érték. Végül az akkumulátor tartalmát a nulláslap adott regiszterében tároljuk. A tároló címét az X-regiszter tartalmazza.
Az ASL utasítás $FF rekesz tartalmát 1 hellyel balra tolja, így ha az eredeti érték %00010000 volt, ASL után %00100000-t kapunk. Ezzel várjuk a következő helyiérték értékét. Három jel után a bit a $FF tárolórekeszben a legmagasabb helyre kerül. Ez az az állapot, amikor a programunk visszatér a BASIC-be.

```
1 REM *** P 24 ***
2 REM
100 FOR I=828 TO 861
110 READ X:POKE I,X:S=S+X:NEXT
120 IF S<>5018 THEN PRINT"HIBA AZ ADATBAN!!":END
130 PRINT CHR$(147)
140 SYS 828
150 FOR I=0 TO2
160 GOSUB 200
170 Z$=Z$+RIGHT$(A$,1)
180 NEXT
190 PRINT CHR$(14);Z$:Z$="":GOTO 140
200 A=PEEK(249+I)
210 IF A=10 THEN A$="-":RETURN
220 IF A=11 THEN A$="↑":RETURN
230 A$=STR$(A)
240 RETURN
250 DATA 120,169, 16,133,255,162
```

```
255 DATA 249,173,  1,221,168, 73
260 DATA 255, 41,240,197,255,208
265 DATA 244,152, 41, 15,149,  0
270 DATA 232,165,255, 10,133,255
275 DATA  16,231, 88, 96
READY.
```

Ebben a BASIC-programban nincs semmi különleges.
A 249...251-es tárolórekeszek értékét egy PEEK utasítással beolvassuk és füzérré alakítjuk. Erre azért van szükség, mert máskülönben a „↑"-jelet, ami a túlcsordulást és a „←" jelet, ami az alulcsordulást jelezné, nem tudnánk megjeleníteni. A BASIC-programot szándékosan készítettük ilyen egyszerűre.

Az ismertetett digitális voltmérő is nagyon sok helyen alkalmazható, de mi egy „valódi" DVM-et is kifejlesztettünk, automatikus méréstartomány kiválasztással, maximum 100 V egyenfeszültségre. A kapcsolás és a program a 8. fejezetben található.

6.4.1 Feszültségmérés

Miután digitális voltmérőnket sikeresen csatlakoztattuk a C 64-es felhasználói portjára, megvizsgáljuk, hogy mire is használhatjuk.

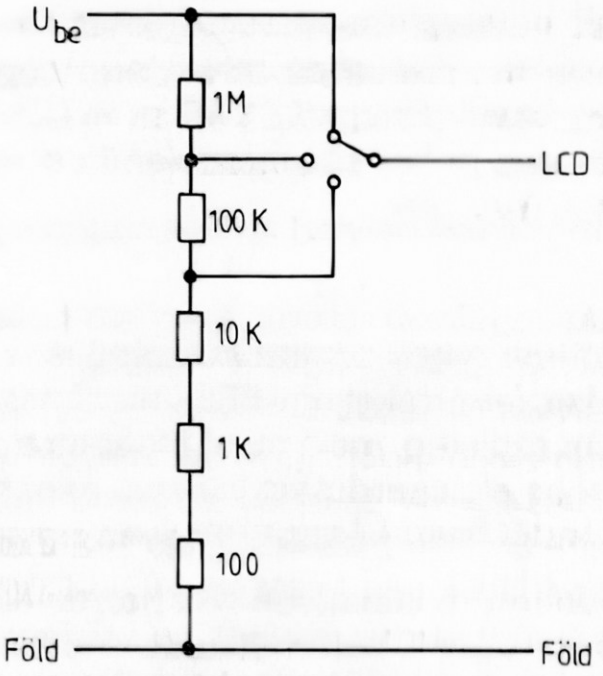
Első és legtermészetesebb alkalmazási lehetőség a feszültségmérés, de minden különösebb ráfordítás nélkül áram és hőmérséklet mérésre is használható. Megfelelő átalakítók beiktatásával ezenkívül még számos más mennyiség (nyomás, fényerősség, nedvesség) mérését megoldhatjuk vele.

Nézzük először a feszültségmérést.
Az előző fejezetben bemutatott modullal önmagában -99...999 mV közötti feszültségek mérhetők. A mérési tartomány növeléséhez feszültségosztót kell beépítenünk. A képlet, amelyből az egyszerű feszültségosztót kiszámítjuk a következő:

$$\frac{U1}{U2} = \frac{R1}{R2}$$

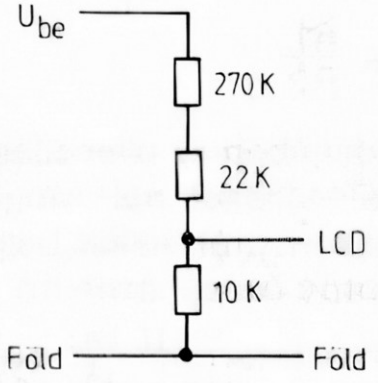
A nagy mérési pontosság érdekében az ellenállásoknak is nagy pontosságúaknak kell lenniük. Az 1%-os ellenállások már megfelelőek, de léteznek 0,1%-os mérőellenállások is. Ha sikerül az utóbbiakat beszereznünk, érdemes a mérési tartomány kiegyenlítését pontos összehasonlító feszültségmérővel elvégezni.

Az ábrán egy pontos, három méréstartományt megvalósító feszültségosztó kapcsolási rajzát mutatjuk be:



A feszültségosztó által az LCD bemenő feszültségtartománya: 1 V, 10 V és 100 V. A bemeneti ellenállás minden esetben 1,111 MΩ.
Mi a szerepe a két alsó, 1 kΩ-os és 100 Ω-os ellenállásnak?
Ha a feszültségosztót csak három ellenállásra számítjuk ki, túl nagy hibát kapunk. Egy 10 kΩ-os és egy 1 kΩ-s ellenállásból álló feszültségosztó esetén 10 V bemenő feszültségnél az 1 kΩ-os ellenálláson nem 1 V, hanem csak 0,909 V feszültségesés jön létre. Számoljunk utána! A megfelelő 1 V feszültségesést 9 és 1 kΩ-os, vagy 10 és 10,111111 kΩ-os ellenállás-kombinációval érnénk el.

Az előző ábrán bemutatott feszültségosztóval egy általánosan alkalmazható voltmérőt valósíthatunk meg. Maximum 30 V méréshatárig a kapcsolás pontossága 1 V. Mivel azonban a C 64-es kijelzi a mért értéket, egyszerű meghatározni a mérési tartományt erre a területre és az máris maximális pontossággal mérhető.
Ha a 30 V-os mérési tartomány elegendő, a feszültségosztó a következőképpen néz ki:



A kapcsolási rajznak megfelelő értékeknél 30 V bemenő feszültség esetén a 10 kΩ-os ellenálláson 1 V feszültségesés jön létre. A voltmérő által adott értéket természetesen most meg kell szorozni 30-cal. Ez a művelet beépíthető a programba.

Ennyit a feszültségmérésről. A bemutatott példák segítségével most már összeállíthatjuk saját feszültségosztónkat.

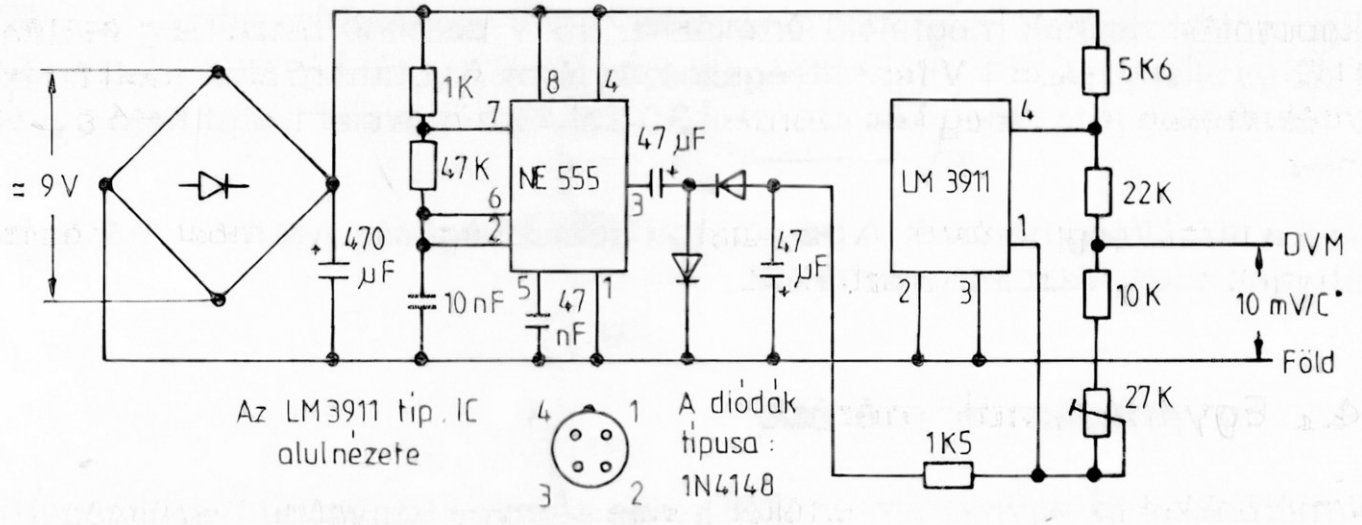
6.4.2 Egyenáramok mérése

Voltmérőnkkel az egyenáram értékét a vele arányos nagyságú feszültségérték mérésével valósíthatjuk meg. Ehhez tehát az áramerősséget feszültséggé kell alakítani. Erre a célra egy közönséges ellenállást használunk. Az ellenálláson, a rajta átfolyó áram erősségével arányos feszültségesést mérhetünk. Mekkora legyen az ellenállás értéke? Nos, ennek a kiszámítása nagyon egyszerű. Mivel a DVM legkisebb méréstartománya 1 V; meghatározott áramerősséghez éppen 1 V feszültségesésnek kell tartoznia. Ha pl. maximum 1 A áramerősséget akarunk mérni, az ellenállásnak az Ohm-törvény értelmében pontosan 1 Ω-osnak kell lennie.

Az áramerősség méréséhez az ellenállást a mérendő körben sorba kötjük. Mivel a feszültségesés mindig 1 V, az ellenállás nagyságának egyenlőnek kell lennie a mérendő áramerősség reciprok értékével (10 A esetén 0,1 Ω, 0,1 A esetén 10 Ω).

6.4.3 Hőmérsékletmérés

A hőmérséklet méréséhez már nem elegendő egy egyszerű ellenállás. Erre a célra az elektronikai iparban különböző speciális alkatrészeket fejlesztettek ki. A legismertebbek az NTC és a PTC. Ezeket ellenállásoknak is tekinthetjük, de lényegük az, hogy kifejezetten hőfokfüggők.
Így a PTC ellenállása a hőmérséklet emelkedésével nő, az NTC-é pedig csökken. Sajnos ezek az alkatrészek nem lineárisak, ezért a jó linearitás megteremtéséhez körülményes kiegyenlítésre van szükség.
A modern félvezető ipar azonban már gyárt olyan hőmérséklet-feszültség átalakítókat, amelyek csak egy hitelesítési ponttal rendelkeznek és rendkívül pontos értékeket adnak.
Számunkra megfelel az LM3911 típusú IC, amivel igazán egyszerűen építhetünk egy megfelelően pontos digitális hőmérsékletmérőt.



A kapcsolás a CA3162 típusú kiegészítő előosztó nélkül –9,9...+99 °C mérési tartományban használható.

Az IC kétféle tokozásban kapható. Egyrészt egy 8 pólusú DIL, másrészt egy 4 pólusú fémházban. Az utóbbinak az az előnye, hogy maga a hőmérsékletérzékelő a tok tetején helyezkedik el. Ezzel szinte mindenhová hozzáférhetünk, és pl. más félvezetők felületi hőmérsékletét is mérhetjük vele. A csatlakozások rövidzárlattal szemben védettek.

A kiegészítő NE555 típusú IC-t negatív tápfeszültség előállítására használjuk. Ezáltal 0 fok alatti (negatív) hőmérsékletek is mérhetők.

A kiegyenlítés a lehető legegyszerűbben történik. Két módszer is létezik:
Az első az, hogy veszünk egy tál jégkockát és hagyjuk felolvadni. Kb. 5-10 perc után a víz hőmérséklete éppen 0 °C. Az érzékelőket a vízbe merítjük (csatlakozásait előzőleg megfelelően szigetelni kell) és a kimenő feszültséget 0 V-ra állítjuk. Ez az ún. jéghidegvíz-módszer.
A másik lehetőség egy pontos referenciahőmérséklet-mérő használata. Ilyen például a minden háztartásban megtalálható lázmérő. A lázmérőket a gyártók 0,1 °C-ra hitelesítették. Az érzékelőket most kézmeleg vízbe tesszük, aminek pontos hőmérsékletét a lázmérővel állapítjuk meg. A mért értéknek megfelelően beállítjuk a kimenő feszültséget, mégpedig fokonként 10 mV-ot. Ha tehát a lázmérő 37,5 °C-ot mutat, a kimenő feszültséget 375 mV-ra állítjuk be.

7. ALAPVETŐ DOLGOK A MEGSZAKÍTÁSI TECHNIKÁHOZ

Sok számítógép felhasználó idegenkedik a megszakítási technikától. Ebben bizonyára jelentős szerepe van annak, hogy a megszakítás programozása a gépi nyelv ismeretét feltételezi. Pedig aki a megszakítási technikát megfelelően tudja alkalmazni, olyan eszközökkel rendelkezik, ami sok, egyébként bonyolult vagy éppen megoldhatatlan feladat elvégzését teszi lehetővé.

A 3.4.2 fejezetben már megismerkedtünk a megszakítás két típusával, az NMI-vel és az IRQ-val. Az ott ismertetett stopper a megszakítási technika alkalmazása nélkül egyáltalán nem lenne megvalósítható.

Mit is értünk pontosan megszakítás alatt?

Ez a kifejezés azt jelenti, hogy a programfutás során bekövetkező valamilyen esemény hatására megszakad a főprogram végrehajtása. A processzor egy megfelelő rutinnal ellenőrzi, hogy mi váltotta ki a megszakítást és kiszolgálja a megszakítást kérő egységet. Ezzel megszűnik a megszakítás forrása és a főprogram folytatódhat.
A folyamatot a következő példával szemléltetjük.
Tételezzük fel, hogy a C 64-esünk egy riasztóberendezést ellenőriz. Ha a számítógépet kizárólag erre a feladatra programozzuk be, más feladatok elvégzésére eközben nincs ideje. Emiatt ez a megoldás nem is gazdaságos. Itt jut szerephez a megszakítási technika, ami lehetővé teszi számunkra, hogy mialatt a számítógépünket valamilyen feladatra igénybe vesszük, elláthassa a riasztóberendezés felügyeletét is.
Megfelelő hardverrel megoldható, hogy a riasztóberendezés működésbe lépése megszakítást váltson ki. Ekkor az éppen futó program megszakad és a számítógép hang vagy/és optikai jelzést ad. Ezután a főprogram folytatódik.

Megszakítás más módszerekkel is kiváltható. Így pl. a megszakítás történhet az eltelt idő függvényében, meghatározott időközönként. A megszakítási rutin ekkor ellenőrizheti, hogy bekövetkezett-e valamilyen előre meghatározott esemény és az eredménytől függően folytathatja a program végrehajtását.

A megszakítási technika alkalmazásának egy másik példája, amivel mindenki nap mint nap kapcsolatba kerül, az ún. időzítő által kiváltott megszakítás (timer interrupt).
Ezalatt történik a billentyűzet lekérdezése. A CIA1 A időzítője minden 1/60 másodpercben (16 ms) megszakítást kezdeményez, majd a megszakítási rutin ellenőrzi, hogy a programfutás közben történt-e billentyűlenyomás (pl. RUN/

STOP). A megszakítási rutin gondoskodik ezenkívül a TI és TI$ rendszerváltozók léptetéséről is.

Mindezt a következő programmal ki is próbálhatjuk:

```
1 REM *** P 25 ***
2 REM
100 PRINT CHR$(147)
110 C1=56320
120 IR=C1+13
130 PRINT"GEPELJEN BE EGY SZOVEGET!"
140 TI$="000000"
150 POKE IR,127
160 FOR I=1 TO 1000
170 : GET A$:Z$=Z$+A$
180 NEXT I
190 PRINT"MOST ELEG, KOSZONOM"
200 FOR I=1 TO 1000
210 NEXT I
220 POKE IR,129
230 PRINT Z$
240 PRINT TI$
READY.
```

Ha a programot lefuttatjuk, tapasztalni fogjuk, hogy a billentyűzetről nem tudunk adatokat bevinni és a TI$ óra is végig 000000-án áll. Ismételjük meg a futtatást, de előzőleg töröljük a 150-es és 220-as sort. Így máris látható, hogy a megszakítás BASIC-ből is kikapcsolható. Mivel azonban nem tilthatjuk le közvetlenül magát a megszakítást (gépi kódban ez a SEI utasítással megvalósítható), ezért a megszakítás forrását szüntetjük meg. Ehhez törölnünk kell a CIA megszakításvezérlő regiszterét. Az A időzítő által kért megszakítás újbóli engedélyezése a 220-as sorban történik.
A 6510-es mikroprocesszor háromféle megszakítási típust különböztet meg. Ezeknek egymáshoz képest különböző elsőbbségi joguk (prioritás) van.
Mindegyik másikkal szemben elsőbbségi joga van a RESET-nek. Igen, a RESET a megszakítás egyik fajtája. Ha a 6510-es 40-es érintkezője alacsony szintre kerül, az éppen futó program megszakad. A megszakított programba nem lehet visszatérni.

A következő az ún. nem maszkolható megszakítás, az NMI. Ez szoftver útján nem tiltható le, de a megszakítás után a programba visszatérhetünk. Az NMI azonnal létrejön, amint a mikroprecesszor 4-es érintkezőjét alacsony szintre helyezzük. A C 64-esben két különböző módon is kiválthatunk NMI-t. Az egyiket nagyon gyakran használjuk. Ez a RESTORE billentyű lenyomása. Fontos tény, hogy NMI-t csak a RESTORE lenyomásával válthatunk ki. A megszakítási rutin egyúttal ellenőrzi, hogy a RUN/STOP billenytűt is lenyomtuk-e.

A másik módszer a CIA U2 megfelelő programozását jelenti. Itt minden lehetőséget felhasználhatunk, amit a CIA leírásánál ismertettünk.
A harmadik megszakítási mód az IRQ (Interrupt ReQuest). Ez a mikroprocesszor 3-as érintkezőjére adott alacsony szintű jellel váltható ki. Érdekessége az, hogy gépi kódban a SEI (SEt Interruptflag) utasítással letiltható, ami annyit jelent, hogy az utasítás hatására a processzor nem veszi figyelembe a 3-as érintkezőre adott alacsony szintű jeleket. A SEI utasítás hatása a CLI (CLear Interruptflag) utasítással oldható fel. Így a programban előírhatjuk, hogy egy adott időpontban engedélyezett-e a megszakítás vagy nem.
A C 64-esben IRQ a CIA U1 által váltható ki.
Ide tartozik az időzítő által kiválasztott megszakítás is, amiről már szó volt.

Ugyancsak válthat ki IRQ-t a videovezérlő is (VIC). Ezáltal nyílik mód pl. a sprite-ok ütközésvizsgálatára és kijelzésére, valamint az ún. rasztersor-megszakításra. Ez utóbbi lényege az, hogy amikor a képernyő felépítése során az elektronsugár egy előre megadott rasztersorba ér, megszakítás váltható ki.

A teljesség kedvéért megemlítjük még a szoftver útján történő megszakítás lehetőségét. Ez a gépi kódú BRK utasítás. Ha egy program ezt az utasítást végrehajtja, a processzor úgy viselkedik, mint egy IRQ esetén. Mivel rendszerint csak gépi nyelvű programok ellenőrzéséhez használjuk, ismertetésétől itt eltekintünk.

Az NMI és IRQ vezetékeknek a bővítő porton is van kimenete, így ezen a porton keresztül külső egységekről is válthatunk ki megszakítást.

7.1 Egyszerre két dolog működik

Miután az előző fejezetben megismerkedtünk a megszakítási technikával, bemutatunk néhány alkalmazási példát. A közölt programok segítségével remélhetőleg minden érthetőbbé válik.

Mielőtt azonban hozzáfognánk, néhány dologra még ki kell térnünk.

A megszakítások kezeléséhez a C 64-es néhány tárolórekesze mutatóként (vektorként) szolgál. Ezek a RAM-ban találhatók és a tulajdonképpeni megszakítási rutin kezdőcímét tartalmazzák. Az IRQ mutató címe $0314/$315, az NMI mutatóé pedig $0318/$0319.

Különös jelentősége van annak, hogy a megszakítási mutatók a RAM-ban vannak. Így ugyanis módosíthatjuk őket és saját megszakítási rutinokat írhatunk és használhatunk.
Ennyi elmélet után most már igazán térjünk rá a gyakorlatra. Tanulmányozzuk át alaposan a következő programot.

```
60 ; *** P 26 ***
70 ;
90 *=$C000
100 FEHER =1;ITT TUD ON
110 KEK =6 ;SZINT VALASZTANI
120 IRQREGI =$EA31;RENDSZERMEGSZAKITAS
130 IRQVECT =$0314;VEKTOR A RAMBAN
140 RASTER =$D012; RASZTERSORREG. 6569
150 IRQREG =$D019;MEGSZAKITASREGISZTER
160 MASZK =$D01A;MEGSZAKITASI MASZK
170 KERET =$D020;KERETSZIN
180 HATTER =$D021;HATTERSZIN
190 RZ1 =98;RASZTERSOR,
200 RZ2 =201;MEGSZAKITAS A
210 ;                    PROGRAMINDITASHOZ
220 SETUP SEI;KOVETKEZO MEGSZAKITAS:;          TILTASA
230 LDA #<IRQUJ;MEGSZAKITAS VEKTOR
240 STA IRQVECT;A RUTIN LETILTASAHOZ
250 LDA #>IRQUJ
260 STA IRQVECT+1
270 LDA #RZ1;ELSO RASZTER SOR
280 STA RASTER;KEZDOERTEK ADASA
290 LDA RASTER-1; SZOVEG
300 AND #$7F
310 STA RASTER-1
320 LDA #$81;RASZTER MEGSZAKITAS
330 STA MASZK;TILTVA
340 CLI ;MEGSZAKITAS ENGEDELYEZVE
350 RTS ; ES VISSZATERES
360 ;ITT KEZDODIK EGY UJ
370 ;MEGSZAKITASI RUTIN
380 IRQUJ LDA IRQREG;JOTT IRQ
390 STA IRQREG;A 6569-ROL
400 AND #1;IGEN VAGY NEM
410 BNE DOIT; HA IGEN, AKKOR DOIT
420 LDA $DC0D;CIA-IRQ TORLES
430 CLI ;ES A RENDSZER RUTINHOZ
440 JMP IRQREGI;ELAGAZAS
450 DOIT LDA RASTER;IRQ-N AT A MASODIK
460 CMP #RZ2;RASZTERSOR-E
470 BCS MASODIK;EGYEZES ESETEN A MASODIKRA
480 LDA #FEHER;KULONBEN KERET ES HATTER
490 STA KERET;FEHERRE KAPCSOL
500 STA HATTER
510 LDA #RZ2;IRQ-N AT A MASODIK RZ
520 STA RASTER;ELOKESZIT ES
```

```
530 JMP $FEBC;AZ  IRQ RUTIN ELENGED
540 MASODIK LDA #KEK;A KERET ES HATTER
550 STA KERET;KEKRE KAPCSOL ES
560 STA HATTER
570 LDA #RZ1;AZ RZ1 ELOKESZIT
580 STA RASTER
590 JMP $FEBC;IRQ RUTIN ELHAGYASA
READY.
```

A programban az ún. rasztersor-megszakítást alkalmaztuk. A képernyőt három részre osztottuk. Az alsó és felső hat sor keret- és háttérszíne kék, a középső 13 soré fehér. Ha az elméletet megértettük, ez könnyen megvalósítható.

Betöltés után a programot SYS 12 * 4096 utasítással indítjuk.
Azok számára, akik nem rendelkeznek assemblerrel, a következőkben közöljük a program BASIC-betöltő listáját is. Az aláhúzott értékek a színekhez és a rasztersorokhoz tartoznak. Ha módosítani akarjuk ezeket, ne felejtsük el, hogy a színkódok értéke 0...15, a rasztersorok értéke 0...255 között lehet, valamint változik az ellenőrző összeg (S). Az új ellenőrző összeget a program indítása után PRINT S-sel tudhatjuk meg.
A rasztersorok átviteli bittel rendelkeznek a $D011 tárolórekeszben, ezt azonban a programunk nem használja. Már az inicializáskor törlődik.

```
1 REM *** P 27 ***
2 REM
100 FOR I=49152 TO 49238
110 READX:POKE I,X:S=S+X:NEXT
120 DATA 120,169, 31,141, 20,  3
125 DATA 169,192,141, 21,  3,169
130 DATA  98,141, 18,208,173, 17
135 DATA 208, 41,127,141, 17,208
140 DATA 169,129,141, 26,208, 88
145 DATA  96,173, 25,208,141, 25
150 DATA 208, 41,  1,208,  7,173
155 DATA  13,220, 88, 76, 49,234
160 DATA 173, 18,208,201,201,176
165 DATA  16,169,  1,141, 32,208
170 DATA 141, 33,208,169,201,141
175 DATA  18,208, 76,188,254,169
180 DATA   6,141, 32,208,141, 33
185 DATA 208,169, 98,141, 18,208
190 DATA  76,188,254
200 IF S<>10594 THEN PRINT"HIBA AZ ADATBAN!!":END
210 PRINT "OK"
220 SYS 49152
READY.
```

Fenti példánk nagyon jól szemlélteti a megszakítások programozását, de mérési és ellenőrzési feladatoknál mindez egy kicsit másképpen alakul. Ha egy A/D átalakítóval feszültséget ellenőrzünk, nincs módunk a közvetlen megszakításra, mert az A/D-átalakító erre nem alkalmas. Ebben az esetben más utat kell választanunk.

Az időzítő által kiváltott megszakítást használhatjuk fel erre a célra. Mint tudjuk, az időzítő minden 16 ms-ban megszakítást kér. Ekkor ellenőrizhetjük, hogy az ellenőrizendő feszültség az engedélyezett tartományban van-e. Ha igen, máris visszatérhetünk a régi megszakítási rutinba.
Az általunk írt kis program (P27) minden megszakításkor beolvassa a felhasználói porton lévő adatokat. Ha ennek értéke 0 és 150 között van, a keret zöldre vált. 150 és 200 közötti tartományban világospiros, ennél magasabb értékek esetében pedig sötétpiros lesz.
A program a 6.3 fejezetben ismertetett A/D-átalakítóval működik.

```
30 ;*** P 28 ***
35 ;
40 *= $C000
45 ZOLD =5
50 VPIROS =10
55 PIROS =2
60 IRQREGI =$EA31
65 IRQVECT =$0314
70 CIA2 =$DD00
75 PB =CIA2+1
80 DR =CIA2+3
85 KERET =$D020
90 ;INICIALIZALAS
95 SETUP SEI
100 LDA #<IRQUJ;INTERRUPTVEKTOR A
105 STA IRQVECT;RUTINUNKRA
110 LDA #>IRQUJ;
115 STA IRQVECT+1
120 LDA #$FF;USER-PORT BITEK BEMENETRE
125 STA DR;KAPCSOLASA
130 CLI;INTERRUPT ENGEDELYEZESE
135 RTS;VISSZATERES AZ INICIALIZALASBOL
136 ;SAJAT IRQ-RUTIN
140 IRQUJ LDA PB;USER-PORT OLVASASA
145 CMP #200;OSSZEHASONLITAS
150 BCC W1;NEM ESETEN W1
155 LDX #PIROS;IGEN ESETEN X-REG.-BE 2
160 BNE WW;ES UGRAS WW-RE
165 W1 CMP #150;OSSZEHASONLITAS
170 BCC W2;NEM ESETEN W2
```

```
175 LDX #VPIROS;IGEN ESETEN VPIROS
180 BNE WW
185 W2 LDX #ZOLD;ERTEK<150,A ZOLD
190 WW STX KERET;A KERETSZIN REGISZTERBE
195 JMP IRQREGI;ES VISSZA A REGI IRQ-RA

READY.
```

Most valóban elmondhatjuk; két programunk fut „egyszerre".

Betöltés után SYS 12 * 4096 utasítással bármikor lehívhatjuk ezt a programot és beolvashatjuk vele a felhasználói porton lévő adatokat. Ettől függetlenül minden további nélkül betölthetünk és futtathatunk egy másik programot is. Csupán arra kell ügyelnünk, hogy ez a program nehogy megváltoztassa a megszakítási mutatókat, mert ez nemkívánatos dolgokat eredményezhet.

Végül közöljük a program BASIC-nyelvű betöltőjét:

```
1 REM *** P 29 ***
2 REM
100 FOR I=49152 TO 49196
110 READ X:POKE I,X:S=S+X:NEXT
120 DATA 120,169, 18,141, 20,  3
125 DATA 169,192,141, 21,  3,169
130 DATA 255,141,  3,221, 88, 96
135 DATA 173,  1,221,201,200,144
140 DATA   4,162,  2,208, 10,201
145 DATA 150,144,  4,162, 10,208
150 DATA   2,162,  5,142, 32,208
155 DATA  76, 49,234
160 IF S<>5085 THEN PRINT"HIBA AZ ADATBAN!!":END
170 PRINT"OK"
180 SYS12*4096
READY.
```

A közölt programokat csak kísérletezéshez használjuk! Ha a megszakítási technika lényegét megértettük, valóban sok érdekes probléma megoldására nyílik lehetőségünk.

8. KOMPLETT KAPCSOLÁSOK SAJÁT CÉLJAINKRA

Könyvünk utolsó fejezetéhez értünk. Szeretnénk befejezésül néhány különösen érdekes és tudomásunk szerint a C 64-eshez ebben a formában még nem közölt kapcsolást bemutatni.

A kapcsolásokhoz kissé bővebb programokat írtunk, hogy a munkát minél kényelmesebbé tegyük. Igyekeztünk BASIC-nyelven programozni, csak az egyébként nagyon időigényes részeket tettük át gépi kódba.

A hardver költségeket szándékosan alacsony szinten tartottuk (esetenként még a minőség rovására is), hogy kispénzű olvasóinknak se kelljen lemondaniuk a megvalósításról.

8.1 EPROM programozó készülék

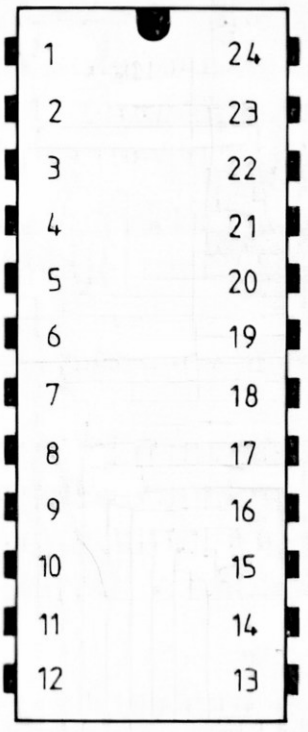
Felvetődhet a kérdés, hogy mi szükség van a C 64-es esetében a programozó készülékre.
Nos, a számítógép technika rohamos fejlődésével egyre összetettebbé válnak az alkalmazási feladatok. Rövidesen bekövetkezhet az az állapot, amikor már egy komplex problémát nem tudunk, vagy nem akarunk teljes egészében a C 64-essel megoldani. Ezekre a feladatokra az ún. egykártyás számítógépek alkalmazhatók a legcélszerűbben. Ezek a kis számítógépek kiválóan használhatók pl. vezérlési célokra. A szükséges szoftvernek azonban ebben az esetben EPROM-ban kell rendelkezésre állnia.
Assemblerrel, monitorprogrammal és egy ún. EPROM-égetővel (itt nem szó szerint égetésről van szó, de a számítástechnikában ez a kifejezés terjedt el, így mi is ezt használjuk) a C 64-es egészen elfogadható fejlesztőrendszerré válik.

Ezenkívül az EPROM-égető egyéb célokra is nagyon jól használható. Az EPROM-ban elhelyezhető pl. egy gépi nyelvű monitorprogram, megtakarítva ezzel a hosszú betöltési időt. Ezt a játékmodulokhoz hasonlóan, egyszerűen a bővítő portonhoz csatlakoztathatjuk és máris használhatjuk a programot. Az EPROM-ban különböző ellenőrző-rutinokat is elhelyezhetünk, pl. RAM-tesztet, periféria IC-tesztet, és a különböző csatlakozók ellenőrzését végző rutinokat. Így, ha a C 64-es hibát jelez, csak behelyezzük az EPROM-modult a bővítő csatlakozóba és a hibaüzenet oka máris megjelenik a képernyőn. Természetesen itt az auto-start lehetőségét is kihasználhatjuk. Erről a későbbiekben még bővebben is beszélünk.

Az általunk készítendő EPROM-égető a három leggyakrabban alkalmazott EPROM típus, a 2716-os, a 2532-es és a 2732-es programozására használható.

A kapcsolás kifejlesztése során nagy súlyt fektettünk a rendkívül egyszerű kezelésre. Kapcsolóval pl. nem rendelkezik. Minden funkció megvalósítása szoftver által történik. Az éppen használt EPROM típusát egy kis csatlakozó modul segítségével határozzuk meg. Ez egy 14 pólusú IC-aljzatból készíthető. A vezérlőprogram egy kis trükkel megállapítja, hogy éppen melyik modul van behelyezve és automatikusan a megfelelő égetési rutint hajtja végre.

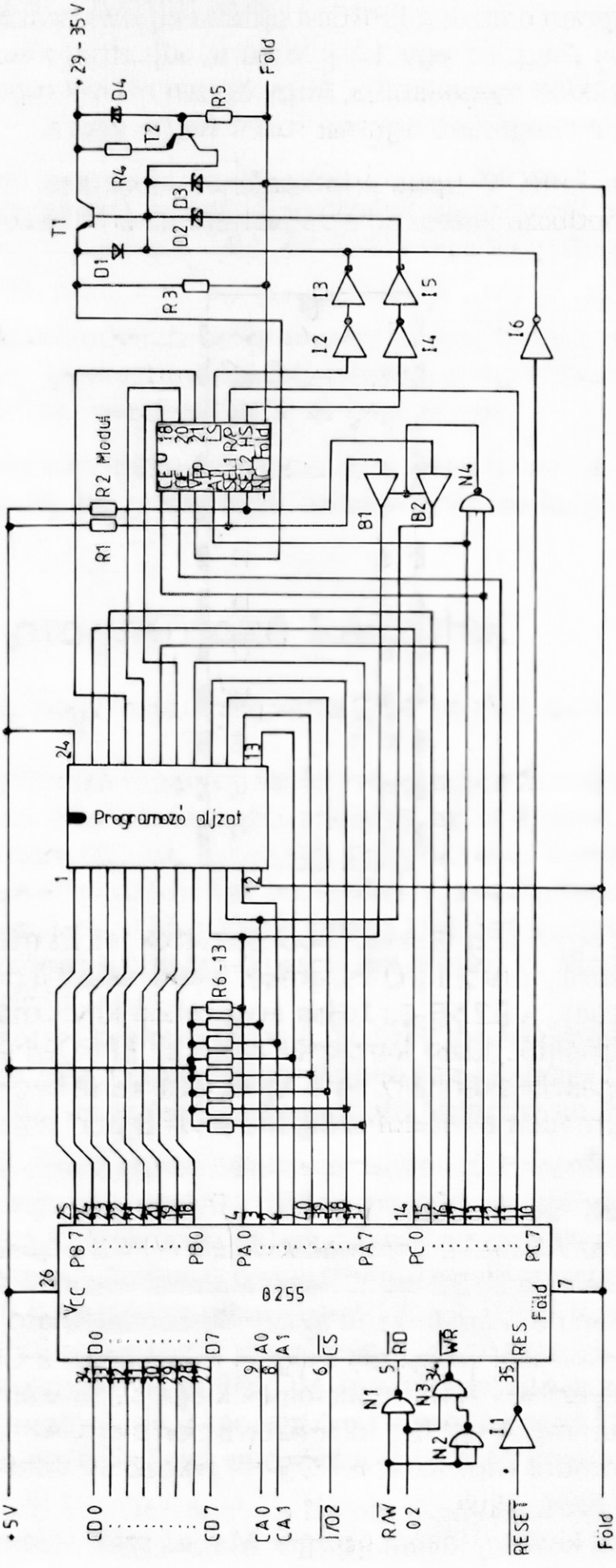
Az ábrán a három EPROM-típus érintkezőinek kiosztása közti különbségeket szemléltetjük. A modulon keresztül elvégezhető az érintkezők megfelelő átkapcsolása.



Az EPROM-égetőt egy 8255 típusú IC-ből építettük fel. Ez mint tudjuk 3 db 8 bites porttal rendelkezik, ami 24 I/O vezetéket jelent. Mivel a programozandó EPROM-ok 24 pólusúak, a 8255-ös teljes egészében kihasználható. Az A portot adatbuszként használjuk. Ezen keresztül zajlik az EPROM és a C 64-es közti adatforgalom. Kiegészítésként a 0. és 1. bitek által kerül beolvasásra az EPROM típusa. Ezt az információt a modul szolgáltatja. A B port adja az EPROM 8 alsó címbitjét.

A C port különböző feladatokat lát el.
A PC0...PC2 bitek az A8...A10 címbiteket adják. A PC3 adja az A11 címbitet, de erre csak a 2532-es és a 2732-es IC használatánál van szükség. Mivel ez a címbit a két IC esetében nem ugyanazon az érintkezőn található, a helyes csatlakozást a kiegészítő modullal valósítjuk meg. A PC4 képezi a CE jelet (Chip Enable = az IC engedélyezése). A CE funkcióinak kiegészítéseként a programozó 50 ms-onként programimpulzust hoz létre ezen a portbiten. Az IC típusától függően ezt a vonalat is a modul kapcsolja. A PC5-ről érkező OE jelet csak a 2716-os IC programozásánál használjuk.
A PC6 az üzemmód kiválasztására szolgál. Magas szint esetén az EPROM-égető

programozás üzemmódban működik, míg alacsony szint esetén az EPROM adatai olvashatók.



A PC7 kapcsolja be és ki a programozási feszültséget, ami 25 V. Ha 21 V-os EPROM-unk van, cseréljük ki a 27 V-os Z-diódát (D3) 22 V-osra. Így a T1 tranzisztor bázis-emitterfeszültsége által a programozási feszültség a kívánt 21 V körüli (valamivel több, mint 21 V) értékre csökken.

Ezzel eljutottunk az EPROM-égetőnk tényleges hardver leírásához. Kapcsolásunk 8255-ös típuson kívül még 74LS125, 7400 és 7416 típusú IC-kből, néhány diódából, tranzisztorból és ellenállásból épül fel.
A T1 és T2 tranzisztorok a programozási feszültséget stabilizálják. Ez a fokozat a kimeneten 0 V, 5 V vagy 25 V feszültséget állít elő. 5 V-ra a 2716-os és a 2532-es IC-k programozásakor van szükség. Ezeknél az EPROM-oknál az adatok olvasásakor a programozási feszültségcsatlakozót 5 V-on kell tartani. A 2732-es IC viszont olvasáskor 0 V-ot igényel a programbemeneten. A feszültség átkapcsolása az 5 db inverterrel valósítható meg.

Ha a modul HS kapcsát a testre kötjük és PC7 alacsony szinten van, akkor a D2 Z-dióda az I5 inverteren keresztül testpotenciálra kerül. Ez a Z-dióda 5 V kimenő feszültséget hoz létre.
Ha az LS és a PC7 érintkező kerül alacsony szintre, a D3 Z-dióda az I3 invertert rövidre zárja.
PC7 alacsony és LS vagy HS, tehát PC6 magas szintje esetén a kimeneten 25 V programozási feszültség áll rendelkezésünkre.
Ha PC7 magas szinten van, akkor PC6-tól függetlenül a kimeneten 0 V jelenik meg, mivel az I6 inverter a T1 tranzisztor bázisát alacsony (test) szintre állítja és ezzel járja a tranzisztort.

A különböző feszültségszintek előállításához a T2 tranzisztor szolgál áramforrásként. A T2 maximális kollektoráramát a D4 Z-dióda és az R5 ellenállás 5 mA-re korlátozza.
Az EPROM típusának meghatározására az N4 jelű ÉS-kapu és két darab 74LS125 típusú tri-state-puffer IC-ből álló fokozat szolgál. PC3 és PC4 magas szintje esetén az adatok a beviteli puffertárolóból a kimenetre kerülnek (kapcsolódnak). Ha megnézzük az EPROM-ok programozási görbéit, láthatjuk, hogy ez az állapot minden szóba jöhető EPROM-nál az EPROM letiltását eredményezi. Ezáltal nem hamisítja meg az adatokat.

Ezek után nézzük a programot:
A RUN utasítás után először egy kis gépi kódú rutin kerül beírásra a tárolóba. Ezt a rutint használja majd a CIA az 50 s-os programozó impulzus előállítására. BASIC-ben ilyen mértékű időzítést nem lehet megvalósítani.
Ezután következik a behelyezett modul típusának megállapítása. Ha a modult elfelejtettük betenni, a program egy hibaüzenettel befejeződik. Ha minden rendben van, a képernyőn megjelenik a programválaszték (menü).
Minden programozás előtt érdemes ellenőrizni, hogy az EPROM törölve van-e? Egy, már programozott EPROM a tárolóba beolvasható, megduplázható, vagy módosítható.

A programozás végén automatikusan összehasonlításra kerül a tároló és az EPROM tartalma. Ha valamilyen eltérés van, a képernyőn hibaüzenet jelenik meg. Az összehasonlítást a 3. menüpont kiválasztásával végezhetjük el.

Ha másik EPROM típusra akarunk áttérni, a 6-os menüpontot kell kiválasztanunk. Először cseréljük ki a modult, majd nyomjuk le a 6-os billentyűt, vagy indítsuk el a programot újból RUN-nal.

Az 5-ös menüpont kiválasztásával egy monitorprogramba ugorhatunk. Ezzel byte-onként kijavíthatjuk a programot. Természetesen előzőleg a monitorprogramot is a tárolóba kell tölteni.

Néhány szó az áramellátásról. Az 5 V-os üzemi feszültséget a C 64-es szolgáltatja. A programozási feszültséget viszont egy külső áramforrásról kell venni. Mivel a kapcsolófokozat stabilizálja a feszültséget, 28 V és kb. 40 V között bármilyen egyenfeszültséget használhatunk. Kiválóan alkalmas erre a célra egy nyomtatott áramkörre szerelhető kis transzformátor, amelynek szekunderfeszültsége 24 V. Egyenirányítás és szűrés után kb. 33 V-os egyenfeszültséget kapunk. A programozás alatt az áramfelvétel kb. 50 mA, tehát a kereskedelemben kapható legkisebb transzformátor is megfelel erre a célra.

```
1 rem *** p 30 ***
2 rem
5 rem       eprom egeto v1.2
10 rem ********************************
15 rem **                              **
20 rem **     egeto  software          **
25 rem **                              **
30 rem **     verzio 04.12.1983        **
35 rem **                              **
40 rem **        c 64-hez              **
45 rem **                              **
50 rem ********************************
52 if gg=0 then gg=1:load"profimon 64",8,1
55 poke 56,128:rem basic-vege
60 pp=57088:rem eprom-port
65 hr=52992:rem $cf00 kezdes 50 millisec
70 ra=32760:rem ram kezdocime eprom olvasashoz
80 for i=1 to 40:st$=st$+chr$(162):next
90 cu$=chr$(145):cd$=chr$(17):cr$=chr$(29):cl$=chr$(157)
95 for i=1 to 24:fl$=fl$+cr$:next
96 fl$=cu$+fl$+"      "+cl$+cl$+cl$+cl$+cl$
100 dim pm(3),ep(3),bz(3),ma(3)
110 pm(1)=112  :pm(2)=64    :pm(3)=64
120 ep(1)=2716:ep(2)=2532:ep(3)=2732
130 bz(1)=2048:bz(2)=4096:bz(3)=4096
140 ma(1)=96:ma(2)=80:ma(3)=80
150 tx$(1)="eprom olvasasa"
151 tx$(2)="eprom programozasa"
152 tx$(3)="eprom ellenorzese"
```

```
153 tx$(4)="eprom = ures ?"
154 tx$(5)="profimonhoz"
155 tx$(6)="ujrainditas"
160 for i=0 to 41
170 read a
180 poke hr+i,a
190 next
250 poke pp+3,144:rem minden kimenet
300 poke pp+2,184:rem %11110000
350 ep=peek(pp): rem modul olvasas
400 ep=-ep+255:rem tipusellenorzes
500 if ep<1 or ep>3 then 49000
510 rem modul nincs beteve
550 print"Sq            eprom tipusa";ep(ep)
600 print:print st$
700 for i=1 to 6
750 print"   "i". "tx$(i)
800 next
1000 a=0
1050 print"qq         Kerem valasszon funkciot"
1060 print:input"                    ";a
1065 if a <1 or a >6 then 550
1500 on agosub 2000, 3000, 4000, 5000, 6000, 7000
1950 goto250
2000 poke pp+3,144
2050 poke pp+1,0
2150 poke pp+2,0
2350 gosub 2000
2400 za=an
2500 for bt=es to es+an
2520 print f1$;za
2550 ha=int(bt/256):la=bt-ha*256
2600 poke pp+1,la
2650 poke pp+2,ha
2700 w=peek(pp)
2710 poke rs+bt,w
2800 za=za-1:next bt
2999 return
3000 poke pp+3,128
3050 poke pp+1,0
3150 poke pp+2,0
3200 poke 53001,pm(ep)
3350 gosub 20000
3400 za=an
3500 for bt=es to es+an:print f1$;za
3550 ha=int(bt/256):la=bt-ha*256
3560 ha=ha or ma(ep)
3600 poke pp+1,la
3650 poke pp+2,ha
3700 w=peek(rs+bt)
3710 poke pp,w
```

```
3750 sys 52992:rem programozo impulzus
3800 za=za-1:next bt
4000 poke pp+3,144
4050 poke pp+1,0
4150 poke pp+2,0
4300 if a=2 then print cd$;cd$;"    ";tx$(3);
4305 if a=2 then print cu$;cu$;cu$:goto 4400
4350 gosub 20000
4400 za=an
4500 for bt=es to es+an:print fl$;za
4550 ha=int(bt/256):la=bt-ha*256
4600 poke pp+1,la
4650 poke pp+2,ha
4700 w=peek(pp)
4710 if w<>peek(rs+bt) then bt=bz(ep):fl=1
4750 if fl=1 then printcd$;cd$"    eprom nem azonos"
4800 za=za-1:next bt
4850 if fl=1 then fl=0:goto 4950
4900 print cd$;cd$;"    jo az eprom"
4950 gosub 50000
4999 return
5000 poke pp+3,144
5050 poke pp+1,0
5150 poke pp+2,0
5350 gosub 20000
5400 za=an
5500 for bt=es to es+an:print fl$;za
5550 ha=int(bt/256):la=bt-ha*256
5600 poke pp+1,la
5650 poke pp+2,ha
5700 w=peek(pp)
5710 if w<>255 then print cd$;cd$;"    az eprom nem
ures!!!!":bt=bz(ep)
5715 if w<>255 then print"    az eprom nem ures!"
5720 if w<>255 then bt=bz(ep)
5800 za=za-1:next bt
5850 if w=255 then print cd$;cd$;"  az eprom ures"
5950 gosub 50000
5999 return
6000 sys 12*4096
7000 run
20000 print chr$(147):print"              eprom
tipus";ep(ep)
20020 print:print"          ";:print tx$(a):print:print
st$:print
20030 if a=4 then goto 20100
20050 print"q   ram kezdocim      0";
20055 print cl$;cl$;cl$;:input rs:rs=rs+32768
20100 print"q   eprom kezdocim    0";
20105 print cl$;cl$;cl$;:input es
20150 print"q   byte-ok szama      ";bz(ep)-es;
```

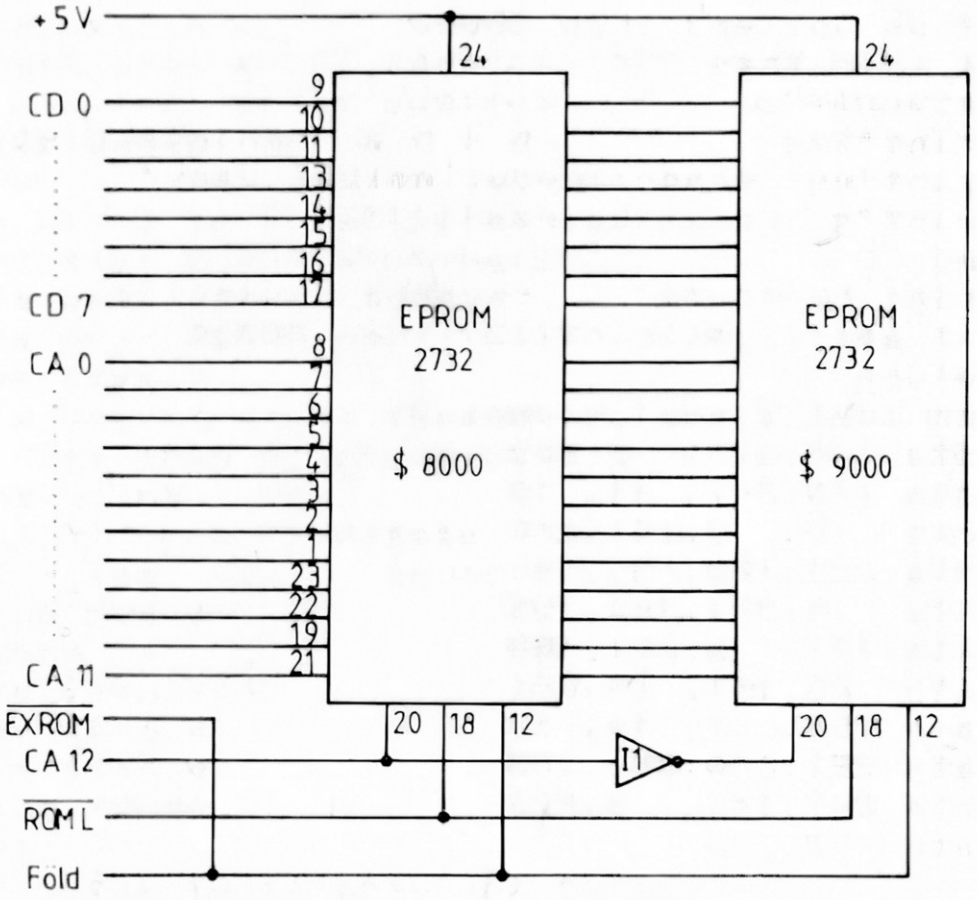
```
20155 for i=1 to 7:print cl$;:next
20200 input an:an=an-1
20250 if es >bz(ep)then 20000
20300 if es<0then250
20350 if an >bz(ep) then 20000
20400 if an<0 then 250
20450 return
49000 print"Sqq            h i b a ":print:print st$
49020 print"qq  programmodul nelkul,vagy"
49040 print"q  rossz beiras!!!!"
49100 end
50000 print cd$;cd$;"    tovabbi bevitel?"
50010 get a$:if a$<>chr$(13) then 50010
50020 return
60000 rem adat a gepi kodhoz
60010 data 120,173,  2,223
60020 data 133,247, 41, 15
60030 data   9,  0,141,  2
60040 data 223,169, 80,141
60050 data   4,221,169,195
60060 data 141,  5,221,169
60070 data  25,141, 14,221
60080 data 169,  1, 44, 13
60090 data 221,240,251,165
60100 data 247,141,  2,223
60110 data  88, 96
```

8.2 EPROM-kártya a C 64-eshez

Miután első EPROM-unkat sikeresen beprogramoztuk, nézzük meg, hogyan csatlakoztathatjuk a C 64-eshez.
Csatlakozási helyként a bővítő port jöhet szóba. Itt minden szükséges jel rendelkezésünkre áll.
A C 64-es dokumentációja szerint a modulok a $8000–$9FFF tárolótartományt foglalják le. Ez a 8 k terjedelmű programok számára is elegendő.
Ezt nevezik a számítógép autostart állományának. Az itt tárolt programok a számítógép bekapcsolásakor azonnal indulnak (erről a későbbiekben még szó lesz).
Ezenkívül a GAME és a ROML csatlakozóknak köszönhetően nincs szükség speciális címdekódolásra. Mihelyt a GAME vezeték alacsony szintre kerül, a ROML jelet az EPROM IC kiválasztó jeleként használhatjuk. Ilyenkor a normál esetben itt található RAM kikapcsolódik.

Mivel a számítógép modultartománya 8 k nagyságú, a kapcsolásban két 2732-es típusú EPROM-mal számoltunk.
A $8000–$8FFF tartományt használó EPROM esetében nincs szükség kiegészítő kapukra. A címzés a ROML jellel a CS-en keresztül (IC kiválasztása) és kiegészítésként a 12-es címbittel, mint OE (bemenet engedélyezése) jellel történik.

Ha egy második 2732-es EPROM által a $9000–$9FFF tartományt is igénybe akarjuk venni, a 12-es címbitet invertálással kell OE jelként használni.



Térjünk át az autostart magyarázatára!

Bekapcsolás után a C 64-es végrehajtja a bekapcsolási vagy más néven RESET rutint. Ez először letiltja a megszakítást és beállítja a Verem-mutatót (STACK). Közvetlenül ezután ellenőrzésre kerül a kérdéses tartományban a modul. Az ellenőrzés abból áll, hogy a $8004 címtől kezdődő 5 tárolórekeszben megtalálható-e a „CBM80" jelzés. Ez hexadecimálisan a következőképpen néz ki:
C3 C2 CD 38 30
Ha a vizsgálat eredménye pozitiv, a program elágazik az első két byte-ban ($8000/$8001) tárolt címre. Az adott címen található önindító programnak tehát mindent, pl. a megszakítások, a képernyő és a tároló inicializálását, önállóan kell elvégezni. Ehhez természetesen használhatjuk az operációs rendszer rutinjait is, de saját rutinokat is írhatunk.

Akit az autostart programozása közelebbről is érdekel, annak ajánljuk, hogy a ROM-lista segítségével alaposan tanulmányozza át a gép operációs rendszerét.

Mi itt lemondunk az autostartról, így nincs szükségünk különleges eljárásokra. A programot egyszerűen beégethetjük az EPROM-ba. Mivel a bekapcsolás után az autostart ellenőrzés megállapítja, hogy a szükséges jelsorozat nincs a tárolóban, a számítógép a szokásos módon inicializálódik. Az EPROM programja a SYS utasítással indítható.

Az EPROM-ot már tudjuk programozni és a C 64-essel működtetni is. De mi legyen azokkal az EPROM-okkal, amiknek a tartalmára már nincs szükségünk? Az EPROM-ban tárolt információk ultraibolya fény hatására eltűnnek, természetesen ettől azért még használhatók az Alpokban és a Földközi tenger mentén is.

Maga a folyamat elég hosszadalmas, mert még közvetlen napsugárzásnál is több napot vagy hetet vesz igénybe. Az viszont tény, hogy minél erősebb a napsugárzás, az információk annál gyorsabban törlődnek. Az általunk választott megoldással az EPROM maximum 2 óra alatt törölhető. Ehhez persze nem a valódi napsugárzást használtuk fel, hanem egy kereskedelemben kapható ultraviola-lámpát. Típusa Philips TUV 6W. Ezzel kb. 1 cm távolságból, az EPROM-ok 35–40 perc alatt törlődnek.

A törlési művelet során feltétlenül tartsuk be a következő szabályokat:

SOSE NÉZZÜNK A BEKAPCSOLT LÁMPA FÉNYÉBE, MERT EZ SÚLYOS SZEMKÁROSODÁST OKOZHAT!

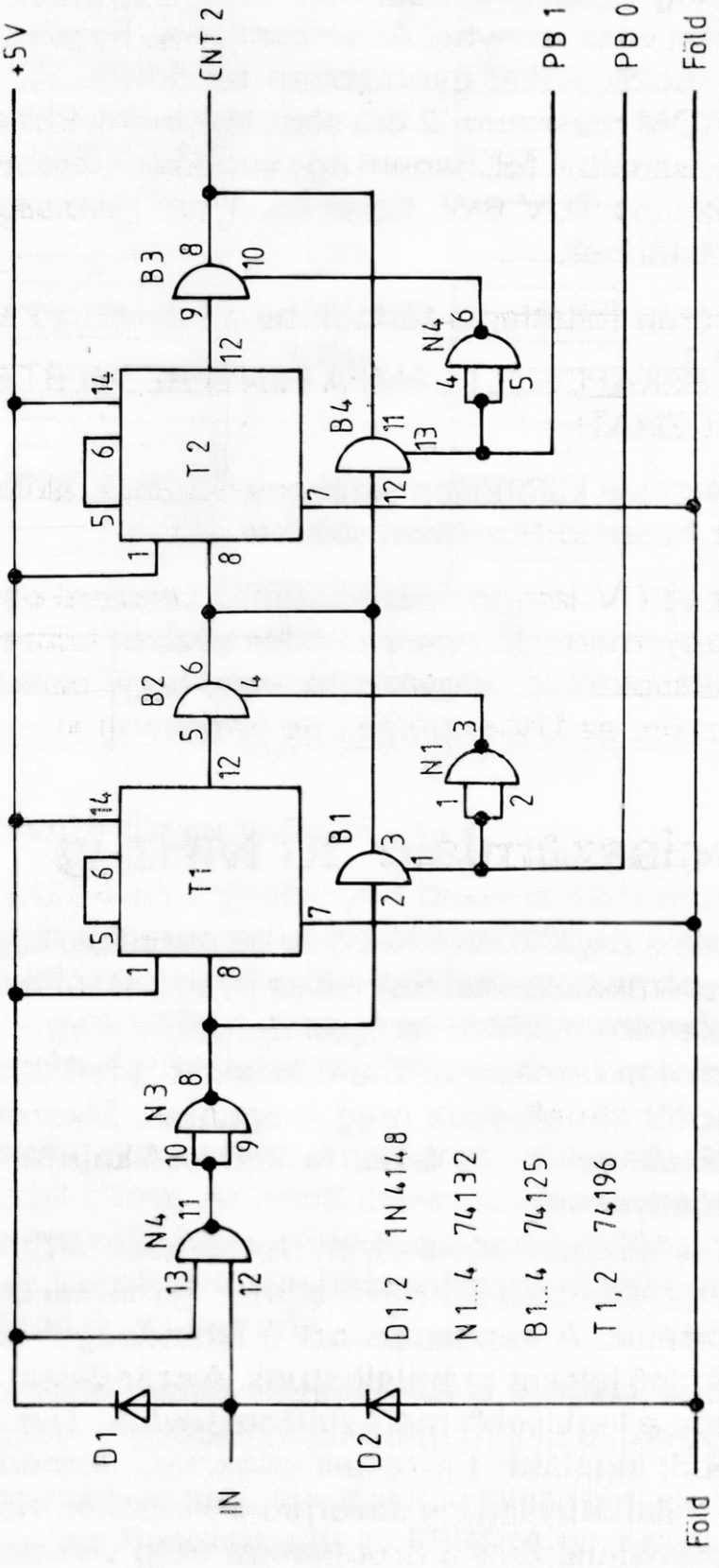
Vannak az UV-sugárzásra különösen érzékeny egyének, akiknél bőrbántalmakat vagy napszúráshoz hasonló tüneteket idézhet elő.

Óvatosságból ezért az UV-lámpa használatához célszerű egy szekrényt építeni. Ez legyen eléggé nagy, mert a lámpa működés közben felmelegszik, és gondoskodjunk szellőzőnyílásokról is. Vigyázzunk arra, hogy ezeken keresztül csak a meleg levegő távozzon, az UV-sugárzás ne juthasson ki.

8.3 Frekvenciaszámláló 30 MHz-ig

A frekvenciaszámláló a digitális voltmérő és az oszcilloszkóp mellett a hobbilabor legértékesebb mérőkészüléke. Sajnos az ilyen készülékek, a félvezetők árának rohamos csökkenése ellenére is, igen drágák.
Erre a célra azonban a már meglévő C 64-esünket is felhasználhatjuk és így jelentős plusz költségtől kímélhetjük meg magunkat. Sikerült létrehoznunk egy olyan általánosan alkalmazható és árban is kedvező kapcsolást, ami eleget tesz a megnövekedett igényeknek.
Mindössze 4 db IC felhasználásával olyan frekvenciamérőt kapunk, ami három méréstartományban, maximum 30 MHz-et mér. A mérési tartomány átkapcsolása programból történik. A kapcsolás azt a lehetőséget használja ki, hogy a CONT-bemeneten külső jeleket számlálhatunk. A számlálást a CIA U2 A időzítője végzi. Ahhoz, hogy a legkisebb méréstartományban 1 Hz pontossággal mérhessünk, a mérési időt legalább 1 s-ra kell választani. A mérési időt a CIA U1 A időzítője állítja elő. Pillanatnyilag ne zavarjon bennünket, hogy eredetileg ezt a megszakításhoz használjuk. Erre a problémára még visszatérünk.
Mivel az időzítő csak 16 bites, előosztó nélkül csak legfeljebb 65535 Hz-es frekvencia mérhető. Emiatt nemcsak az A, hanem a B időzítő is részt vesz a számlá-

lásban. Ezzel már egy 32 bites számlálónk van, amivel 2 ↑ 32 = 4294967296 különböző értéket ábrázolhatunk.
Így a frekvenciaszámlálónk felső méréshatára elméletileg 4 gigaHz, de sajnos a hardver miatt ez nem valósítható meg.



Kísérleteink szerint a CNT-bemenet maximális bemenő frekvenciája pontosan a CIA ütemfrekvenciájának fele.

Ez a C 64-esen 458 kHz. Ennél nagyobb frekvenciák mérésekor külső elosztót kell alkalmazni. Ezt a feladatot látja el az ábrán bemutatott kapcsolás.
Két Schmitt–trigger-bemenetű NAND kapuból (74LS132), két decimális számlálóból (74LS196) és négy tri-state kimenetű tárolóból (74LS125) áll. A be- és kikapcsoláshoz szükséges vezérlőjeleket a felhasználói port PB0 és PB1 bitjei szolgáltatják.

A programhoz még a következőket kell tudni.
A tulajdonképpeni számlálást egy gépi rutin végzi, amit a BASIC-program tölt be a tároló $CF00 címétől kezdődően. Továbbá ez a rutin tölti be az időzítő kezdőértékét, ami a CIA2 A és B időzítőjére vonatkozóan $FFFF.
Mint azt a 3. fejezetből már tudjuk, az időzítők visszafelé számlálnak, tehát a megjelenő impulzusok a számláló tartalmát csökkentik.
A CIA1 A időzítőjének kezdőértéke $FFFF, a B időzítőé $0010. Ebből könnyen kiszámithatjuk, hogy mennyi idő alatt érik el a 0-t. Az A időzítőnek ehhez 65536 ütemimpulzusra van szüksége. Mivel folyamatos (CONTINOUS) üzemmódban működik, nulla átmenetnél egy jelet ad és a számlálást elölről kezdi. A leadott jeleket a B időzítő számlálja, tehát az A időzítő minden nulla-átmeneténél csökken eggyel a tartalma. Így összesen 16*65536 ütemimpulzusra van szükség ahhoz, hogy a B időzítő is nullára fusson. Ez valamivel több, mint 1 s, tehát mérési időnek éppen megfelel.
Miután a program elindította az időzítőt, egy ciklussal kivárjuk a mérési idő végét. Ha lejárt, leállnak a CIA2-ben az impulzusszámlálóink és értékeik a nullaslap (Zero-Page) $F7, $F8, és $F9 című rekeszeibe íródnak.
A 32 bites időzítő felső 8 bitjét nem kell tárolni, mert ott, a fent említett hardverokok miatt, úgysem történik változás.
Felmerülhet a kérdés, hogy az adatokat miért kell a nullaslapon tárolni, hiszen a CIA regiszterei BASIC-ből PEEK utasítással kiolvashatók. Ez igaz, de mivel az CIA1 A időzítőjét is használjuk, amit eredetileg a megszakítások kiváltásához veszünk igénybe, a mérés után vissza kell adnunk az operációs rendszernek az időzítőn és a megszakításon keresztül a vezérlést.
Ez egyszerűen úgy történik, hogy lehívjuk a $FDA3 címen található rendszerrutint. Ez az időzítőket feltölti a bekapcsolás utáni szokásos értékkel, így a CNT-impulzus által létrehozott számláló érték felülíródna.

A BASIC-program a mérést SYS 52992-vel indítja. A mérési idő lejárta után a számláló értéke a megadott tárolórekeszekbe kerül. A kapott adatok az F és F$ változókba kerülnek, majd nagy, jól látható számok formájában megjelennek a képernyőn. A kivitel füzérként történik.
Ha az értékek 320 000 felett vannak, a következő mérési tartomány kerül bekapcsolásra. 30 000 alatt a visszakapcsolás történik az alacsonyabb mérési tartományra.
Ilyen típusú hiszterézissel az átkapcsolás problémája két mérési tartomány között az átkapcsolási határon megszűnik.

Ha a mért érték 0, tehát nincs bemenő váltófeszültség, automatikusan a legfelső méréstartomány lép érvénybe. Ez első pillantásra érthetetlennek tűnik, de a 6526-os IC felépítéséből adódik. 10 MHz feletti frekvenciáknál előfordulhat, hogy az impulzusok nem lesznek megszámlálva. Így a program nem kapcsol magasabb méréstartományra. A legfelső tartományban azonban ez nem fordulhat elő, így a lefelé kapcsolás biztosan megtörténik.

A CIA állandó inicializálása miatt a vezérlési információkat az előosztó számára a SYS utasítás előtt POKE-val újra be kell írni. Itt a következő szabály érvényesül:

PB1	PB0	
0	0	$= 0_{dec}$ – közvetlen mérés
0	1	$= 1_{dec}$ – mérés 10-es osztással
1	0	$= 2_{dec}$ – mérés 10-es osztással (nem használjuk)
1	1	$= 3_{dec}$ – mérés 100-as osztással

Összesen tehát két vezérlővezetékre, két tápvezetékre és a CNT-bemenet vezetékére van szükségünk. Mivel a kapcsolás csak négy IC-ből áll, helyigénye minimális, kis hengerben helyezhetjük el és egy kissé hosszabb, 5 eres kábellel csatlakoztathatjuk a felhasználói portra. A kis henger egyúttal tapintófejként is használható.

Ezek után nézzük a programot! A mérőkészülék elkészítéséhez és használatához sok sikert kívánunk.

```
1 rem *** p 31 ***
2 rem
100 rem ******************************
110 rem *                            *
120 rem *   frekvencia szamlalo      *
130 rem *                            *
140 rem *       1.3 variacio         *
150 rem *                            *
160 rem *    rolf brueckmann         *
170 rem *    friedhelm kuech         *
180 rem ******************************
190 poke 53280,0
200 poke 53281,0
210 print chr$(5)
220 print chr$(147):print:print" frekvencia mero "
230 print:print:print"egy kis turelmet kerek!"
240 mb=2
250 ml=52992:rem $cf00
260 mh=53078
270 v=53248:rem vic kezdocim
280 y=110:rem y-sprite pozicio
290 d1=1024+12*40+18
300 d2=d1+6:rem tizedespont pozicionalas
310 busy=1024+13*40+36
```

```
320 for i=1 to 39:b1$=b1$+" ":next
330 f1$="    "
340 for i=1 to 32:f2$=f2$+chr$(96):next
350 f3$=chr$(98)+left$(b1$,32)+chr$(98)
360 led(0)=1024+10*40+36
370 led(1)=1024+8*40+36
380 led(2)=1024+6*40+36
390 ma(0)=0:ma(1)=1:ma(2)=3
400 poke v+16,0
405 rem sprite legnagyobb helyierteku bit
410 poke v+23,255:rem sprite y iranyu nagyitas
420 poke v+29,0:rem no sprite x iranyu nagyitas
430 poke v+28,0:rem normal szin
440 poke v,250:poke v+1,y:poke v+2,225:poke v+3,y
450 poke v+4,200:poke v+5,y:poke v+6,175:poke v+7,y
460 poke v+8,150:poke v+9,y
465 poke v+10,125:poke v+11,y
470 poke v+12,100:poke v+13,y:poke v +14,75
475 poke v+15,y
480 for i=v+39 to v+46
490 :poke i,1:rem sprite szine=feher
500 next
510 rem read es poke  prg
520 for i=m1 to mh
530 :read a
540 :poke i,a
550 next
560 rem read es poke sprites
570 for i=12800 to 13440
580 :read a
590 :poke i,a
600 next
610 rem mutatok beallitasa
620 for i=0 to 9
630 :sp(i)=200+i:rem start
640 next i
650 print chr$(147):print:print
660 printf1$;chr$(176);f2$;chr$(174)
670 printf1$;chr$(98);
675 print"  30 mhz frekvenciamero c64      ";chr$(98)
680 print f1$;chr$(171);f2$;chr$(179)
690 print f1$;chr$(98);"
mhz";chr$(215);chr$(98)
700 print f1$;f3$
710 print f1$;chr$(98);"
khz";chr$(215);chr$(98)
720 print f1$;f3$
730 print f1$;chr$(98);"
hz";chr$(215);chr$(98)
740 print f1$;f3$
750 print f1$;f3$
```

```
760 print f1$;chr$(98);"
bsy";chr$(215);chr$(98)
770 print f1$;f3$
780 print f1$;chr$(173);f2$;chr$(189)
790 print:print:print:print
800 print"        "chr$(18)"f1";
805 printchr$(146)" informacio";
810 print"        "chr$(18)"f7"chr$(146)" vege"
820 get a$
830 if a$=chr$(133) then goto 1090:rem informacio
840 if a$=chr$(136) then poke v+21,0
845 if a$=chr$(136) then print chr$(147):print
846 if a$=chr$(136) then print"   vege":end
850 poke busy,81:rem jelzes be
860 poke 56579,255:poke 56577,ma(mb)
870 sys 52992:rem meres
880 poke busy,87:rem jelzes ki
890 a=peek(247):b=peek(248):c=peek(249)
900 zw=((c*65536+b*256+a+1)-2↑24)*-1
910 k1%=(zw/100)*6.041:f=zw-k1%
920 if f> 320000 then if mb<2 then mb=mb+1
930 if f<30000 then if mb>0 then mb=mb-1
940 if f=0 then mb=2
950 for i=0 to 2:poke led (i),87:next
955 poke led(mb),81
960 if mb=0 or(mb=2 and f=0) then poke d1,32
965 if mb=0 or(mb=2 and f=0) then poke d2,32
966 if mb=0 or(mb=2 and f=0) then goto 990
970 if mb=1 then poke d1,32:poke d2,81
980 if mb=2 then poke d1,81:poke d2,32
990 rem ****frekvencia****
1000 :
1010
f$=str$(f):le=len(f$)-1:f$=right$(f$,le):f$=left$(f$,8)
1020 for i=8 to1 step-1
1030 ::ifi>le thenpokev+21,peek(v+21)and(255-2↑(i-1))
1035 if i>le then goto 1060
1040 ::poke 2040+(le-i),sp(val(mid$(f$,i,1)))
1050 ::poke v+21,peek(v+21)or2↑(i-1)
1060 next
1070 goto 820
1080 rem ***informacio***
1090 poke v+21,0:poke 53280,1
1100 printchr$(147)"           rinformacioRqq"
1110 print" a program es a hozza tartozo    ":print
1120 print"hardver segitsegevel a c64 egy":print
1130 print"nagyteljesitmenyu frekvenciamerove"
1140 print"qalakul at.  a frekvenciat a user-port"
1150 print"q6-os kivezetesen merjuk "
1160 print
1170 print"a felso frekvenciahatar kb. 30 mhz"
```

```
1180 print"qa bemeno feszultseg maximum 5 volt"
1190 print
1192 print
1200 print
1210 print
1220 print "q       rtovabb=barmely billentyuR"
1230 get a$:if a$=""then 1230
1240 goto 650
1250 data 120,169,255,141,  4,221,141,  5
1260 data 221,141,  6,221,141,  7,221,141
1270 data   4,220,141,  5,220,169, 16,141
1280 data   6,220,169,  0,141,  7,220,160
1290 data  17,162, 89,169, 81,141, 15,221
1300 data 169, 49,141, 14,221,140, 14,220
1310 data 142, 15,220,173,  6,220,208,251
1320 data 141, 14,221,141, 15,221,173,  4
1330 data 221,133,247,173,  5,221,133,248
1340 data 173,  6,221,133,249,173,  7,221
1350 data 133,250, 88, 76,163,253,  0, 15
1360 data 248,  0, 24, 12,  0, 56, 14,  0,120
1370 data  15,  0,120, 15,  0,120, 15,  0,120
1380 data  15,  0,120, 15,  0,120, 15,  0,120
1390 data  15,  0,120, 15,  0,120, 15,  0,120
1400 data  15,  0,120, 15,  0,120, 15,  0,120
1410 data  15,  0,120, 15,  0,120, 15,  0, 56
1420 data  14,  0, 24, 12,  0, 15,248,  0,  0
1430 data   0,224,  0,  1,224,  0,  3,224,  0
1440 data   6,224,  0,  0,224,  0,  0,224,  0
1450 data   0,224,  0,  0,224,  0,  0,224,  0
1460 data   0,224,  0,  0,224,  0,  0,224,  0
1470 data   0,224,  0,  0,224,  0,  0,224,  0
1480 data   0,224,  0,  0,224,  0,  0,224,  0
1490 data   0,224,  0,  0,224,  0, 15,254,  0
1500 data   0, 15,248,  0, 28, 28,  0, 56, 14
1510 data   0,120, 15,  0,120, 15,  0,  0, 15
1520 data   0,  0, 30,  0,  0, 60,  0,  0, 120
1530 data   0,  0,240,  0,  1,192,  0,  3,128
1540 data   0, 14,  0,  0, 28,  0,  0, 56,  0
1550 data   0,120, 12,  0,112, 30,  0,127,255
1560 data   0,126,255,  0,120, 62,  0,112, 12
1570 data   0,  0, 15,248,  0, 28, 28,  0, 56
1580 data  14,  0,120, 15,  0,120, 15,  0,  0
1590 data  15,  0,  0, 15,  0,  0, 14,  0,  0
1600 data  12,  0,  1,248,  0,  1,248,  0,  0
1610 data  12,  0,  0, 14,  0,  0, 14,  0,  0
1620 data  15,  0,  0, 15,  0,120, 15,  0,120
1630 data  15,  0, 56, 14,  0, 28, 28,  0, 15
1640 data 248,  0,  0,  0,224,  0,  1,224,  0
1650 data   3,224,  0,  7,224,  0, 14,224,  0
1660 data  28,224,  0, 56,224,  0,112,224,  0
1670 data 112,224,  0,224,224,  0,224,224,  0
```

```
1680 data 224,224,  0,255,254,  0,  0,224,  0
1690 data   0,224,  0,  0,224,  0,  0,224,  0
1700 data   0,224,  0,  0,224,  0,  0,224,  0
1710 data  15,254,  0,  0,127,255,  0,120,  0
1720 data   0,120,  0,  0,120,  0,  0,120,  0
1730 data   0,120,  0,  0,120,  0,  0,120,240
1740 data   0,127,252,  0,  0, 62,  0,  0, 30
1750 data   0,  0, 15,  0,  0, 15,  0,  0,  7
1760 data   0,  0,  7,  0,  0,  7,  0,112,  7
1770 data   0,120, 15,  0, 56, 14,  0, 28, 28
1780 data   0, 15,248,  0,  0, 15,252,  0, 28
1790 data  14,  0, 56, 15,  0,112,  0,  0,112
1800 data   0,  0,112,  0,  0,112,  0,  0,112
1810 data   0,  0,119,240,  0,124, 60,  0,124
1820 data  30,  0,120, 15,  0,112, 15,  0,112
1830 data   7,  0,112,  7,  0,112,  7,  0,112
1840 data   7,  0,120, 15,  0, 56, 14,  0, 28
1850 data  28,  0, 15,248,  0,  0,127,255,  0
1860 data 120, 15,  0,120, 15,  0,120, 15,  0
1870 data   0, 15,  0,  0, 14,  0,  0, 30,  0
1880 data   0, 60,  0,  0,120,  0,  0,112,  0
1890 data   0,224,  0,  0,224,  0,  1,192,  0
1900 data   1,192,  0,  1,192,  0,  1,192,  0
1910 data   1,192,  0,  1,192,  0,  1,192,  0
1920 data   1,192,  0,  1,192,  0,  0, 15,248
1930 data   0, 28, 28,  0, 56, 14,  0,112,  7
1940 data   0,112,  7,  0,112,  7,  0,112,  7
1950 data   0, 56, 14,  0, 28, 28,  0, 15,248
1960 data   0, 15,248,  0, 28, 28,  0, 56, 14
1970 data   0,112,  6,  0,112,  7,  0,112,  7
1980 data   0,112,  7,  0,120, 15,  0, 56, 14
1990 data   0, 28, 28,  0, 15,248,  0,  0, 15
2000 data 252,  0, 28, 14,  0, 56, 15,  0,112
2010 data   7,  0,112,  7,  0,112,  7,  0,112
2020 data   7,  0,112,  7,  0, 56,  7,  0, 28
2030 data  15,  0, 15,255,  0,  0,  7,  0,  0
2040 data   7,  0,  0,  7,  0,  0,  7,  0,  0
2050 data   7,  0,  0,  7,  0,120, 15,  0, 56
2060 data  14,  0, 28, 28,  0, 15,248,  0,  0
2070 data 92
```

8.4 Automata digitális voltmérő

A 6.4 fejezetben említett CA3162 típusú IC kiválóan alkalmas digitális voltmérő építésére. A kísérleteink eredményeként kifejlesztettünk egy olyan készüléket, amelyik a méréstartományt automatikusan kiválasztja.
Az esetek többségében (több mint 90%) egyenfeszültség mérésére alkalmaztuk, ezért, valamint a költségek kímélése érdekében, fejlesztésünket egyenfeszültség mérésére korlátoztuk. A teljes méréstartomány −10 V...+100 V.

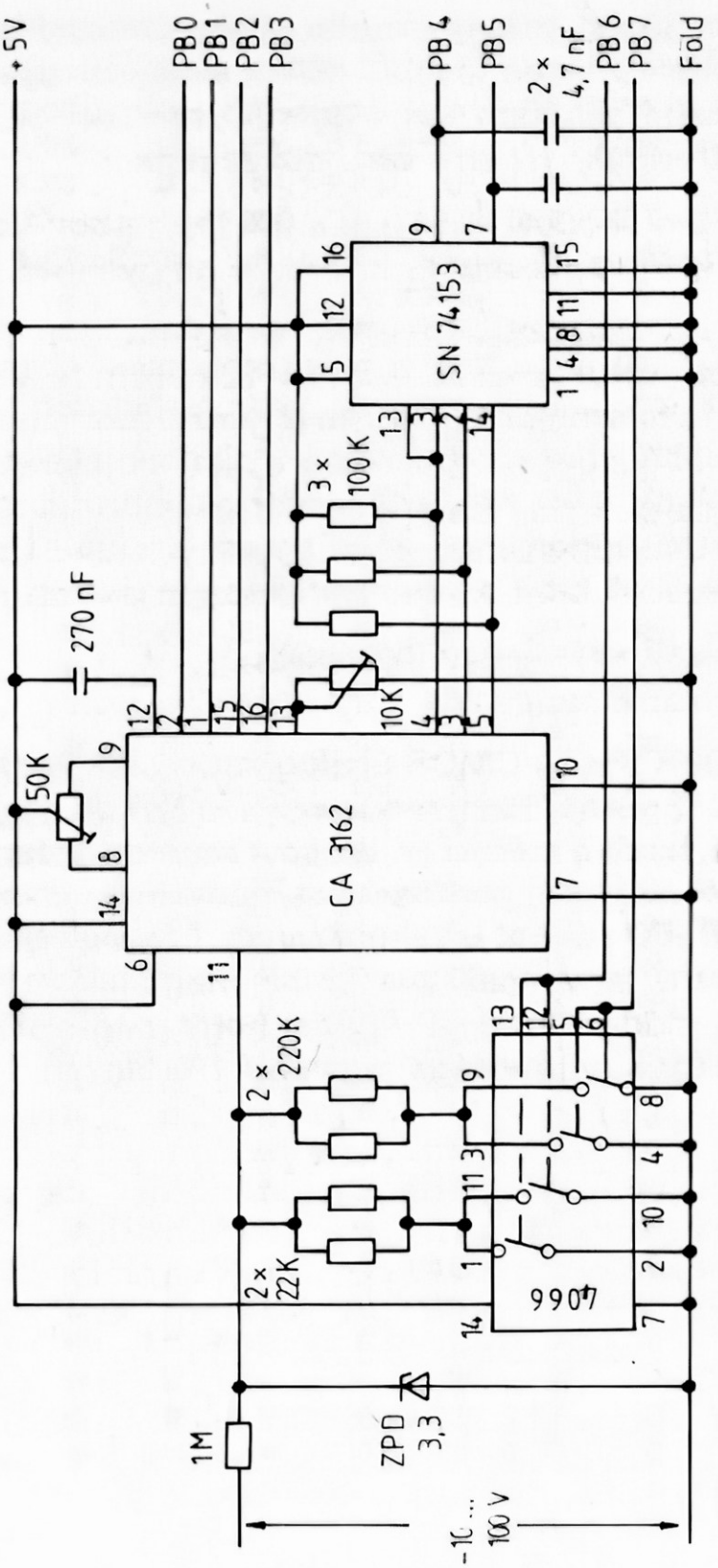
Az előző fejezetben közölt frekvenciaszámlálóhoz hasonlóan ez a kapcsolás is minimális helyen elfér. Minden további nélkül elhelyezhetjük egy nyomógombházban. Az 5 eres kábel azonban nem elegendő, mert ahhoz, hogy a felhasználói portra csatlakoztathassuk, 10 érre van szükségünk.

A CA3162 típusú IC működési elvét már a 6.4 fejezetben leírtuk, ezért itt csak a kapcsolás néhány különlegességére hívjuk fel a figyelmet.

A voltmérő által szolgáltatott adatokat a számítógép a felhasználói port PB0–PB3 bitjein keresztül veszi át. A DVM–IC három helyiérték vezetéke nem kerül közvetlenül a felhasználói portra. Mivel három méréstartomány esetén a kiválasztáshoz legfeljebb 2 bitre van szükség, a digitális jeleket egy 74LS153 típusú IC-vel multiplexeljük. Így a helyiérték-információhoz is csak két vezeték kell. Ez összesen 8 vezeték, éppen annyi, mint amennyi a felhasználói porton rendelkezésünkre áll. A felső 4 bitet a következőképpen osztottuk fel:

PB4–PB5: multiplexelt számjegyinformációk
PB6–PB7: méréshatár átkapcsolás

Az átkapcsolást 4066 típusú CMOS analóg kapcsolók végzik. Ezek bekapcsolt állapotban kb. 200 Ω ellenállással rendelkeznek. Nyitott állapotban ellenállásuk több, mint 100 mΩ, tehát a mérési eredményt számottevően nem befolyásolják. Azoknak, akik ennél nagyobb pontosságot igényelnek, javasoljuk, hogy analóg kapcsolók helyett REED reléket alkalmazzanak. Ezeknek bekapcsolt állapotban mindössze néhány mΩ az ellenállásuk. Ennek megfelelően természetesen a vezérlő kapcsolást is módosítani kell. Ahhoz, hogy megfelelő mérési eredményt kapjunk, az ellenállások pontossága legalább 1% legyen.


+5V
PB0
PB1
PB2
PB3
PB4
PB5
2 × 4,7 nF
PB6
PB7
Fold
SN 74153
3 × 100 K
10K
270 nF
50K
CA 3162
2 × 220K
2 × 22K
4066
1M
ZPD 3,3
-10 ... 100 V

```
1 rem *** p 32 ***
2 rem
100 rem*********************************
110 rem*                               *
120 rem*automatikus digitalis voltmero*
130 rem*                               *
140 rem*        1.3.2 valtozat         *
150 rem*                               *
160 rem*       rolf brueckmann         *
170 rem*       friedhelm kuech         *
180 rem*********************************
190 print chr$(147):print:print"    automatikus
digitalis voltmero"
200 print"qq        egy kis turelmet kerek!"
210 me(1)=0:me(2)=128:me(3)=64
220 ml=52992
230 mh=53032
240 v=53248
250 y=110:rem sprite y pozicio
260 busy=1024+13*40+36
270 z$="000"
280 for i=1 to 32:f0$=f0$+" ":f2$=f2$+chr$(96)
285 next
290 f1$="    "
300 f3$=chr$(98)+f0$+chr$(98)
310 f4$=chr$(19)
320 for i=1 to 12:f4$=f4$+chr$(17):next
330 for i=1 to 22:f4$=f4$+chr$(29):next
340 f5$=chr$(19)
350 for i=1 to 10:f5$=f5$+chr$(17):next
360 for i=1 to 11:f5$=f5$+chr$(29):next
370 f6$=chr$(157)+chr$(157)+ch  $(157)+chr$(145)
375 f6$=f6$+chr$(145)+ch  $(32)+chr$(157)
380 f6$=f6$+chr$  )+chr$(32)+chr$(17)
38  f6$=f6$+chr$(17)+chr$(157)+chr$(32)+chr$(17)
39  f6$=f6$+chr$(157)+chr$(32)
400 poke 56577,0:me=1:rem also hatar
410 poke 53281,0:poke 53280,1
420 poke v+16,0:rem sprite helyzet felso bit
430 poke v+23,255:rem sprite y iranyu nagyitas
440 poke v+29,0:rem sprite x iranyu nagyitas
450 poke v+28,0:rem egyszin uzemmod
460 poke v,225:poke v+1,y:poke v+2,200:poke v+3,y
470 poke v+4,175:poke v+5,y
475 poke v+6,150:poke v+7,y
480 poke v+8,125:poke v+9,y:poke v+10,100
485 poke v+11,y
490 poke v+12,100:poke v+13,y:poke v+14,75
495 poke v+15,y
500 for i=v+39 to v+46
510 : poke i,1      :rem sprite szin=feher
```

```
520 next
530 rem read es poke prg
540 for i=ml to mh
550 : read a:cs=cs+a
560 poke i,a
570 next
580 rem read es poke sprites
590 for i=12800 to 13440
600 : read a:cs=cs+a
610 :poke i,a
620 next
630 if cs<>37591 then print"hiba az adatban!":end
640 poke v+21,1:rem 0-as sprite bekapcsolva
650 poke 2040,200:rem elso digit=0
660 rem a mutatok beallitasa
670 for i=0 to 3
680 ::sp(i)=200+i:rem start
690 next i
700 print chr$(147),chr$(5):print:print
710 print f1$;chr$(176);f2$;chr$(174)
720 print f1$;chr$(98);
722 print " automatikus digitalis voltmero ";
724 print chr$(98)
730 print f1$;chr$(171);f2$;chr$(179)
740 print f1$;f3$
750 print f1$;f3$
760 print f1$;chr$(98);"                                          v
"chr$(98)
770 print f1$;f3$
780 print f1$;chr$(98);"                                          mv
";
785 print chr$(98)
790 print f1$;f3$
800 print f1$;f3$
810 print f1$;chr$(98);
815 print "                                    bsy "chr$(98)
820 print f1$;f3$
830 print f1$;chr$(173);f2$;chr$(189)
840 print:print:print:print
850 print"       "chr$(18);"f1";
855 print chr$(146);" informacio";
860 print"       "chr$(18);"f7";chr$(146);" vege";
870 get a$:if a$=""then 870
880 if a$=chr$(133) then goto 1320:rem informacio
890 if a$=chr$(136) then poke v+21,0
895 if a$=chr$(136) then print chr$(147)
896 if a$=chr$(136) then print chr$(5);"vege":end
900 poke busy,81:rem jelzes be
910 sys 52992:rem meres
920 poke busy,87:rem jelzes ki
930 z$=""
```

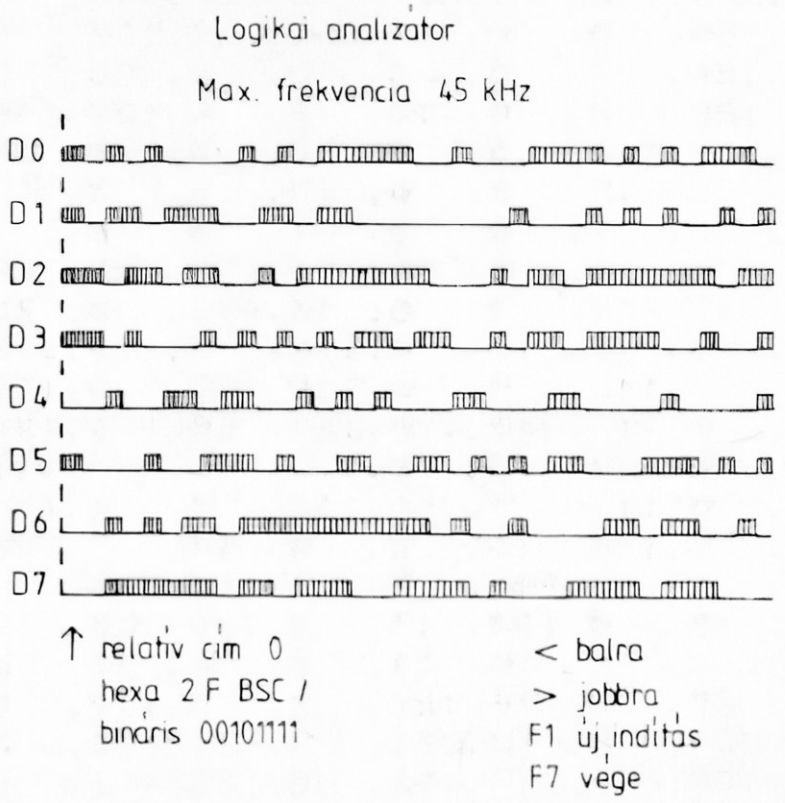
```
940 :
950 for i=1 to 2
960 :gosub 1480
970 :z$=z$+right$(a$,1)
980 next i
990 if z$="---" then me=me+1:goto 1030
1000 if z$="↑↑↑" then me=me+1:goto 1030
1010 if val(z$)>0 then if val(z$)<100 then me=me-1
1020 if val(z$)<0 then if val(z$)>-10 then me=me-1
1030 if me<1 then me=1
1040 if me>3 then me=3:gosub 1270:goto 870
1050 poke 56577,me(me):rem mereshatar
1060 print f4$;
1070 if me=1 then print"      "
1080 if me=2 then print chr$(5);chr$(113);"      "
1090 if me=3 then print chr$(5);"      ";
1095 if me=3 then print chr$(113);"    "
1100 if me=1 then poke 1024+400+33,81
1105 if me=1 then poke 1024+320+33,32:goto 1120
1110 poke 1024+320+33,81:poke 1024+400+33,32
1120 if left$(z$,1)="-" then z$=right$(z$,2)
1125 if left$(z$,1)="-" then gosub 1210:goto 1140
1130 gosub 1240:rem pozitiv elojel
1140 le=len(z$)
1150 for i=3 to step -1
1160 ::ifi>lethenpokev+21,peek(v+21)and(255-2↑(i-1))
1165 next
1170 ::poke 2040+(le-i),sp(val(mid$(z$,i,1)))
1180 poke v+21,peek(v+21) or 2↑(i-1)
1190 next
1200 goto 870:rem foprogram
1210 rem ****negativ elojel kiiratas****
1220 print f5$;chr$(31);chr$(18);
1225 print "      "chr$(146);f6$
1230 return
1240 rem ****pozitiv elojel kiiratas****
1250 print f5$;chr$(28);chr$(18);"      ";f6$
1260 return
1270 rem ****tulcsordulas*****
1280 rem villogtatas, meres kijelzese
1290 print f5$;chr$(29);"hataron tul"
1300 poke 53281,1:for p=1 to 100:next
1305 poke 53281,0:return
1310 rem ****informacio****
1320 poke v+21,0:poke 53280,1
1330 print chr$(147);chr$(5);
1335 print "              rinformacioR"
1340 print"q  a megfelelo hardverrel a commodere 64"
1360 print"q  ";chr$(18);
1365 print" automatikus digitalis voltmero lesz R";
1366 printchr$(146)
```

```
1370 print:print
1380 print"    a maximalis bemeno feszultseg "
1390 print"q    ";chr$(18);
1395 print" 220 volt egyenaram    ! "chr$(146)
1400 print"qq    ha villog a jelzes,veszelyes   "
1410 print"q    nagy bemeno feszultseg van      "
1420 print"q    a bemeno kapcsokon !!!           "
1430 print"q                                    "
1440 print:print
1455 print "    rtovabb=barmely billentyuR"
1456 printchr$(146)
1460 get a$:if a$="" then 1460
1470 goto 640
1480 rem ****kijelzo string****
1490 a=peek(243+i)
1500 if a=10 then a$="-":return
1510 if a=11 then a$="↑":return
1520 a$=str$(a)
1530 return
1540 rem **** ml program ****
1550 data 120,169,000,133,255,162,249,169
1560 data 192,141,003,221,173,001,221,168
1570 data 041,048,197,255,208,246,152,041
1580 data 015,149,000,232,165,255,024,105
1590 data 016,201,048,133,255,208,229,088,096
1600 data   0, 15
1610 data 248,  0, 24, 12,  0, 56, 14,  0,120
1620 data  15,  0,120, 15,  0,120, 15,  0,120
1630 data  15,  0,120, 15,  0,120, 15,  0,120
1640 data  15,  0,120, 15,  0,120, 15,  0,120
1650 data  15,  0,120, 15,  0,120, 15,  0,120
1660 data  15,  0,120, 15,  0,120, 15,  0, 56
1670 data  14,  0, 24, 12,  0, 15,248,  0,  0
1680 data   0,224,  0,  1,224,  0,  3,224,  0
1690 data   6,224,  0,  0,224,  0,  0,224,  0
1700 data   0,224,  0,  0,224,  0,  0,224,  0
1710 data   0,224,  0,  0,224,  0,  0,224,  0
1720 data   0,224,  0,  0,224,  0,  0,224,  0
1730 data   0,224,  0,  0,224,  0,  0,224,  0
1740 data   0,224,  0,  0,224,  0, 15,254,  0
1750 data   0, 15,248,  0, 28, 28,  0, 56, 14
1760 data   0,120, 15,  0,120, 15,  0,  0, 15
1770 data   0,  0, 30,  0,  0, 60,  0,  0,120
1780 data   0,  0,240,  0,  1,192,  0,  3,128
1790 data   0, 14,  0,  0, 28,  0,  0, 56,  0
1800 data   0,120, 12,  0,112, 30,  0,127,255
1810 data   0,126,255,  0,120, 62,  0,112, 12
1820 data   0,  0, 15,248,  0, 28, 28,  0, 56
1830 data  14,  0,120, 15,  0,120, 15,  0,  0
1840 data  15,  0,  0, 15,  0,  0, 14,  0,  0
1950 data  12,  0,  1,248,  0,  1,248,  0,  0
```

```
1860 data  12,  0,  0, 14,  0,  0, 14,  0,  0
1870 data  15,  0,  0, 15,  0,120, 15,  0,120
1880 data  15,  0, 56, 14,  0, 28, 28,  0, 15
1890 data 248,  0,  0,  0,224,  0,  1,224,  0
1900 data   3,224,  0,  7,224,  0, 14,224,  0
1910 data  28,224,  0, 56,224,  0,112,224,  0
1920 data 112,224,  0,224,224,  0,224,224,  0
1930 data 224,224,  0,255,254,  0,  0,224,  0
1940 data   0,224,  0,  0,224,  0,  0,224,  0
1950 data   0,224,  0,  0,224,  0,  0,224,  0
1960 data  15,254,  0,  0,127,255,  0,120,  0
1970 data   0,120,  0,  0,120,  0,  0,120,  0
1980 data   0,120,  0,  0,120,  0,  0,127,240
1990 data   0,127,252,  0,  0, 62,  0,  0, 30
2000 data   0,  0, 15,  0,  0, 15,  0,  0,  7
2010 data   0,  0,  7,  0,  0,  7,  0,112,  7
2020 data   0,120, 15,  0, 56, 14,  0, 28, 28
2030 data   0, 15,248,  0,  0, 15,252,  0, 28
2040 data  14,  0, 56, 15,  0,112,  0,  0,112
2050 data   0,  0,112,  0,  0,112,  0,  0,112
2060 data   0,  0,119,240,  0,124, 60,  0,124
2070 data  30,  0,120, 15,  0,112, 15,  0,112
2080 data   7,  0,112,  7,  0,112,  7,  0,112
2090 data   7,  0,120, 15,  0, 56, 14,  0, 28
2100 data  28,  0, 15,248,  0,  0,127,255,  0
2110 data 120, 15,  0,120, 15,  0,120, 15,  0
2120 data   0, 15,  0,  0, 14,  0,  0, 30,  0
2130 data   0, 60,  0,  0,120,  0,  0,112,  0
2140 data   0,224,  0,  0,224,  0,  1,192,  0
2150 data   1,192,  0,  1,192,  0,  1,192,  0
2160 data   1,192,  0,  1,192,  0,  1,192,  0
2170 data   1,192,  0,  1,192,  0,  0, 15,248
2180 data   0, 28, 28,  0, 56, 14,  0,112,  7
2190 data   0,112,  7,  0,112,  7,  0,112,  7
2200 data   0, 56, 14,  0, 28, 28,  0, 15,248
2210 data   0, 15,248,  0, 28, 28,  0, 56, 14
2220 data   0,112,  6,  0,112,  7,  0,112,  7
2230 data   0,112,  7,  0,120, 15,  0, 56, 14
2240 data   0, 28, 28,  0, 15,248,  0,  0, 15
2250 data 252,  0, 28, 14,  0, 56, 15,  0,112
2260 data   7,  0,112,  7,  0,112,  7,  0,112
2270 data   7,  0,112,  7,  0, 56,  7,  0, 28
2280 data  15,  0, 15,255,  0,  0,  7,  0,  0
2290 data   7,  0,  0,  7,  0,  0,  7,  0,  0
2300 data   7,  0,  0,  7,  0,120, 15,  0, 56
2310 data  14,  0, 28, 28,  0, 15,248,  0,  0
2320 data 92
```

8.5 Logikai analizátor

A logikai analizátorokat nagy, komplex digitális kapcsolások fejlesztéséhez és javításához használják. A készülék feladata leginkább egy többcsatornás oszcillográfhoz hasonlítható. Az ilyenek rendszerint legalább 8 adatbemenettel rendelkeznek. A bemeneteken keresztül az adatok egy meghatározott ideig az analizátorba áramlanak, ott tárolódnak és a mérés végén kiértékelhetők.



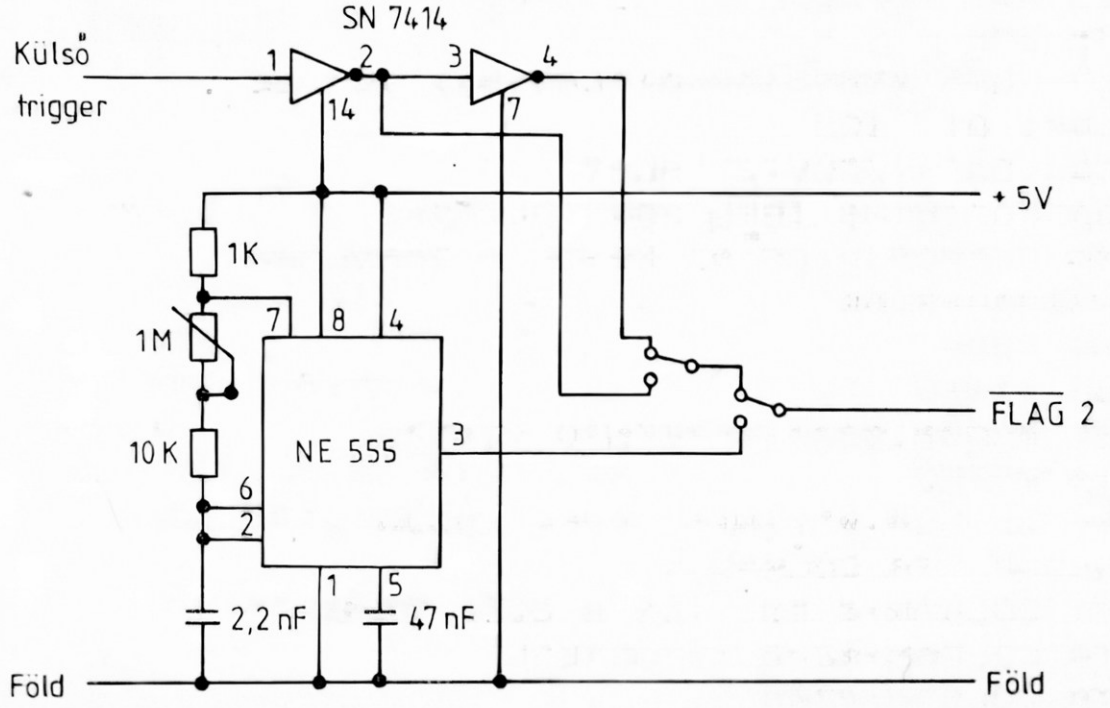
A logikai analizátor és az oszcillográf különbsége a több bemenetben, tehát az adatok tárolásában nyilvánul meg. Az oszcillográf a bemenetére érkező adatokat azonnal kijelzi, semmilyen érték nem tűnik el. Aki már megpróbálkozott egy processzor adatbuszára érkező jelek mérésével, tisztában van azzal, hogy a kiértékelés milyen nehézségekkel jár.

Sajnos logikai analizátorunk nem alkalmas arra, hogy a C 64-es adatbuszára érkező jeleket mérje.
Mivel a munka legnagyobb részét a program végzi, lényegesen kisebb munkafrekvenciát kapunk, mint pl. a C 64-es ütemfrekvenciája.
A maximális munkafrekvencia kb. 40 kHz. Ebből következik, hogy nagyobb frekvenciájú bemenő jelek esetén bizonyos információk egyszerűen elvesznek. A kapott kép, ami tulajdonképpen egy idődiagram, már nem tartalmaz minden beérkező jelet.

Ez a korlátozás azonban az analizátor hasznosságát döntően nem befolyásolja. E viszonylag alacsony frekvenciatartományban is számos feladat megoldására alkalmazható.

A sebességi szempontokat figyelembe véve a programot nagyrészt gépi kódban írtuk. Az alábbiakban közölt assembler listát megfelelő magyarázatokkal láttuk el, ezért itt a programot csak vázlatosan ismertetjük.

Maga a mérés egy cikluson belül történik. A $\overline{\text{FLAG}}$-bemenet negatív éle esetén a felhasználói porton lévő adatok a tárolóba kerülnek. A mérés ideje alatt a képernyő kikapcsolt állapotban van. Ez azért szükséges, mert enélkül a video-vezérlő minden rasztersor elején megszakítást kérne a processzortól. Egy-egy ilyen megszakítás kb. 40 µs-ig tartana, ami megközelítően duplája a legmagasabb méréstartományhoz tartozó ciklus idejének, így könnyen adatvesztéshez vezethet.



Miután a puffertárolóban 256 byte van, az összes benne lévő adatot nem célszerű egyszerre a képernyőre vinni. Ezért először csak az első 37 byte tartalma jelenik meg. Ezután a program visszatér a BASIC-be. Ahhoz, hogy a többi adatot is kijelezhessük, a program két beugrási ponttal rendelkezik (JOBB, BAL). A kiválasztott irányban a kijelzés 1 byte-tal eltolódik, így mind a 256 byte a képernyőre kerülhet és idődiagramként ábrázolható.
Kiegészítésként a hexadecimális és bináris érték, valamint a kijelzés bal szélén lévő mérési adatok képernyőkódja kerül kiírásra.

```
70 : *** P 33  ***
80 ;
100 CIA2 =$DD00
120 DDRB =CIA2+3
130 PORTB =CIA2+1
140 ICR =CIA2+13
150 VIC =$D000
160 STREG =VIC+17
```

```
170 POINTER =$F7
180 SOR =$F9
190 SZIN =1;FEHER
200 COLRAM =$D800
205 PUFFER =$C100
210 *=$C000
220 READLOOP SEI;MEGSZAKITAS BIT BEALLITAS
230 LDA STREG
240 AND #%11101111;KEP KIKAPCSOLASA
250 STA STREG
260 LDX #0;SZAMLALO NULLAZASA
270 LDA #$7F;MEGSZAKITAS TORLES
280 STA ICR
290 LOOP LDA #%00010000;FLAG-BIT ICR-BE
300 LOOP1 BIT ICR
310 BEQ LOOP1;JELVIZSGALAT
320 LDA PORTB;HA IGEN,PORT-OLVASAS
330 STA PUFFER,X;ES AZ ERTEK A TAROLOBA
340 INX;256-SZOR
350 BNE LOOP
360 LDA STREG
370 ORA #%00010000;KEPERNYO VISSZA
380 STA STREG
390 FRFILL LDA #SZIN;A FEHER KODJA
400 FRLOOP STA COLRAM,Y
410 STA COLRAM+$100,Y;ES A SZIN-RAMBA
420 STA COLRAM+$200,Y;TOLTESE
430 STA COLRAM+$300,Y
440 INY
450 BNE FRLOOP
460 LDA #<PUFFER;CIM A TAROLOBA
470 STA POINTER;A NULLAS LAPRA IRODIK
480 LDA #>PUFFER
490 STA POINTER+1
500 CLI;MEGSZAKITAS ENGEDELYEZVE
510 JMP OUT;ELSO 'IDO' KIJELZESE
520 LEFT DEC POINTER;ELUGRAS <
530 JMP OUT
540 RIGHT INC POINTER;ELUGRAS >
550 ;KIVITEL A KEPERNYORE
560 OUT LDY #36; A MENNYISEG ALLANDO
570 OUTL1 LDX #7;BITSZAM=IDODIAGRAM
580 LDA (POINTER),Y;A BYTE JELENTESEHEZ
590 OUTLOOP ROR;ISMERNI KELL A CARRY TARTALMAT
600 PHA
```

```
610 BCS BE;CARRY-TARTALALOM=BIT TARTALOM
620 LDA #100;ITT A BITTOL
630 BNE KI;FUGGOEN BEIRJUK VAGY TOROLJUK
640 BE LDA #102;A MEGFELELO RAJZJELET
650 KI PHA
660 TXA
670 PHA;A KEPERNYON LEVO JEL
680 ASL;POZICIOJANAK SZAMITASA
690 TAX
700 LDA TABL,X
710 STA SOR
720 LDA TABL+1,X
730 STA SOR+1
740 PLA
750 TAX
760 PLA
770 STA (SOR),Y
780 PLA
790 DEX ;KOVETKEZO BIT
800 BPL OUTLOOP
810 DEY;KOVETKEZO BYTE
820 BPL OUTL1
830 LDA (POINTER),Y;RELATIV CIM
840 STA 1919;BSC KIVITEL
850 PHA
860 ROR;KIVITEL HEX ERTEKBEN
870 ROR
880 ROR
890 ROR
900 AND #$0F
910 CLC
920 ADC #48
930 CMP #58
940 BCC HEXOUT
950 CLC
960 SBC #56
970 HEXOUT STA 1912
980 PLA
990 PHA
1000 AND #$0F
1010 CLC
1020 AND #48
1030 CMP #58
1040 BCC HEXOUT1
1050 CLC
1060 SBC #56
```

```
1070 HEXOUT1 STA 1913
1080 PLA
1090 LDY #7:KIVITEL BINAR ERTEKBEN
1100 BTL ROR
1110 PHA
1120 LDA #48
1130 ADC #0
1140 STA 1952,Y
1150 PLA
1160 DEY
1170 BPL BTL
1180 RTS: VISSZA A BASIC-HOZ
1190 TABL .WORD 1024+(18*40)+3;
1200 .WORD 1024+(16*40)+3;EBBEN A LISTABAN
1210 .WORD 1024+(14*40)+3;NYOLC CIM VAN
1220 .WORD 1024+(12*40)+3; AZ ELSO CIM
1230 .WORD 1024+(10*40)+3;A KEPERNYORE
1240 .WORD 1024+( 8*40)+3;KERUL
1250 .WORD 1024+( 6*40)+3
1260 .WORD 1024+( 4*40)+3
```

Ezt a programot egy kis BASIC-programba (P. 34) építettük be. A DATA – sorokban szerepel a gépi kódú rutin, amit a már ismert módon, POKE utasítással írunk be a megfelelő tárolóterületre ($C000 kezdőcímtől). Ebben a programban nincs semmi különös. Gyakorlott BASIC programozóknak semmi problémát nem okozhat.

A program kezelése rendkívül egyszerű. Elindítása után megjelenik a képernyőmaszk, majd az F1 billentyű lenyomásával elindítjuk a mérést. Miután a FLAG-bemeneten a 256 jel megjelent, a képernyőn kiírásra kerül az első 37 byte tartalma. A < és > billentyűkkel a képernyő jobbra és balra gördíthető. F1 lenyomásával új mérés indítható, F7-tel, pedig a program befejezhető.

```
1 rem *** p 34  ***
2 rem
100 poke 650,128:poke247,0
110 gosub 450
120 for i=1 to 16:cc$=cc$+chr$(29):next
130 for i=1 to 20:cb$=cb$+chr$(17):next:cb$=chr$(19)+cb$
140 lm$="  "+chr$(182):cl$=chr$(29)
150 for i=0 to 40:poke 49664+i,0:next
160 print chr$(144);chr$(147);
165 print "          logikai analizator    "
170 print:print"          max. frekvencia 45 khz"
180 if f then sys 49152
190 print lm$:print "d0";chr$(182)
200 print lm$:print "d1";chr$(182)
```

```
210 print lm$:print "d2";chr$(182)
220 print lm$:print "d3";chr$(182)
230 print lm$:print "d4";chr$(182)
240 print lm$:print "d5";chr$(182)
250 print lm$:print "d6";chr$(182)
260 print lm$:print "d7";chr$(182)
270 print:print"    ↑ rel cim..              <  balra"
280 print"                              >  jobbra"
290 print"     hex";
300 print cl$;cl$;cl$;cl$;"bsc";
310 print"           f1 uj inditas"
320 print"     bin";
330 print"                    f7 vege"
340 print cb$;
350 ad=peek(247):ad$=str$(ad)
360 ad$=chr$(5)+ad$+"    "+chr$(144)
370 print cc$;ad$
380 print "QQ"
390 get a$:ifa$="" then 390
400 if f then if a$=chr$(60) or a$=chr$(44)then
sys49223:goto 350
410 if f then if a$=chr$(62) or a$=chr$(46)then
sys49228:goto 350
420 if a$=chr$(133) then f=1:goto 150
430 if a$=chr$(136) then print chr$(147);"logikai
analizator prg vege":end
440 goto 390
450 for i=49152 to 49350
460 read x:pokei,x:s=s+x:next
470 data 120,173, 17,208, 41,239
475 data 141, 17,208,162,  0,169
480 data 127,141, 13,221,169, 16
485 data  44, 13,221,240,251,173
490 data   1,221,157,  0,193,232
495 data 208,240,173, 17,208,  9
500 data  16,141, 17,208,169,  1
505 data 160,  0,153,  0,216,153
510 data   0,217,153,  0,218,153
515 data   0,219,200,208,241,169
520 data   0,133,247,169,193,133
525 data 248, 88, 76, 78,192,198
530 data 247, 76, 78,192,230,247
535 data 160, 36,162,  7,177,247
540 data 106, 72,176,  4,169,100
545 data 208,  2,169,102, 72,138
550 data  72, 10,170,189,182,192
555 data 133,249,189,183,192,133
560 data 250,104,170,104,145,249
565 data 104,202, 16,222,136, 16
570 data 215,160,  0,177,247,141
575 data 127,  7, 72,106,106,106
```

```
580 data 106, 41, 15, 24,105, 48
585 data 201, 58,144,  3, 24,233
590 data  56,141,120,  7,104, 72
595 data  41, 15, 24,105, 48,201
600 data  58,144,  3, 24,233, 56
605 data 141,121,  7,104,160,  7
610 data 106, 72,169, 48,105,  0
615 data 153,160,  7,104,136, 16
620 data 243, 96,211,  6,131,  6
625 data  51,  6,227,  5,147,  5
630 data  67,  5,243,  4,163,  4, 59
640 if s<> 23552 then print"hiba az adatban!!":end
650 return
```

8.6 A C 64-es beszélni tanul

Számítógépünk a szövegszerkesztéstől kezdve a mérőkészülékekig már eddig is sok mindent megtanult, de ezenkívül még beszélni is megtaníthatjuk. Egy kapcsolás és egy mikrofon segítségével ez egyszerűbb, mint gondolnánk. Egy szöveget csak egyszer kell elmondanunk neki ahhoz, hogy (majdnem) pontosan elismételje.

Erről a problémáról az utóbbi időben elég gyakran hallunk. Mi is többször foglalkoztunk már a hangkeltés elvével és ismertettük az ún. hangszintetizátort is. Ez az eljárás sok érdekességgel bír, ezért most itt is foglalkozunk vele.

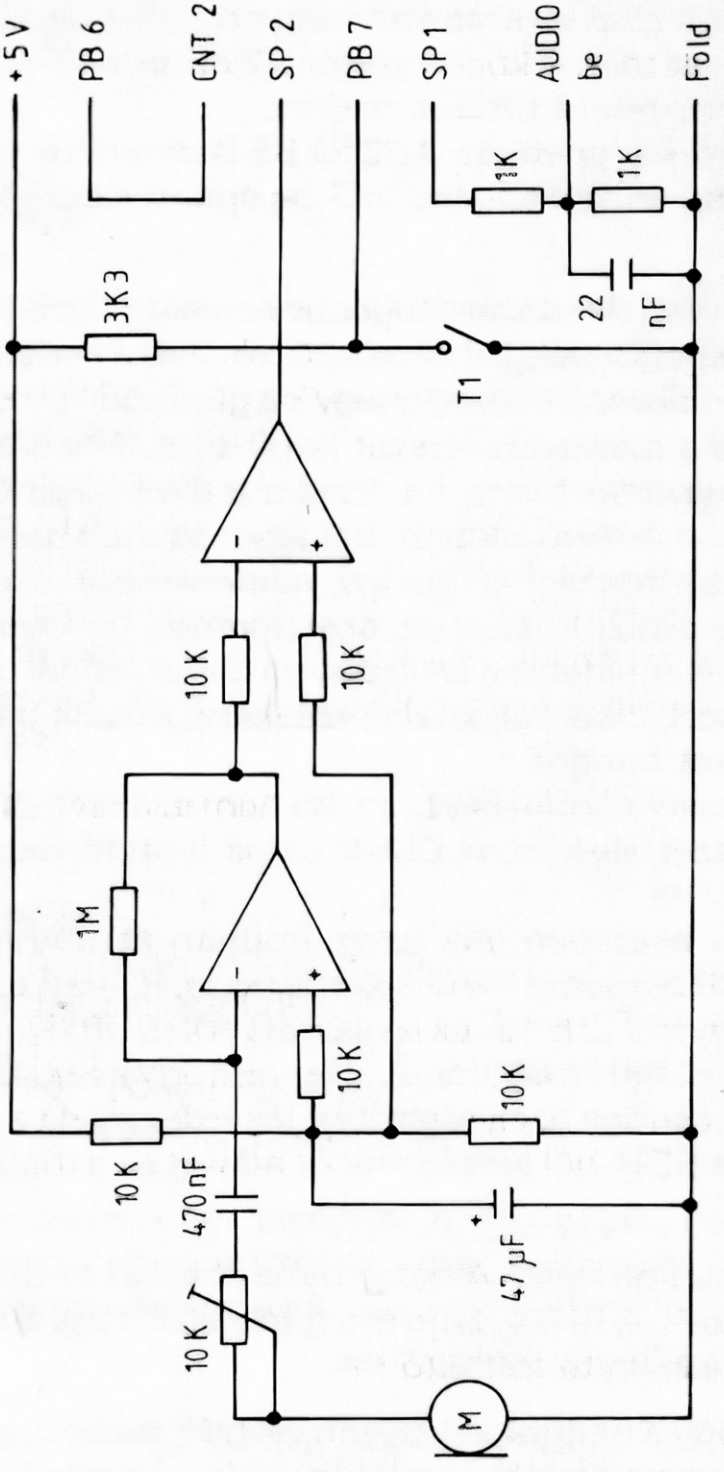
A hangszintetizálás alapja az, hogy a kimondott szó kis hangelemekre (hangkép) bontható fel. Ezek, a kémiai elemekhez hasonlóan, egy alapállományt képeznek, amelyből minden anyag, esetünkben minden szó felépíthető. A hangelemek száma több, mint ahány betű van. A jól érthető nyelv létrehozásához kb. 60 hangelem szükséges.

A félvezető ipar már gyárt olyan chipeket, amelyek az érthető nyelv minden szükséges komponensét tárolják. A hangkép kiválasztása mindössze egy byte-ot igényel, így egy kimondott szó tárolóigénye rendkívül kicsi.
Sajnos, ezek az IC-k még mindig nagyon drágák és esetenként nehezen beszerezhetők.

A hangkeltésnek egy másik, lényegesen olcsóbb módszere is létezik. Az ún. beszédszintetizátor az előzőnek kb. egy tizedébe kerül.

Az általunk alkalmazott módszer a diktafonnal hasonlítható össze. A szavakat először mikrofonba mondjuk, majd a számítógépben tároljuk. A mikrofon által küldött jelek először egy erősítőbe jutnak, ahol kiszűrhető az amplitudó és minden más információ. Végeredményként tiszta négyszögjelet kapunk, ami minden további nélkül tárolható a számítógépben. Ha a tárolt jelét egy aluláteresztő szűrőn keresztül ismét kiadjuk, jól érthető, az eredetihez nagyon hasonló hangjelzést kapunk.

Ennyit a hangképzés elméletéről. Nézzük, hogy valósul meg mindez a gyakorlatban. Először is felhívjuk a figyelmet arra, hogy az itt közölt kapcsolás és a programok a lehető legegyszerűbb változatai az adott témának. Ösztönzésül szántuk további kísérletekhez. Az érthetőségen a kapcsolás és a program továbbfejlesztésével még sokat javíthatunk.



A kapcsolásban LM324 típusú erősítőket alkalmaztunk. A mikrofontól érkező jelek az elsőre kerülnek. A 10 kΩ-os potméterrel a mikrofonnak megfelelően állítható az érzékenység. A beállítás fülre történik, tehát kísérletezni kell.
A második erősítő egy komparátor, ami az erősített jelek lehatárolását végzi. Kimenetén tiszta négyszögjeleket kapunk, amelyek már alkalmasak a C 64-esen történő további feldolgozásra.

A T1 kapcsolót felvételkor zárni kell. Az RC kör egy aluláteresztő szűrő, amelyen keresztül a számítógép által visszaadott négyszögjelet kissé ellapítjuk, így a kiadott hang kevésbé lesz torz. A kondenzátor 22 nF-os értéke csak kiinduló érték. Növelhető vagy csökkenthető tetszés szerint.
Az aluláteresztő szűrő kimenete az AUDIO BE bemenetre kerül az audio-video csatlakozón. Így a hang érthetőségét a SID-be épített szűrőkkel is befolyásolhatjuk. Kísérletezzünk!

Hogy kerülnek a négyszögjelek a számítógépbe és vissza? Erre elvileg több lehetőségünk is van. Rákapcsolhatjuk például az erősítő fokozat kimenetét a felhasználói port egyik bitjére. Ezután a felhasználói portot egy kis gépi kódú programmal folyamatosan lekérdezhetjük és a leolvasott értéket tárolhatjuk. Mivel azonban információt csak egy bit hordoz, pazarlás lenne, ha mind a 8 bitet tárolnánk.
Jobb megoldás, ha a lekérdező ciklusban 8 egymás után beolvasott bitet 1 byteban rögzítenénk. A lekérdezési ciklus így valamivel bonyolultabb és hosszabb lesz. Sajnos, éppen a ciklus hossza az, ami döntően befolyásolja a hangvisszaadás minőségét. Minél rövidebb a lekérdezési ciklus, annál többször kérdezhetjük le a felhasználói port állapotát, következésképp a tárolt bitminta annál pontosabban követi a felvett hangot.
De miért programozzunk körülményesen, ha nem szükséges. A C 64-es hardvere a munka nagy részét elvégzi. A CIA-k soros léptető regiszterei tökéletesen megfelelnek erre a célra.
A léptető regiszterek esetében írás üzemmódban az ütemjelzéstől függően a CNT bemeneten az SP bemenet kerül lekérdezésre. 8 ütem után a regiszter megtelt és a 8 bitből álló byte a CIA 1 2-es regiszteréből (SDR) kiolvasható. Eközben a 13-as (ICR) regiszter 4. bitje beállítódik, jelezve, hogy a regiszter tele van. Így elegendő a programnak csupán ezt a regisztert lekérdeznie és a 4. bitet ellenőriznie. Ha ennek értéke 1, az SDR-ből kiolvassuk az adatot és a megfelelő tárolórekeszbe töltjük.

Lejátszáskor a folyamat hasonló. A hang-byte-ot most az SDR-be írjuk be, amelyet a CIA sorossá alakít. A CIA 4. bitje most azt jelzi, hogy a regiszterből minden bitet kiolvastunk, új adatbyte tölthető be.

Miután tisztáztuk, hogy a hangokat hogyan visszük be és hogyan kapjuk vissza a C 64-estől, felmerül a megfelelő tárolótartomány kérdése.
Mivel a CIA programozása elfogadható sebességgel csak gépi kódban valósítható meg, erre a célra a BASIC–ROM, a jelgenerátor–ROM és a KERNAL alatti tel-

jes RAM területet felhasználhatjuk. Ez összesen 24 k tárterületet jelent, ami kb. 20 másodpercnyi hanganyag tárolásához elegendő.

Az általunk használt rutin felvételkor is és lejátszáskor is 160 byte hosszúságú, tehát elfér a kazetta-pufferban. A gépi kódú programok által legtöbbször igénybe vett $C000–$CFFF tartományt azonban nem használhatjuk, mert ezt a területet is a hangok tárolására tartjuk fenn.

```
60 ; *** P 35 ***
70 ;
100 *=$33C
120 PROCPORT =1
130 POINTER =$F7
140 VEGE =$F9
150 CIA1 =$DC00
160 CIA2 =$DD00
170 IDOZ1 =CIA1+4
180 IDOZ2 =CIA2+4
190 SDR1 =CIA1+12
200 SDR2 =CIA2+12
210 ICR1 =CIA1+13
220 ICR2 =CIA2+13
230 CRA1 =CIA1+14
240 CRA2 =CIA2+14
250 PORT =CIA2+1
260 DR =CIA2+3
270 IDO1 =50
280 IDO2 =100
290 KERET =$D020
300 ;KEZDODIK A FELVETEL-RUTIN
310 HEAR SEI;CIA U2 HASZNALATA
320 LDA #<IDO2;
330 STA IDOZ2;AZ IDOZITO MINT
340 LDA #>IDO2;UTEMADO DOLGOZIK
350 STA IDOZ2+1
360 LDA ICR2;MEGSZAKITASVEZERLO ELLENORZESE
370 LDA #%00110011;IDOZITO ES SDR MUKODTETES
380 STA CRA2;MEGHATAROZASA
390 LDY #0
400 LDX #$09;
410 BILL LDA PORT;FELVETEL BILLENTYU
420 BMI BILL;ELLENORZESE
430 PRELL INY;BILLENTYU LENYOMVA
440 BNE PRELL;PERGESMENTESITES
450 DEX
460 BNE PRELL
```

```
470 LDA KERET;KERETSZIN BEALLITAS
480 STA $FB;KEZDETHEZ FEHER
490 LDA 1;
500 STA KERET
510 ;ITT INDUL A FELVETEL
520 LOOP LDA PORT;BILLENTYU LE VAN-E NYOMVA
530 BMI KILEPES;HA NEM,AKKOR VEGE
540 LDA #%1000;SDR-BIT ICR-BEN
550 SDRFULL BIT ICR2;EGY BYTE KESZEN VAN-E
560 BEQ SDRFULL
570 LDX SDR2;ERTEK OLVASAS SDR-BOL
580 LDA #$30;ATKAPCSOL RAM-RA
590 STA PROCPORT;AB $A00
600 TXA
610 STA (POINTER),Y;SDR ERTEK A RAM-BA
620 LDA #$37;VISSZAKAPCSOLAS
630 STA PROCPORT;ROM-RA
640 INC POINTER;
650 BNE LOOP;CIMSZAM EMELES
660 INC POINTER+1;MAX.$0000-IG
670 BNE LOOP;
680 DEC POINTER;HA $0000,AKKOR
690 DEC POINTER+1;VISSZA $FFFF-RE
700 KILEPES LDA $FB;REGI KERETSZIN VISSZA
710 STA KERET
720 CLI;ES VISSZA BASIC-RA
730 RTS
740 ;KEZDODIK A VISSZAJATSZAS
750 TALK SEI;CIA U1
760 LDA #<IDO1;IDOZITO M1NT BAUD-RATE
770 STA IDOZ1;GENERATOR
780 LDA #>IDO1
790 STA IDOZ1+1
800 LDA #$7F;ICR MASZK TORLESE
810 STA ICR1
820 LDA ICR1;ICR TORLESE
830 LDA #%01010001;IDOZITES ES SDR INDUL
840 STA CRA1
850 LDY #0;SDR NULLAZVA
860 STY SDR1
870 TALKLOOP LDA #$30;ATKAPCSOLAS A RAM-RA
880 STA PROCPORT
890 LDA (POINTER),Y;BYTE TAROLOBOL ADAT
900 TAX
910 LDA #$37;VISSZAKAPCSOLAS
920 STA PROCPORT
```

```
930 LDA #%1000;SDR-BIT ICR-BEN
940 SDREMPTY BIT ICR1;SDR URES"?"
950 BEQ SDREMPTY
960 STX SDR1;BESZED BYTE AZ SDR-BOL
970 INC POINTER ;CIMMUTATO NOVELES
980 BNE LVEGE
990 INC POINTER+1
1000 LVEGE LDA POINTER;VEGE A LEJATSZASNAK
1010 CMP VEGE;
1020 BNE TALKLOOP
1030 LDA POINTER+1
1040 CMP VEGE+1
1050 BNE TALKLOOP
1060 JMP $FDA3;MEGSZAKITAS IDOZITESE

READY.
```

Hogy megkönnyítsük a munkát, a következőkben közlünk egy olyan BASIC-programot, amelyikkel max. 20 kis szöveg adható be. Minden szöveget egy számmal azonosítunk, amelynek alapján később visszahallgatható.
A felvétel rögtön le is játszható, ha nem sikerült, megismételhető.

A paraméterek beviteléhez, a RAM-terület kijelöléséhez és a szöveg befejezéséhez a nulláslap 247...250 címeit vesszük igénybe, a következők szerint:

1. Felvétel:

A 247/248 tárolórekeszekben alsó/felső byte alakban a következő felvétel tárolóbeli kezdőcíme található. A 247-es tárolórekesz legkisebb értéke 0, a 248-asé pedig 160 lehet. Ez hexadecimálisan $A000, vagyis decimálisan 40960. Felvétel után ez az utolsó, azaz a következő felvétel kezdőcímének felel meg.

A 249/250 címeket felvételnél nem használjuk.

2. Lejátszás:

A 247/248 címeken alsó és felső byte alakjában az első lejátszandó byte címe található. A 249/250-es tárolórekeszek az utolsóként még lejátszandó hangbyte címét tartalmazzák. A formátum az előzővel azonos.

A BASIC-program 430...450-es soraiban a SID néhány paraméterét állítjuk be. A 430-as sorban meghatározzuk a beépített szűrők frekvenciáját. A 440-es sorban a 8-as értékkel a bemenetet befolyásoljuk. A 240-es érték adott szűrőfrekvencia esetén megadja a rezonancia magasságát. A 450-es sorban határozzuk meg a hangerősséget és a szűrő típusát.
Ezekkel az adatokkal mindenképpen kísérleteznünk kell a megfelelő hangkép és érthetőség eléréséhez.

Ez a BASIC-program csak a legalapvetőbb dolgokat nyújtja ahhoz, hogy a kapcsolást kipróbálhassuk és kedvünkre kísérletezhessünk. Komolyabb programot azért nem érdemes készíteni, mert a hangképzésnek önmagában nincs értelme, csak más feladatokkal kapcsolatban. Itt megemlíthetjük pl. a játékokat, vagy esetleg egy beszélő órát... stb. Felhasználói programokban elképzelhető, hogy a számítógép nem kiírja, hanem „kimondja" az üzeneteket, udvariasan felszólítva a kezelőt arra, hogy javítsa ki a beviteli hibát vagy tegye be a lemezt, stb.

```
1 rem *** p 36  ***
2 rem
100 dim s(20)
110 s(1)=40960
120 i=1
130 sid=54272:gosub 510
140 print chr$(147):print:print
150 print"     f1 felvetel"
160 print"     f3 lejatszas"
170 print"     f7 vege      "
180 get a$:if a$="" then 180
190 if a$=chr$(133) then goto 230
200 if a$=chr$(134) then goto 350
210 if a$=chr$(136) then goto 490
220 goto 180
230 print chr$(147)
240 h=s(i)/256:poke248,h
250 l=s(i)-h*256:poke247,l
260 print"szoveg szama   #=";i;"cim=";s(i)
270 sys 828
280 h=peek(248)
290 l=peek(247)
300 s(i+1)=(h*256)+l
310 print"tarolo #";i;"cim";s(i+1)
320 input "egyforman hallhato";a$
330 if a$<>"j" then i=i+1:goto 140
340 n=i:ga=1:goto 370
350 print chr$(147)
360 input"Kerem a szoveg #-t beirni";n
370 ifs(n+1)<40960 then print"ez a # szoveg
nelkuli":er=1
380 if er=1 then er=0
385 if er=1 then er=0:input"ismetles?
";a$:ifa$<>"j"then140
390 h=s(n)/256:poke248,h
400 l=s(n)-h*256:poke 247,l
410 h=s(n+1)/256:poke250,h
420 l=s(n+1)-h*256:poke 249,l
430 poke sid+22,150
440 poke sid+23,8+240
450 poke sid+24,15+32
460 sys 917
```

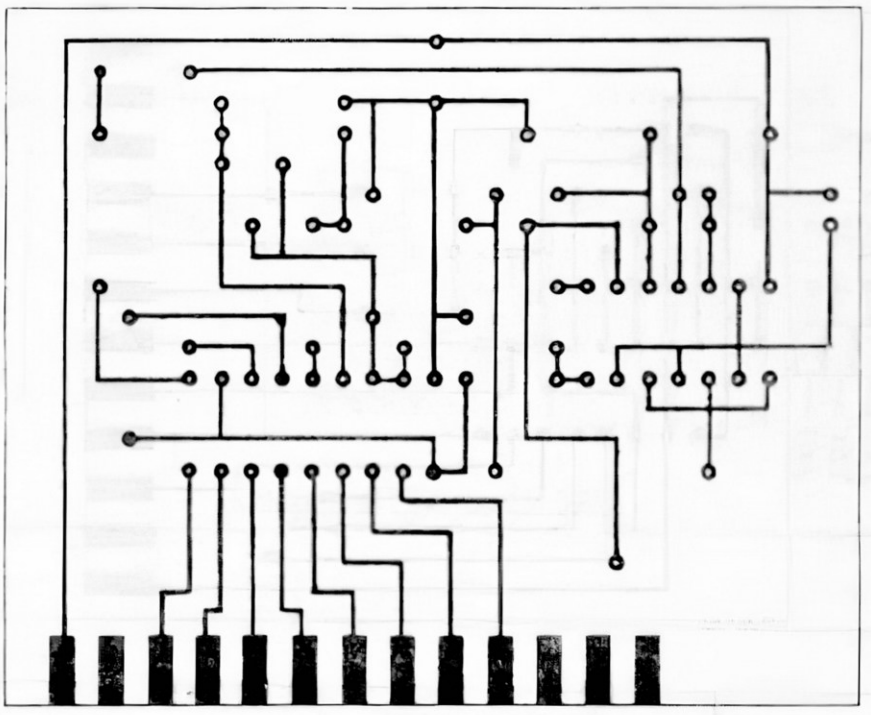
```
470 if ga=1 then ga=0:input"szoveg oK";a$
475 if ga=1 then if a$="j"theni=i+1
480 goto 140
490 print chr$(147)"vege a programnaK"
500 end
510 forii=828 to 987
520 read x:pokeii,x:s=s+x:next
530 data 120,169,100,141,  4,221,169,  0,141,  5,221,173
540 data  13,221,169, 19,141, 14
545 data 221,160,  0,162, 16,173
550 data   1,221, 48,251,200,208
555 data 253,202,208,250,173, 32
560 data 208,133,251,165,  1,141
565 data  32,208,173,  1,221, 48
570 data  33,169,  8, 44, 13,221
575 data 240,251,174, 12,221,169
580 data  48,133,  1,138,145,247
585 data 169, 55,133,  1,230,247
590 data 208,226,230,248,208,222
595 data 198,247,198,248,165,251
600 data 141, 32,208, 88, 96,120
605 data 169, 50,141,  4,220,169
610 data   0,141,  5,220,169,127
615 data 141, 13,220,173, 13,220
620 data 169, 81,141, 14,220,160
625 data   0,140, 12,220,169, 48
630 data 133,  1,177,247,170,169
635 data  55,133,  1,169,  8, 44
640 data  13,220,240,251,142, 12
645 data 220,230,247,208,  2,230
650 data 248,165,247,197,249,208
655 data 223,165,248,197,250,208
660 data 217, 76,163,253
670 if s<>23043 then print "hiba az afdatban!":end
680 return
```

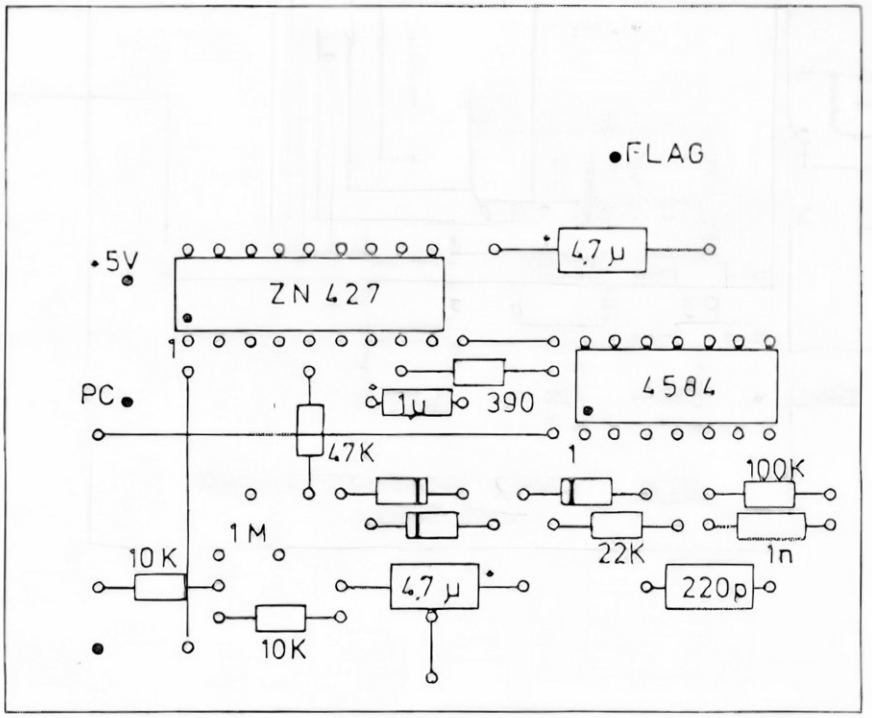
FÜGGELÉK

A könyv végén közlünk néhány NYÁK és beültetési rajzot. Azoknak, akik rendelkeznek gyártási lehetőséggel, az 1:1 arányú NYÁK rajzok megkönnyítik a könyvben lévő kapcsolások felépítését.

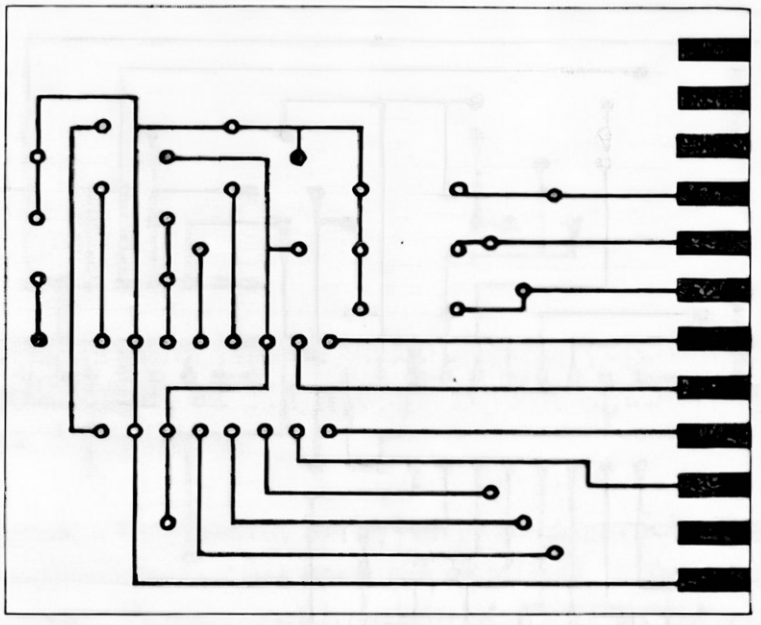
Nyomatékosan felhívjuk a figyelmet arra, hogy az áramköröket lyukraszter lemezen, WIRE–WRAP technikával célszerű elkészíteni, mert ezzel általában helyet és időt takaríthatunk meg. Ezért mondtunk le pl. az EPROM-égető NYÁK tervéről.

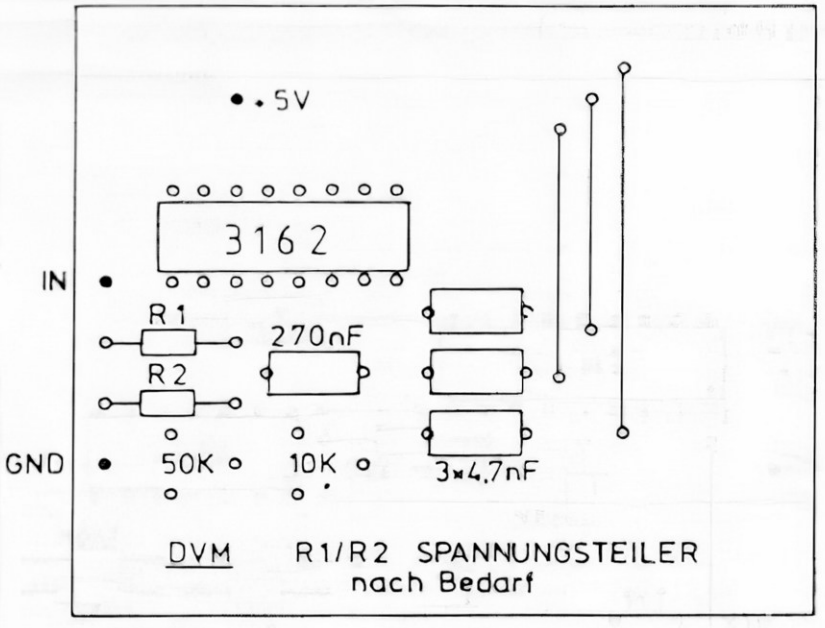
Az EPROM-panelhoz, hasonlóan az EPROM-égetőhöz, kétoldalas lemezre van szükség. A kétoldali csatlakozási pontok pontos illesztése meghaladja a legtöbb hobbi-műhely lehetőségeit.

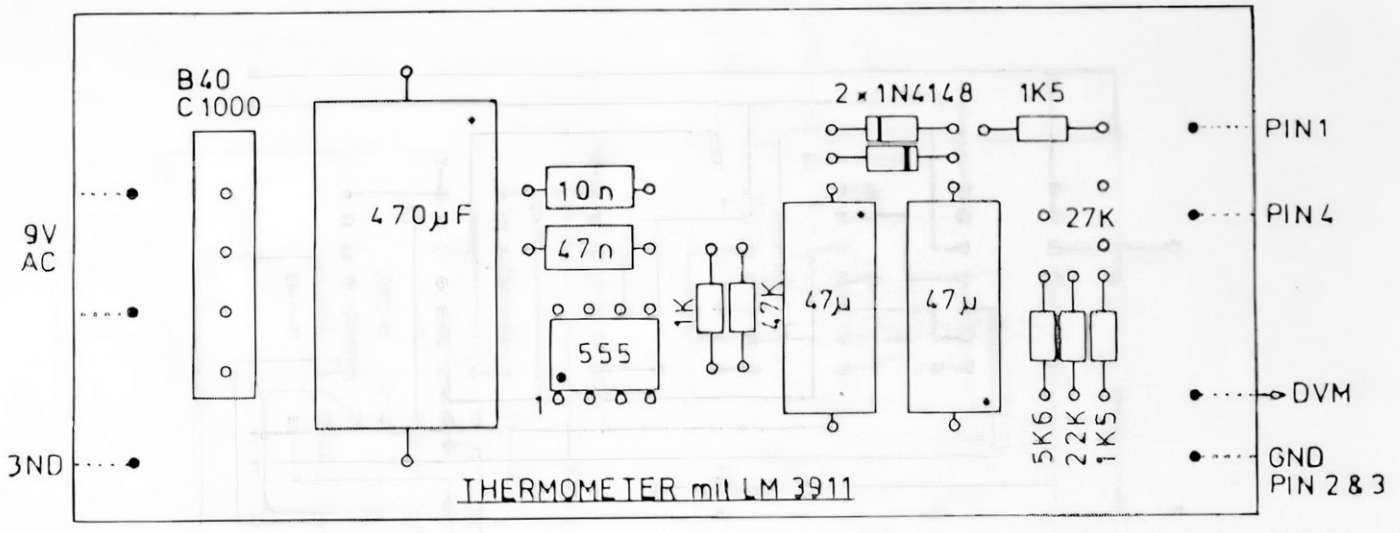


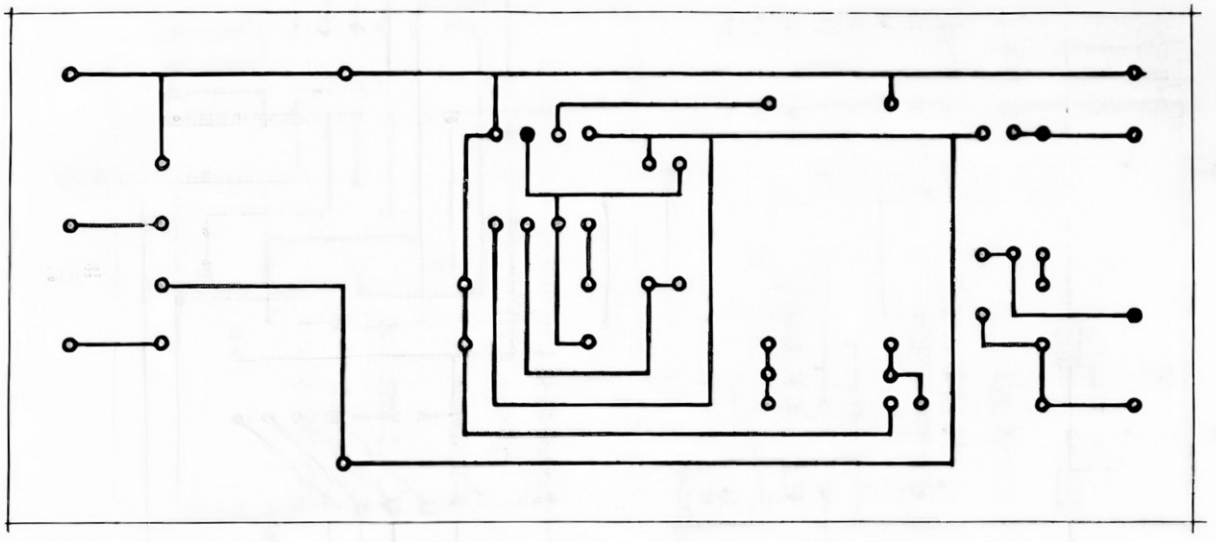

FLAG
5V
ZN 427
4.7 µ
4584
PC
1µ
390
47K
100K
1 M
10 K
22K
1n
4.7 µ
220p
10K

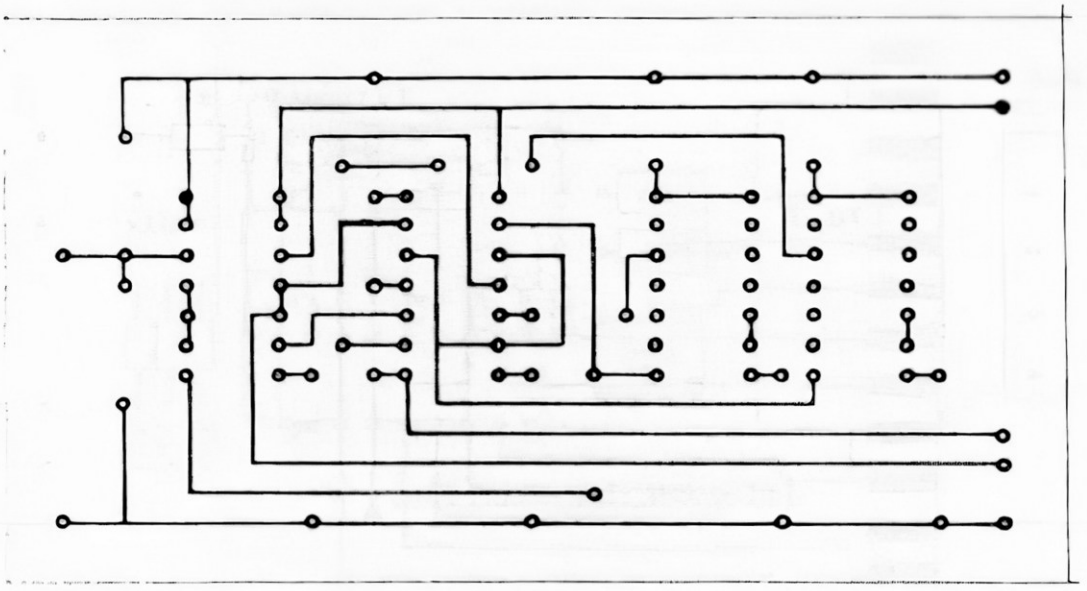
IN

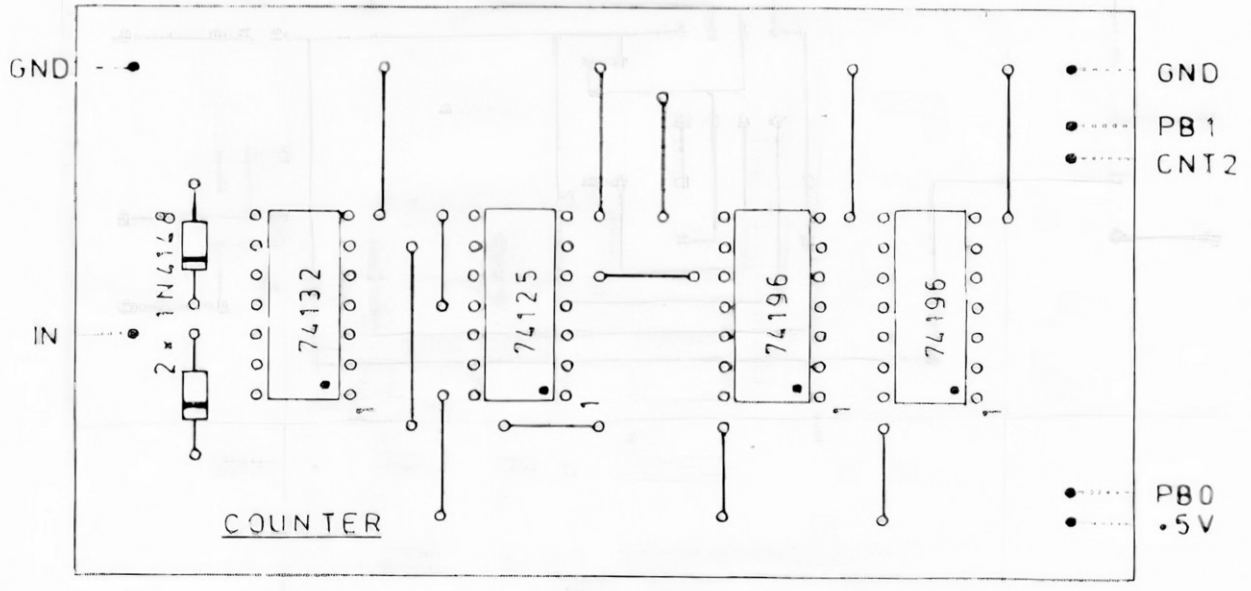


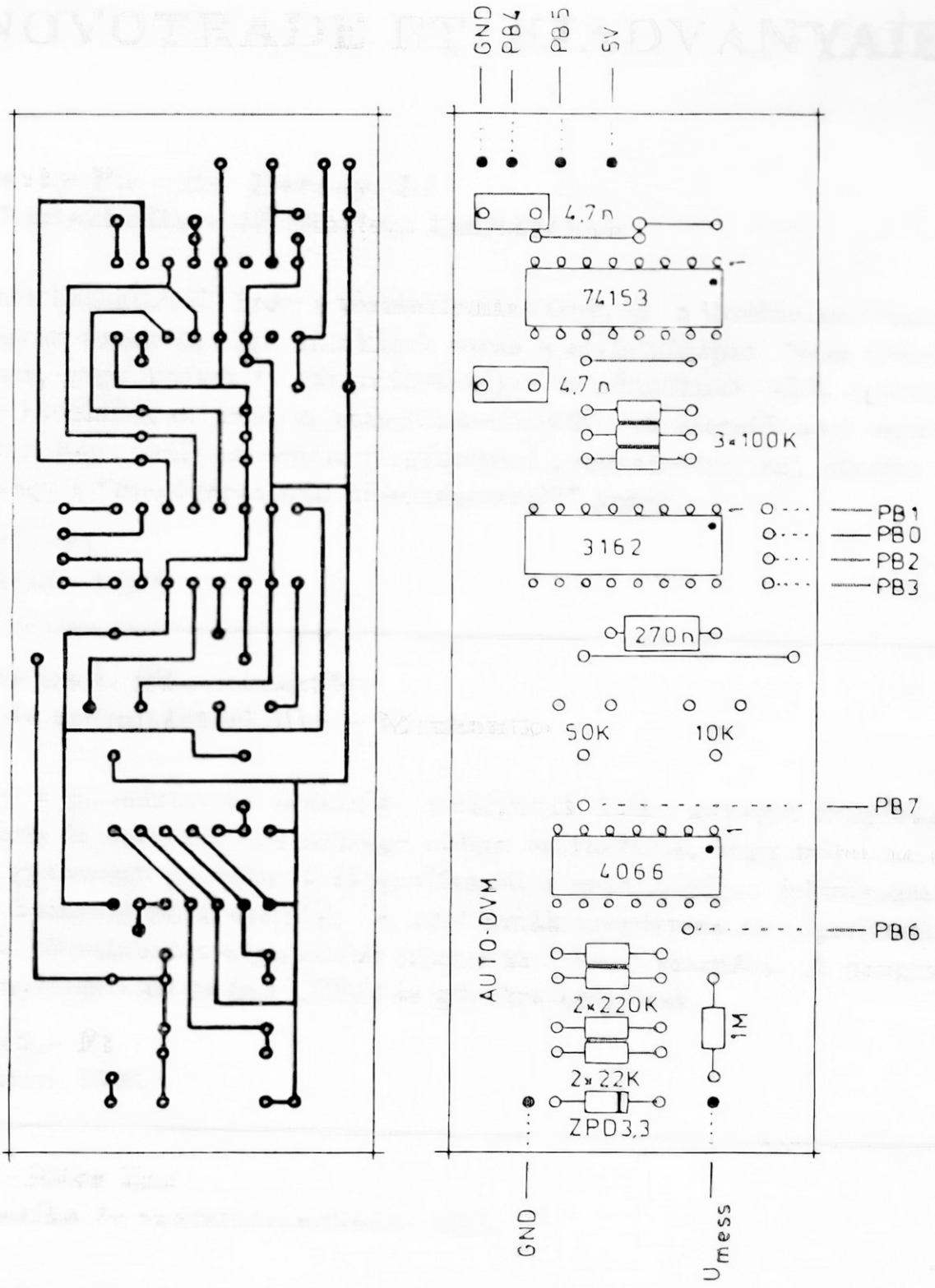

+5V
3162
IN
R1
270nF
R2
GND
50K
10K
3*4,7nF
DVM
R1/R2 SPANNUNGSTEILER
nach Bedarf


B40
C1000
470µF
10n
47n
555
1
1K
47K
47µ
47µ
2 x 1N4148
1K5
27K
5K6
22K
1K5
9V
AC
3ND
PIN1
PIN4
DVM
GND
PIN 2&3
THERMOMETER mit LM 3911






GND
IN
2 x 1N4148
74132
74125
74196
74196
GND
PB1
CNT2
PB0
+5V
COUNTER


GND
PB4
PB5
5V
4,7n
74153
4,7n
3×100K
3162
PB1
PB0
PB2
PB3
270n
50K
10K
PB7
4066
PB6
AUTO-DVM
2×220K
1M
2×22K
ZPD3,3
GND
Umess

A NOVOTRADE RT. KIADVÁNYAIBÓL

Dedinszky F.: – dr. Horányi I.:
Számítástechnika a történelem tanításában

A szerzők bemutatják, hogy a történelemtanárok, ill. a történelem iránt érdeklődők hogyan használhatják munkájuk során a számítógépet. Nem titkolt célja a könyvnek, hogy kedvet és bátorságot adjon mindazoknak, akik egyéniségüktől nagyon távolállónak érzik a számítástechnikát. A szerzők sem nyíltan, sem burkoltan nem akarnak senkit megtanítani programozni, azt akarják megmutatni, hogy a "mosógépet sem mosógépszerelő" kezeli.

Ára: 99,–Ft
Megjelent: 1987

---

dr. Kovács I. (főszerkesztő):
Fizika és számítástechnika – Mechanika

A könyv a gimnáziumok második osztályának fizika anyagát dolgozza fel. A feldolgozás és szemlélet újszerűsége abban mutatkozik, hogy mind az elméleti, mind a gyakorlati problémák tárgyalásánál a számítógépes feldolgozás lehetőségét is bemutatják a szerzők. A problémák megértése az algoritmikus gondolkodás következetes végigvitelét jelenti az Olvasó számára. A programlisták BASIC nyelven C64-es és PLUS/4-es gépekre készültek.

Ára: 149,– Ft
Megjelent: 1987

Rácz J.–Rácz Zs.:
Matematika és számítástechnika I-II.

A könyvek a gimnáziumi első és második osztályos matematika anyagot dolgozzák fel. Mind az elméleti, mind a gyakorlati rész integráltan tartalmazza a számítástechnikai megoldásokat, ill. tanítja a számítástechnikát. Messzemenő célja a szerzőknek a matematika élvezetes megtanításán túlmenően algoritmikus gondolkodás elsajátítása.
A programok Commodore 64-es és PLUS/4-es gépekre készültek.

I. kötet ára: 149,– Ft
Megjelent: 1987
II. kötet előkészületben

Rácz M.–Horváth A.:
Fizikomp

Aki megunta már a lövöldözős játékokat, szereti a számítógépet és egy kicsit a fizikát, nagyon sok hasznosat talál majd a könyvben.
A mechanika tárgyköréből vett számos probléma (versenyfeladatok) megoldásának bemutatására vállalkoznak a szerzők.
A problémák, ill. a feladatmegoldások megközelítése a számítógépes megoldáshoz juttatja el az olvasót.
A megoldásokat PLUS/4-es, C 64-es és ZX Spectrum gépekre közlik a szerzők BASIC és Pascal nyelvű programokban.
A Pascal nyelvvel egy külön fejezet foglalkozik. Az itt szerzett ismeretek elegendőek a programok megértéséhez és futtatásához.

Ára: 129,– Ft
Megjelent: 1987

Kampow:
BASIC gyakorlatok a Commodore 64-esen
(Data Becker)

Ez a kötet sokat segíthet azoknak, akik a BASIC programozást magas szinten szeretnék megtanulni és gyakorolni: számos programon keresztül vezeti rá használóját az optimális megoldások felismerésére és alkalmazására.

Ára: 300,– Ft
Megjelent: 1986

Hornig–Trapp–Weltner:
További tippek és trükkök a Commodore 64-esen
(Data Becker)

Az első nagy sikerű kötet folytatása, amely önállóan is használható. A BASIC nyelv alapszintű ismeretét feltételezi, és sok ötletet ad a programozási gyakorlathoz. Ismertet néhány lemezvédelmi eljárást, tippeket ad játékprogramok készítéséhez, megmutatja az egyes rutinok felépítését, és egy táblázatot is tartalmaz a gépi kód használatához.

Ára: 239,– Ft
Megjelent: 1986

Severin:
Tudomány és technika (Commodore 64)
(Data Becker)

E könyv azoknak készült, akik gépükkel műszaki, fizikai, matematikai, kémiai, biológiai stb. kutatásuk problémáira keresnek megoldást. Számos ilyen problémához – pl. integrálszámítás, titrálás, mátrixok, csillagászati számítások, valós gázok termodinamikája stb. – futtatható programokat is talál a könyvben.

Ára: 215,– Ft
Megjelent: 1986

Heift:
CAD – Bevezetés a számítógéppel segített műszaki tervezésbe
(Data Becker)

A Commodore 64-es lehetőségei a műszaki tervezésben is igen széles körűek. A könyv bevezeti Olvasóit a számítógéppel segített tervezés alapjaiba. Közöl egy teljes CAD (Computer Aided Design) rendszerprogramot, és néhány olyan programrészt, amely a műszaki tervezésben használható.

Ára: 290,– Ft
Megjelent: 1987

Angerhausen–Brückmann–Englisch–Gerits:
A Commodore 64-es belső felépítése
(Data Becker)

A könyv nélkülözhetetlen mindazok számára, akik a mikrogép rejtelmeinek mélyére akarnak ásni. A kötet a C 64-es minden szempontból alapos leírását adja. Teljes ROM listát tartalmaz részletes magyarázatokkal. Megismertet a BASIC interpreter, a kernal, ill. az operációs rendszer működésével. Számos példa segíti a gépi nyelvű programozás elsajátítását is.

Ára: 365,– Ft
Megjelent: 1985

Dachsel:
Zenekönyv a Commodore 64-eshez
(Data Becker)

A C 64-es mikroszámítógép kivételes zenei lehetőséggel rendelkezik! Könyvünk három része a zenei programozás, a hangképzés, végül a magasabb szintű, assembler programszerkesztés leírását tartalmazza. A függelékben a gép és a sztereoberendezések összekapcsolásához, illetve más technikai problémák megoldásához kapunk segítséget.

Ára: 323,– Ft
Megjelent: 1986

Voss:
Bevezetés a statisztikai számításokba C 64-esen
(Data Becker)

A könyv megismerteti Olvasóit a statisztikai módszerek alapelveivel, mellékelve az egyes módszerek BASIC nyelvű programját. A könyv a rendszerré összefűzött programokból egy teljes statisztikai programcsomagot is bemutat.

Ára: 348,– Ft
Megjelent: 1986

Liesert:
PEEK-ek és POKE-ok a C 64-esen
(Data Becker)

Ez az a két utasítás, amellyel szinte minden megoldható.
Az alapismeretek elsajátítása után részletesen megismerkedhetünk a finom felbontású grafika és a sprite-ok használatával, ill. még számos újdonsággal.

Ára: 120,– Ft
Megjelent: 1987

Ára: 198,– Ft

SZÁMÍTÁSTECHNIKA A KÖNYVESBOLTOKBAN

NOVOTRADE – 2 C ÁRUHÁZ
1136 Bp., Balzac u. 35. Tel.: 402-954

ÁLLAMI KÖNYVTERJESZTŐ V. – NOVOTRADE 2C

BUDAPEST

Táncsics Könyvesbolt
1073 Lenin krt. 17.
Telefon: 422–178

BUDAPEST

Műszaki Könyváruház
1061 Liszt Ferenc tér 9.
Telefon: 420-353

MŰVELT NÉP KÖNYVTERJESZTŐ V. – NOVOTRADE 2C

PÉCS

Zrínyi Miklós Könyvesbolt
7621 Jókai u. 25.
Telefon: 72-12835

VESZPRÉM

Kölcsey Ferenc
Könyvesbolt
8200 Cserhát út 7

SZEGED

Tömörkény Könyvesbolt
6720 Lenin krt. 48.
Telefon: 62-21453

DEBRECEN

Szak- és ismeretterjesztő
Könyváruház
4024 Hunyadi u. 8.
Telefon: 52-23237

BÉKÉSCSABA

Radnóti M. könyvesbolt
5600 Tanácsköztársaság
út 2.
Telefon: 25-207

SZOLNOK

Szigligeti Könyvesbolt
5000 Ságvári krt. 35.
Telefon: 56-11133

SZOMBATHELY

Savaria Könyvesbolt
9700 Mártírok tere 1.
Telefon: 94-12341

GYŐR

Pattantyús Á. Géza Szak-
könyvesbolt
9021 Molnár Ferenc u. 9.

MISKOLC

Chip-kuckó
3530 Tanácsház tér 14.

KECSKEMÉT

Művelt Nép Könyváruháza
6000 Március 15. u.
Széchenyiváros
Telefon: 06-76-28157