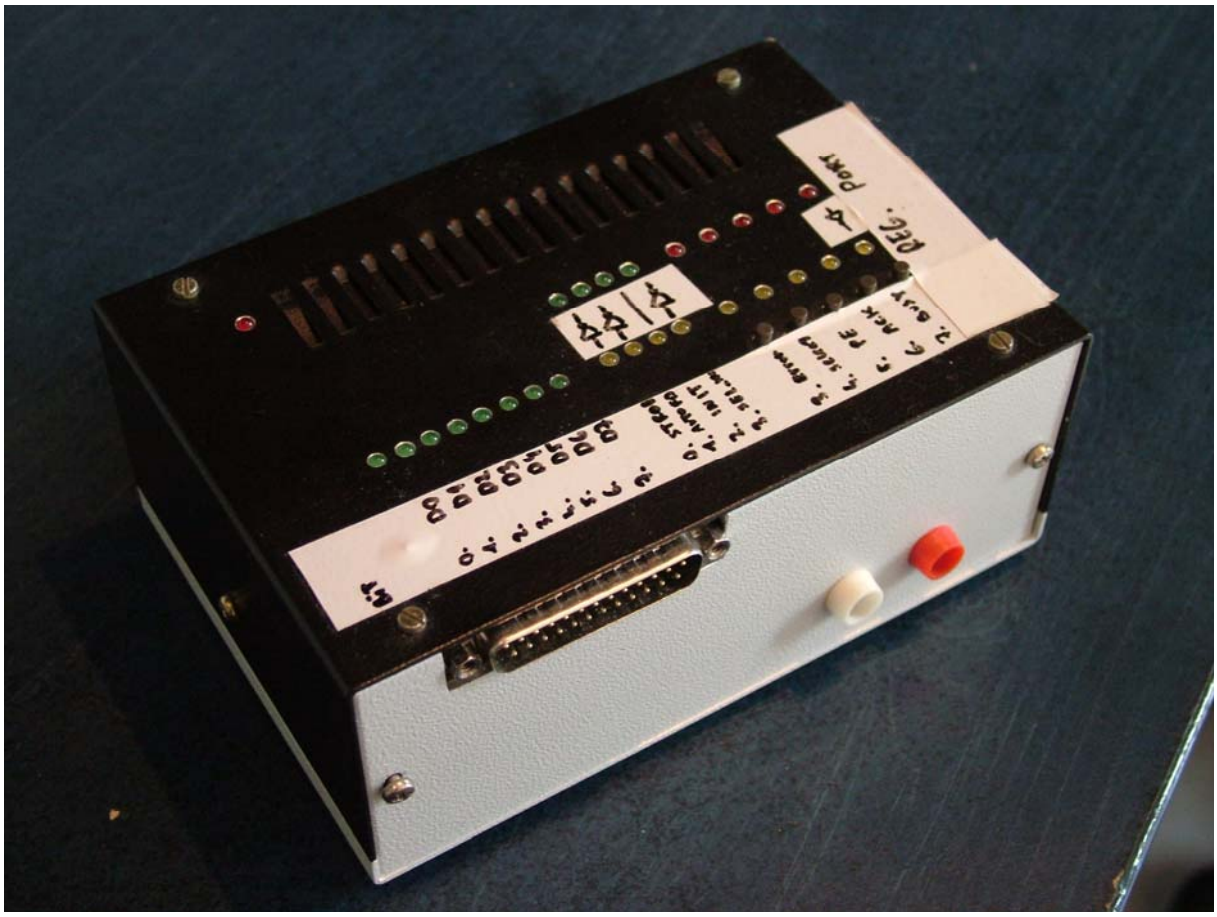
A Párhuzamos port



Készítette:

Gurbán László Sándor
Szajkó Roland

Dr. Puklus Zoltán felügyelete alatt

Mielőtt még belemerülnénk a PC párhuzamos port-jának kivesézésének, szükségesnek tartok pár alapfogalom és alapdolog megtárgyalását. Az anyag tartalmaz többek között olyan kifejezéseket is mint pl.: TTL (tranzisztor tranzisztor logika), regiszterek, báziscím(ek), bit stb. Ebben a kis bevezetőben elsősorban ezen alapdolgokat szeretném leírni nagy vonalakban, hogy mindenki lássa, és legyen elképzelése róla hogy miről is beszélünk.

TTL alapok

Szerintem jelen esetben a legfontosabb dolog amivel első körben meg kell ismerkedünk az a TTL másnéven *Tranzisztor Tranzisztor Logika* (Transistor Transistor Logic). Mint köztudott, a számítástechnika alapja a kettes számrendszer, tehát az értékek lehetséges halmaza a 0 és 1 számokból áll. Mivel ezzel a két értékkel kell mindent megoldanunk ezért nem árt átnézni néhány fontosabb alapfogalmat.
A digitális készülékek első pillanatban viszonylag bonyolultnak tűnhetnek, pedig alig néhány logikai alapkapcsolás többszöri felhasználásával egyszerűen kialakíthatóak. A digitális technika egyik legszükségesebb eleme az ún. Boole-algebra melyet esetenként kapcsolásalgebrának is szoktak hívni. Nézzük ennek alapjait.
A hagyományos algebrával ellentétben a logikai változók csak két diszkrét értéket vehetnek fel, melyeket általában L-nek (logikai 0) illetve H-nak (logikai 1) szokás nevezni. Bármi ami digitális technikával illetve a számítástechnikával kapcsolatos, mind ezt az algebrát illetve ennek szabályait alkalmazza. Például a mai asztali gépek processzorai bizonyos tekintetben „buták”, ugyanis csak két számot ismernek, a 0-t és az 1-et, viszont egyáltalán nem nevezhetők butának olyan tekintetben, hogy mindössze ennek a két számnak a felhasználásával (0:nincs áram, 1:van áram) szinte bármilyen számítási műveletet meg tudnak csinálni.

A Boole algebra három fő műveletet ismer:

Konjunkció (másnéven ÉS művelet):

A	B	A ÉS B
0	0	0
0	1	0
1	0	0
1	1	1

Diszjunkció (másnéven VAGY művelet):

A	B	A VAGY B
0	0	0
0	1	1
1	0	1
1	1	1

Negálás:

A	NEG A
0	1
1	0

Aztán léteznek még ezek kombinációi is. Például: NAND (nem és), NOR (nem vagy), XOR (kizáró vagy)

A kis matematikai kitérő után, lássuk hogy mit is takar az a TTL fogalom.

TTL JELSZINT

Az előbbiekben említett 1-es és 0-s értékek megjelenítését a 60-as években a TEXAS cég által létrehozott TTL rendszer (Tranzisztor Tranzisztor Logika) működési okokból a 0 és az 5 V-hoz kötötte. Mivel ezek az értékek nem tarthatók tökéletesen, ezért az értékek pontos megjelenése a következő:

Boole érték	Határ:	Feszültség:
0	Alsó	0
	Felső	0,4-0,5 nagyonmax 1
1	Alsó	2,4
	Felső	5,0

Láthatjuk a táblázatból, hogy a rendszer 0 és 1 értéke között nagy eltérés van, ez az úgyevezett tiltott sáv. Elméletileg ebben a tartományban nem lehetnek a jelek, de ha rövid

időre jelentkezik egy bemeneten akkor bizonytalan értékűvé válik a kapcsolás és előre nem meghatározott értéket vesz fel a kimenet. Természetesen ez is0 vagy 1 lesz.
A TTL-nél a nulla értéket L-el jelölik, utalva a LOW, tehát alacsony jelszintre, az 1-et pedig H-val, ami a HIGH (magas) rövidítése. A továbbiakban mi is így fogjuk őket jelölni.

Ha a TTL logikát és a Boole algebrát összekapcsoljuk, akkor egy nagyon egyszerűen tervezhető rendszert kapunk, mert csak az egyszerű alapszabályokat kell követnünk. Tehát ha azt mondjuk, hogy egy 2 bemenetű ÉS kapcsolásra van szükségünk, akkor azt a következőképpen írhatjuk fel:

A	B	Q	TTL
0	0	0	0V
0	1	0	0V
1	0	0	0V
1	1	1	1V

A táblázatból látszik, hogy egy olyan kapcsolást kell építenünk, amelynél csak akkor lesz a kimenet (Q) értéke 5 V-os értéken, ha az A és B bemeneten is logikai 1, tehát 5 V érték van.
Természetesen ki kell emelnünk, hogy az eltelt évek folyamán a technológia fejlesztéseknek köszönhetően már korántsem szükséges ekkora feszültségszint a működéshez, de az ipari szabvánnyá vált TTL értékei a bemeneteken és kimenteken állandóak maradtak, míg a belső működés más feszültségeken zajlik, így energiatakarékosabb rendszerek építhetők.

A TTL eszközök kivitelezése ún. bipoláris tranzisztorokkal történik. Ezek főbb jellemzője hogy jelszintekkel dolgozik. Mit jelent ez?!A TTL eszközök **0V-5V**-os intervallumban üzemelnek. Ha a TTL eszközünkbe bemenő vagy abból kijövő feszültség 0.8V~1V körüli érték akkor az logikai 0-nak (Low szintnek) felel meg, ha 2,4V-nál magasabb a feszültség akkor az logikai 1-esnek (High szintnek) felel meg.
Nézzük a következő TTL-es **NAND** (nem és) kapcsolást: Melléklet!
Ha minden bemenet H állapotú, akkor az R1-en átfolyó áram a bemeneti tranzisztor kinyitott bázis-kollektor diódáján át folyik R2 bázisára és azt kinyitja. Ha akár csak egy bemenetre is alacsony feszültségszintet adunk, akkor a hozzá tartozó bázis-emitter dióda kinyit, és T2 bázisárama megszakad. T2 lezár és a kimenet H szintre kerül.

REGISZTEREK

A regiszterek gyors írható - olvasható munkatárak. A különböző regiszterek szigorúan meghatározott feladatokhoz vannak hozzárendelve, emiatt korlátozott funkció betöltésére alkalmasak. A belső sínrendszeren keresztül tartanak kapcsolatot a processzor más részeivel. Felépítésük: A regiszterek felépítése lehet statikus memóriaelemek (pl.: flip - flop áramkörök) valamilyen rendszere vagy egy RAM memória része, ami lehet dinamikus vagy statikus. Néhány mikroprocesszor típusnál egyetlen chipben mind a két megoldást alkalmazzák úgy, hogy pl. az akkumulátort statikus memóriaelemekből állítják elő, míg az összes többi regisztert egy általános, dinamikus RAM-ba lehet kombinálni.

A regiszterek a belső sínrendszeren keresztül tartanak kapcsolatot a processzor más részeivel .

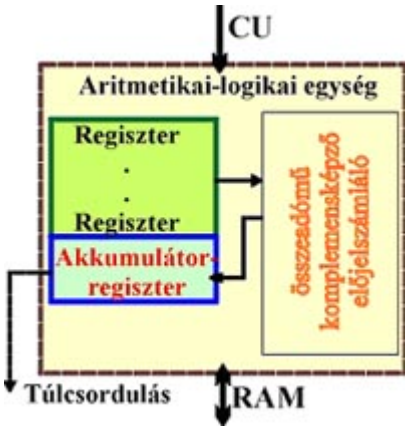
A regiszterek gyors működésű tárak, amelyek hossza általában az adatsín szélességével egyezik meg (1-17 byte).

A regiszterek a processzor legkülönbözőbb részeiben (ALU (aritmetikai logikai egység), CU (központi egység)) vagy önállóan tömbökbe szervezetten fordulhat elő. A regiszterkészlet processzor függő.

A regiszterek egy része a felhasználó által közvetlenül hozzáférhető, míg egy másik része a felhasználó által közvetlenül hozzá nem férhető (csak a processzor használja).

Regiszterek fajtái:

Akkumulátor regiszter:



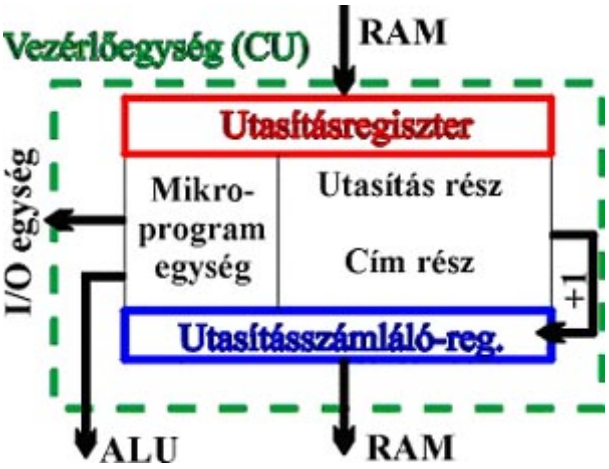
Az akkumulátor az aritmetikai és logikai műveletek operandusait, vagyis a műveletek tárgyát képező mennyiségeket vagy azoknak az eredményeit tárolja. A közbenső, részeredmények tárolására is alkalmas és minden műveletben részt vesz. Az ALU az összes aritmetikai műveletet a regiszterekben tárolt két bináris szám ill. azok komplemenseinek összeadására vezeti vissza. A korszerű számítógépekben az akkumulátor helyett már egy vagy több regisztertömb van, amelyben akár 512 regiszter is elhelyezkedhet. Így csökkenthető a tárhozfordulások száma, illetve növelhető a végrehajtás sebessége.

Adatszámláló regiszter:

Az adatok kiolvasásakor vagy beírásakor azonosított memóriarekesz címét tárolja. Mérete függ a mikroprocesszor által címezhető memóriakapacitástól. Egyszerre több is lehet belőle a CPU-ban.

Utasításregiszter (IR):

A vezérlő egységhez tartozó regiszter, amelyben a memóriából lehozott utasítás tárolódik, amíg a CU az utasítás műveleti jelrésze alapján meghatározza az elvégzendő műveletet és elindítja a mikroprogramot. A pipelining feldolgozásra alkalmas processzorok esetében az IR ebben a formában már nem létezik.



Utasításszámláló regiszter (PC v. IP):

A soron következő utasítás címét tárolja. Az utasításszámláló tartalmát a program maga is változtathatja. Fontos különbség az utasításszámláló és az adatszámláló regiszter között, hogy az utasításszámláló regiszternél a problémamegoldás az utasításkódok címeinek sorrendjében megy végbe. Vagyis az utasításszámláló által címzett első memóriarekesz elérésekor kihozunk a memóriából egy utasításkódot, így az utasításszámláló tartalma eggyel nő és így a memória következő rekeszét címezi, ahol a program szerint a következő utasításkód található. Az adatszámláló regiszter csak akkor fut végig az adatcímeken, ha az adatok sokszavas egységben vagy táblázatban vannak tárolva.

Bázis(cím)regiszter (BR):

Az operandusok címzéséhez felhasznált regiszter, amely nem általános használatú. A báziscím egy alapcím, amelyhez viszonyítva adhatjuk meg az utasításban az operandus helyét. Nem minden processzornál használják.

Indexregiszterek:

Szintén nem minden processzorban találhatók és ezek is az operandusok címzését segítik elő, különösen adatsorok feldolgozásánál.

Állapotregiszterek, vezérlő regiszterek:

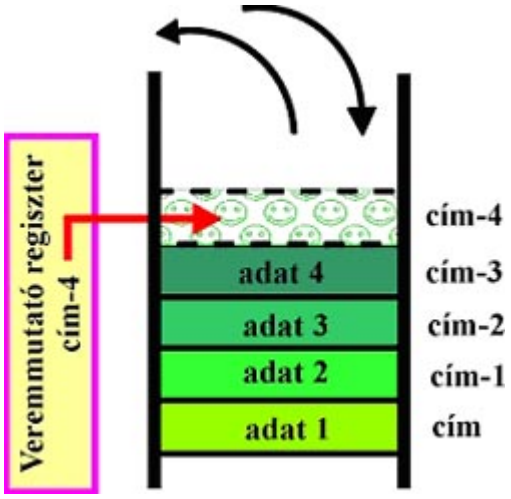
Egy vagy több regiszteren belül tárolnak vezérlő és ellenőrző jeleket. Az állapotregiszter az ALU műveleti eredményeit jellemzi, a műveletek végrehajtásának eredménye alapján

bekövetkező állapotot tükrözi vissza. Pl. az eredmény 0 volta, vagy ha túlcsordulás keletkezik.

Jellegzetes állapotbitek:

- előjelbit - S (sign)
- nulla bit - Z
- túlcsordulásbit - O (overflow)
- átvitelbit - C (carry)
- félbyte - átvitelbit - H
- megszakításbit - I
- paritásbit - P
- stb.

Veremmutató regiszter (SP):



A verem legfelső elemét jelöli ki. A veremtároló egy speciális tároló, amely elsősorban az alprogramok kezelését segíti. A verem nem része a belső regisztereknek, általában a főtárolóban kerül kialakításra. Szervezése LIFO (Last in first out) jellegű, ami azt jelenti, hogy az utoljára bekerült adat vehető ki először, és amit legelőször tettünk be, azt vehetjük ki utoljára (Több szintű verem létezhet, több SP is lehet). A "verem instrukciók" (PUSH, POP) automatikusan hivatkoznak az SP-re és automatikusan állítják

Már tudjuk mi is az a TTL, tudjuk mik a regiszterek, báziscím, mostmár belefoghatunk a párhuzamos port felépítésének, programozásának ismertetésébe.

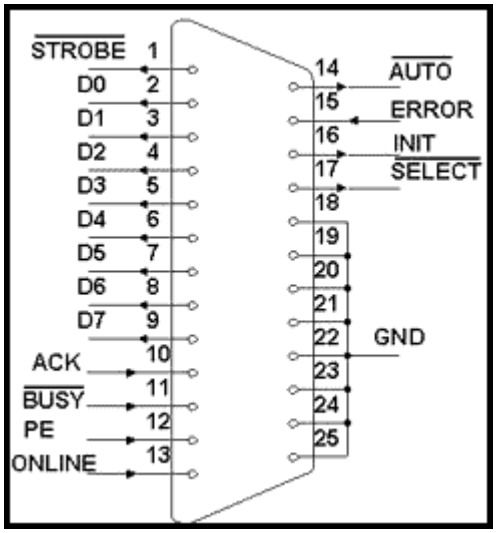
A PÁRHUZAMOS PORT

Nézzük meg először is, hogy a számítógépünkhöz milyen interfészeken keresztül illeszthetünk különböző eszközöket:

- **Soros port**: Eléggé mostohán kezelt, de igen hasznos port. Általában a régebbi egereket, egyelőre még a modemeket, mobiltelefonokat kötjük rá, esetleg gépek összekötésére, mi több hálózatba kacsolására is alkalmazható. Számos egyéb területen hasznos, mint pl. a párszáz forintból házilag megépíthető mikrovezérlő és **EEPROM**-programozók illesztése. Egy gépben egyszerre alapesetben 4db lehet, de trükközéssel 8-at is lehet üzemeltetni. Számunkra egyetlen hátránya, hogy +/-12V-os, jelszintekkel dolgozik, amit csak több alkatrésszel tudunk illeszteni. Másik hátránya, hogy közvetlen vezérlésnél (logikai áramkörök, mikrovezérlők alkalmazása nélkül) csupán a modemvezérlő bitek állnak rendelkezésre, ami kevés felületet, játékteret ad.
- **USB**: Nagysebességű soros elérésű port. Előnye, hogy a csatlakozóhoz ki van vezetve a szükséges tápfeszültség, így akár tápáram forrása is lehet. Sajnos speciális áramkörök nélkül (mikrovezérlők, USB-I2C adapter) nehéz boldogulni vele.
- **IRDA**: Infraport néven sokkal ismertebb. Gyakorlatilag ez is egy soros megoldás, viszont ami nagy előnye, hogy vezeték nélküli kapcsolatot tud biztosítani a vevő és az adó között.
- **GAME** port: a régebbi játékvezérlőket erre az interfészre lehet(ett) rákötni. Előnye szintén hogy 5V tápfeszültség kinyerhető belőle.
- **Billentyűzet port**: Azt hiszem sok magyarázást nem igényel a dolog.
- **Printer port**: gyakorlatilag ez az anyag erről a témáról fog szólni a továbbiakban.

A nyomtató portból (a későbbiekben csak LPT az egyszerűség miatt) egy számítógépben alapesetben 4db lehet, de trükközéssel szintén elérhető nagyobb szám. Hátránya a többivel szemben hogy ebből nem nyerhető ki 12V de még 5V tápfeszültség sem, így ha erre a portra építünk valamit, akkor a tápfeszt bizony máshonnan kell beszereznünk.
Az LPT portot eredetileg egyetlen nyomtató elérésére/vezérlésére szánták, ezért kialakításából adódóan sem volt képes kétirányú kommunikációra. Gyakorlatilag annyiból állt az egész, hogy a számítógép kiküldte az adatokat a portra aztán a nyomtató vagy kezdett velük valamit vagy nem, és erről a számítógép semmi visszajelzést nem kapott.
Ezt a Centronics standardot illetve ennek továbbfejlesztett változatát az IEEE1284-es szabványként vezette be az Institute of Electrical and Electronik Engineers szabványügyi

hivatal. Ebben a szabványban rögzítettek szerint a port már kétirányú interfészként működött. Az LPT port a fentebb leirt TTL jelszintekkel dolgozik, igy digitális kezelése nem túl bonyolult. Most nézzük meg hogyan is néz ki egy ilyen port.



Így néz ki a portunk kívülről. Most nézzük meg kicsit közelebbről, hogy az egyes feliratok, lábak mit is jelentenek. A képen azok a lábak vannak L szinttel, melyek feliratán vonal van.

Láb	Megnevezés	Irány (PC felől)	Alapértelmezett jelszint (H:high;L:low)
1	Strobe	O	L
2	D0	i/o	H
3	D1	i/o	H
4	D2	i/o	H
5	D3	i/o	H
6	D4	i/o	H
7	D5	i/o	H
8	D6	i/o	H
9	D7	i/o	H
10	ACK	I	L
11	BUSY	I	H
12	PAPER OUT	I	H
13	SELECT	I	H
14	AUTO FEED	O	L
15	ERROR	I	L
16	RESET,INIT	O	L
17	SELECT IN	O	L
18	EXT	I	-
19-25	GND	-	-

1. Vezérlőjel adatvételhez. Adatok átvétele a nyomtatótól egy LOW impulzussal történik.
2. 2-9. Adatok küldése printerre vagy más eszközre. 8 adatbit áll rendelkezésre, tehát gyakorlatilag bármilyen információ kiküldhető a portra direktben.
10. Kézfogással nyugtáható ha az adatok vétele megtörtént/sikeres volt. Ha ezt megkapta a számítógép akkor küldheti a többi adatot.
11. Ha az eszköz még az előzőleg elküldött adatok feldolgozásával foglalkozik, akkor ezt a BUSY jellel közli a számítógéppel. Ha H akkor leáll az átvitel míg L nem lesz a jelszint. Ha L-re változik akkor akkor ACK H impulzussal tudatja az eszköz a számítógéppel, hogy megérkeztek az adatok, küldhető a többi.
12. Nyomtatóból kifogyott a papír. A nyomtatók többsége átvált off-line módba, a SELECT jelet pedig L szintre állítja. Ha a hiba a nyomtatóban van, akkor az ERORR lábat is L-re húzza.
13. Közli a számítógéppel, hogy a nyomtató éppen milyen állapotban van. Ha H akkor on-line és képes adatokat fogadni. A nyomtatón lévő kapcsoló közvetlenül befolyásolja.
14. Ha fogadja a 0Dh (CR: kocsivissza) jelet, akkor automatikusan váltson sort (line feed).
15. Ha hiba van a nyomtatónál akkor L lesz. FAULT-nak is szokás hívni.
16. Alaphelyzetbe állítja a nyomtatót.
17. Nyomtató leválasztása.

Fentebb említettem hogy egy számítógépen belül több párhuzamos port is lehet. Nos a különböző párhuzamos portokhoz különböző báziscímek tartoznak, amik alapján el lehet őket érni közvetlenül a memóriában.

Port	**Báziscím**
1.	378h
2.	278h
X	3bch

Báziscím	0-7 bit	Adatregiszter
Báziscím+1	Állapotregiszter	
	7. bit	BUSY 0
	6. bit	ACK 0
	5. bit	Paper Out 1
	4. bit	Select 0
	3. bit	ERORR 0
	2-0. bitek	Fenntartott
Báziscím+2	Vezérlőregiszter	
	7-5. bitek	Fenntartott
	4. bit	Megszakításkérés
	3. bit	Select In 1
	2. bit	Reset 0
	1. bit	Auto Feed 1
	0. bit	Strobe 0

Mindezek ismeretében gyakorlatilag mindent tudunk ahhoz, hogy építsünk akármilyen eszközt a portunkra, illetve azon keresztül vezéreljük számítógépünket. Nézzünk egy egyszerű és gyors megoldást a sikerélményhez. Ehhez szükségünk lesz 1db LED-re (akármilyen színű lehet), valamint egy körülbelül 150-200Ohm-os ellenállásra is. Az ellenállás azért kell, mert a diódánk feszültségvezérelt, tehát minél több feszültséget kap, annál jobban világít, ha túl sok feszültség jut a diódára, akkor egyszerűen elfüstöl vagy elpukkan. Az ellenállást a ledünkel sorba kötve a feszültségosztó alapján kevesebb feszültség esik rá, így védjük alkatrészünket is. Bár egy darab led ára párszor 10 forint, de azért vigyázni kell az alkatrészeinkre. Szóval van sorba kötve egy ledünk és egy ellenállásunk. A kapcsolás egyik végét nemes egyszerűséggel dugjuk a párhuzamos portunk 2-es számú csatlakozójába, a másikat pedig kössük földre (19-25-ös csatlakozók). Ha ezzel megvagyunk nincs más hátra mint kipróbálni. Ez azt jelenti, hogy programozni fogjuk a portot, méghozzá úgy hogy a legelső adatbitre küldünk jelet, ezáltal a 2-es lábra kötött led felgyullad és világítani fog. Mivel 8 adatbitünk van, ami ugye 1 egész byte, ezért először megnézzük ennek felépítését.

Bitek:	8	7	6	5	4	3	2	1
2 hatványai:	7	6	5	4	3	2	1	0

Ha például az első adatbitre (2-es láb) szeretnénk küldeni, akkor 2^0 azaz 1-et küldünk a portra. Ha a másodikra (3-as láb), akkor 2^1 azaz 2-t kell küldenünk. Ha egyszerre többet akkor az egyes értékeket összeadva kapjuk meg azt az értéket amit ki szeretnénk irni a portra. Ha például az első, második és negyedik adatbitet szeretnénk aktívvá tenni, akkor $1*2^0 +1*2^1+0*2^2+1*2^3 =1+2+8=11$, a többi érték természetesen nulla, ugyanúgy mint a 3-ik tag. Mivel kijelentettük, hogy a regiszter aktuális értékével megegyezik a port lábán a jelszint, így láthatjuk azt, hogy a 11 kiíratása a D0, D1, D3 lábakon fog jelet eredményezni. Most akkor ott tartunk hogy van egy azaz 1 darab ledünk a port D0-ás lábára kötve. Villantsuk fel a következő módok valamelyikén. A legegyszerűbb és leghatásosabb, illetve minden rendszeren megtalálható az ún. DEBUG parancs. Parancssorból indítsuk el. Adjuk ki a következő két parancsot (feltéve hogy portunk a 378h címen van):

- O 378 01 (ekkor felgyullad a fényünk)
- O 378 00 (kialszik)

Portunk vezérlését gyakorlatilag bármelyik programozási nyelvből megtehetjük, legyen az BASIC, PASCAL, ASSEMBLY (ez az egyik leghardverközelibb nyelv), C, C++ és ezeknek objektumorientált változatai (Visual BASIC, Visual C++, stb)

A C nyelv parancsszavai (a port kezeléséhez szükséges a DOS.H headerfájl behúzása a programunkba):

- port olvasása: adat=inportb(portcim); /*portcim: alapesetben 378h*/

- port írása: outportb(portcim, adat); /*portcim: mint fejlebb, adat: amit ki szeretnénk irni a portra*/

Mivel a Pascal-nál fejlettebb nyelvek szintaktikája és megértése bonyolultabb, ezért maradunk a Pascal-os szintaktikánál.

- írás: PORTW[portcim]:=adat

- olvasás: status:=PORT[portcim]

Az egy ledes vezérlésre a programrészletünk a következőképpen néz ki:

```
Portw[$378]:=1; {ez a D0-ás bitet teszi aktívvá}

Portw[$378]:=0; {lenullázza az összes adatbitet}
```

A kicsivel bonyolultabb megoldást az 1. számú melléklet tartalmazza. A bal oldalon található a port csatlakozója, melyen fel vannak tüntetve a kivezetések sorszámai, igy a fenti táblázat alapján meg lehet állapítani, hogy melyik milyen feladatot lát el a kapcsolásban. Van benne egy eddig számunkra ismeretlen elem. Ez a 74LS244-es IC. Miért is kell ez nekünk?
A legfontosabb dolgunk mindig az legyen, ha külső perifériát vagy eszközt tervezünk a számítógépünkhöz, hogy megfelelő biztonsággal rendelkezzen, és az első adandó alkalommal ne tegye tönkre gépünket. Különböző leválasztási, védelmi stratégiák léteznek számítógépünk portjának, portjainak védelmére. Ezek közül talán a legegyszerűbb és legismertebb az ún. optocsatolós megoldás. Ennek lényege hogy van egy led diódánk, amit „szembeteszünk” egy fényérzékelő diódával. A dolog úgy működik, hogy amikor a dióda fényt bocsájt ki magából, akkor a fényérzékelő dióda ad, tehát olyan mintha vezetékkel fizikálisan össze lenne kötve egymással a kettő, de nincs szükség a vezetékre. Ezzel biztosíthatjuk azt, hogy ha esetleg rosszul terveztük, vagy raktuk össze a kapcsolásunkat, akkor a számítógépet semmilyen káros visszacsatolás (zárlat, viszáram) nem érheti. Ezeket a védelmeket manapság már tokozva megkaphatjuk pár száz forintért. Egy ilyen meghajtó, védelmi IC ez az LS244-es. A mellékletben látható a felépítése, lábkiosztása, működési elve. Gyakorlatilag a kapcsolás csak egyetlen mozzanatban tér el az előbbi, 1 ledes, megoldástól. Mivel beraktuk ezt az IC-t, ennek működéséhez szükségünk van külső +5V tápfeszültségre a működtetéshez. A dolog ugyanúgy működik mint a fenti példánál, de itt az LPT port adatregiszterébe beirt adatok, értékek már

egy az egyben kiolvashatóak a ledek segítségével. Nézzük egy másik megközelítésből hátha az érthetőbb. Van 8 ledünk, ami a port 8 adatbitjének felel meg. A számítástechnikában ez 256 különböző jel, érték tárolására alkalmas egység. Mondjuk legyen az a feladat, hogy egy adott 0-255 intervallumba eső számnak adjuk meg a bináris, azaz 2-es számrendszerbeli alakját. Normál esetben ehhez szükségünk van némi matematikai ismeretre. Egy ilyen eszközzel számolás nélkül bármelyik az adott intervallumba eső szám bináris alakját megadhatjuk, ha egyszerűen csak fogjuk és az adott számot kiírjuk a port adatregiszterébe. Ugyanis az egyes ledek biteket jelölnek, mely bitek értéke akkor 1, ha a hozzátartozó led világít. A következő programrészlet bekér egy számot, majd azt kiírja a port adatregiszterébe.

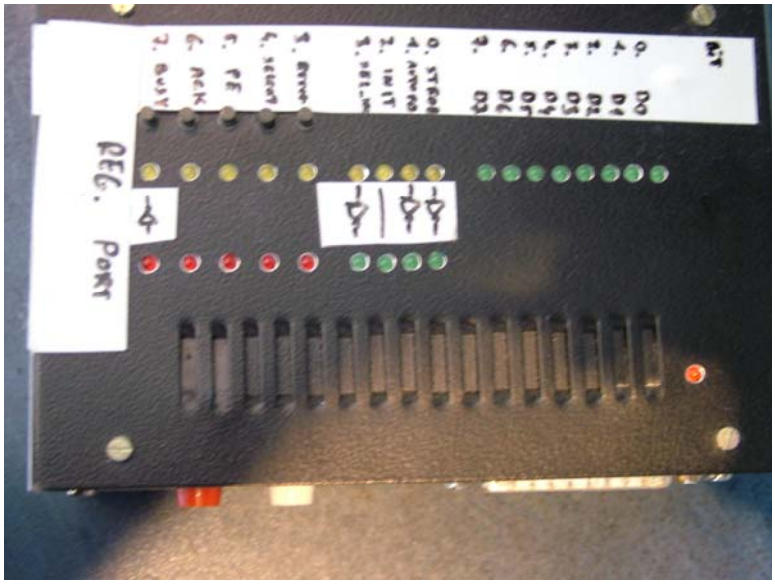
```
Program binaris;
uses crt;
const lpt=$0378;
var szam:byte;
begin
        readln(szam);
        portw[lpt]:=szam;
end.
```

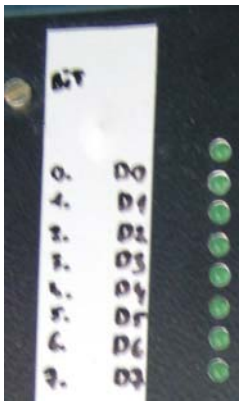
A 2-es számú mellékletben látható kapcsolás körülbelül fele ugyanaz mint a fenti. A másik felében a port bemeneteit tettük felhasználhatóvá, és ezáltal vezérelni tudjuk gépünket. A vezérlést kapcsolók segítségével valósíthatjuk meg. A kapcsolókat, mivel ezek a számítógép felől nézve bemenetnek számítanak, mindenképpen valamelyik bemeneti lábhoz kell hozzárendelnünk. Ezeket a bemeneteket az állapotregiszter biztosítja számunkra, melyek a 14, 10, 11, 12, 13-as (ERROR, ACK, PE, SELECT, BUSY) lábakon találhatóak meg.

Említést érdemel még a 3. mellékletben szereplő kapcsolás is. A címből és a kapcsolásból is kiderülhet, hogy itt bizony valami nem stimmel. Normál esetben van 8 darab adatbitünk amik kimenetként szolgálnak, itt azonban mégis 24 kimenetre tettünk szert. A mágikus szó amit erre használunk az a MULTIPLEXÁLÁS és DEMULTIPLEXÁLÁS. Ez azt jelenti, hogy különböző jeleket bocsátunk ki egyidőben az átviteli közegre, úgy hogy az egyes információkat később vissza tudjuk nyerni. A demultiplexálás ennek a folyamatnak a fordítottja, tehát a már nyalábolt adatok különválasztása. Ezt a dolgot a kapcsolásban úgy oldották meg, hogy beraktak egy 74LS374-es IC-t, amely tokozva tartalmaz mindent amire csak szükség lehet. Láthatjuk hogy a D0-D7-es lábakhoz ugyanazon vezetékek futnak, viszont az IC-k CLK lábához rendre a STROBE, AUTO FEED, SELECT IN vezetékek futnak. Ezek a vezérlőregiszterben található egyes bitek. Erről az IC-ről tudni kell, hogy alapesetben akkor aktív, tehát akkor ad, amikor a CLK lábán 0-ról 1-re történik változás. Ezeket figyelembe

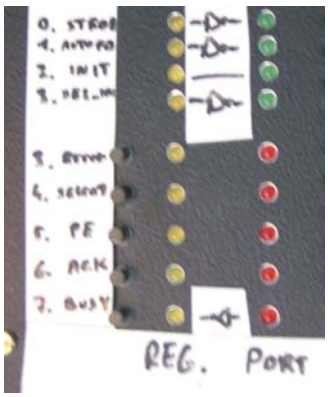
véve kijelenthetjük, hogy ezzel az egyszerű trükkel, külön tudunk vezérelni 3x8 azaz 24 darab különálló kimenetet, mindössze annyival, hogy megadjuk, hogy éppen melyik IC-nk adja ki a jeleket.

Hogy bonyolítsuk az életünket, Sütő István kollégának hála, van egy ún. diákbiztos eszközünk, ami így néz ki:





Ezen a részen található az a 8 bizonyos led, amik az adatregiszter tartalmát reprezentálják. Példaprogramunkban ezt a részt fogjuk vezérelni, demonstrálva a lehetőségeket.



A 0-3 jelölt sárga illetve zöld színű ledek a vezérlőregiszter bitjeit jelképezik. A dokumentum elején lévő táblázatban ugye ez a báziscím+2, ahol a 4,5,6,7-es bitek fenntartott bitek, azokat vezérlésre nem tudjuk használni.

A 3-7-ig terjedő részen, ahol a sárga illetve piros ledek láthatóak, ezek az állapotregiszter egyes bitjeit jelölik. Az állapotregisztert a báziscím+1 címen lehet elérni és ugye jól kivehető hogy a 0,1,2-es bitek fenntartott, illetve nem használt bitek, ezért az eszközön nincsenek is feltüntetve. Eszközünk diákbiztos tápcsatlakozóval rendelkezik, melynek

lényege, hogy üzemeltethető mind 5V-os mind 12V-os tápfeszültséggel. A 4-es számú melléklet tartalmaz egy programot két részletben. Az első a főprogram, a második pedig egy a párhuzamos porton való tevékenykedésünket segítő függvényeket és eljárásokat tartalmazó Pascal unit. Illetve van még egy Alap nevű unitunk is ami a szükséges típusokat és konstansokat tartalmazza.

A program ismertetését először az Alap unittal kezdeném.

Az érdemi rész a 6. sorral kezdődik. Itt létrehozunk egy egydimenziós tömbtípust, mely 8 elemű, és 1-8-ig számozzuk. A 7. sortól kezdődően különböző konstans, azaz a program futása során állandó, értékeket határozunk meg, hogy ezzel is könnyítsük a munkát. A legelsővel megadjuk az első alapértelmezett párhuzamos port báziscímét, ami szinte mindig $0378 lesz. Van egy Kesleltetes nevű dolgunk is, ami majd arra fog kelleni, hogy egyáltalán lássuk, hogy mit csinál az eszközünk. A LED nevű konstansunk egy 8 elemű tömb, ami egymásután tartalmazza 2-nek 0-7-ig a hatványait. Ez azért lesz segítségünkre, mert egyetlen hivatkozással fogunk tudni rámutatni a 8 közül tetszőleges ledre. Pl. ha kell nekünk a 3. led, akkor csak annyit írunk a programba hogy led[3], és máris a 3. leddel tudunk dolgozni.

Most térjünk át az LPT_UNIT nevű unitunkra

A programunk érdemi része a 19. sornál kezdődik. Ebben a szakaszban irtam egy NULLAZ nevű eljárást, ami nemes egyszerűséggel lenullázza a porthoz tartozó összes regisztert, amit a 22. sorban lévő ciklusunk tesz meg. És itt láthatjuk is azt az értékadási formát amit már a legelején említettem: **portw[lpt]:=0**. LPT helyére természetesen bekerül az a szám, amit az ALAP unitban megadtunk. Tehát sehol nem fog LED világítani az eszközön.

A 25. sorban kezdődik egy leptetes nevezetű eljárásunk, ami azt csinálja, hogy 1 ledes futófényt generál az eszközön, és a ’+’ ’-’ jelekkel befolyásolhatjuk a futófény sebességét. A késleltetésre pont emiatt van szükségünk, hisz ha nem adunk meg ilyet, akkor a másodperc tört része alatt lefut a folyamat, és mi nem látunk belőle semmit. A 32 és 46. sorok között zajlik a kiírás a portra, illetve a kilépő billentyű figyelése. A 33. sorban kiírjuk az aktuális értéket a portra, tehát az 1. led fog világítani, majd az ii értékét növelve folyamatosan haladunk odébb és odébb a ledsoron. A leptetes_vissza eljárás ugyanezt a technikát követi, csak itt ellenkező irányban történik a mozgás.

A 73. sortól kezdve érdekessé válik a dolog, ugyanis itt már bevetjük az ún. maszkolás trükkjét is. Az előző részben látható volt, hogy a ciklusunk addig futott, míg a ’Q’ billentyűt meg nem nyomta valaki. Itt a ciklusból való kilépési feltétel:

until ((port[lpt+1] and led[4])=led[4]) or (ki='q');

Itt láthatunk egy szép ÉS kapcsolatot két csúnyán kinéző érték között. Maszkolással tudjuk vizsgálni például azt, hogy adott bit változott-e. Ez ugye a nyomógombnál azt jelenti hogy megnyomták-e. Ha megnyomták akkor a megnyomás idejére 1-es impulzust kapott a számítógép portja.

	7	**6?**	**5**	**4**	**3**	**2**	**1**	**0**
AND	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
=	**0**	**1/0**	**0**	**0**	**0**	**0**	**0**	**0**

A táblázat legfelső sora az egyes biteket jelzi. A 6-os melletti kérdőjel azt jelenti, hogy arra vagyunk kíváncsiak, hogy egyes-e. A második sorban található az a feltétel, ami meghatározza, hogy melyik bit bekapcsolt állapotára vagyunk kíváncsiak. Jelen esetben ez a 6-os bit. A 3. sorban lévő bitek jelentik az ÉS művelet eredményét. Mint látható a 6-os bit attól függően lesz 1 vagy 0, hogy milyen feltétel volt, és hogy teljesült-e vagy nem.

Van még egy művelet, amit szintén sokszor lehet alkalmazni, ez a bitek be és kikapcsolását segíti elő:

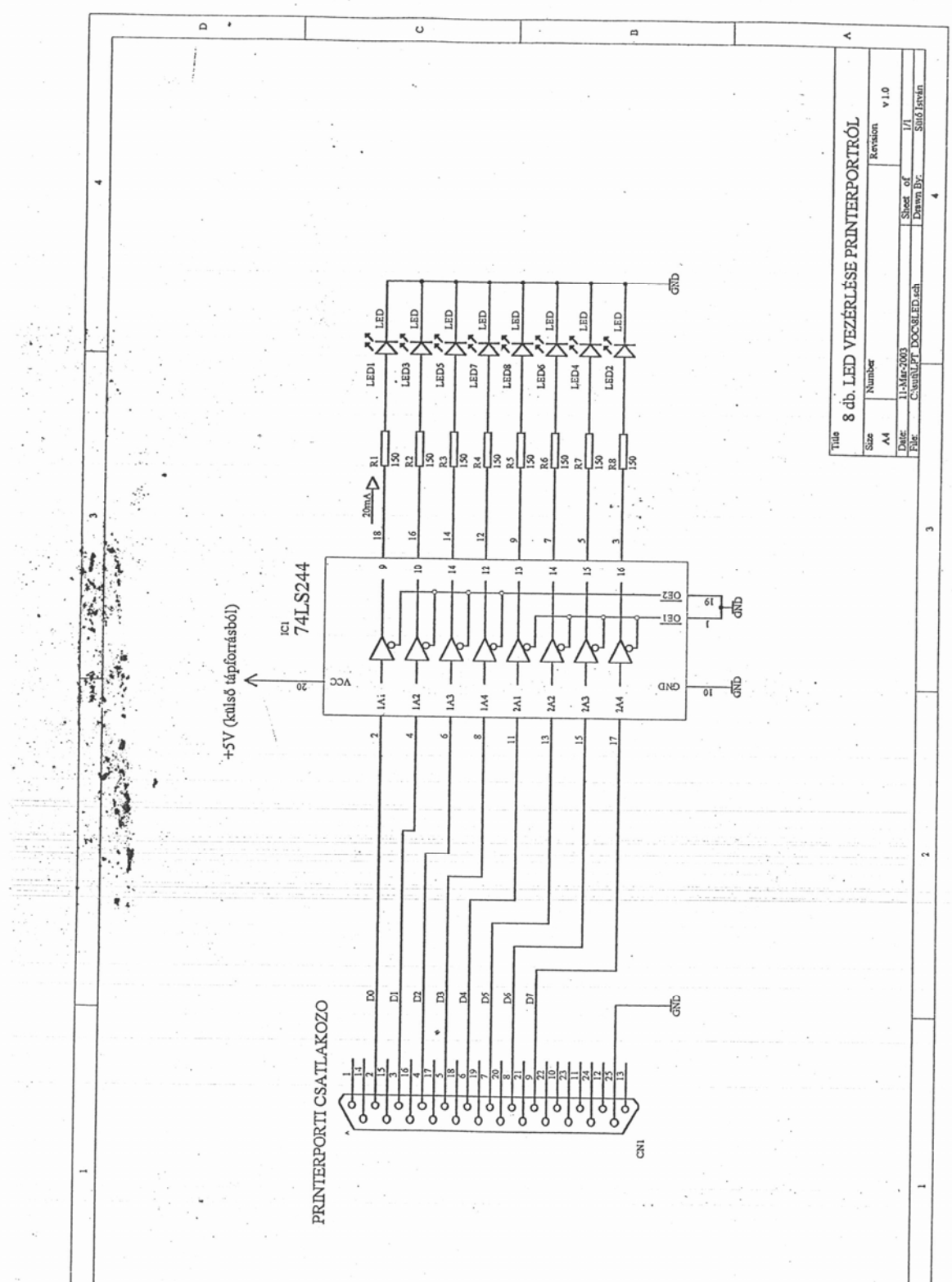
	X	**X**	**X**	**X**	**X**	**X**	**X**	**X**
OR	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
=	**X**	**X**	**X**	**1**	**X**	**X**	**X**	**1**

Az X-ek azt hivatottak jelölni, hogy teljesen mindegy, mi áll azokon a helyeken, 0 vagy 1-es. Tehát ahol 0 van a feltételben, az nem változik, ahol 1-es, az változik 1-esre. Így bekapcsolhatóak a bitek.

A 121. sortól kezdődően ezeknek a maszkolásoknak a használatával kommunikálunk a portunkkal, illetve az eszközünkkel.

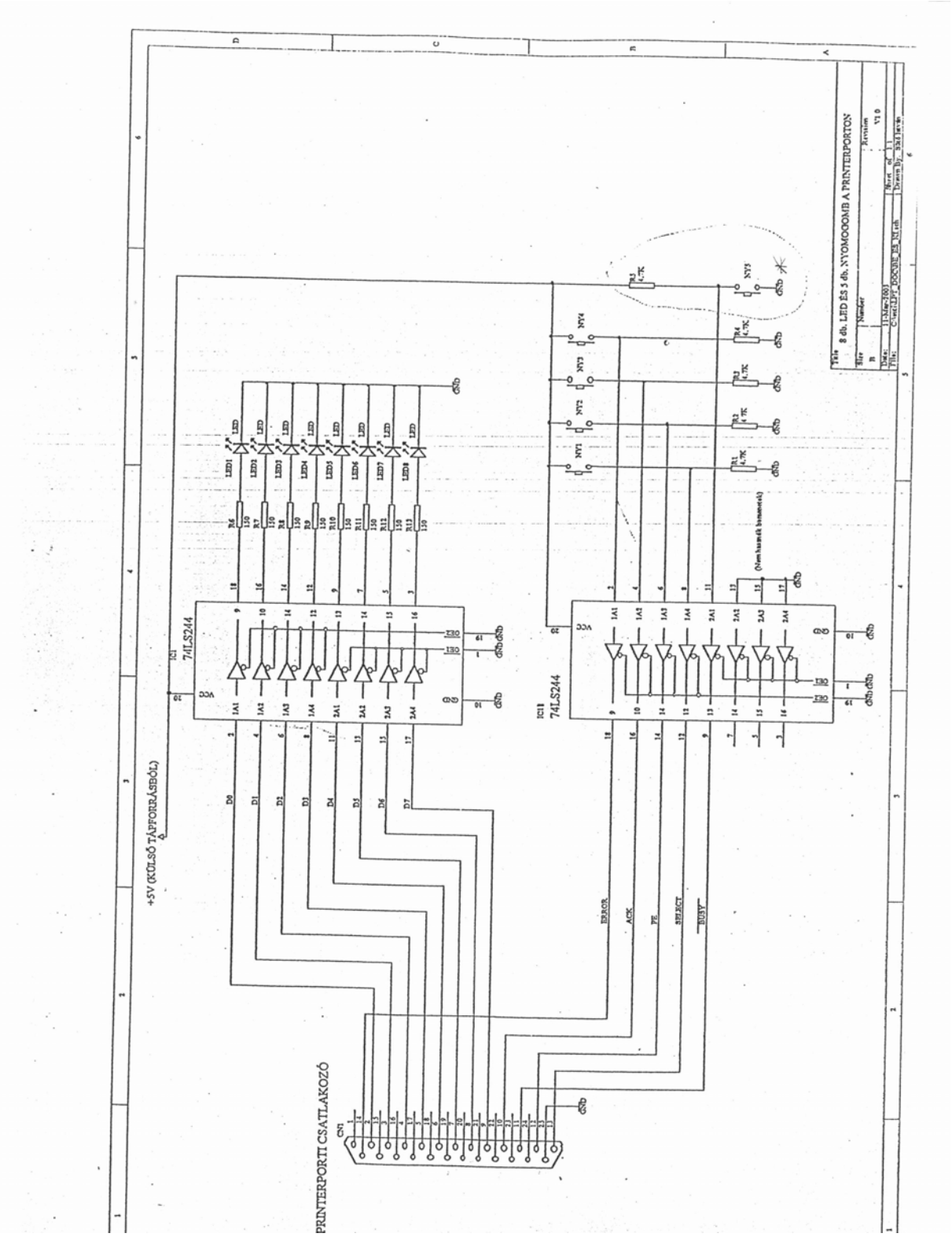
MELLÉKLETEK

1.melléklet



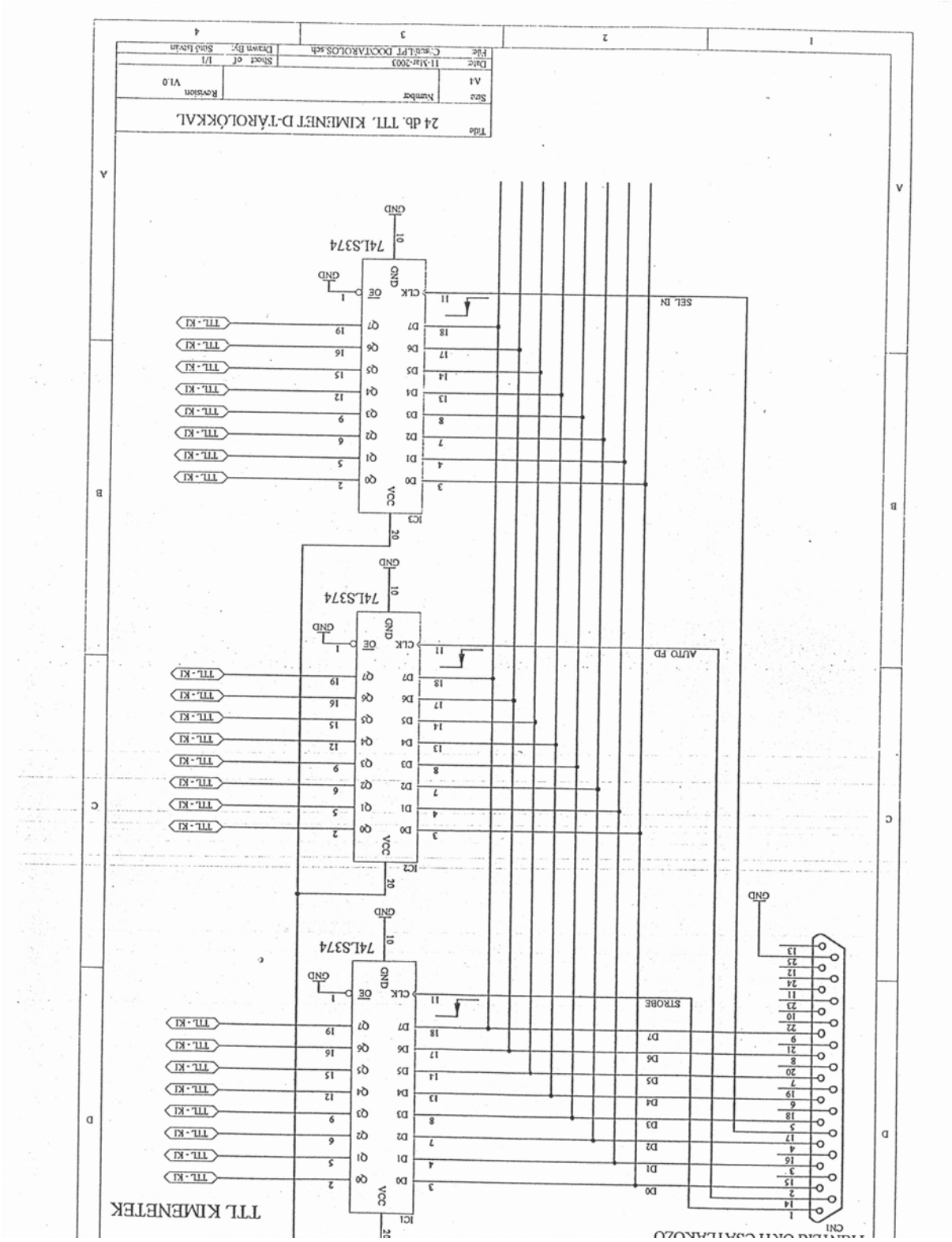
A rajzokat és az eszközt Sütő András kollégának köszönhetjük

2. melléklet



A rajzokat és az eszközt Sütő András kollégának köszönhetjük

3. melléklet



A rajzokat és az eszközt Sütő András kollégának köszönhetjük