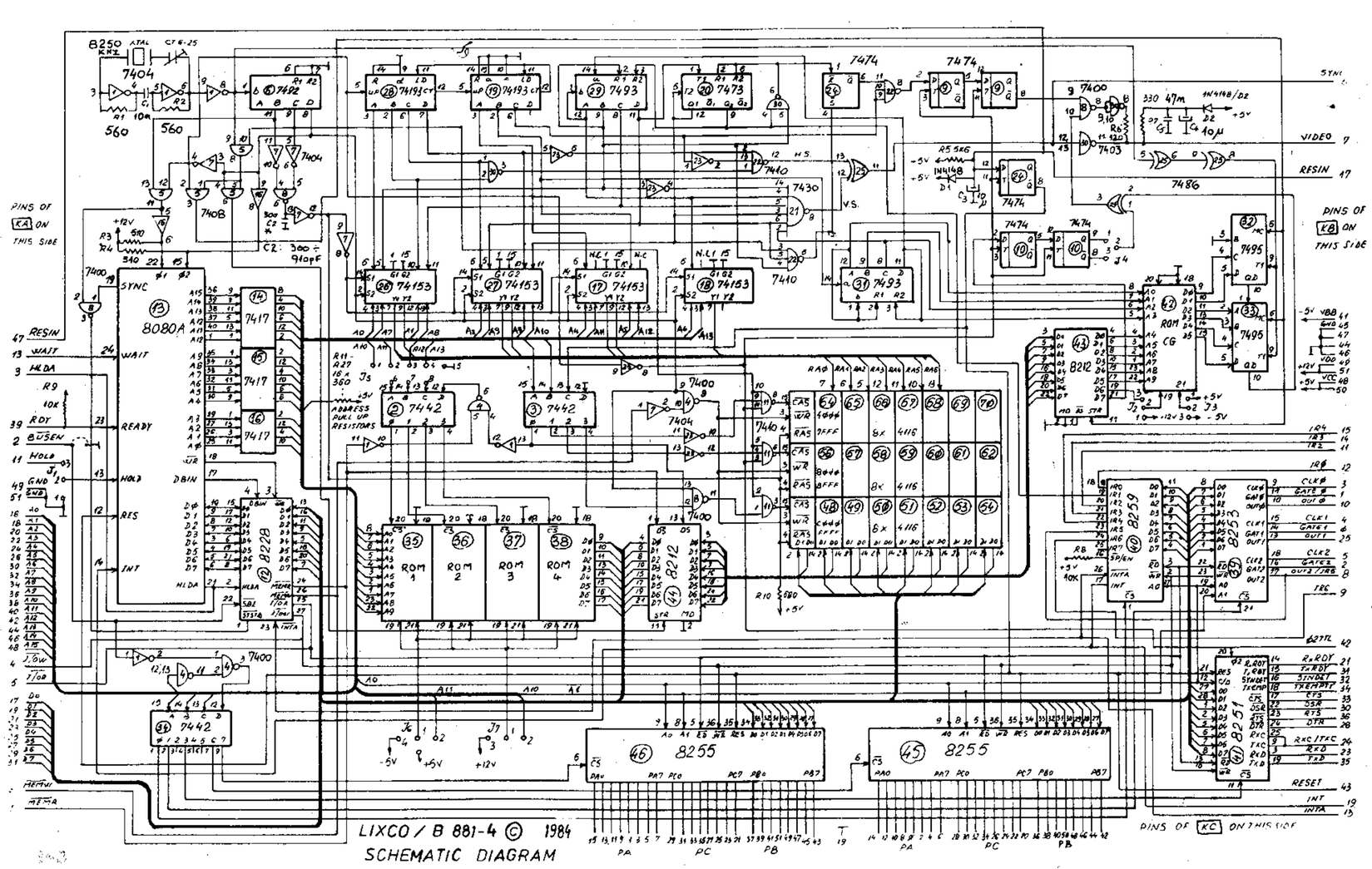
Lix Nicoara Paulian

Gheorghe Chita
Ion Rusovici
Catalin Bratu

Stefan Bordeanu
Liviu Ionescu
Romeo Burada

MICROCALCULATORUL

L/B 881



1983-1986

TEHNICĂ MODERNĂ

MICROCALCULATORUL

GHEORGHE CHITA
NICOARA PAULIAN YO3NP

Progresele inregistrate de tehnologiile integrării pe scară largă şi foarte largă, pe lingă implicaţiile remarcabile pe care le au in industrie, au determinat apariţia unui nou instrument de lucru in viaţa de zi cu zi: microcalculatorul.

După seria de articole pe care revista "Tehnium" a publicat-o avind ca temă structura şi particularităţile microprocesorului, vă propunem acum realizarea practică a unui microcalculator. Se impun insă citeva explicaţii pentru cititorii mai puţin avizaţi asupra modului in care lucrează calculatoarele in general.

Calculatorul poate fi privit ca o cutie neagră ce prelucrează informaţii. Informaţiile au sens doar pentru utilizatorul uman; ele reprezintă cunoştinţe dobindite despre lumea inconJurătoare. Omul işi reprezintă informaţiile prin **simboluri**: cuvinte, numere, diagrame etc; aceste simboluri sint proprii modului in care lucrează creierul uman: el stochează informaţiile folosind coduri cu redundanţă mare şi rezolvă problemele pe căi paralele folosind metode euristice.

Calculatorul nu poate prelucra decit date; datele sint reprezentări specifice calculatorului, ale informaţiei prelucrate; el foloseşte cel mai simplu mod de reprezentare, cu doar două simboluri notate convenţional cu cifrele 0 şi 1; acestea sint singurele cifre de care este nevoie in sistemul binar pentru a reprezenta orice număr.

O cifră binară se numeşte bit (engl. BInary digiT - cifră binară); 8 biţi formează un octet (byte); 1024 octeţi (bytes) formează un **kilooctet** (kilobyte) etc.

Folosirea acestui extrem de simplu sistem de reprezentare -sistemul binar- are avantaje şi dezavantaje.

avantaje:

- din punct de vedere tehnic - este mult mai simplu a se construi sisteme cu doar două stări stabile (decit cu 10, de exemplu): un contact poate fi inchis sau deschis, un tranzistor poate conduce sau poate fi blocat, o tensiune poate avea o anumită valoare (prestabilită), sau poate fi zero etc;
- din punct de vedere matematic - există un instrument puternic de tratare a logicii binare: algebra Boole;

dezavantaje:

- pentru a reprezenta aceeaşi cantitate de informaţie este necesară o succesiune mai lungă de simboluri in sistemul binar decit in alte sisteme; la nivelul utilizatorului uman acest dezavantaJ este eliminat prin folosirea sistemelor de numeraţie in bază 8 (octal) sau 16 (hexazecimal sau hex); sistemul hexazecimal foloseşte 16 cifre, iar corespondenţa dintre el şi sistemul binar este următoarea:

0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

- sint necesare frecvente conversii intre sistemul binar şi sistemul zecimal (sau alte forme de reprezentare a informaţiei pentru om).

Avantajele compensează cu prisosinţă dezavantajele (datorită progreselor in domeniul microelectronicii) şi de aceea, imensa majoritate a calculatoarelor actuale sint calculatoare numerice binare.

Datele prelucrate pot fi de mai multe tipuri; de exemplu:

- **date numerice:** numere intregi, reale, complexe; asupra lor se efectuează toate operaţiile cunoscute in matematică;
- **date logice:** acestea pot lua doar două valori, adevărat sau fals; asupra lor se efectuează operaţiile de negare, conJuncţie, disJuncţie etc. cunoscute in logica booleană;
- **date alfanumerice:** şiruri de caractere reprezentind litere, cifre, semne de punctuaţie; asupra acestor date se execută operaţiile de comparare, concatenare, substituire etc.

Calculatorul singur nu poate face distincţie intre aceste tipuri de date (pentru că ele sint reprezentate in acelaşi mod - prin numere in sistemul binar); de aceea, este necesar ca utilizatorul să-i indice cum să prelucreze datele de intrare pentru a obţine datele de ieşire (rezultatele). Acest lucru se face prin intermediul unui **algoritm**.

Algoritmul este un concept intuitiv şi inseamnă ceva de genul "metodă de rezolvare" sau "reţetă"; el desemnează o mulţime finită de operaţii cunoscute, care, executate intr-o ordine bine stabilită, pornind de la un set de valori iniţiale, produc in timp finit un set de valori la ieşire [1].

Un algoritm este deci "o mulţime finită de operaţii cunoscute", operaţii ce trebuie să aibe un caracter determinist şi eficace; pentru a se obţine aceste caracteristici, s-au creat **limbaJele de programare**. Limbaţele de programare sint limbaJe artificiale ce servesc la descrierea algoritmilor prin instrucţiuni. Instrucţiunile sint executate de calculator, se desfăşoară intr-un interval finit de timp şi au un efect bine definit. Algoritmul descris intr-un limbaJ de programare se numeşte **program**; programarea este activitatea de elaborare a programelor şi este executată de **programator**.

HARDWARE

Se pot vedea acum cele două componente ce alcătuiesc oricare calculator: ceea ce este care realizează prelucrarea informaţiei executind instrucţiunile programului, iar cea de-a doua calitate - programele inseşi. Cele două componente se numesc **hardware** şi **software**.

"Hardware" este un termen englezesc care inseamnă la origine "articole de fierărie şi menaJ" (!); semnificaţia sa a fost extinsă insă la totalitatea circuitelor, dispozitivelor şi echipamentelor calculatorului. Practic, prin hardware se inţelege tot ceea ce are legătură cu partea electronică, electrică sau mecanică (circuite, piese electronice, relee, cabluri, şasie etc.).

"Software" este tot un termen englezesc, construit prin antiteză cu hardware (hard = tare, soft = moale; ware = marfă). Prin el se desemnează totalitatea programelor cu care este echipat un calculator.

In calculatoarele actuale, software-ul reprezintă cam 80%, iar hardware-ul, 20% (atit valoric, cit şi ca performanţe) şi disproporţia continuă să crească. Principalul motiv este flexibilitatea software-ului; schimbind programele - operaţie ce se poate face in fracţiuni de secundă - se pot rezolva numeroase probleme din diferite domenii de activitate, cu aceeaşi configuraţie hardware.

HARDWARE

Calculatorul prelucrează informaţii codificate numeric in sistemul binar; prelucrarea se efectuează prin operaţii aritmetice, funcţii logice şi transferuri de informaţie şi are loc automat in urma executării unui şir de instrucţiuni - alcătuind un program care specifică operaţiile elementare efectuate şi operanzii folosiţi. Ca urmare, calculatoarele trebuie să indeplinească anumite condiţii:

- existenţa unui mediu de intrare de la care se pot introduce un număr nelimitat de date şi instrucţiuni;
- utilizarea unei memorii in care se păstrează datele de intrare şi instrucţiunile şi in care se depozitează rezultatele in ordinea dorită;
- existenţa unei unităţi funcţionale capabilă să execute operaţii aritmetice şi logice asupra oricărui operand citit din memorie;
- existenţa unui mediu de ieşire prin intermediul căruia se furnizează utilizatorului un număr nelimitat de rezultate;
- existenţa unei unităţi funcţionale capabile să ia decizii, cu scopul de a stabili ordinea de execuţie a instrucţiunilor in funcţie de rezultatele obţinute;
- memorarea instrucţiunilor şi datelor in aceeaşi formă, astfel incit să se poată prelucra programe (ca date de intrare) pentru a obţine alte programe (ca rezultate).

In figura 1 este dată organizarea generală a hardware-ului calculatorului, organizare ce urmăreşte tocmai realizarea cerinţelor de mai sus.

UCP - Unitatea Centrală de Prelucrare (CPU - Central Processing Unit)

UCP are rolul de a:

- efectua prelucrarea propriu-zisă prin execuţia instrucţiunilor;
- dirija activitatea celorlalte dispozitive.

UCP este compusă din:

- unitatea aritmetică-logică UAL (ALU Arithmetic Logic Unit);
- registre - Reg;
- unitatea de comandă UCd (Control Unit).

UAL efectuează operaţii aritmetice, logice, de comparare, decizie, etc. asupra operanzilor citiţi din memorie in registrele generale, sub acţiunea semnalelor de comandă şi control primite de la UCd.

Registrele sint circuite speciale destinate păstrării informaţiei binare in timpul prelucrării ei in UCP. Unele registre sint accesibile programatorului, altele nu. Registrele accesibile programatorului:

- registre generale (general purpose registers) păstrează operanzii prelucraţi de UAL;
- registrul de stare a programului (program status register) - conţine informaţii asupra rezultatului ultimei operaţii din UAL: rezultat zero, pozitiv/negativ, depăşire, trunchiere, transport etc;
- numărătorul de program (program counter) - conţine adresa următoarei instrucţiuni de executat;
- registre de bază, index, indicatorul de stivă (base, index registers, stack pointer) - registre folosite in diferite moduri de acces la instrucţiuni şi date aflate in memorie.

Registrele neaccesibile programatorului:

- registrul de instrucţiuni - păstrează instrucţiunea pe timpul execuţiei sale in UCP;
- registrul de adrese - folosit pentru formarea adresei necesare accesului la memorie sau U I/O.

MEM - Memoria (memory, storage)

Memoria păstrează programele (pe durata execuţiei lor de către UCP) şi informaţiile asociate acestora (date de intrare, rezultate); este compusă dintr-un ansamblu de locaţii, fiecăreia fiindu-i asociată o adresă, prin intermediul căreia i se specifică accesul la locaţia respectivă; la fiecare locaţie poate fi memorat un cuvint de lungime fixă, care reprezintă unitatea adresabilă, şi numit in general, cuvint al memoriei. Asupra memoriei se pot efectua operaţii de citire şi scriere a informaţiei pe baza unei adrese; citirea presupune obţinerea informaţiei de la locaţia specificată prin adresă, iar scrierea - depunerea ei in locaţia corespunzătoare adresei.

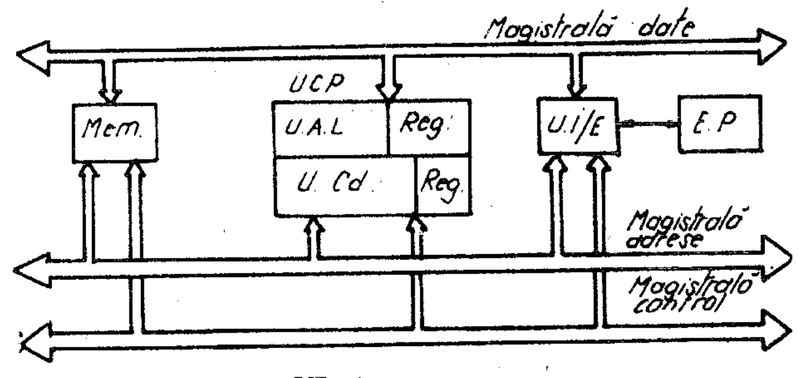


FIG. 1

Există, in mare, două tipuri de memorii:

- memorii RAM (Random Access Memory - memorie cu acces aleator), asupra cărora se pot executa operaţii de scriere şi citire a conţinutului, fiind astfel destinate păstrării datelor intermediare, rezultatelor, programelor a căror prezenţă este necesară pentru un timp limitat. DezavantaJul RAM-urilor este că işi pierd conţinutul la oprirea alimentării, făcind astfel necesară prezenţa ROM-urilor;
- memorii ROM (Read Only Memory - memorie numai pentru citire), care pot fi doar citite şi al căror conţinut nu se pierde la oprirea alimentării. Ele sint programate "pe viaţă", fie de către fabricant, fie de utilizator; in acest din urmă caz, memoria se numeşte programabilă (PROM); dacă beneficiarul poate anula conţinutul memoriei şi inscrie un altul, atunci memoria se numeşte reprogramabilă (EPROM - Erasable PROM).

U I/O - Unităţile de Intrare/Ieşire (I/O C - Input/Output Channels)

U I/O au rolul de a controla activitatea echipamentelor periferice (EP), asigurind astfel transferul datelor intre acestea şi UCP sau Mem, prin:

- memorarea temporară a datelor aflate in transfer;
- conversia reprezentării datelor;
- serializarea sau deserializarea datelor;
- comanda şi controlul unor operaţii specifice EP (deplasarea capetelor discurilor, rebobinarea benzilor magnetice etc.);
- păstrarea unor informaţii de stare referitoare la EP (nepregătit, neoperaţional, ocupat etc.) sau la operaţia in curs de desfăşurare (detectarea unei erori la controlul datelor).

U I/O permit astfel creşterea eficienţei utilizării UCP, aceasta nemaifiind nevoită să se sincronizeze cu operaţiile de I/O (viteza de execuţie a acestora fiind mult diferită de viteza de execuţie a instrucţiunilor in UCP). Tot in acest scop, s-au creat U I/O programabile ce lucrează ca un calculator specializat de I/O: UCP introduce un program (cu datele necesare: adresa şi lungimea mesaJului) in U I/O şi iniţiază execuţia sa, putind apoi să continue prelucrarea informaţiilor aflate deJa in memorie; U I/O indică UCP, printr-un semnal de control, terminarea transferului sau apariţia unui eveniment neprevăzut pentru ca UCP să ia deciziile corespunzătoare; de asemenea, UCP poate interveni in timpul transferului pentru testarea stării acestuia sau a EP şi poate opri transferul.

EP - Echipamentele Periferice (Peripheral Devices)

EP asigură introducerea/extragerea informaţiei in/din calculator prin conversia intre formele de reprezentare (accesibile calculatorului) şi formele externe (accesibile utilizatorului, memoriei externe sau liniilor de comunicaţie).

EP pot fi:

- EP de intrare - asigură conversia spre calculator (cititorul de cartele perforate, cititorul de bandă perforată etc.);
- EP de ieşire - asigură conversia dinspre calculator (perforatorul de cartele, perforatorul de bandă, imprimanta, plotter-ul etc.);
- EP de intrare/ieşire - asigură conversia in ambele sensuri (terminalul video, teleimprimatorul, discul magnetic, caseta magnetică, floppy-disk-ul etc.);

Magistralele (Buses)

Magistralele [1] sint grupuri de conductoare folosite in comun de unităţile funcţionale pentru transmiterea semnalelor ce codifică un vector binar. După semnificaţia semnalelor transmise, magistralele pot fi magistrale de adrese, de date sau de control.

Pe magistrala de date se transmit atit operanzii necesari execuţiei instrucţiunilor, cit şi instrucţiunile inseşi. De capacitatea ei depinde puterea de calcul.

Adresele locaţiilor de memorie sau ale U I/O se transmit pe magistrala de adrese. Capacitatea sa determină cantitatea maximă de memorie internă ce poate fi folosită şi numărul maxim de U I/O ce pot fi conectate (dacă n este numărul de conductoare ale magistralei, atunci UCP va putea folosi maximum 2^n adrese).

Magistrala de control este folosită la transmiterea de comenzi şi informaţii despre starea unităţilor funcţionale interconectate.

SOFTWARE

Există, in mare, două tipuri de software: software de aplicaţie şi software de sistem.

Prin software de aplicaţie se inţeleg programele scrise de utilizator pentru rezolvarea problemelor sale specifice cu ajutorul calculatorului. Scrierea, depanarea şi execuţia acestora sint mult uşurate de software-ul de sistem.

Rolul software-ului de sistem (sau software de bază) este de a asigura gestiunea resurselor calculatorului (memorie, timp al UCP, U I/O), eliberind astfel utilizatorul de această sarcină şi permiţindu-i să se concentreze asupra problemei specifice ce o are de rezolvat. Printre programele de sistem se află:

- asamblorul [3]. După cum se ştie, programele scrise in limbaJul calculatorului (sau limbaJ-maşină), sint lungi şiruri de biţi; calculatorul le manipulează cu uşurinţă, oamenii insă nu. Programele in limbaJ maşină le apar oamenilor lungi, obositoare, confuze şi fără inţeles. O imbunătăţire evidentă este de a atribui un nume fiecărui cod de instrucţiune (care este un număr binar); acest nume se numeşte mnemonic şi el descrie intuitiv cam ce face instrucţiunea. Se obţine astfel un limbaJ de asamblare, care este mai uşor de inţeles de către om. Asamblorul este un program de translatare a programului din limbaJ de asamblare (numit program sursă) in secvenţele binare ale limbaJului maşină (numit program obiect). De asemenea, asamblorele permit şi alte facilităţi - cum ar fi asocierea de nume simbolice locaţiilor de memorie, registrelor UCP şi unităţilor I/O. Asamblorele au şi dezavantaJe legate de dependenţa lor de limbaJul maşină al unui anumit tip de calculator: programatorul trebuie să cunoască in detaliu calculatorul pe care il foloseşte, să se concentreze asupra setului de instrucţiuni al acestuia mai degrabă decit asupra problemei ce o are de rezolvat, de aici decurgind şi lipsa de portabilitate a programelor (programele nu se pot executa decit pe un anumit tip de calculator);
- compilatorul [3]. Soluţia la multe din dificultăţile asociate folosirii limbaJului de asamblare este folosirea unui limbaJ "de nivel inalt" (limbaJ "orientat pe procedură"). Aceste limbaJe permit descrierea rezolvării unei probleme in forme ce sint orientate mai degrabă spre acea problemă decit spre calculatorul folosit. Compilatorul este programul care traduce programe din limbaJul de nivel inalt in limbaJul maşină. Exemple de limbaJe de nivel inalt sint FORTRAN (FORmula TRANslation - pentru calcule tehnico-ştiinţifice), COBOL (COmmon Bussines Oriented Language - pentru domeniul economic administrativ), LISP (LISt Processor - pentru inteligenţă artificială), PASCAL (un limbaJ structurat), C (parte integrantă a sistemului de operare UNIX cu avantaJe in ce priveşte generarea de cod compact) ADA (un limbaJ complex, structurat, aplicabil in foarte multe domenii) etc;
- interpretorul - este un program care acceptă instrucţiuni şi comenzi de nivel inalt, pe care le interpretează in momentul execuţiei fără să genereze cod. Au viteză scăzută, dar un grad mai inalt de interactivitate, fiind foarte populare mai ales printre amatori (ex. BASIC);
- editorul de legături - permite legarea mai multor programe compilate sau asamblate anterior (scrise eventual, in limbaJe diferite şi aflate intr-o bibliotecă de programe) intr-unul singur;
- incărcătorul - incarcă un program in limbaJ maşină de pe un mediu extern in memorie şi il lansează in execuţie;
- monitorul - cel mai important program de sistem avind rolul de a supraveghea intreaga activitate a calculatorului; el primeşte ordine de la utilizator pentru a lansa in execuţie diferite programe (de sistem sau de aplicaţie), gestionează resursele calculatorului şi asistă programul utilizatorului in timpul execuţiei (facilitindu-i in special operaţiile de intrare-ieşire şi tratarea erorilor ce pot apare).

BIBLIOGRAFIE

1. * * * - Dicţionar de informatică, Ed. ştiinţifică şi enciclopedică, Bucureşti, 1981
2. Knuth D. E. - Tratat de programarea calculatoarelor, Vol.I - Algoritmi fundamentali, Ed. Tehnică, Bucureşti, 1974
3. Lance A. Leventhal - 8080A/8085 Assembly Language Programming, Osborne & Associates, Inc. Berkeley, California, 1978

PENTRU AMATORI

MICROCALCULATORUL L/B 881

PAULIAN NICOARA YO3NP
RUSOVICI ION YO3JF

Microcalculatorul L/B881 a fost realizat in ideea minimizării componentelor inglobate, fără insă a renunța la o serie de facilități considerate ca strict necesare. El utilizează intr-o proporție covirșitoare componente de fabricație românească. Incepind din acest număr, vom descrie modul de realizare practică a calculatorului, urmind apoi să demonstrăm și citeva aplicații și programe.

GENERALITATI

Caracteristicile principale ale microcalculatorului L/B881 sint:

* **unitatea centrală:** microprocesor de 8 biți tip 8080A;
* **memoria ROM:** 4 buc. a cite 1, 2 sau 4 kocteți (maxim 16 kocteți);
* **memoria RAM:** dinamică, maxim 48 kocteți;
* **periferice:**

interfață serială programabilă;
2 interfețe paralele programabile;
3 timere programabile;
controlor de video display (64 de caractere și 26 de rinduri cu posibilități grafice 128/78 pixeli);
interfață pentru tastatură;
interfață pentru casetofon;
interfață pentru imprimantă serie V24 (RS 232);

* **sistem intreruperi:** 8 nivele programabile, cu tabelă de salturi in memoria RAM.

Toate componentele microcalculatorului sint montate pe o singură placă de circuit imprimat, iar comunicarea cu exteriorul se face prin trei conectori ale căror semnale au fost grupate pe funcțiuni.

Placa se montează intr-o eutie impreună cu claviatura, sistemul de interconectare și sursa de alimentare.

PRINCIPIUL DE FUNCTIONARE

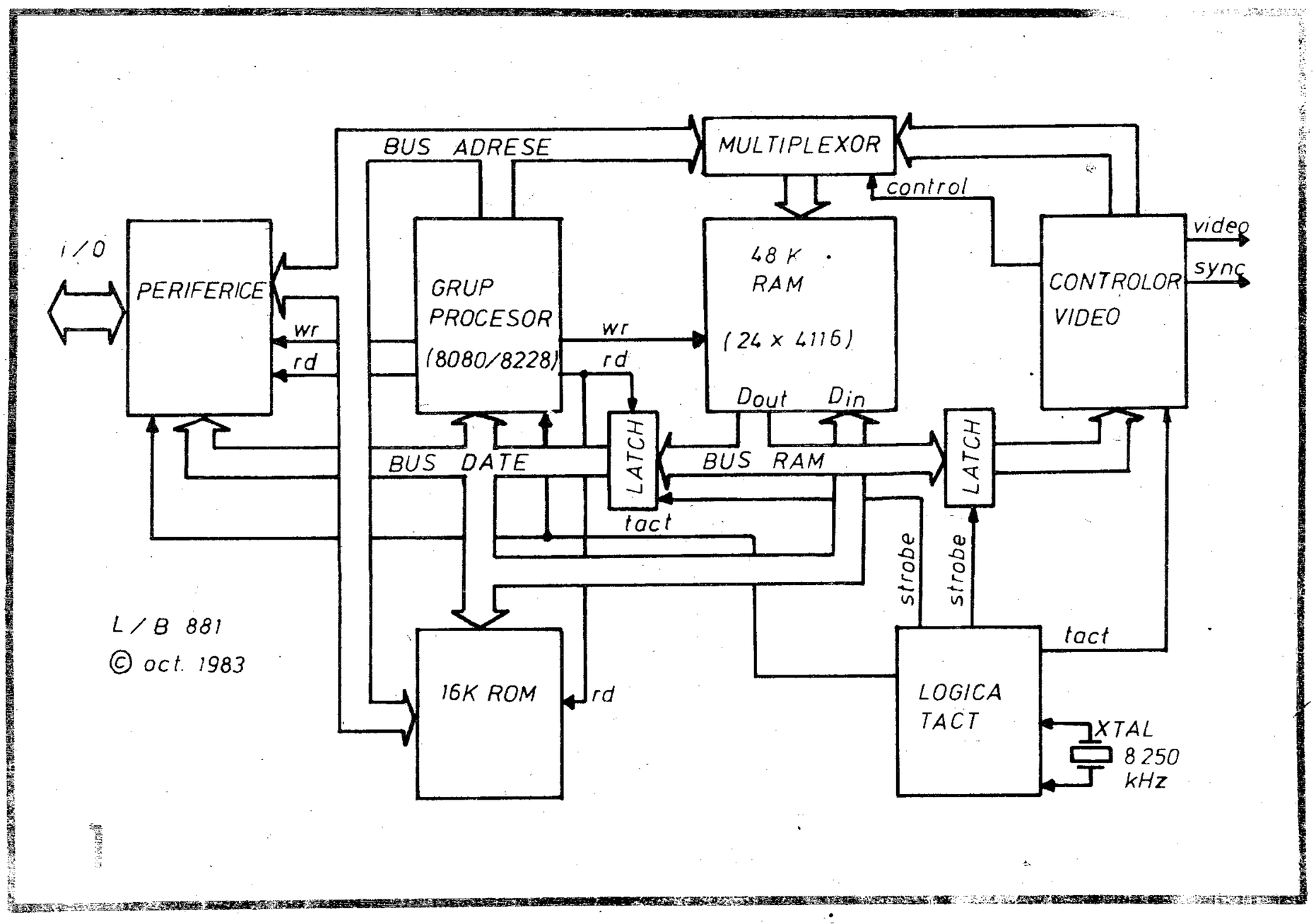
In vederea clarificării unor aspecte generale de funcționare, cele ce urmează se vor referi la figura 1.

Rezolvarea problemelor de refresh al memoriei dinamice și a ecranului s-a făcut prin multiplexare, astfel incit pe un ciclu de procesor (aprox. 720 ns) au loc două cicluri de acces la memorie: unul pentru procesor și unul pentru controlorul video. Datele din memorii sint strobate alternativ in două registre de 8 biți și pot fi accesate după nevoie de procesor sau controlor. In timp ce procesorul nu are nevoie de date la fiecare ciclu (pentru el memoria pare statică prin intermediul latch-ului), controlorul video trebuie in mod absolut necesar ca la fiecare ciclu de 720 ns să ia o nouă dată din memorie pentru a o afișa pe ecran. De fapt, ciclul de 720 ns este impus tocmai de viteza de succedare a caracterelor pe un rind (respectiv 64).

In această situație, la un ciclu de memorie de 360 ns și utilizind o tehnică de selecție prin pinul de CAS (Column Address Strobe) al memoriilor, reimprospătarea acestora se face automat la fiecare două rinduri de ecran (64x2=128 rinduri reimprospătate, adică intregul banc de 48 kbytes).

In figura 2 este prezentată schema electrică a plăcii L/B881. Partea de timing pleacă de la oscilatorul cu cristal pe frecvența de 8250 kHz, realizat cu o parte din porțile circuitului integrat U1. Urmează un divizor cu 6 realizat cu U6 din care se obțin principalele semnale ale sistemului (U5, U7, U8 și U16): timpul de RAS (Row Address Strobe) și de CAS (Column Address Strobe) atit pentru accesul procesorului cit și al controlorului video, cele două semnale de tact ale procesorului (care sint aduse la nivelele MOS prin doi operatori open-collector din U16), semnalul de Status Strobe pentru 8228, precum și semnalele care comandă multiplexorul de adrese ale memoriei.

Se observă in continuare lanțul de divizoare pentru obținerea adreselor de linie (U28 și U19) și rinduri (U29, U20 și U31), plus grupul de porți care asigură resetarea numărătoarelor și obținerea semnalelor de sincronizare pe verticală și orizontală (U21, U22, U23 și U30). Multiplexorul de adrese este realizat cu circuitele 74153 (U17, U18, U26 și U27). El transmite alternativ către memorie cele două rinduri de adrese de 2x7 biți (deci 4 la 1). La rindul ei, memoria are pe ieșire două latch-uri de 8 biți care rețin datele pentru procesor (U44) și controlor video (U43). Pe partea video, datele sint transmise generatorului de caractere (U42), care la rindul său le transmite serializatorului format din cele două registre de deplasare (U32 și U33), in timp ce pe partea procesorului, datele sint depuse pe bus-ul principal de date al micro-



sistemului atunci cind este necesar (linia MEMR).

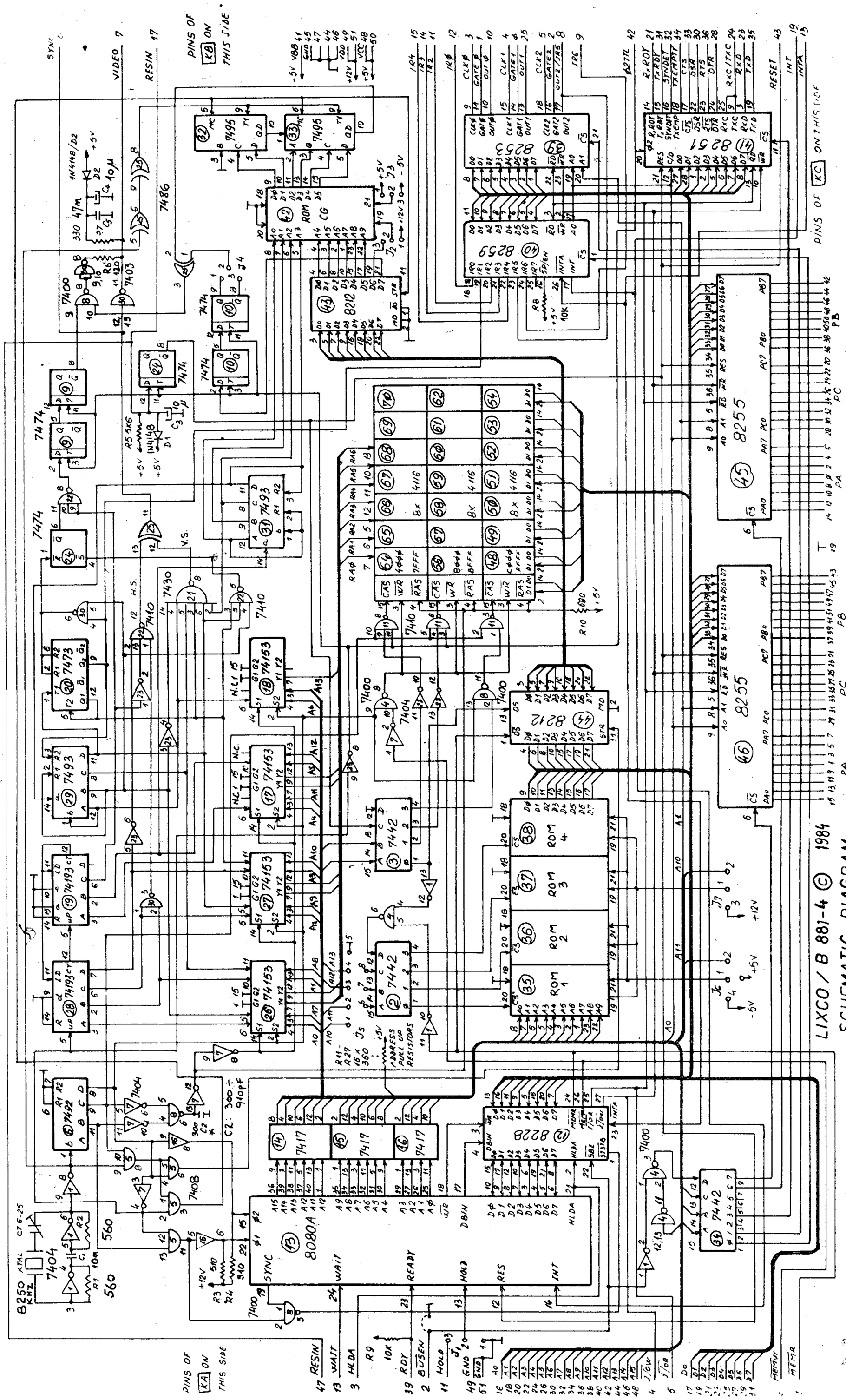
Semnalul video serializat este trecut printr-un operator XOR care are posibilitatea să-l inverseze funcție de bitul 7 al memoriei, iar apoi este combinat cu semnalele de sincronizare TV și scos in afara plăcii folosind un NAND open-collector. Se observă că atit porțiunile de blanking de margine a ecranului cit și reversarea video sint intirziate cu două perioade de tact (U9 și U10), ca urmare a faptului că ele survin direct din RAM, fără o intirziere suplimentară pe generatorul de caractere.

Pe partea de procesor, sistemul este compus din grupul CPU (8080A - U13 și 8228 - U12), la care se remarcă absența generatorului de tact 8224 datorită unor impedimente care ar fi apărut din forma asimetrică a lui FI2 și ar fi afectat sistemul de utilizare multiplexată a memoriei dinamice (8224 generează un semnal FI2 cu raport de umplere 4/5, in timp ce logica utilizată la L/B881 generează un raport 3/3 folosind o divizare cu 6); un tampon de buffere pe magistrala de adrese realizat cu operatori open-collector (U14, U15 și U16), decodificatoare de adrese pentru memorii (U2 și U3) și dispozitive I/O (U34), memoriile ROM (U35 la U38) și RAM (U48 la U70). Sistemul de periferice include un USART 8251 pentru comunicații serie (U41), un timer programabil 8253 (U39), un contrólor de intreruperi programabil 8259 (U40) și două interfețe paralele programabile PPI 8255 (U45 și U46).

Pe placă se mai remarcă prezența unor jumperi care configurează:

* diferite tipuri de memorii EPROM folosite;
* semnale utilizabile in afara plăcii;
* inversarea video.

Descrierea modului de poziționare a acestor jumperi va fi făcută in numărul viitor, odată cu alte detalii constructive. Cei interesați pot lua legătura cu autorii prin intermediul redacției pentru obținerea unor informații legate de documentația plăcii imprimate.



MICROCALCULATORUL L/B 881

NICOARA PAULIAN YO3NP
ION RUSOVICI YO3JP
STEFAN BORDEANU YO3DP

In acest număr sînt tratate cîteva probleme legate de interfaţa cu lumea exterioară a plăcii principale a microcalculatorului L/B881 şi anume claviatura, casetofonul şi sursa de alimentare.

INTERFATA CONECTORULUI KB

Pe conectorul KB este fixată o placă de circuit imprimat (de fapt placa este infiptă in picioruşele conectorului), care conţine circuitele de formare a semnalului provenit din casetofon, un repetor pentru Bell (clopot), un potenţiometru pentru reglajul nivelului video şi interfaţa standard serie RS232c (fig. 1).

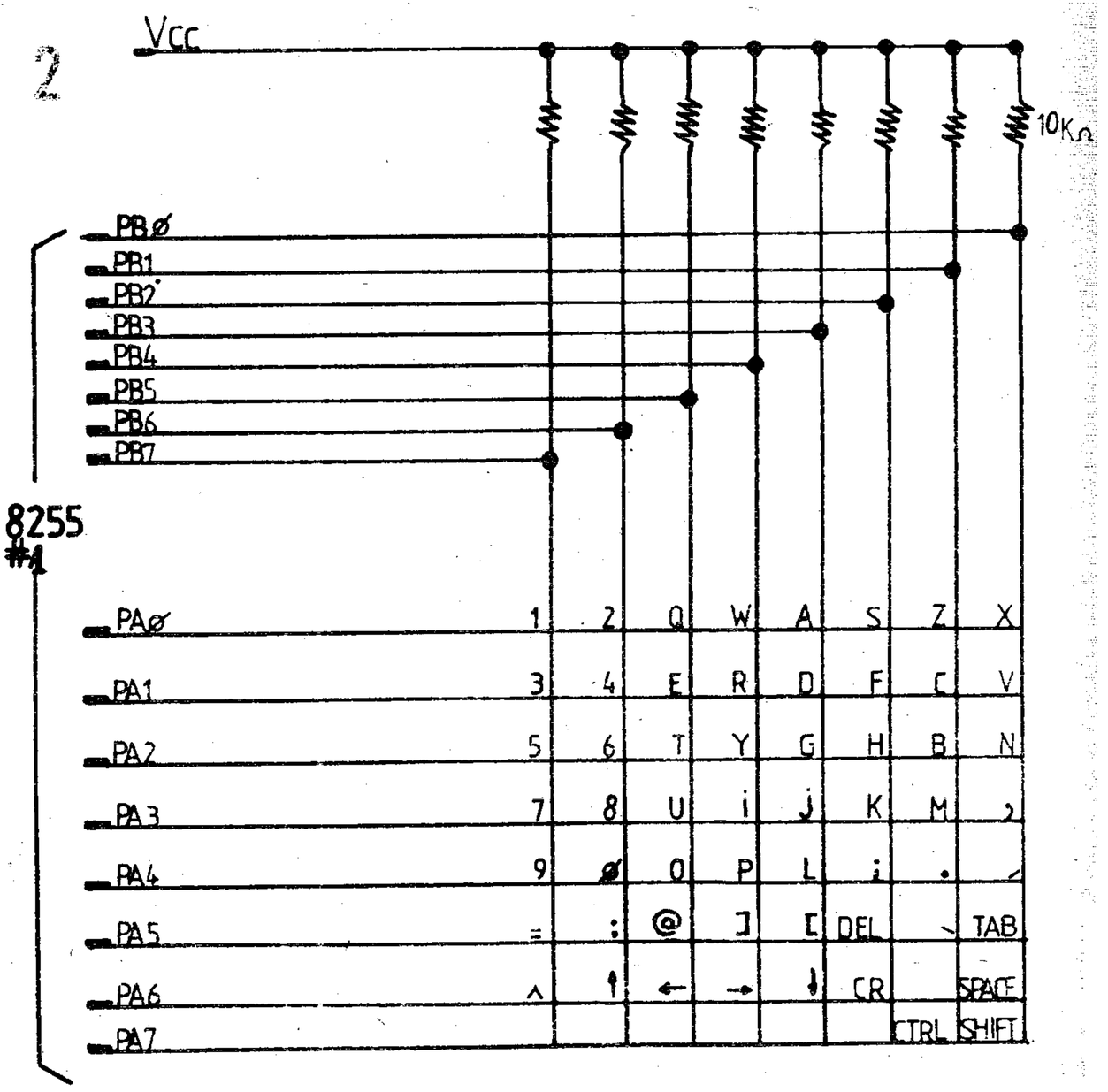
Pentru partea de casetofon, semnalele culese de la mufa de line-out sint amplificate cu un 741 şi apoi aplicate pe de o parte unui monostabil cu perioada de aprox. 0,3 ms din care se obţine un semnal de intreruperi pentru 8259, nivelul 4 şi pe de altă parte direct unei intrări (PC0) a PPI 8255. Analizarea şi reconstituirea datelor de pe banda magnetică se fac prin software de către monitorul rezident 881/Mon (care va fi descris in numărul viitor), la fiecare intrerupere generată de un front crescător provenit din inregistrare.

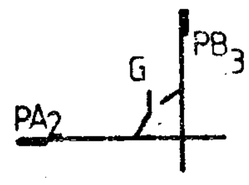
Poziţionarea jumperilor de pe această placă se face astfel:

- J1,J2 şi J3 se ştrapează. Ele furnizează tactul necesar celor trei timere din cadrul 8253, din FI2. In cazul in care este necesară utilizarea la unul din aceste timere a unei frecvenţe de tact externă, se desface jumperul corespunzător şi se aplică semnalul pe intrarea respectivă (CLK0, CLK1 sau CLK2).
- J4 şi J5 se ştrapează. Ele reprezintă semnalele de RXRDY şi TXEMPTY ale USART-ului care sint in mod normal cuplate pe nivelele de intreruperi 2 şi 3. Dacă nu se urmăreşte modul de lucru cu USART-ul pe intreruperi sau este necesară utilizarea nivelelor respective de intreruperi in alte scopuri, atunci jumperii nu se mai montează şi se cuplează semnalele externe.
- J6 şi J7 reprezintă cuplarea intrărilor USART-ului fie pe nivel TTL fie pe nivel RS232c, prin circuitele de interfaţă MC1488 şi MC1489. Dacă se urmăreşte utilizarea USART-ului in ambele situaţii, se recomandă montarea unui mic comutator pe spatele cutiei care să realizeze trecerea rapidă de la un mod la celălalt.

CLAVIATURA

Claviatura reprezintă interfaţa cu utilizatorul şi a fost realizată prin codificare prin program folosind o reţea de taste cuplate intre ele sub forma unei matrice. In figurile 2, 3 şi 4 sint date detaliile de amplasare a tastelor, schema electrică şi schema de interconectare. Se recomandă păstrarea dimensiunilor date intrucit acestea sint standardizate după maşinile obişnuite de scris. După realizarea fixării mecanice, se execută cablajul folosind fire subţiri sau realizind o placă de circuit special in acest scop. Legătura cu placa de bază se va face printr-un mănunchi de 16 fire care va avea la celălalt capăt conectorul ce se va infige in

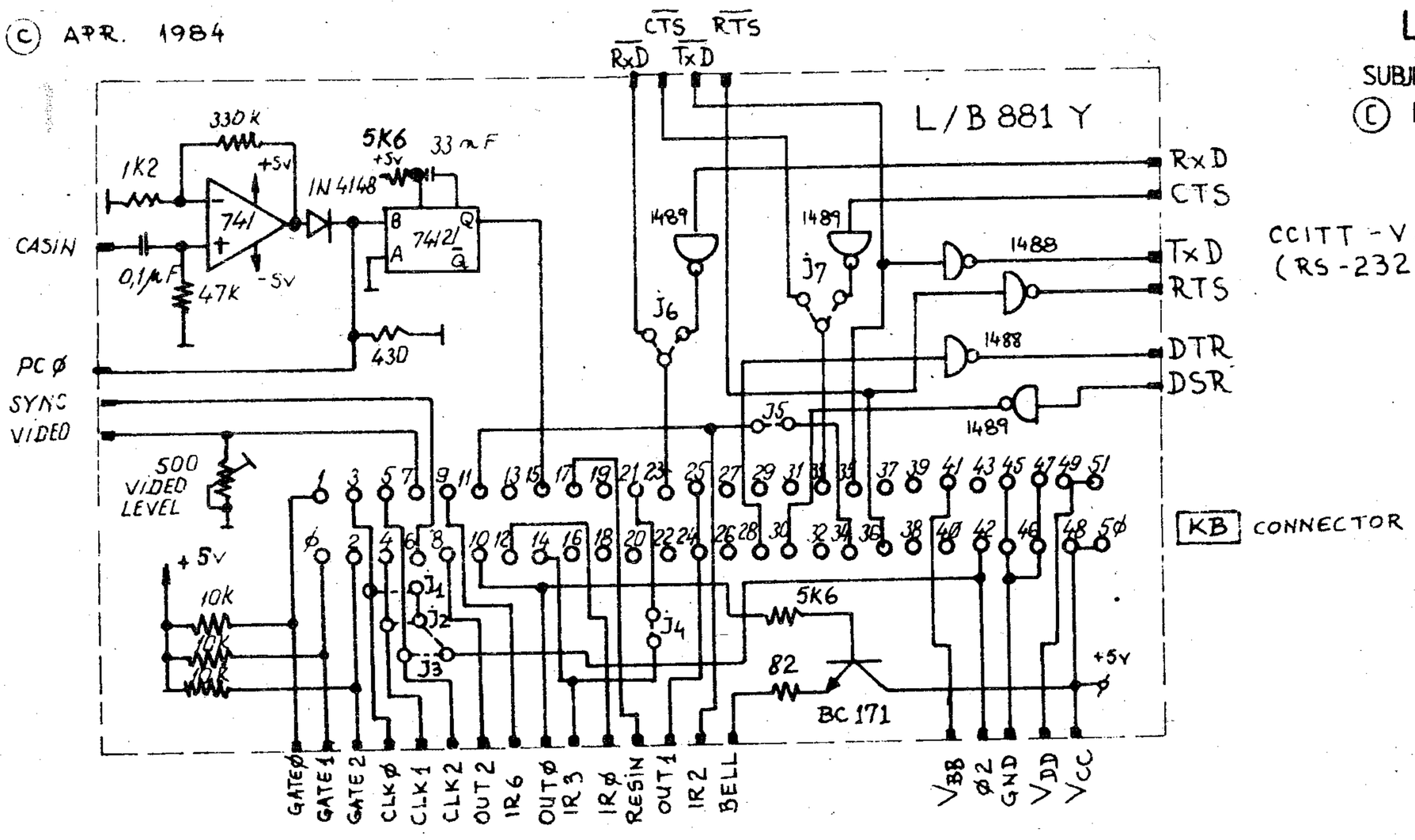


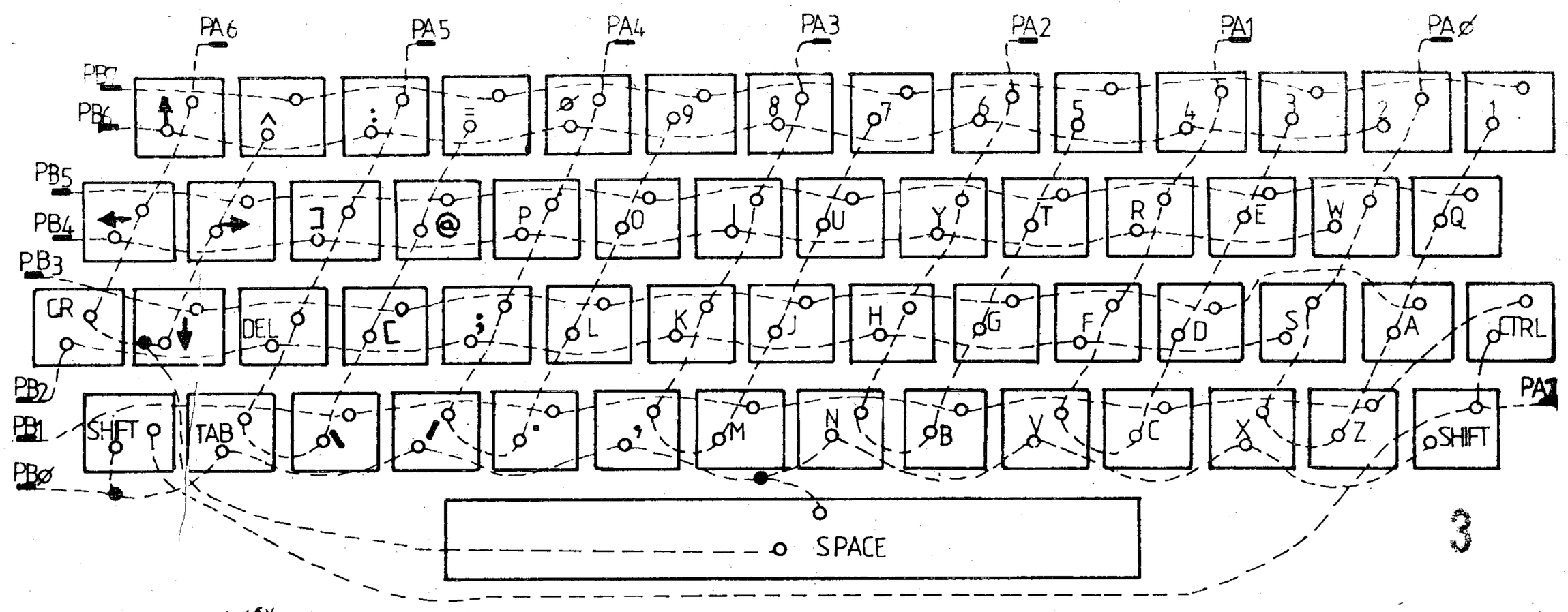
NOTA: 

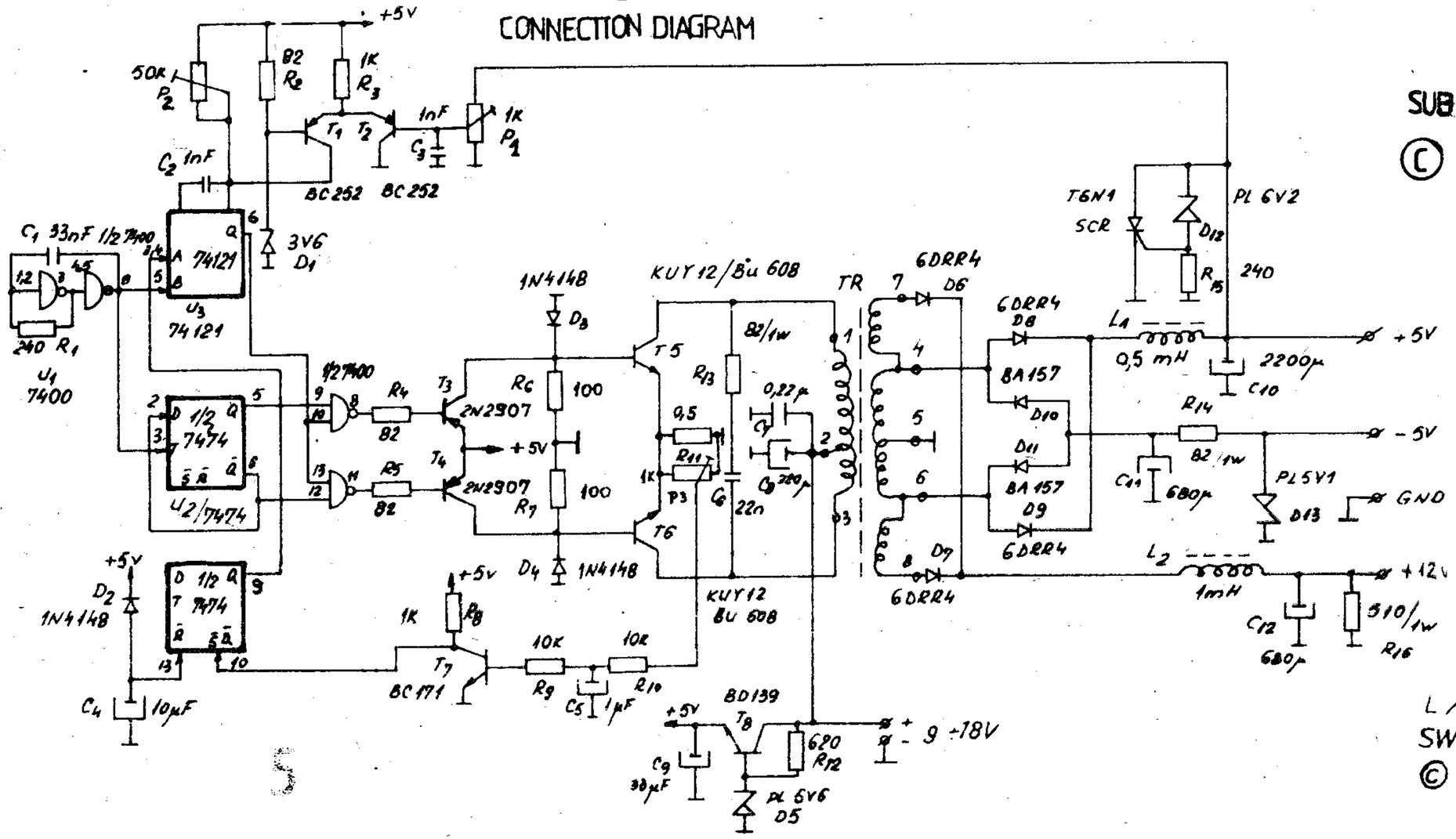
DENOTĂ UN ÎNTRERUPĂTOR NORMAL DESCHIS

L/B 881
SUBJECT: KEY BOARD SECTION-SCHEMATIC DIAGRAM
© DEC. 1983

L/B 881
SUBJECT: L/B 881 Y INTERFACE BOARD
© APR. 1984



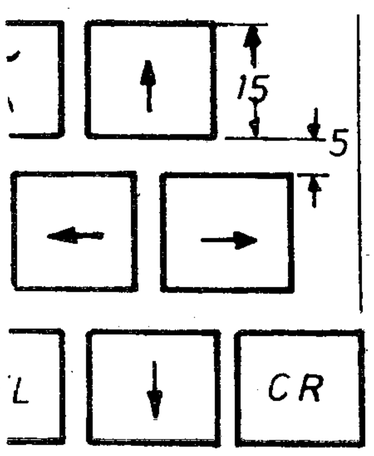




L/B 881

SUBIECT KEYBOARD SECTION – SWITCH

© DEC 1983



L/B 881 PSPY-2
SWITCH MODE POWER SUPLY
© Aug. 1985

placă (KC). Tot pe acest conector se pot cupla şi cele 8 rezistenţe. Pentru marcarea clapelor se pot folosi diferite procedee ca pantograful, letrasetul, rondele de plastic adeziv etc.

SURSA DE ALIMENTARE

Un aspect deosebit de important (şi critic) al microcalculatorului este sursa de alimentare. Mai Jos este dat consumul total al plăcii L/B881:

1,6 A la 5 v;
0,7 A la 12 v;
0,1 A la -5 v.

Rezolvarea problemei alimentării poate fi făcută in două moduri: sursă liniară sau in comutaţie. Descriem in continuare o variantă de sursă in comutaţie. Randamentul sursei prezentate nu este foarte mare, in Jur de 57%, funcţie de tensiunea de alimentare şi asta datorită sistemului de comandă in lăţime a impulsurilor, care este realizat cu trei circuite integrate TTL şi care trebuie alimentate la 5v. DezavantaJul este compensat de simplitatea schemei şi posibilităţile ei de a lucra la tensiuni de alimentare intre 9 şi 18 volţi.

Principiul de .funcţionare este următorul: semnalul preluat de la oscilatorul local pe aprox. 64 kHz realizăt cu două porţi NAND este divizat pe de o parte de către un bistabil tip D şi pe de alta este aplicat monostabilului U3. Perioada de declanşare a monostabilului este determinată de rezistenţa echivalentă a Joncţiunii CE a tranzistorului T1 din etaJul diferenţial format din T1-T2 şi condensatorul C2. EtaJul diferenţial este tocmai amplificatorul de eroare prin care se inchide bucla de reglaJ a convertorului. Prin două porţi NAND, semnalele din ieşirile bistabilului sint modulate in lăţime proporţional cu variaţia sarcinii şi aplicate etaJului prefinal şi apoi final in push-pull. S-a ales soluţia cu etaJ final in contratimp tocmai pentru a putea asigura o funcţionare corectă a sursei la tensiuni de alimentare reduse. In colectoarele tranzistoarelor finale se află un transformator pe miez de ferită al cărui secundar este proiectat de aşa manieră incit să se obţină toate tensiunile necesare microcalculatorului. Se remarcă sistemul de filtraj cu bobine ca şi grupul de deparazitare format din C6 şi R13. Sursa este protejată la scurtcircuit prin tranzistorul T7 care monitorizează curentul etaJului final şi acţionează bistabilul D in cazul unei depăşiri exagerate (peste 3 amperi); protecţia la supratensiune este realizată prin tiristorul SCR şi dioda Zenner din poartă care pun sursa in scurtcircuit urmind să acţioneze in continuare fie protecţia la supracurent, fie siguranţa fuzibilă.

Se menţionează faptul că acest convertor nu generează spike-uri la pornire sau oprire pe nici una din alimentări, iar riplul nu depăşeşte 100 mV. Din punct de vedere mecanic, piesele sint montate pe două plăcuţe de circuit imprimat cu dimensiunile de aprox. 120x70 mm şi aşezate "sandwich" cu nişte distanţieri de alamă intre două table de aluminiu ce servesc de radiatoare pentru tranzistoarele finale şi respectiv diodele redresoare de pe ramura de 5v.

Transformatorul este bobinat pe un tor de ferită (de tipul T34x23x12,7 A5 - ICSITE) şi are 2x11 spire in primar cu sirma de 0,9mm CuEm, iar secundarul are 2x11 spire cu sirmă de 0,9mm CuEm pentru 5v la care sint inseriate infăşurările pentru 12v care au cite 13 spire cu sirmă de 0,6mm CuEm. L1 şi L2 sint bobinate pe două oale de ferită (de tipul 025x16A1400 . A5 - ICSITE) pină la umplere, L1 cu sirma de 0,9mm CuEm iar L2 cu sirma de 0,5 - 0,6mm CuEm.

Reglarea se face cu o sarcină cuplată pe 5v şi una pe 12v care să fie cit mai apropiate de condiţiile reale de lucru (respectiv 2,5 ohmi/5W pe 5v şi 24 ohmi/5W pe 12v), urmărind că umplerea maximă (limitată din P2) să fie de 45%. Tensiunea de 5v se reglează din P1, iar cea de 12v rezultă automat. Protecţia la supracurent se reglează din potenţiometrul P3 pentru un curent de 3 Amperi.

(CONTINUARE ÎN NR. VIITOR)

MICROCALCULATORUL L/B 881

(URMARE DIN NR. TRECUT)

NICOARA PAULIAN YO3NP
ION RUSOVICI YO3JF
ROMEO BURADA YO9CFR

CUPLAREA CU TELEVIZORUL

Comunicarea dintre utilizator şi microcalculator se face prin intermediul claviaturii şi a monitorului TV (interfaţa dintre om şi maşină). In numărul trecut am văzut cum se realizează claviatura, acum vom rezolva problema cuplării semnalului video la un televizor obişnuit. Desigur, se poate folosi un monitor TV profesional, dar cum acest lucru nu este posibil intotdeauna, vom face apel la televizorul din casă.

Interfaţarea microcalculatorului cu televizorul se poate face in două moduri: fie cuplind semnalul video direct la intrarea amplificatorului video al televizorului, fie prin intermediul unui modulator RF la borna de antenă. In primul caz se va obţine o calitate foarte bună a imaginii dar este necesară o intervenţie in televizor pentru a monta o mufă de cuplare; deasemeni, operaţiunea nu se poate face decit pe un televizor care are transformator de separaţie faţă de reţeaua de curent alternativ (de ex. tip "Sport"). **Pentru a evita accidentele verificaţi cu atenţie dacă tipul de televizor pe care il veţi utiliza respectă condiţia de separare galvanică faţă de reţea!** In cazul televizoarelor portabile tip "Sport", sint date două posibilităţi de cuplare: pentru TV cu tranzistoare (fig. 1) şi pentru TV cu circuite integrate (fig.2).

In cazul in care modificările de mai sus nu sint posibile, in fig. 3 este prezentat un modulator RF care poate fi realizat intr-o cutie din tablă de conserve de mici dimensiuni montată chiar pe mufa BNC exterioară microcalculatorului. Elementele de circuit sint date pentru canalul 8 TV, dar pot fi uşor modificate pentru alte canale. Evident, calitatea imaginii va fi mai scăzută datorită spectrului larg de frecvenţă al semnalului video generat de microcalculator (8 MHz), faţă de lărgimea amplificatorului de medie frecvenţă a televizorului (5,5 MHz), dar posibilitatea cuplării la orice televizor compensează acest dezavantaj.

MONTAREA MECANICĂ ŞI CABLAREA MICROCALCULATORULUI

In figura 4 sint date cîteva sugestii de montare mecanică iar in figura 5 este dat planul de cablare generală a microcalculatorului. Se recomandă respectarea riguroasă a tipului de conectoare utilizat pentru a se asigura compatibilitatea. Intr-o primă fază nu este necesară montarea şi cablarea tuturor conectoarelor, cel mai important fiind insă cel notat in schemă cu "MISC".

Se va asigura o ventilaţie prin convecţie naturală suficientă prin găurirea plăcii de bază şi a capacului. In ce priveşte construcţia mecanică propriu-zisă, este recomandată folosirea tablei de aluminiu pentru a evita pătrunderea cimpurilor de radiofrecvenţă generate de emiţătoare, pe de o parte sau afectarea unor instalaţii de recepţie din apropiere, datorită sursei de alimentare sau a calculatorului propriu-zis. Montarea intrerupătorului de reţea şi a butonului de RESET se va face pe spatele cutiei, pentru a se evita acţionarea lor accidentală. Transformatorul de reţea utilizat poate fi cel de la televizorul "Sport".

POZITIONAREA JUMPERILOR

Inainte de montare, sau in cazul in care se schimbă tipul de EPROM utilizat, este necesară o repoziţionare a jumperilor din placă. Mai jos este dată semnificaţia şi modul de legare a acestora:

- J1 = linia de HOLD a procesorului; se leagă 1 cu 2 pentru lucru normal sau 2 cu 3 in conjuncţie cu un alt BUS-MASTER exterior.
- J2, J3 = tipul de EPROM utilizat pentru generatorul de caractere; pentru 2708 se leagă J2, 1 cu 2 şi J3, 2 cu 3; pentru 2716 se leagă J2, 2 cu 3 şi J3, 1 cu 2.
- J4 = semnalul video: 1 cu 2 normal; 2 cu 3 inversat.
- J5, J6 şi J7 = configurarea după tipul de EPROM utilizat in sistem. Atenţie! nu se poate utiliza decit un singur tip de EPROM in toate soclurile de pe placă (adică ori 2708 de la un capăt la altul ori 2716 etc.).

se leagă	2708	2716	2732
J5	1 cu 6	2 cu 6	3 cu 6
	2 cu 7	3 cu 7	4 cu 7
	3 cu 8	4 cu 8	5 cu 8
J6	1 cu 4	1 cu 3	1 cu 2
J7	1 cu 3	1 cu 2	1 cu 2

Placa de bază a microcalculatorului L/B881 poate fi expandată extern folosind conectorul KA, care conţine toate BUS-urile şi semnalele importante ale sistemului. Organizarea memoriei interne a microcalculatorului este pe 4 pagini de cite 16 kbytes, astfel:

* de la 0 la 3FFF - pagina de ROM (EPROM)
* de la 4000 la 7FFF - prima pagină de RAM
* de la 8000 la BFFF - a doua pagină de RAM
* de la C000 la FFFF - a treia pagină de RAM in care se află şi zona ecranului, stiva şi variabilele monitorului, fiind singura pagină absolut necesară microcalculatorului.

In ce priveşte expandarea cu circuite de intrare/ieşire, subliniem că BUS-urile de date şi control sint de tipul three-state, in timp ce cel de adrese este open-collector, lucruri de care trebuie ţinut seama cind se proiectează un controlor extern. Deasemeni, se va urmări ca adresele perifericelor externe să nu se suprapună peste cele ale perifericelor din sistem, care sint:

* de la 0 la 1F (int. controller şi timer)
* de la 30 la 3F (USART)
* de la 60 la 7F (interfeţe paralele)

Tot pe conectorul KA este scos şi semnalul BUSEN (către 8228), care are un ştrap la masă pe placă ce trebuie eliminat in cazul utilizării externe.

In continuare sint date semnificaţiile diferitelor conectoare şi insistăm asupra importanţei respectării standardizării propuse la legarea lor, pentru a permite interconectarea unor periferice externe.

Semnificaţia pinilor conectoarelor la placa L/B881

Nr.pin	KA	KB	KC
0	MEMW	GATE1	PA42
1	MEMR	GATE0	PA41
2	BUSEN	GATE2	PA52
3	HLDA	CLK0	PA51
4	I/OW	CLK1	PA62
5	I/OR	CLK2	PA61
6	NC	SYNC	PA72
7	NC	VIDEO	PA71
8	NC	OUT2	PA32
9	NC	IR6	PA31
10	NC	OUT0	PA22
11	HOLD	IR2	PA21
12	NC	IR0	PA12
13	WAIT	INTA	PA11
14	NC	IR3	PA02
15	NC	IR4	PA01
16	A0	NC	NC
17	D0	RESIN	NC
18	A1	NC	NC
19	D1	INT	GND
20	A2	NC	PC72
21	D2	RXRDY	PC71
22	A3	NC	PC62
23	D3	RXD	PC61
24	A4	RX/TXCLOCK	PC52
25	D4	OUT1	PC51
26	A5	NC	PC42
27	D5	NC	PC41
28	NC	DTR	PC02
29	D6	NC	PC01
30	A6	DSR	PC12
31	D7	TXRDY	PC11
32	A7	SYNDET	PC22
33	NC	CTS	PC21
34	A8	TXEMPTY	PC32
35	NC	TXD	PC31
36	A9	RTS	PB02
37	NC	NC	PB01
38	A10	NC	PB12
39	READY	NC	PB11
40	A11	NC	PB22
41	NC	-5V (VBB)	PB21
42	A12	FI 2 (CLK)	PB72
43	NC	RESET OUT	PB71
44	A13	GND	PB62
45	NC	GND	PB61
46	A14	GND	PB52
47	RESIN	GND	PB51
48	A15	+5V (VCC)	PB42
49	GND	+12V (VDD)	PB41
50	NC	+5V (VCC)	PB32
51	GND	+12V (VDD)	PB31

Semnificaţia pinilor conectoarelor externe

* Conectorul de BUS

1.	MEMW	20.	MEMR
2.	BUSEN	21.	HLDA
3.	I/OW	22.	I/OR
4.	A0	23.	HOLD
5.	A1	24.	WAIT
6.	A2	25.	D0
7.	A3	26.	D1
8.	A4	27.	D2
9.	A5	28.	D3
10.	A6	29.	D4
11.	A7	30.	D5
12.	A8	31.	D6
13.	A9	32.	D7
14.	A10	33.	READY
15.	A11	34.	RESIN
16.	A12	35.	GND
17.	A13	36.	RESET
18.	A14	37.	FI2 TTL
19.	A15		

* Conectorul MISC

1.	+Vcc	14.	OUT0
2.	GATE0	15.	OUT1
3.	GATE1	16.	OUT2
4.	GATE2	17.	IR0
5.	CLK0	18.	IR2
6.	CLK1	19.	IR3
7.	CLK2	20.	IR6
8.	FI2 TTL	21.	N.C.
9.	N.C	22.	PC4
10.	PC1	23.	PC5
11.	PC2	24.	PC6
12.	PC3	25.	PC7
13.	GND		

* Conectorul V24 serial port

1. Protective ground
2. Tx Data
3. Rx Data
4. RTS
5. CTS
6. DSR
7. Signal Ground
20. DTR

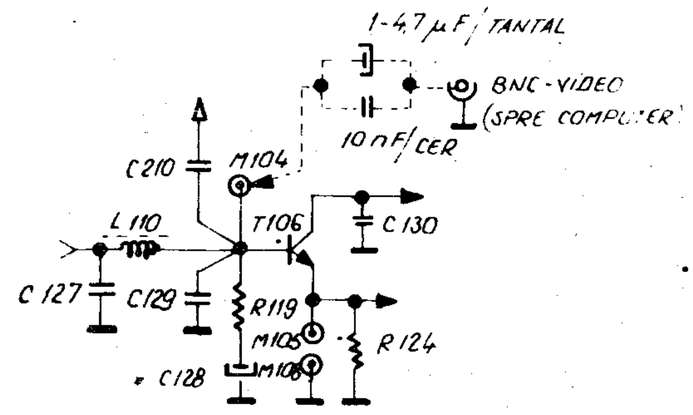
* Conectorul portului paralel opţional

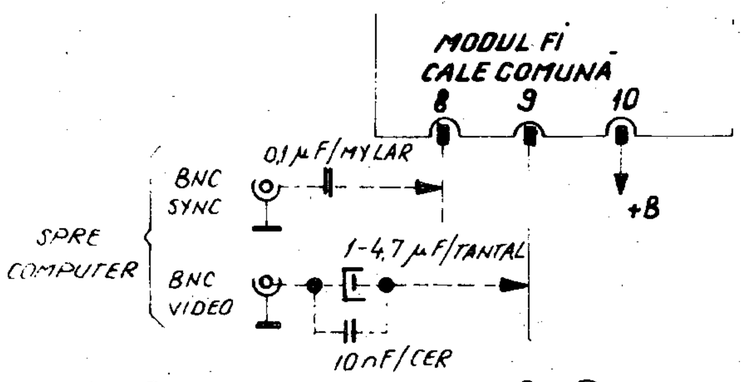
1.	PA0	14.	PB0
2.	PA1	15.	PB1
3.	PA2	16.	PB2
4.	PA3	17.	PB3
5.	PA4	18.	PB4
6.	PA5	19.	PB5
7.	PA6	20.	PB6
8.	PA7	21.	PB7
9.	PC0	22.	PC4
10.	PC1	23.	PC5
11.	PC2	24.	PC6
12.	PC3	25.	PC7
13.	GND		

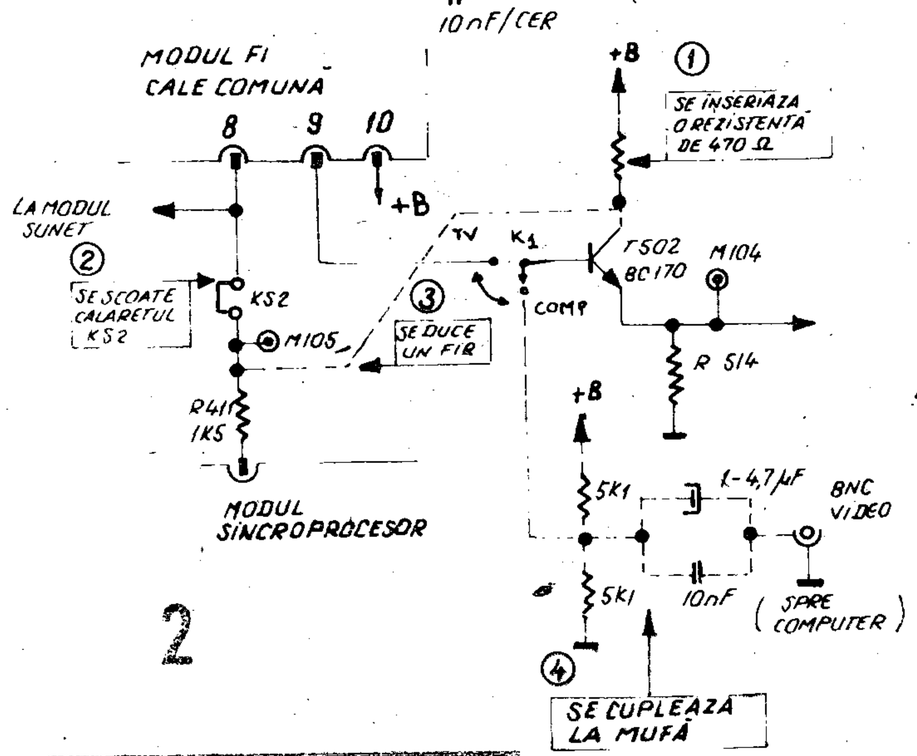
* Conectorul PSPY
(pentru alimentări in exterior)

1. -5v
3. +5v
5. +12v
7. GND
8. GND

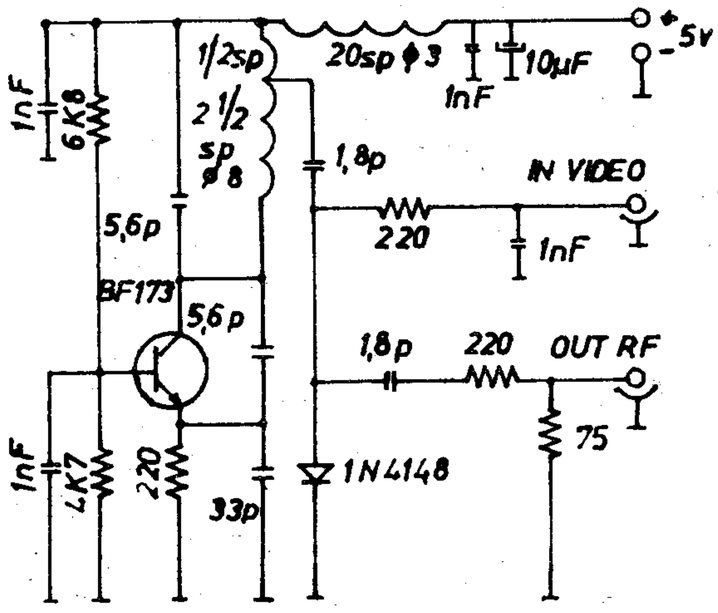
In numărul viitor vom continua cu descrierea etapelor de punere in funcţiune a microcalculatorului prin publicarea listingului generatorului de caractere şi a unui program de test.





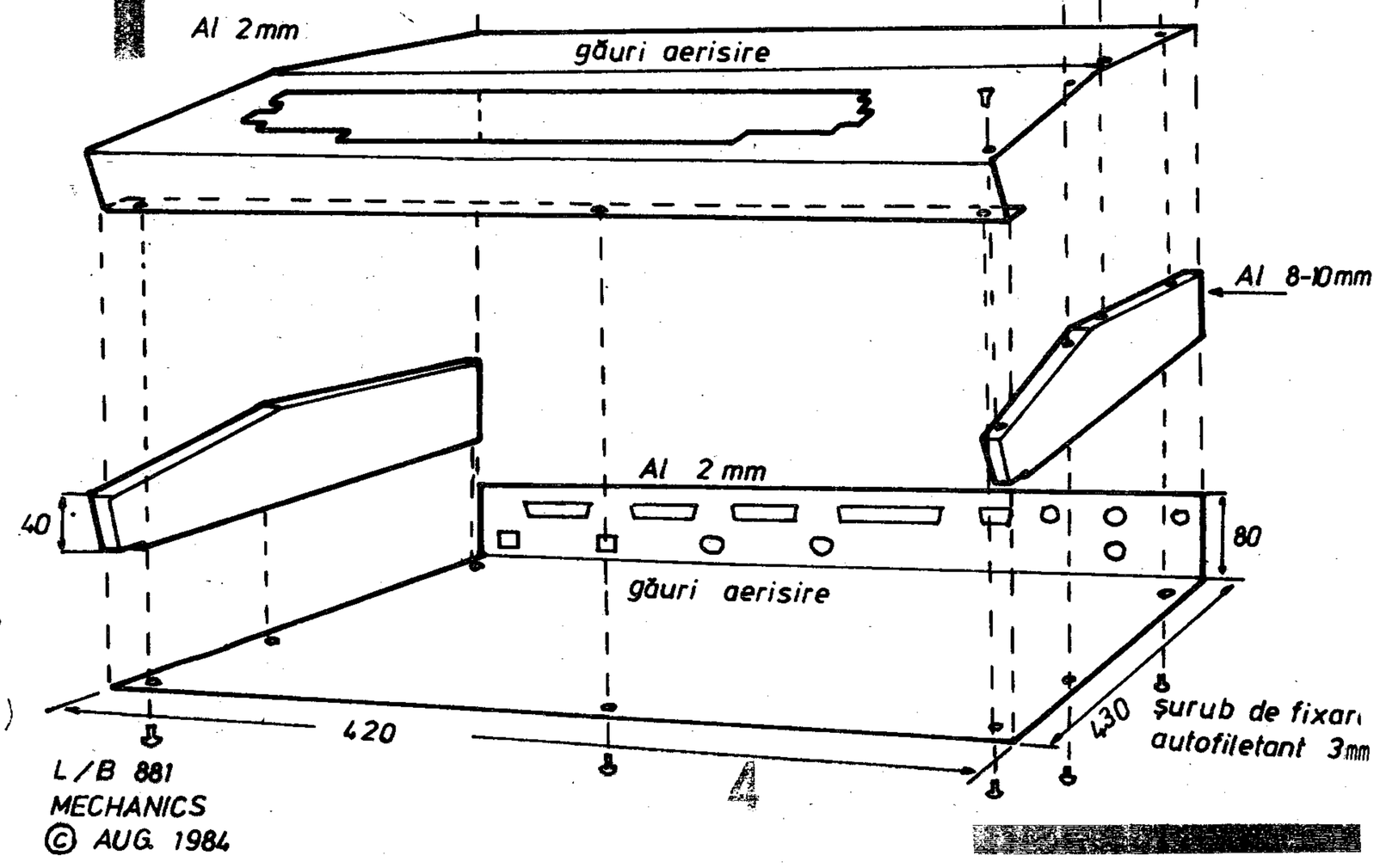


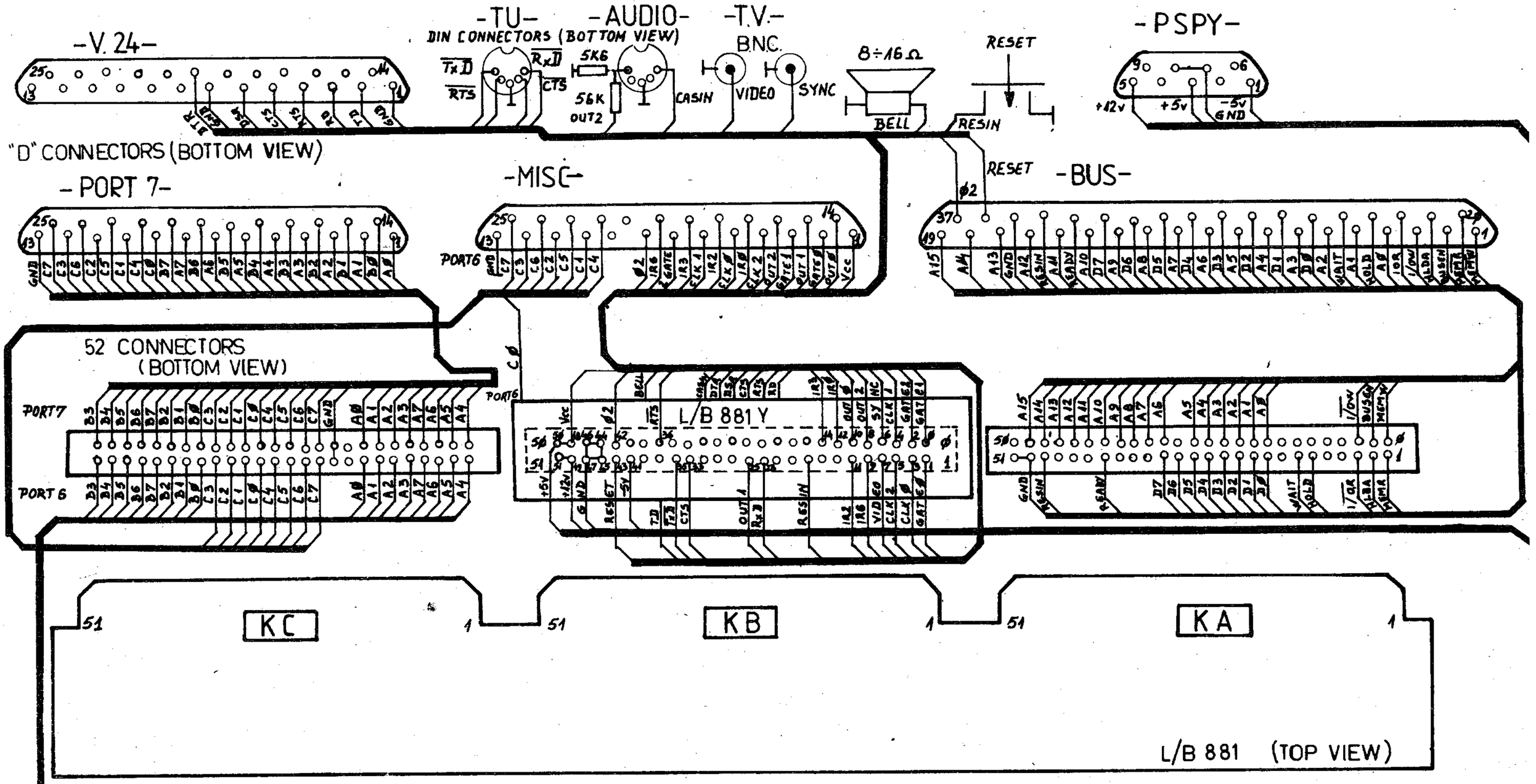
2

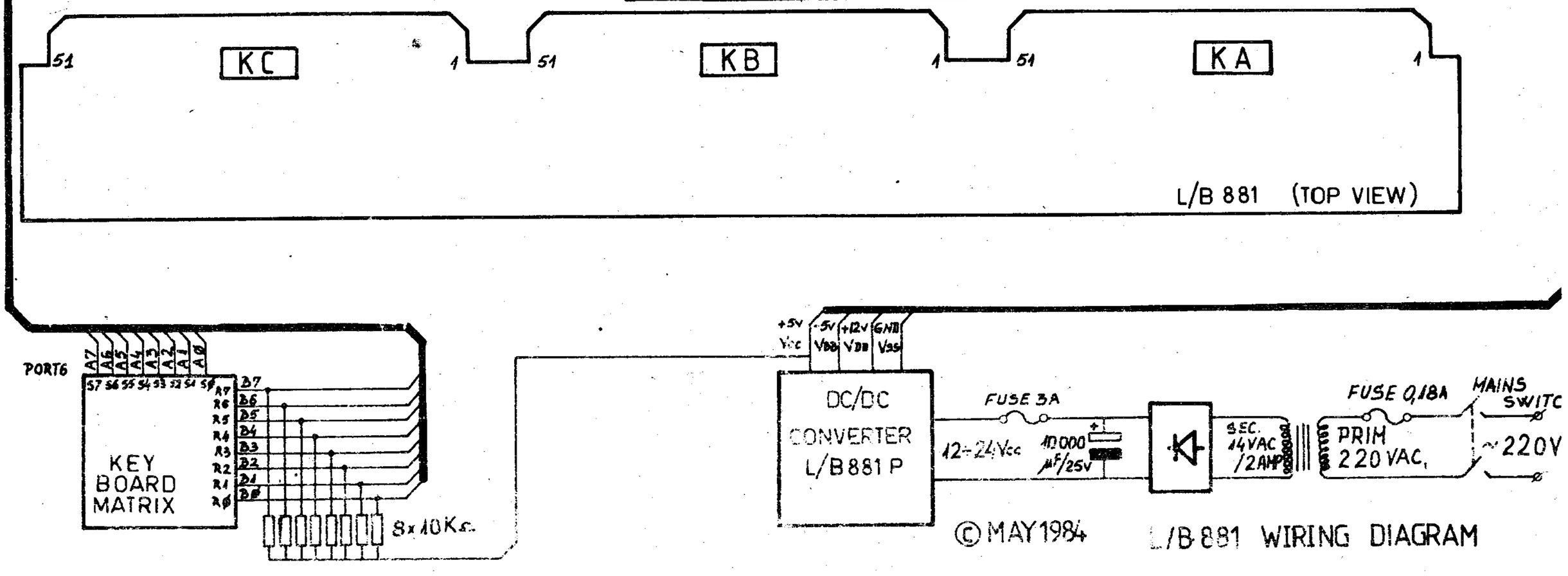


3

(CONTINUARE ÎN NR. VIITOR)







MICROCALCULATORUL L/B 881

NICOARA PAULIAN
ION RUSOVICI
GHEORGHE CHITA
LIVIU IONESCU

Realizarea practică a plăcii poate fi făcută fie in wrapping fie pe un cablaj imprimat. Față de cele publicate in numărul din decembrie 1985, facem precizarea că in schema de principiu nu au fost trecute condensatoarele de decuplare de pe alimentare. Cei ce vor realiza placa trebuie să țină seama că trebuie montat cite un condensator multistrat (10-100 nF) la fiecare grup de cite 5 TTL-uri și la fiecare memorie dinamică. Deasemeni, cite un condensator cu tantal la fiecare banc de memorii dinamice (pe fiecare tensiune) și din loc in loc răspindiți pe placă, in funcție de configurația existentă. Placa originală realizată de autori conține 18 condensatori cu tantal și 41 de condensatori multistrat. Incă un sfat: evitați pe cit posibil utilizarea de socluri la circuitele integrate (mărginiți-vă numai la EPROM-uri).

Vom descrie in continuare modul de punere in funcțiune a microcalculatorului L/B881. Se presupune că placa a fost realizată, sursa de alimentare a fost testată și furnizează toate tensiunile corecte, cuplajul cu televizorul a fost făcut ca și toate celelalte conexiuni ale plăcii cu exteriorul. Inainte de a cupla tensiunile, este necesară implantarea generatorului de caractere; conținutul acestuia este dat in tabela de pe această pagină. Se observă că este vorba de un EPROM de tipul 2716 organizat intr-o matrice 9x6.

Se trece la testarea propriu-zisă. Se incepe prin a cupla tensiunile de alimentare, măsurind consumurile pe cele trei tensiuni. Acestea sint:

- intre 1,5 și 1,8 A pe 5 volți;
- intre 250 și 800 mA pe 12 volți (diferența mare se datorește numărului de memorii implantate care poate varia intre 8 și 24).
- aprox. 5 mA pe -5 volți (in cazul in care EPROM-urile folosite in sistem sint de tipul 2708, consumul pe această ramură va crește pină la aprox 150 mA). Dacă există diferențe mari față de cele de mai sus, se vor urmări traseele (scurt-circuite sau intreruperi), condensatorii cu tantal sau eventuala piesă defectă.

Urmează apoi testarea controlorului video. Se verifică oscilatorul cu cristal, lanțul de numărătoare pe orizontală și verticală. Cu ajutorul unui osciloscop se verifică prezența semnalelor de sincronizare pe verticală (pinul 12 de la U25) și orizontală (pinul 13 de la U25). In acest moment, ar trebui deja ca pe ecranul televizorului să apară niște semne (alfanumerice sau grafice); in caz contrar, se verifică toată partea de timing a microsistemului, ca și serializatorul cu cele două 7495 (U32 și U33), blanking-ul ecranului (U9), generatoarele semnalelor RAS și CAS ale memoriilor dinamice și latch-ul video (U43). Dacă totul este insă normal, se poate trece la partea a doua a testărilor și anume punerea in funcțiune a procesorului (U13).

Pentru ușurarea acestei sarcini, se va programa un EPROM cu un program de test special conceput ca prin rularea lui să se poată identifica și elimina rapid eventualele probleme. Programul de test prezentat in continuare in format sursă, are și rolul de a demonstra cititorilor modul de abordare și rezolvare a unei probleme de software in limbaj de asamblare. Pentru cititorii neavizați: dacă programul pare greu sau de loc de ințeles, nu trebuie să vă alarmați, cu timpul lucrurile se vor lămuri. Pentru moment, trebuie să introduceți codul obiect generat prin asamblarea programului sursă intr-un EPROM de 16 Kocteți (2716). Citeva lămuriri: pe prima coloană a listingului sint date adresele absolute din memoria EPROM, iar in a doua și a treia conținutul lor (respectiv codurile instrucțiunilor și operanzii). Atenție insă, macroasamblorul M80 MicroSoft cu ajutorul căruia a fost asamblat programul sursă prezintă particularitatea că listează operanzii de doi octeți intr-un format diferit de standardul Intel, adică mai intii cel mai semnificativ octet și apoi cel mai puțin semnificativ, cum de altfel pare mai normal (toate procesoarele Intel extrag mai intii cel mai puțin semnificativ octet din memorie și apoi pe cel mai semnificativ); practic, va trebui ca la toți operanzii pe doi octeți să faceți o inversare la introducerea in memoria EPROM.

Exemplu (vezi listingul):

la adresa 0 se introduce 21
la adresa 1 se introduce 00
la adresa 2 se introduce F8
la adresa 3 se introduce 01
la adresa 4 se introduce 80
la adresa 5 se introduce 06
la adresa 6 se introduce 36
la adresa 7 se introduce 00

și așa mai departe.

In cazul in care constructorul are probleme cu programarea EPROM-urilor, poate lua legătura cu autorii (in scris) prin intermediul redacției.

Se introduce EPROM-ul in soclul marcat pe schemă ROMI (U35) și se repornește microcalculatorul.

Pentru a vedea ce ar trebui să se intimple, vom consulta listingul programului sursă. La inceput, programul presupune că totul in sistem este defect (mai puțin procesorul și memoria EPROM), drept care după o prealabilă ștergere a ecranului trece la testarea zonei de memorie folosită pentru stivă, operație care durează aprox. 3 secunde (de notat folosirea unui macro pentru afișarea locației defecte, ca urmare a inexistenței stivei). Dacă testul reușește, se afișează acest lucru și se trece la testarea memoriei din ecran (care se vede ca o agitație de caractere "ciudate" pe ecran). In cazul in care există o eroare la testarea stivei, programul va afișa mesajul "??? Memory error at XXXX: YY/ZZ", unde XXXX reprezintă adresa octetului eronat, YY valoarea citită și ZZ valoarea ce ar fi trebuit citită. Se vor verifica adresele de la multiplexoare și memorii, precum și semnalele de date care vin/pleacă la/dela memoriile dinamice; o altă sursă de erori poate proveni și din selectorul de memorii (U3).

In numărul următor vom continua cu partea a doua a listingului programului de test și vom comenta totodată fazele de punere la punct a microcalculatorului, in paralel cu descrierea funcționării programului.

2716 JOB: display data buffer <CR>

```
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010 38 38 38 00 00 00 00 00 00 00 00 00 00 00 00 00
0020 07 07 07 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 3F 3F 3F 00 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 38 38 38 00 00 00 00 00 00 00 00 00 00
0050 38 38 38 38 38 38 00 00 00 00 00 00 00 00 00 00
0060 07 07 07 38 38 38 00 00 00 00 00 00 00 00 00 00
0070 3F 3F 3F 38 38 38 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 07 07 07 00 00 00 00 00 00 00 00 00 00
0090 38 38 38 07 07 07 00 00 00 00 00 00 00 00 00 00
00A0 07 07 07 07 07 07 00 00 00 00 00 00 00 00 00 00
00B0 3F 3F 3F 07 07 07 00 00 00 00 00 00 00 00 00 00
00C0 00 00 00 3F 3F 3F 00 00 00 00 00 00 00 00 00 00
00D0 38 38 38 3F 3F 3F 00 00 00 00 00 00 00 00 00 00
00E0 07 07 07 3F 3F 3F 00 00 00 00 00 00 00 00 00 00
00F0 3F 3F 3F 3F 3F 3F 00 00 00 00 00 00 00 00 00 00
0100 00 00 00 00 00 00 38 38 38 00 00 00 00 00 00 00
0110 38 38 38 00 00 00 38 38 38 00 00 00 00 00 00 00
0120 07 07 07 00 00 00 38 38 38 00 00 00 00 00 00 00
0130 3F 3F 3F 00 00 00 38 38 38 00 00 00 00 00 00 00
0140 00 00 00 38 38 38 38 38 38 00 00 00 00 00 00 00
0150 38 38 38 38 38 38 38 38 38 00 00 00 00 00 00 00
0160 07 07 07 38 38 38 38 38 38 00 00 00 00 00 00 00
0170 3F 3F 3F 38 38 38 38 38 38 00 00 00 00 00 00 00
0180 00 00 00 07 07 07 38 38 38 00 00 00 00 00 00 00
0190 38 38 38 07 07 07 38 38 38 00 00 00 00 00 00 00
01A0 07 07 07 07 07 07 38 38 38 00 00 00 00 00 00 00
01B0 3F 3F 3F 07 07 07 38 38 38 00 00 00 00 00 00 00
01C0 00 00 00 3F 3F 3F 38 38 38 00 00 00 00 00 00 00
01D0 38 38 38 3F 3F 3F 38 38 38 00 00 00 00 00 00 00
01E0 07 07 07 3F 3F 3F 38 38 38 00 00 00 00 00 00 00
01F0 3F 3F 3F 3F 3F 3F 38 38 38 00 00 00 00 00 00 00
0200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0210 00 04 04 04 04 04 00 04 00 00 00 00 00 00 00 00
0220 00 0A 0A 0A 00 00 00 00 00 00 00 00 00 00 00 00
0230 00 0A 0A 1F 0A 1F 0A 0A 00 00 00 00 00 00 00 00
0240 00 04 0F 14 0E 05 1E 04 00 00 00 00 00 00 00 00
0250 00 18 19 02 04 08 13 03 00 00 00 00 00 00 00 00
0260 00 08 14 14 08 15 12 0D 00 00 00 00 00 00 00 00
0270 00 02 04 08 00 00 00 00 00 00 00 00 00 00 00 00
0280 00 04 08 10 10 10 08 04 00 00 00 00 00 00 00 00
0290 00 04 02 01 01 01 02 04 00 00 00 00 00 00 00 00
02A0 00 00 04 15 0E 15 04 00 00 00 00 00 00 00 00 00
02B0 00 00 04 04 1F 04 04 00 00 00 00 00 00 00 00 00
02C0 00 00 00 00 00 00 04 04 08 00 00 00 00 00 00 00
02D0 00 00 00 00 1F 00 00 00 00 00 00 00 00 00 00 00
02E0 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 00
02F0 00 00 01 02 04 08 10 00 00 00 00 00 00 00 00 00
0300 00 0E 11 13 15 19 11 0E 00 00 00 00 00 00 00 00
0310 00 04 0C 04 04 04 04 0E 00 00 00 00 00 00 00 00
0320 00 0E 11 01 02 04 08 1F 00 00 00 00 00 00 00 00
0330 00 0E 11 01 06 01 11 0E 00 00 00 00 00 00 00 00
0340 00 02 06 0A 12 1F 02 02 00 00 00 00 00 00 00 00
0350 00 1F 10 1E 01 01 11 0E 00 00 00 00 00 00 00 00
0360 00 02 04 08 0E 11 11 0E 00 00 00 00 00 00 00 00
0370 00 1F 01 01 02 04 08 10 00 00 00 00 00 00 00 00
0380 00 0E 11 11 0E 11 11 0E 00 00 00 00 00 00 00 00
0390 00 0E 11 11 0E 02 04 08 00 00 00 00 00 00 00 00
03A0 00 00 00 04 00 00 00 04 00 00 00 00 00 00 00 00
03B0 00 00 00 04 00 00 04 04 08 00 00 00 00 00 00 00
03C0 00 02 04 08 10 08 04 02 00 00 00 00 00 00 00 00
03D0 00 00 00 1F 00 1F 00 00 00 00 00 00 00 00 00 00
03E0 00 08 04 02 01 02 04 08 00 00 00 00 00 00 00 00
03F0 00 0E 11 02 04 04 00 04 00 00 00 00 00 00 00 00
0400 00 0E 11 15 17 16 10 0F 00 00 00 00 00 00 00 00
0410 00 04 0A 11 11 1F 11 11 00 00 00 00 00 00 00 00
0420 00 1E 09 09 0E 09 09 1E 00 00 00 00 00 00 00 00
0430 00 0E 11 10 10 10 11 0E 00 00 00 00 00 00 00 00
0440 00 1E 09 09 09 09 09 1E 00 00 00 00 00 00 00 00
0450 00 1F 10 10 1E 10 10 1F 00 00 00 00 00 00 00 00
0460 00 1F 10 10 1E 10 10 10 00 00 00 00 00 00 00 00
0470 00 0F 10 10 10 13 11 0F 00 00 00 00 00 00 00 00
0480 00 11 11 11 1F 11 11 11 00 00 00 00 00 00 00 00
0490 00 0E 04 04 04 04 04 0E 00 00 00 00 00 00 00 00
04A0 00 01 01 01 01 01 11 0E 00 00 00 00 00 00 00 00
04B0 00 11 12 14 18 14 12 11 00 00 00 00 00 00 00 00
04C0 00 10 10 10 10 10 10 1F 00 00 00 00 00 00 00 00
04D0 00 11 1B 15 15 11 11 11 00 00 00 00 00 00 00 00
04E0 00 11 11 19 15 13 11 11 00 00 00 00 00 00 00 00
04F0 00 0E 11 11 11 11 11 0E 00 00 00 00 00 00 00 00
0500 00 1E 11 11 1E 10 10 10 00 00 00 00 00 00 00 00
0510 00 0E 11 11 11 15 12 0D 00 00 00 00 00 00 00 00
0520 00 1E 11 11 1E 14 12 11 00 00 00 00 00 00 00 00
0530 00 0E 11 10 0E 01 11 0E 00 00 00 00 00 00 00 00
0540 00 1F 04 04 04 04 04 04 00 00 00 00 00 00 00 00
0550 00 11 11 11 11 11 11 0E 00 00 00 00 00 00 00 00
0560 00 11 11 11 0A 0A 04 04 00 00 00 00 00 00 00 00
0570 00 11 11 11 15 15 1B 11 00 00 00 00 00 00 00 00
0580 00 11 11 0A 04 0A 11 11 00 00 00 00 00 00 00 00
0590 00 11 11 0A 04 04 04 04 00 00 00 00 00 00 00 00
05A0 00 1F 01 02 04 08 10 1F 00 00 00 00 00 00 00 00
05B0 00 1F 18 18 18 18 18 1F 00 00 00 00 00 00 00 00
05C0 00 00 10 08 04 02 01 00 00 00 00 00 00 00 00 00
05D0 00 1F 03 03 03 03 03 1F 00 00 00 00 00 00 00 00
05E0 00 00 00 04 0A 11 00 00 00 00 00 00 00 00 00 00
05F0 00 00 00 00 00 00 00 1F 00 00 00 00 00 00 00 00
0600 00 08 04 02 00 00 00 00 00 00 00 00 00 00 00 00
0610 00 00 00 0E 01 0F 11 0F 00 00 00 00 00 00 00 00
0620 00 10 10 1E 11 11 11 1E 00 00 00 00 00 00 00 00
0630 00 00 00 0E 10 10 10 0E 00 00 00 00 00 00 00 00
0640 00 01 01 0F 11 11 11 0F 00 00 00 00 00 00 00 00
0650 00 00 00 0E 11 1F 10 0E 00 00 00 00 00 00 00 00
0660 00 02 04 04 0E 04 04 04 00 00 00 00 00 00 00 00
0670 00 00 00 0F 11 11 0F 01 06 00 00 00 00 00 00 00
0680 00 10 10 1E 11 11 11 11 00 00 00 00 00 00 00 00
0690 00 04 00 0C 04 04 04 0E 00 00 00 00 00 00 00 00
06A0 00 02 00 06 02 02 02 12 0C 00 00 00 00 00 00 00
06B0 00 10 10 12 14 18 14 12 00 00 00 00 00 00 00 00
06C0 00 0C 04 04 04 04 04 0E 00 00 00 00 00 00 00 00
06D0 00 00 00 1A 15 15 15 15 00 00 00 00 00 00 00 00
06E0 00 00 00 1E 11 11 11 11 00 00 00 00 00 00 00 00
06F0 00 00 00 0E 11 11 11 0E 00 00 00 00 00 00 00 00
0700 00 00 00 1E 11 11 1E 10 10 00 00 00 00 00 00 00
0710 00 00 00 0F 11 11 0F 01 01 00 00 00 00 00 00 00
0720 00 00 00 16 18 10 10 10 00 00 00 00 00 00 00 00
0730 00 00 00 0E 10 0E 01 1E 00 00 00 00 00 00 00 00
0740 00 04 04 0E 04 04 04 02 00 00 00 00 00 00 00 00
0750 00 00 00 11 11 11 11 0E 00 00 00 00 00 00 00 00
0760 00 00 00 11 11 11 0A 04 00 00 00 00 00 00 00 00
0770 00 00 00 11 11 15 15 0A 00 00 00 00 00 00 00 00
0780 00 00 00 11 0A 04 0A 11 00 00 00 00 00 00 00 00
0790 00 00 00 11 11 11 0F 01 06 00 00 00 00 00 00 00
07A0 00 00 00 1F 02 04 08 1F 00 00 00 00 00 00 00 00
07B0 00 02 04 04 08 04 04 02 00 00 00 00 00 00 00 00
07C0 00 04 04 04 00 04 04 04 00 00 00 00 00 00 00 00
07D0 00 08 04 04 02 04 04 08 00 00 00 00 00 00 00 00
07E0 00 00 00 01 0E 10 00 00 00 00 00 00 00 00 00 00
07F0 15 2A 15 2A 15 2A 15 2A 15 00 00 00 00 00 00 00
```

```
                        title   881/Test V2.1 (C) 1986 Lixco Software
                        subttl  Hardware Test


                ;       created         sep. 1985

                ;       last revision   18 feb. 1986

                ;       Programul 881/Test se găseşte intr-un EPROM de la
                ; adresa 0 şi execută următoarele teste (in ordine):
                ; - testarea memoriei stivă;
                ; - testarea memoriei de ecran;
                ; - testarea pe rind a celor trei bancuri de memorie;
                ; - testarea perifericelor (in ordine: 8259, 8255, 8253, 8251).
                ; In cazurile de insucces, programul afişează eroarea (locaţia
                ; de memorie defectă sau perifericul necorespunzător); la tes-
                ; tarea stivei, programul se opreşte in caz de insucces, in
                ; rest insă, trece la testul următor. Oprirea se face intr-o
                ; buclă care incrementează adresele pe magistrală de la 0 la
                ; FFFF hex, pentru o eventuală testare a lor cu osciloscopul.



                ;nmout  este o macroinstrucţiune folosită pentru tipărirea unui
                ;       număr hex de 8 biţi pe ecran. S-a recurs la un macro şi
                ;       nu la o subrutină pentru că stiva nu poate fi utilizată.
                ;       Input: A = octetul de afişat
                ;              HL = adresa unde urmează a fi afişat
                ;       Output: D = octetul de afişat
                ;       Distruge: AF, D, HL
                nmout   macro
                        mov     d,a     ; Salvare număr de tipărit
                        rrc             ; Conversie in ASCII a celor mai
                        rrc             ;  semnificativi 4 biţi
                        rrc
                        rrc
                        ani     00001111B
                        adi     '0'
                        cpi     '9'+1   ; Mai mare decit 9 ?
                        jc      $+5     ; Nu, salt (cifra este intre 0 şi 9)
                        adi     7       ; Altfel, este intre A şi F hex
                        mov     m,a     ; Afişare in memoria ecranului
                        inx     h
                        mov     a,d     ; Conversie in ASCII a celor mai puţin
                        ani     00001111B       ; semnificativi 4 biţi
                        adi     '0'
                        cpi     '9'+1
                        jc      $+5
                        adi     7
                        mov     m,a     ; Notă: pointerul pe ecran rămine
                        inx     h       ;       pregătit pentru o nouă afişare
                        endm

                        .phase  0

                ; *** Punctul de intrare in program ***

                ;       Sterge ecanul
0000  21 F800           lxi     h,rowA  ; Inceput ecran in HL
```



```
0003  01 0680                   lxi     b,rowlng*nrows  ; Contor lungime ecran in BC
0006                    loop1:
0006  36 00                     mvi     m,0     ; Umple memoria ecran cu 0
0008  23                       inx     h
0009  0B                       dcx     b
000A  78                       mov     a,b
000B  B1                       ora     c
000C  C2 0006                  jnz     loop1

                        ; *** Test RAM stivă şi variabile ***

000F  AF                       xra     a       ; Byte de test iniţial = 0
0010                    loop15:
0010  21 FFFF                  lxi     h,hotram
0013  01 017F                  lxi     b,hotram-endscr ; Test zona RAM intre sfirşit
0016                    loop2:                  ;  ecran şi ultima locaţie
0016  77                       mov     m,a     ; Inscrie byte de test
0017  BE                       cmp     m       ; ... şi apoi verifică
0018  C2 021C                  jnz     error   ; Eroare dacă nu se potriveşte
001B  2B                       dcx     h
001C  0B                       dcx     b
001D  5F                       mov     e,a     ; Salvare byte test
001E  78                       mov     a,b
001F  B1                       ora     c
0020  7B                       mov     a,e     ; Refacere byte test
0021  C2 0016                  jnz     loop2
0024  3C                       inr     a       ; Schimbă valoare byte test
0025  C2 0010                  jnz     loop15  ; Dacă nu am trecut prin toate valorile
                                                ;  continuă testul
0028  31 FF00                  lxi     sp,stack        ; RAM stivă OK, deci se poate
002B  21 F800                  lxi     h,rowA          ;  iniţializa
002E  22 FF35                  shld    mcurs           ; Iniţializare cursor
0031  21 0375                  lxi     h,stkmsg        ; Test reuşit, afişează mesaj
0034  CD 0334                  call    outstr

                        ; *** Test RAM ecran ***

0037  21 F840                  lxi     h,rowB          ; Inceput zonă test in HL
003A  11 FE80                  lxi     d,endscr        ; Sfirşit zonă test in DE
003D  CD 02BC                  call    ramtst
0040  DA 021C                  jc      error   ; Dacă ecranul este okay, ştergere
0043  21 F800                  lxi     h,rowA  ; Inceput ecran in HL
0046  01 0680                  lxi     b,rowlng*nrows  ; Contor lungime ecran in BC
0049                    clsc05:
0049  36 00                     mvi     m,0     ; Umple memoria ecran cu 0
004B  23                       inx     h
004C  0B                       dcx     b
004D  78                       mov     a,b
004E  B1                       ora     c
004F  C2 0049                  jnz     clsc05
0052  21 F800                  lxi     h,rowA          ; Test primul rind
0055  11 F83F                  lxi     d,rowB-1
0058  CD 02BC                  call    ramtst
005B  D2 0064                  jnc     tram05          ; Test reuşit, salt
005E  CD 0348                  call    teserr          ; Altfel, afişează locaţia
0061  C3 006A                  jmp     tram10          ;  defectă
0064                    tram05:
0064  21 038E                  lxi     h,scrok         ; Afişare rezultat favorabil
0067  CD 0334                  call    outstr
006A                    tram10:
006A  21 F800                  lxi     h,rowA          ; Sterge primul rind
006D  06 40                    mvi     b,rowlng
006F                    tram13:
006F  36 20                     mvi     m,' '
```



```
0071  23                       inx     h
0072  05                       dcr     b
0073  C2 006F                  jnz     tram13
0076  06 40                    mvi     b,rowlng        ; Afişare "miră" pentru
0078  21 FE40                  lxi     h,lastrw        ;  verificare video controler
007B                    pict05:
007B  3E 30                     mvi     a,'0'
007D                    pict10:
007D  77                        mov     m,a             ; Ultimul rind va fi numerotat
007E  23                        inx     h
007F  05                        dcr     b
0080  CA 008C                   jz      pict15
0083  3C                        inr     a
0084  FE 3A                     cpi     '9'+1
0086  C2 007D                   jnz     pict10
0089  C3 007B                   jmp     pict05
008C                    pict15:
008C  06 1A                     mvi     b,nrows         ; Se numerotează şi sfirşitul
008E  21 F83F                   lxi     h,rowB-1        ;  tuturor rindurilor
0091  11 0040                   lxi     d,rowlng
0094                    pict17:
0094  3E 30                     mvi     a,'0'
0096                    pict20:
0096  77                        mov     m,a
0097  19                        dad     d
0098  05                        dcr     b
0099  CA 00A5                   jz      pict25
009C  3C                        inr     a
009D  FE 3A                     cpi     '9'+1
009F  C2 0096                   jnz     pict20
00A2  C3 0094                   jmp     pict17
00A5                    pict25:
00A5  21 FDC0                   lxi     h,row22         ; Afişează miră grafică pe
00A8  0E 1F                     mvi     c,(rowlng/2)-1  ;  rindul 22
00AA  16 20                     mvi     d,' '
00AC  1E 9F                     mvi     e,graph
00AE                    pict30:
00AE  72                        mov     m,d
00AF  23                        inx     h
00B0  73                        mov     m,e
00B1  23                        inx     h
00B2  0D                        dcr     c
00B3  C2 00AE                   jnz     pict30
00B6  21 F800                   lxi     h,rowA          ; Se aduce cursorul pe
00B9  22 FF35                   shld    mcurs           ;  primul rind
00BC  21 0375                   lxi     h,stkmsg        ; ... şi se reface primul masaj
00BF  CD 0334                   call    outstr
00C2  21 03A8                   lxi     h,patmsg        ; Avertizează utilizatorul
00C5  CD 0334                   call    outstr          ;  de durata testului următor
00C8  21 F880                   lxi     h,rowA+(2*rowlng)       ; Poziţionare cursor
00CB  22 FF35                   shld    mcurs                   ;  pe rindul următor

                        ; *** Test bancuri memorie ***

00CE  21 C000                   lxi     h,bank1         ; Verificare primul banc
00D1  11 F7FF                   lxi     d,rowA-1        ;  pină in ecran
00D4  CD 02BC                   call    ramtst
00D7  D2 00E0                   jnc     tram15          ; Test reuşit, salt
00DA  CD 0348                   call    teserr          ; Altfel, afişează locaţia
00DD  C3 00E6                   jmp     tram20          ;  defectă
00E0                    tram15:
00E0  21 03E8                   lxi     h,bnk1ok        ; Afişare primul bank ok
00E3  CD 0334                   call    outstr
00E6                    tram20:
```



```
00E6  21 8000                   lxi     h,bank2         ; Verificare al doilea banc
00E9  11 BFFF                   lxi     d,bank1-1
00EC  CD 02BC                   call    ramtst
00EF  D2 00F8                   jnc     tram30
00F2  CD 0348                   call    teserr
00F5  C3 00FE                   jmp     tram35
00F8                    tram30:
00F8  21 041D                   lxi     h,bnk2ok
00FB  CD 0334                   call    outstr
00FE                    tram35:
00FE  21 4000                   lxi     h,bank3         ; Verificare al treilea banc
0101  11 7FFF                   lxi     d,bank2-1
0104  CD 02BC                   call    ramtst
0107  D2 0110                   jnc     tram45
010A  CD 0348                   call    teserr
010D  C3 0116                   jmp     tint
0110                    tram45:
0110  21 043C                   lxi     h,bnk3ok
0113  CD 0334                   call    outstr

                        ; *** Test controler intreruperi ***

0116                    tint:
0116  21 FA00                   lxi     h,row7  ; Poziţionare cursor
0119  22 FF35                   shld    mcurs
011C  3E C3                     mvi     a,(jmp) ; Iniţializare vector intreruperi
011E  32 FF04                   sta     ir1
0121  21 02B0                   lxi     h,irout1
0124  22 FF05                   shld    ir1+1
0127  3E 16                     mvi     a,icw1  ; Iniţializare PIC 8259
0129  D3 00                     out     intct0
012B  3E FF                     mvi     a,icw2
012D  D3 01                     out     intct1
012F  3E FD                     mvi     a,mask1
0131  D3 01                     out     intct1
0133  AF                       xra     a       ; Valoare iniţială pentru test
0134  32 FFF0                   sta     tesloc
0137  FB                       ei
0138  CD 033E                   call    delay   ; ... aşteaptă o intrerupere
013B  3A FFF0                   lda     tesloc  ; Verifică dacă s-a executat rutina
013E  3C                       inr     a       ;  pe intrupere
013F  CA 014B                   jz      tint05  ; Da, salt (totul este ok)
0142  21 045B                   lxi     h,interr; Nu, afişare eroare
0145  CD 0334                   call    outstr
0148  C3 0151                   jmp     tppi
014B                    tint05:
014B  21 0474                   lxi     h,irlok
014E  CD 0334                   call    outstr

                        ; *** Test porturi paralele ***

0151                    tppi:
0151  3E 80                     mvi     a,pcw   ; Configurare toate porturile pe output
0153  D3 63                     out     parsta
0155  3E 55                     mvi     a,vala  ; Inscriere porturi
0157  D3 60                     out     porta
0159  3E AA                     mvi     a,valb
015B  D3 61                     out     portb
015D  3E 5A                     mvi     a,valc
015F  D3 62                     out     portc
0161  DB 60                     in      porta   ; Citire + verificare
0163  FE 55                     cpi     vala    ; Port A corect ?
0165  C2 017F                   jnz     tppi05  ; Nu, salt
0168  DB 61                     in      portb   ; Port B corect ?
```

MICROCALCULATORUL L/B 881

NICOARA PAULIAN
ION RUSOVICI
GHEORGHE CHITA
LIVIU IONESCU

Continuăm cu descrierea punerii la punct a microcalculatorului.

In cazul in care la reset nu se intimplă nimic (adică ştergerea ecranului), este de presupus că ansamblul microprocesor-PROM nu funcţionează corect. Se vor verifica cele două circuite (8080 şi 8228) precum şi logica de selecţie a PROM-urilor şi generatoarele de ceas.

După trecerea testelor de memorie a stivei şi ecranului, programul incepe testarea sistematică a memoriei RAM principale (bancul de la C000 la F7FF hex). Si in acest caz, prima eroare detectată este afişată după care programul se opreşte intr-o buclă ce uşurează testarea mai departe cu osciloscopul (se execută o instrucţiune MOV A,M incrementind registrul index HL - vezi listingul la adresa 2ABh, eticheta TEND).

Inainte de testarea RAM-urilor, programul afişează o "miră" constituită din numerotarea pe verticală şi orizontală a numărului de caractere a ecranului; se poate intimpla să lipsească unele cifre, sau să fie dublate, caz in care defecţiunea trebuie căutată in lanţul de numărătoare care baleiază ecranul sau in multiplexoare. De exemplu, dacă lipsesc primele două rinduri de sus ale ecranului, problema apare datorită unui 7493 necorespunzător in locul lui U29. In unele cazuri se poate remedia fără inlocuirea circuitului montind un condensator de aprox. 10 nF intre pinul 3 şi masă. Dacă insă apare o puternică desincronizare pe orizontală, cauza poate fi in grupul U19-U28; şi aici se poate incerca montarea unui condensator de aprox. 500 pF intre pinul 11/U23 şi masă. Dacă defecţiunea persistă, se vor schimba circuitele.

Tot pe "miră" se va constata prezenţa unor caractere grafice in partea de jos a ecranului; succesiunea lor a fost aleasă in aşa fel incit să pună in evidenţă eventualele imperfecţiuni de timing in comutarea pe video-reverse a caracterelor serializate (intirzieri diferite pe porţile care constituie partea de ieşire video a microcalculatorului). Dacă punctele ce formează şirul de caractere grafice sint curate, fără liniuţe verticale, sistemul funcţionează corect; in caz contrar trebuie executată o compensare folosind condensatori de valori intre 180 şi 360 pF ce se vor monta prin tatonare fie intre pinul 10/U33 şi masă, fie intre pinul 3/U25 şi masă (există situaţii cind trebuie montate ambele condensatoare).

In continuare programul testează şi celelalte bancuri de memorie (in cazul in care ele există), apoi trece la testarea perifericelor din sistem. Trebuie remarcat că pentru o corectă executare a acestei porţiuni a programului, este necesar să se cupleze exterior (pe conectorul KB) semnalele de TxD şi RxD impreună, precum şi semnalul CTS la masă. Deasemeni, este necesar ca CLK1, 2 şi 3 să fie legate la FI2, GATE0, 1 şi 2 la +Vcc prin rezistenţele de 10 K, şi OUT1 la Rx/TxClock (USART).

Testarea perifericelor include (in ordine) controlorul de intreruperi (8259), interfaţa paralelă programabilă de la adresa 60h (8255/U46), numărătorul programabil (8253) şi interfaţa serie programabilă USART (8251). De semnalat faptul că nefuncţionarea circuitului 8253 va semnala eroare şi pentru 8251, intrucit frecvenţa de ceas a acestuia din urmă este asigurată de către numărătorul programabil.

Funcţionarea incorectă a circuitelor periferice (unul sau mai multe) se poate datora fie unor defecte interne (se vor schimba circuitele), fie unor defecte provenite din modul de adresare şi selecţie a acestora. Se vor verifica in acest sens semnalele de CS (Chip Select), decodificatorul de adrese pentru periferice (U34), apariţia corectă a adreselor A0 şi/sau A1 pentru selecţia registrelor interne, semnalele I/OR şi I/OW, precum şi a bus-ului de date.

In momentul depăşirii acestor probleme puteţi considera operaţia de punere la punct a microcalculatorului terminată. Incepind din numărul viitor vom publica programul monitor standard al microcalculatorului L/B881. Evident, se poate folosi orice alt monitor, adaptat sau scris special in acest scop, dar pentru a se păstra compatibilitatea cu programele deja scrise este recomandată utilizarea monitorului 881/Mon.

Cu articolul din acest număr se incheie descrierea părţii hardware a microcalculatorului L/B881; mulţumiri din partea autorilor se cuvin pentru Rodica Avram, Tatiana Rusovici pentru numeroasele desene executate şi Neamţu Napoleon pentru sugestii valoroase la partea de mecanică şi design; deasemeni realizatorului primei machete, Gigi Alexandrescu, un pasionat al tehnicii microprocesoarelor, care ar fi citit cu bucurie rindurile de faţă.

```
881/Test V2.1 (C) 1986 Lixco Software  MACRO-80 3.36  17-Mar-80  PAGE  1-4
Hardware Test

016A  FE AA          cpi     valb
016C  C2 017F        jnz     tppi05
016F  DB 62          in      portc   ; Port C corect ?
0171  FE 5A          cpi     valc
0173  C2 017F        jnz     tppi05
0176  21 0489        lxi     h,ppiok ; Afişare rezultat ok
0179  CD 0334        call    outstr
017C  C3 0185        jmp     ttimer
017F                 tppi05:
017F  21 049E        lxi     h,ppibad        ; Afişare eroare
0182  CD 0334        call    outstr

                     ; *** Test timere ***
0185                 ttimer:
0185  3E 3E          mvi     a,cwt0; Iniţializare timere ca generatoare
0187  D3 13          out     timsta
0189  3E 7E          mvi     a,cwt1
018B  D3 13          out     timsta
018D  3E BE          mvi     a,cwt2
018F  D3 13          out     timsta
0191  AF             xra     a       ; Incărcare şi pornire timere
0192  D3 10          out     timer0
0194  D3 10          out     timer0
0196  D3 11          out     timer1
0198  D3 11          out     timer1
019A  D3 12          out     timer2
019C  D3 12          out     timer2
019E  3E 3E          mvi     a,cwt0  ; Opreşte timerele pentru citire
01A0  D3 13          out     timsta
01A2  3E 7E          mvi     a,cwt1
01A4  D3 13          out     timsta
01A6  3E BE          mvi     a,cwt2
01A8  D3 13          out     timsta
01AA  DB 10          in      timer0  ; Verificare valoare oprire
01AC  FE 92          cpi     val1
01AE  C2 01DD        jnz     ttim10
01B1  DB 10          in      timer0
01B3  FE FF          cpi     val0
01B5  C2 01DD        jnz     ttim10
01B8  DB 11          in      timer1
01BA  FE 98          cpi     val2
01BC  C2 01DD        jnz     ttim10
01BF  DB 11          in      timer1
01C1  FE FF          cpi     val0
01C3  C2 01DD        jnz     ttim10
01C6  DB 12          in      timer2
01C8  FE 9E          cpi     val3
01CA  C2 01DD        jnz     ttim10
01CD  DB 12          in      timer2
01CF  FE FF          cpi     val0
01D1  C2 01DD        jnz     ttim10
01D4  21 04B7        lxi     h,timok ; Test timere reuşit
01D7  CD 0334        call    outstr
01DA  C3 01E3        jmp     tusart
01DD                 ttim10:
01DD  21 04CC        lxi     h,timbad
01E0  CD 0334        call    outstr

                     ; *** Test USART ***
01E3                 tusart:
01E3  3E CE          mvi     a,mode  ; Iniţializare USART
01E5  D3 31          out     sersta
```

```
881/Test V2.1 (C) 1986 Lixco Software  MACRO-80 3.36  17-Mar-80  PAGE  1-5
Hardware Test

01E7  3E 37          mvi     a,cmd
01E9  D3 31          out     sersta
01EB  3E 7F          mvi     a,cwt10 ; Iniţializare timer care dă clock-ul
01ED  D3 13          out     timsta
01EF  3E 78          mvi     a,78h
01F1  D3 11          out     timer1
01F3  AF             xra     a
01F4  D3 11          out     timer1
01F6  3E AA          mvi     a,valb  ; Transmitere dată de test
01F8  D3 30          out     serdat  ;  ce va fi aşteptată la recepţie
01FA  CD 033E        call    delay   ; Aşteptare sfirşit emisie
01FD  DB 30          in      serdat  ; Citire dată recepţie
01FF  FE AA          cpi     valb    ; Este aceeaşi ?
0201  CA 020D        jz      tusar7  ; Da, salt (totul este ok)
0204  21 04E5        lxi     h,serbad
0207  CD 0334        call    outstr  ; Nu, afişare eroare
020A  C3 0213        jmp     over
020D                 tusar7:
020D  21 0504        lxi     h,serok
0210  CD 0334        call    outstr
                     ; *** Test terminat ***
0213                 over:
0213  21 051D        lxi     h,okmsg
0216  CD 0334        call    outstr
0219  C3 02AB        jmp     tend


                     ;       Rutina afişare locaţie de memorie defectă
021C                 error:
021C  EB             xchg
021D  21 F81A        lxi     h,rowA+26       ; Pregătire pointer pentru
0220  47             mov     b,a             ;  afişare
0221  1A             ldax    d               ; In B este valoarea inscrisă
0222  4F             mov     c,a             ; In C este valoarea citită
0223  7A             mov     a,d             ; Afişează adresa incriminată
                     nmout
0224  57          +  mov     d,a     ; Salvare număr de tipărit
0225  0F          +  rrc             ; Conversie in ASCII a celor mai
0226  0F          +  rrc             ;  semnificativi 4 biţi
0227  0F          +  rrc
0228  0F          +  rrc
0229  E6 0F       +  ani     00001111B
022B  C6 30       +  adi     '0'
022D  FE 3A       +  cpi     '9'+1   ; Mai mare decit 9 ?
022F  DA 0234     +  jc      $+5     ; Nu, salt (cifra este intre 0 şi 9)
0232  C6 07       +  adi     7       ; Altfel, este intre A şi F hex
0234  77          +  mov     m,a     ; Afişare in memoria ecranului
0235  23          +  inx     h
0236  7A          +  mov     a,d     ; Conversie in ASCII a celor mai puţin
0237  E6 0F       +  ani     00001111B       ; semnificativi 4 biţi
0239  C6 30       +  adi     '0'
023B  FE 3A       +  cpi     '9'+1
023D  DA 0242     +  jc      $+5
0240  C6 07       +  adi     7
0242  77          +  mov     m,a     ; Notă: pointerul pe ecran rămine
0243  23          +  inx     h       ;       pregătit pentru o nouă afişare
0244  7B             mov     a,e
                     nmout
0245  57          +  mov     d,a     ; Salvare număr de tipărit
0246  0F          +  rrc             ; Conversie in ASCII a celor mai
0247  0F          +  rrc             ;  semnificativi 4 biţi
```

```
0248  0F            +         rrc
0249  0F            +         rrc
024A  E6 0F         +         ani     00001111B
024C  C6 30         +         adi     '0'
024E  FE 3A         +         cpi     '9'+1   ; Mai mare decit 9 ?
0250  DA 0255       +         jc      $+5     ; Nu, salt (cifra este intre 0 si 9)
0253  C6 07         +         adi     7       ; Altfel, este intre A si F hex
0255  77            +         mov     m,a     ; Afisare in memoria ecranului
0256  23            +         inx     h
0257  7A            +         mov     a,d     ; Conversie in ASCII a celor mai putin
0258  E6 0F         +         ani     00001111B       ; semnificativi 4 biti
025A  C6 30         +         adi     '0'
025C  FE 3A         +         cpi     '9'+1
025E  DA 0263       +         jc      $+5
0261  C6 07         +         adi     7
0263  77            +         mov     m,a     ; Notă: pointerul pe ecran rămine
0264  23            +         inx     h       ;       pregătit pentru o nouă afişare
0265  23                      inx     h
0266  78                      mov     a,b             ; Afişează valoarea inscrisă
                              nmout
0267  57            +         mov     d,a     ; Salvare număr de tipărit
0268  0F            +         rrc             ; Conversie in ASCII a celor mai
0269  0F            +         rrc             ;  semnificativi 4 biti
026A  0F            +         rrc
026B  0F            +         rrc
026C  E6 0F         +         ani     00001111B
026E  C6 30         +         adi     '0'
0270  FE 3A         +         cpi     '9'+1   ; Mai mare decit 9 ?
0272  DA 0277       +         jc      $+5     ; Nu, salt (cifra este intre 0 si 9)
0275  C6 07         +         adi     7       ; Altfel, este intre A si F hex
0277  77            +         mov     m,a     ; Afişare in memoria ecranului
0278  23            +         inx     h
0279  7A            +         mov     a,d     ; Conversie in ASCII a celor mai putin
027A  E6 0F         +         ani     00001111B       ; semnificativi 4 biti
027C  C6 30         +         adi     '0'
027E  FE 3A         +         cpi     '9'+1
0280  DA 0285       +         jc      $+5
0283  C6 07         +         adi     7
0285  77            +         mov     m,a     ; Notă: pointerul pe ecran rămine
0286  23            +         inx     h       ;       pregătit pentru o nouă afişare
0287  36 2F                   mvi     m,'/'           ; Afişează un separator...
0289  23                      inx     h
028A  79                      mov     a,c             ;  ... şi valoarea găsită
                              nmout
028B  57            +         mov     d,a     ; Salvare număr de tipărit
028C  0F            +         rrc             ; Conversie in ASCII a celor mai
028D  0F            +         rrc             ;  semnificativi 4 biti
028E  0F            +         rrc
028F  0F            +         rrc
0290  E6 0F         +         ani     00001111B
0292  C6 30         +         adi     '0'
0294  FE 3A         +         cpi     '9'+1   ; Mai mare decit 9 ?
0296  DA 029B       +         jc      $+5     ; Nu, salt (cifra este intre 0 si 9)
0299  C6 07         +         adi     7       ; Altfel, este intre A si F hex
029B  77            +         mov     m,a     ; Afişare in memoria ecranului
029C  23            +         inx     h
029D  7A            +         mov     a,d     ; Conversie in ASCII a celor mai putin
029E  E6 0F         +         ani     00001111B       ; semnificativi 4 biti
02A0  C6 30         +         adi     '0'
02A2  FE 3A         +         cpi     '9'+1
02A4  DA 02A9       +         jc      $+5
02A7  C6 07         +         adi     7
02A9  77            +         mov     m,a     ; Notă: pointerul pe ecran rămine
02AA  23            +         inx     h       ;       pregătit pentru o nouă afişare
```



```
                        ;       Test abandonat sau terminat

02AB                    tend:
02AB  7E                        mov     a,m     ; Buclă pentru testare adrese
02AC  23                        inx     h       ;  ... eventual cu osciloscopul
02AD  C3 02AB                   jmp     tend


                        ;irout1 rutină de intrerupere nivel 1. Semnalizează execuţia
                        ;       prin modificarea conţinutului adresei 'tesloc'.
02B0                    irout1:
02B0  F5                        push    psw
02B1  3E FF                     mvi     a,true  ; Ridicare flag
02B3  32 FFF0                   sta     tesloc
02B6  3E FF                     mvi     a,mask  ; Mascare toate IR
02B8  D3 01                     out     intctl  ;
02BA  F1                        pop     psw
02BB  C9                        ret

                        ;ramtst testează o zonă de RAM prin inscrieri şi citiri
                        ;       succesive in tehnică barber-pole.
                        ;       Input: HL = inceput zonă
                        ;              DE = sfirşit zonă
                        ;       Distruge: AF, BC
02BC                    ramtst:
02BC  01 08FE                   lxi     b,8FEh  ; C conţine pattern iniţial, B contor
02BF                    ramt05:                 ;  pentru modificare pattern
02BF  CD 02D4                   call    testit  ; Test RAM
02C2  D8                        rc              ; Return in caz de eroare
02C3  79                        mov     a,c     ; Complementează pattern
02C4  2F                        cma
02C5  4F                        mov     c,a
02C6  CD 02D4                   call    testit  ; Test cu pattern complementat
02C9  D8                        rc
02CA  79                        mov     a,c     ; Refacere şi rotire pentru un nou test
02CB  2F                        cma
02CC  07                        rlc
02CD  4F                        mov     c,a
02CE  05                        dcr     b
02CF  C2 02BF                   jnz     ramt05
02D2  B7                        ora     a
02D3  C9                        ret

                        ;testit face inscrierea şi citirea propriu-zisă a RAM-ului cu
                        ;       un pattern pe care il roteşte la fiecare locaţie. Dacă
                        ;       găseşte o eroare, o afişează şi abandonează procedura.
                        ;       Input: HL = inceput zonă
                        ;              DE = sfirşit zonă
                        ;              C = pattern iniţial
                        ;       Output: CY = 1 locaţie defectă
                        ;               CY = 0 toată zona ok
                        ;       Distruge: AF
02D4                    testit:
02D4  C5                        push    b
02D5  E5                        push    h
02D6  71                        mov     m,c     ; Prima inscriere
02D7                    load:   ; Inscriere zonă
02D7  7E                        mov     a,m     ; Rotire pattern pentru următoarea
02D8  07                        rlc             ;  locaţie
02D9  23                        inx     h
02DA  77                        mov     m,a
02DB  CD 0300                   call    hilo
```



```
02DE  DA 02D7                   jc      load
02E1  79                        mov     a,c     ; Restaurare pattern
02E2  E1                        pop     h       ;  şi adresa de inceput
02E3  E5                        push    h       ;
02E4                    read:   ; Verificare zonă
02E4  BE                        cmp     m       ; Este corect ?
02E5  37                        stc
02E6  C2 02F3                   jnz     read05  ; Nu, abandonare test cu CY = 1
02E9  07                        rlc             ; Rotire pattern pentru următoarea
02EA  23                        inx     h       ;  locaţie
02EB  4F                        mov     c,a
02EC  CD 0300                   call    hilo
02EF  79                        mov     a,c
02F0  DA 02E4                   jc      read
02F3                    read05:
02F3  E1                        pop     h
02F4  C1                        pop     b
02F5  C9                        ret

                        ;hexasc convertesţe un digit hex intr-un caracter ASCII.
                        ;       Input: A = 8 bit hex digit
                        ;       Output: A = 8 bit data, caracter ASCII
                        ;       Distruge: AF
02F6                    hexasc:
02F6  E6 0F                     ani     00001111B       ; Dacă cifra este intre 0-9
02F8  C6 30                     adi     '0'             ;  adună '0'
02FA  FE 3A                     cpi     '9'+1
02FC  D8                        rc
02FD  C6 07                     adi     7               ; Dacă este mai mare
02FF  C9                        ret                     ;  transformă in A-F

                        ;hilo   compară registrele duble HL şi DE.
                        ;       Output: CY=1 DE > HL
                        ;               CY=0 DE <= HL
                        ;               Z=1  DE = HL
                        ;       Distruge: AF
0300                    hilo:
0300  7C                        mov     a,h     ; Compară MSByte
0301  BA                        cmp     d
0302  C0                        rnz
0303  7D                        mov     a,l     ; ... şi eventual şi LSByte
0304  BB                        cmp     e
0305  C9                        ret

                        ;prnum  trimite un octet la consolă transformat in ASCII
                        ;       Input: A = 8 bit data
                        ;       Distruge: AF
0306                    prnum:
0306  F5                        push    psw
0307  0F                        rrc             ; Transformă cei mai semnificativi
0308  0F                        rrc             ;  patru biţi intr-un caracter ASCII
0309  0F                        rrc
030A  0F                        rrc
030B  CD 02F6                   call    hexasc
030E  CD 0315                   call    output  ; Afişează primii patru biţi
0311  F1                        pop     psw     ; Restul de patru biţi
0312  CD 02F6                   call    hexasc  ;  este transformat şi afişat

                        ;output scrie un caracter la poziţia curentă a cursorului.
                        ;       Input: A = caracterul de afişat
0315                    output:
0315  F5                        push    psw
0316  D5                        push    d
0317  E5                        push    h
```



```
0318  2A FF35                   lhld    mcurs
031B  FE 0D                     cpi     cr      ; Trebuie afişat un "return" ?
031D  CA 0325                   jz      crcmd   ; Da, salt
0320  77                        mov     m,a     ; Nu, afişează caracter
0321  23                        inx     h       ; Avansează cursor
0322  C3 032D                   jmp     exit
0325                    crcmd:
0325  3E C0                     mvi     a,0C0h  ; Obţine adresa inceput rind următor
0327  A5                        ana     l
0328  6F                        mov     l,a
0329  11 0040                   lxi     d,rowlng
032C  19                        dad     d
032D                    exit:
032D  22 FF35                   shld    mcurs   ; Salvare noul pointer
0330  E1                        pop     h
0331  D1                        pop     d
0332  F1                        pop     psw
0333  C9                        ret

                        ;outstr tipăreşte un şir de caractere terminat cu null (0) la
                        ;       poziţia curentă a cursorului.
                        ;       Input: HL = adresa de inceput a şirului
                        ;       Distruge: AF, HL
0334                    outstr:
0334  7E                        mov     a,m
0335  B7                        ora     a
0336  C8                        rz
0337  CD 0315                   call    output
033A  23                        inx     h
033B  C3 0334                   jmp     outstr

                        ;delay  este o buclă de aşteptare de aproximativ o secundă.
                        ;       Distruge: AF, HL
033E                    delay:
033E  21 0000                   lxi     h,0
0341                    delay5:
0341  2B                        dcx     h
0342  7D                        mov     a,l
0343  B4                        ora     h
0344  C2 0341                   jnz     delay5
0347  C9                        ret

                        ;teserr afişează adresa, conţinutul şi valoarea iniţială a unei
                        ;       locaţii de RAM găsită necorespunzătoare de "testit".
                        ;       Input: HL = adresa
                        ;              A = valoare iniţială
                        ;       Distruge: AF
0348                    teserr:
0348  F5                        push    psw
0349  E5                        push    h
034A  21 0407                   lxi     h,ermsg ; Afişare mesaj eroare
034D  CD 0334                   call    outstr
0350  E1                        pop     h       ; ... adresa
0351  7C                        mov     a,h
0352  CD 0306                   call    prnum
0355  7D                        mov     a,l
0356  CD 0306                   call    prnum
0359  3E 3A                     mvi     a,':'
035B  CD 0315                   call    output
035E  3E 20                     mvi     a,' '
0360  CD 0315                   call    output
0363  7E                        mov     a,m     ; ... valoarea găsită
0364  CD 0306                   call    prnum
0367  3E 2F                     mvi     a,'/'
```

MICROCALCULATORUL L/B 881

NICOARA PAULIAN
ION RUSOVICI
GHEORGHE CHITA
LIVIU IONESCU

Pentru a putea fi utilizat, un calculator are nevoie de un sistem oarecare de operare care să-i poată asigura compatibilitatea programelor; acesta este de regulă monitorul, care reprezintă o "prelungire" a hardware-ului in software. Este complicat şi inutil ca fiecare program utilizator să-şi construiască propriile lui rutine de intrare/ieşire (claviatură, display) de exemplu, cind acestea pot fi inglobate intr-un program rezident, standardizat şi apelate ori de cite ori este necesar. Pentru L/B881 a fost elaborat un monitor reprezentind minimul de funcţii absolut necesar, dar care poate fi extins cu un editor de texte şi un asamblor pentru mnemonicele microprocesorului 8080. Monitorul minim ocupă 3 Kocteţi (versiunea 2.4) iar cel extins ocupă 8 Kocteţi (881/Sys V1.6). Cele două versiuni sint perfect compatibile intre ele din punct de vedere al programelor aplicative. In continuare referirile se vor face la versiunea minimă, care va fi publicată sub formă de listing hex incepind din numărul viitor.

Monitorul realizează următoarele funcţiuni importante:

- asigură interfaţa cu utilizatorul (claviatura scanată prin soft pe portul paralel 8255 şi terminal video display ocupind o parte din memoria procesorului);
- asigură interfaţa cu suportul de memorie externă (casetă magnetică);
- controlează interfaţa serie şi traductorul acustic (bell);
- asigură funcţiile ceasului de timp real;
- realizează funcţiile minime de examinare/modificare a memoriei şi registrelor procesorului;
- permite inserarea de programe direct in memorie, in format hex;
- asigură lansarea şi asistă execuţia programelor utilizator;
- facilitează depanarea programelor prin implantarea de puncte de oprire (breakpoint).

Monitorul este organizat in două părţi distincte: o colecţie de subrutine (care ocupă aproximativ 1 Koctet) şi procedurile care implementează funcţiile de bază. Subrutinele din prima parte au punctele de intrare fixe (deci compatibile de la o versiune la alta) şi pot fi folosite din programe externe monitorului. Avind in vedere importanţa deosebită a cunoaşterii conţinutului şi modului de operare a acestor subrutine pentru cei ca vor scrie programe, vom publica intr-unul din numerele viitoare listingul sursă al primei părţi a monitorului.

Monitorul comunică cu utilizatorul in mod interactiv. Dialogul constă in introducerea de către utilizator a comenzilor de la claviatură şi afişarea răspunsurilor monitorului pe un terminal CRT, fie in forma unui mesaj tipărit, fie sub forma inceperii unei activităţi. La pornire sau in urma acţionării butonului RESET, monitorul tipăreşte un mesaj de identificare, intrind in modul de aşteptare a comenzilor.

Comenzile sint sub forma unui caracter alfabetic care specifică tipul comenzii, urmat de o listă de parametri alfanumerici. Parametrii numerici sint in general sub formă hexazecimală şi pot avea de la 1 la 4 digiţi. Numere mai lungi pot fi introduse, dar nu vor fi luate in considerare decit ultimele 4 caractere. Caracterele alfabetice pot fi atit cu litere mari, cit şi mici.

Monitorul va afişa caracterele introduse de la claviatură pe prima linie a ecranului, care este constituită intr-un buffer de comenzi numit DISPLAY. Pot fi introduse mai multe comenzi separate prin terminatorul ";". Ultima comandă introdusă trebuie incheiată prin apăsarea tastei RETURN, ceea ce va determina trecerea la analiza şi execuţia fiecărei comenzi de pe DISPLAY in ordinea introducerii.

Dacă in timpul introducerii, utilizatorul doreşte corectarea unor caractere deja introduse, el are la dispoziţie fie cele două taste CURSOR STINGA şi CURSOR DREAPTA cu ajutorul cărora se poate manevra cursorul pe DISPLAY pentru a opera schimbările necesare, fie tasta CANCEL (CTRL X) care va anula toate datele de pe DISPLAY. Introducerea spaţiilor in cimpul comenzilor sau al parametrilor este opţională, monitorul ignorindu-le in timpul analizei. Delimitatorul valid al parametrilor este ",". Dacă litera ce simbolizează o comandă este urmată de virgule consecutive, parametrii respectivi vor fi consideraţi nuli.

In cazul uneia sau mai multor comenzi eronate, monitorul va afişa mesajul "Error" şi concomitent va scoate in evidenţă comanda (sau comenzile) respective prin videoreversare.

Monitorul foloseşte a doua linie a ecranului, numită STATUS exclusiv pentru uz propriu (afişarea unor mesaje, a orei curente, ş.a.).

1. COMENZILE MONITORULUI

1.1. Comanda "F"

a. Format: FBnnnn<cr>

Permite modificarea vitezei de lucru (BAUD RATE) a USART-ului. Tactul de emisie-recepţie pentru USART este furnizat de unul din cele trei numărătoare programabile ale circuitului LSI 8253 P.I.T. (adica TIMER 1). TIMER 1 va diviza un tact de 1375 kHz cu valoarea reprezentată de parametrul nnnn.

Relaţia de calcul este:

nnnn = 85938/Bd.rate

De pildă, dacă se doreşte o viteză de 1200 Bd, utilizatorul va trebui să execute comanda FB72<cr> aceasta asigurind tactul necesar.

b. Format: FChhmm<cr>

Comanda FC permite iniţializarea ceasului de timp real, afişat in extrema dreaptă a zonei STATUS. Parametrul care o insoţeşte reprezintă ora (hh) şi minutul (mm) curent. De exemplu, comanda FC715<cr> va determina afişarea orei 07:15:00. La pornire, ceasul este adus automat la zero in cadrul iniţializărilor monitorului.

c. Format: FI<cr>
sau FInnn<cr>

Are ca efect tipărirea unui header "Lixco 881/Mon" pe linia curentă a imprimantei şi considerarea acestuia ca inceput de pagină. In forma a doua, cele trei caractere (obligatoriu trei!) reprezintă numărul paginii curente, şi se va afişa in header. La fiecare pagină nouă acest număr va fi afişat şi apoi incrementat.

Exemplu: FI 3<cr>

Lungimea unei pagini este de 72 rinduri, utile fiind numai 64, restul fiind folosite astfel: 3 pentru header şi 5 pentru separarea paginilor. Cele două valori (64 şi 5) sint implicite la iniţializare, dar pentru cazul cind se doreşte utilizarea altui format, există posibilitatea modificării unor locaţii de RAM, la adresele FF61 şi FF62, cu noile valori (valori ce trebuie date in hex). Modificarea se va face cu comanda M şi va fi urmată de un FI.

d. Format: FOaaaa<cr>

Comanda FO realizează setarea vectorului OVECT la adresa aaaa. Aceasta va determina trimiterea datelor afişate pe ecran spre unul sau mai multe periferice conectate la microcalculator, ai căror driveri software se găsesc in memorie la adresa aaaa. Pentru aceasta, se pot folosi porturile sistemului (paralel şi/sau serie). In acest scop rutina OUTPUT are prevăzută posibilitatea extinderii canalului de ieşire. Ea face afişarea pe ecran a caracterului din registrul A şi cheamă o subrutină numită OVECT a cărei adresă de intrare este in RAM. Folosind comanda FOaaaa utilizatorul va iniţializa vectorul de output auxiliar la adresa aaaa, unde a fost implementat programul (driverul) de ieşire adiţional (serie sau paralel). Dacă se doreşte intreruperea transmiterii prin OVECT a datelor afişate pe ecran spre porturile auxiliare, o nouă comandă FO<cr> (fără parametru) va realiza resetarea vectorului OVECT şi deci, afişarea numai pe CRT.

e. Format: FP<cr>

Comanda FP setează vectorul OVECT la adresa driverului soft pentru portul serie, conţinut in monitor. Resetarea se face printr-o comandă FO<cr> (fără parametru). De semnalat faptul că după execuţia comenzii FP, comanda M cu doi parametri va tipări şi la imprimantă zona de memorie respectivă. Acelaşi lucru se va intimpla şi cu comanda R.

f. Format: FXaaaa<cr>

Comanda FX este utilă in cazul unor extensii de comenzi la monitor scrise şi executate in RAM. Comanda setează vectorul XVECT la adresa aaaa, ceea ce va determina un salt necondiţionat al monitorului la adresa specificată in comandă, in cazul execuţiei comenzii X.

1.2. Comanda "G"

Format: Gssss(,bbbb,cccc)<cr>

Comanda G transferă controlul procesorului din monitor in programul utilizator. Parametrul ssss trebuie să fie o adresă din memorie, care conţine prima instrucţiune a programului ce se doreşte a se executa. Dacă se doreşte ca la sfirşitul rulării programului utilizator să se efectueze o reintoarcere in monitor, se poate executa o instrucţiune RST7, prin care se asigură şi salvarea registrelor CPU pentru o eventuală analiză ulterioară.

Programul in curs de rulare poate fi intrerupt in orice moment de utilizator prin apăsarea clapei ESCAPE (CTRL [), controlul sistemului fiind trecut monitorului, care va afişa pe STATUS mesajul "Monitor control", şi va trece in modul de aşteptare a comenzilor. Ulterior se poate relua execuţia programului care a fost intrerupt, din locul respectiv, prin execuţia unei comenzi G<cr> (fără parametru).

Parametrii (,bbbb,cccc) sint opţionali şi se utilizează in faza de punere la punct a programelor scrise in limbajul de asamblare. Ei reprezintă două puncte de intrerupere (breakpoint), care pot fi introduse in programul in curs de depanare. Astfel, comanda G4100,41E2<cr> va determina execuţia de la adresa 4100 la 41E2, după care se va afişa mesajul "Break" impreună cu registrele procesorului, monitorul preluind apoi controlul. Programul poate fi reluat din locul in care a fost intrerupt printr-o nouă comandă G cu sau fără breakpoint-uri. Dacă cursul unui program poate lua la un moment dat două căi distincte (ca urmare a unei instrucţiuni de salt condiţionat), se pot insera cele două breakpointuri pe fiecare ramură, urmărind astfel drumul parcurs de procesor (ex. G4100,41B2,41F6<cr>). Sistemul de breakpoint funcţionează corect doar in cazul in care se foloseşte o stivă diferită de a monitorului in programul utilizatorului.

```
0369  CD 0315                  call    output
036C  F1                       pop     psw     ;... şi valoarea inițială
036D  CD 0306                  call    prnum
0370  3E 0D                    mvi     a,cr
0372  C3 0315                  jmp     output


0375                     stkmsg:
0375  20 2A 20 53               db      ' * Stack memory okay...',cr,0
0379  74 61 63 6B
037D  20 6D 65 6D
0381  6F 72 79 20
0385  6F 6B 61 79
0389  2E 2E 2E 0D
038D  00
038E                     scrok:
038E  20 2A 20 53               db      ' * Screen memory okay...',cr,0
0392  63 72 65 65
0396  6E 20 6D 65
039A  6D 6F 72 79
039E  20 6F 6B 61
03A2  79 2E 2E 2E
03A6  0D 00
03A8                     patmsg:
03A8  0D 0D 0D 0D               db      cr,cr,cr,cr,cr,'
03AC  0D 20 20 20
03B0  20 20 20 20
03B4  20 20
03B6  28 42 65 20               db      '(Be patient, it may take a while ... RAM
03BA  70 61 74 69
03BE  65 6E 74 2C
03C2  20 69 74 20
03C6  6D 61 79 20
03CA  74 61 6B 65
03CE  20 61 20 77
03D2  68 69 6C 65
03D6  20 2E 2E 2E
03DA  20 52 41 4D
03DE  20
03DF  74 65 73 74               db      'testing)',0
03E3  69 6E 67 29
03E7  00
03E8                     bnk1ok:
03E8  20 2A 20 52               db      ' * RAM bank C000-FFFF okay...',cr,0
03EC  41 4D 20 62
03F0  61 6E 6B 20
03F4  43 30 30 30
03F8  2D 46 46 46
03FC  46 20 6F 6B
0400  61 79 2E 2E
0404  2E 0D 00
0407                     ermsg:
0407  20 3F 3F 3F               db      ' ??? Memory error at ',0
040B  20 4D 65 6D
040F  6F 72 79 20
0413  65 72 72 6F
0417  72 20 61 74
041B  20 00
041D                     bnk2ok:
041D  20 2A 20 52               db      ' * RAM bank 8000-BFFF okay...',cr,0
0421  41 4D 20 62
0425  61 6E 6B 20
0429  38 30 30 30
042D  2D 42 46 46
```



```
0431  46 20 6F 6B
0435  61 79 2E 2E
0439  2E 0D 00
043C                     bnk3ok:
043C  20 2A 20 52               db      ' * RAM bank 4000-7FFF okay...',cr,0
0440  41 4D 20 62
0444  61 6E 6B 20
0448  34 30 30 30
044C  2D 37 46 46
0450  46 20 6F 6B
0454  61 79 2E 2E
0458  2E 0D 00
045B                     interr:
045B  20 3F 3F 3F               db      ' ??? PIC 8259 defective',cr,0
045F  20 50 49 43
0463  20 38 32 35
0467  39 20 64 65
046B  66 65 63 74
046F  69 76 65 0D
0473  00
0474                     irlok:
0474  20 2A 20 50               db      ' * PIC 8259 okay...',cr,0
0478  49 43 20 38
047C  32 35 39 20
0480  6F 6B 61 79
0484  2E 2E 2E 0D
0488  00
0489                     ppiok:
0489  20 2A 20 50               db      ' * PPI 8255 okay...',cr,0
048D  50 49 20 38
0491  32 35 35 20
0495  6F 6B 61 79
0499  2E 2E 2E 0D
049D  00
049E                     ppibad:
049E  20 3F 3F 3F               db      ' ??? PPI 8255 defective',cr,0
04A2  20 50 50 49
04A6  20 38 32 35
04AA  35 20 64 65
04AE  66 65 63 74
04B2  69 76 65 0D
04B6  00
04B7                     timok:
04B7  20 2A 20 50               db      ' * PIT 8253 okay...',cr,0
04BB  49 54 20 38
04BF  32 35 33 20
04C3  6F 6B 61 79
04C7  2E 2E 2E 0D
04CB  00
04CC                     timbad:
04CC  20 3F 3F 3F               db      ' ??? PIT 8253 defective',cr,0
04D0  20 50 49 54
04D4  20 38 32 35
04D8  33 20 64 65
04DC  66 65 63 74
04E0  69 76 65 0D
04E4  00
04E5                     serbad:
04E5  20 3F 3F 3F               db      ' ??? USART 8251 defective',cr,cr,cr,cr,cr,0
04E9  20 55 53 41
04ED  52 54 20 38
04F1  32 35 31 20
04F5  64 65 66 65
04F9  63 74 69 76
```



```
04FD  65 0D 0D 0D
0501  0D 0D 00
0504                     serok:
0504  20 2A 20 55               db      ' * USART 8251 okay.',cr,cr,cr,cr,cr,cr,0
0508  53 41 52 54
050C  20 38 32 35
0510  31 20 6F 6B
0514  61 79 2E 0D
0518  0D 0D 0D 0D
051C  00
051D                     okmsg:
051D  20 2A 2A 2A               db      ' *** End of test ***',cr,cr,cr,cr
0521  20 45 6E 64
0525  20 6F 66 20
0529  74 65 73 74
052D  20 2A 2A 2A
0531  0D 0D 0D 0D
0535  4C 69 78 63               db      'Lixco 881/Test V2.1
0539  6F 20 38 38
053D  31 2F 54 65
0541  73 74 20 56
0545  32 2E 31 20
0549  20
054A  28 43 29 20               db      '(C) 1986 Lixco Software',0
054E  31 39 38 36
0552  20 4C 69 78
0556  63 6F 20 53
055A  6F 66 74 77
055E  61 72 65 00


00FF                     true    equ     0FFh
000D                     cr      equ     0Dh     ; Carriage Return code
009F                     graph   equ     9Fh     ; Un caracter grafic  .
0016                     icw1    equ     16h     ; Inițializări 8259
00FF                     icw2    equ     0FFh
00FD                     mask1   equ     0FDh    ; IR1 nemascați
00FF                     mask    equ     0FFh    ; Toate intreruperile mascate
0000                     intct0  equ     0       ; Adresa pentru icw1, ocw2/3, IRR, ISR
0001                     intct1  equ     1       ; Adresa pentru ocw1, icw2, IMR
0061                     seoi1   equ     61h     ; Specific End Of Interrupt level 1
0080                     pcw     equ     80h     ; Porturile inițializate ca output
0060                     porta   equ     60h     ; Adresa port A PPI
0061                     portb   equ     61h     ; Adresa port B PPI
0062                     portc   equ     62h     ; Adresa port C PPI
0063                     parsta  equ     63h     ; Adresa PPI status
0055                     vala    equ     55h     ; Data inscrisă in portul A
00AA                     valb    equ     0AAh    ; Data inscrisă in portul B
005A                     valc    equ     5Ah     ; Data inscrisă in portul C
003E                     cwt0    equ     3Eh     ; Control Word Timer 0
007E                     cwt1    equ     7Eh     ; Control Word Timer 1
00BE                     cwt2    equ     0BEh    ; Control Word Timer 2
007F                     cwt10   equ     7Fh     ; Control Word Timer 1 pentru baud rate
0010                     timer0  equ     10h     ; Adresa timer 0
0011                     timer1  equ     11h     ; Adresa timer 1
0012                     timer2  equ     12h     ; Adresa timer 2
0013                     timsta  equ     13h     ; Adresa status timere
00FF                     val0    equ     0FFh    ; Valoare de test timer MSB
0092                     val1    equ     92h     ; Valoare de test timer 0 LSB
0096                     val2    equ     96h     ; Valoare de test timer 1 LSB
009E                     val3    equ     9Eh     ; Valoare de test timer 2 LSB
00CE                     mode    equ     0CEh    ; Cuvînt de mod USART
0027                     cmd     equ     37h     ; Comandă USART
0031                     sersta  equ     31h     ; Adresa status USART
```



```
0030                     serdat  equ     30h     ; Adresa date USART
                         ;
0040                     rowlng  equ     64      ; Lungimea unui rind in bytes
001A                     nrows   equ     26      ; Numărul de rinduri ale ecranului
4000                     bank3   equ     4000h   ; Inceput RAM
8000                     bank2   equ     8000h   ; Inceput banc din mijloc
C000                     bank1   equ     0C000h  ; Inceput primul banc de RAM
F800                     rowA    equ     0F800h  ; Inceput memorie ecran
F840                     rowB    equ     rowA+rowlng     ; Al doilea rind al ecranului
FA00                     row7    equ     0FA00h  ; Alt rind
FE40                     lastrw  equ     0FE40h  ; Ultima linie a ecranului
FE80                     endscr  equ     lastrw+rowlng   ; Primul byte liber după ecran
FDC0                     row22   equ     lastrw-(2*rowlng)       ; Un alt rind
FF00                     stack   equ     0FF00h  ; Stiva de lucru
FF04                     irl     equ     0FF04h  ; Vector pentru intreruperi nivel 1
FF35                     mcurs   equ     0FF35h  ; Pointerul de cursor
FFF0                     tesloc  equ     0FFF0h  ; Adresă de test pentru intreruperi
FFFF                     hotram  equ     0FFFFh  ; Ultima locație de RAM

                                 .dephase

                                 end
```

```
Macros:
NMOUT

Symbols:
BANK1   C000    BANK2   8000    BANK3   4000    BNK1OK  03E8
BNK2OK  041D    BNK3OK  043C    CLSC05  0049    CMD     0037
CR      000D    CRCMD   0325    CWT0    003E    CWT1    007E
CWT10   007F    CWT2    00BE    DELAY   033E    DELAY5  0341
ENDSCR  FE80    ERMSG   0407    ERROR   021C    EXIT    032D
GRAPH   009F    HEXASC  02F8    HILO    0360    HOTRAM  FFFF
ICW1    0016    ICW2    00FF    INTCT0  0000    INTCT1  0001
INTERR  0458    IR1     FF04    IR1OK   0474    IROUT1  03B0
LASTRW  FE40    LOAD    02D7    LOOP1   0006    LOOP15  0010
LOOP2   0016    MASK    00FF    MASK1   00FD    MCURS   FF35
MODE    00CE    NROWS   001A    OKMSG   051D    OUTPUT  0315
OUTSTR  0334    OVER    0213    PARSTA  0063    PATMSG  03A8
PCW     0080    PICT05  007B    PICT10  007D    PICT15  008C
PICT17  0094    PICT20  0096    PICT25  00A5    PICT30  00AE
PORTA   0060    PORTB   0061    PORTC   0062    PPIBAD  049E
PPIOK   0489    PRNUM   0306    RAMT05  02BF    RAMTST  02BC
READ    02E4    READ05  02F3    ROW22   FDC0    ROW7    FA00
ROWA    F800    ROWB    F840    ROWLNG  0040    SCROK   038E
SEOI1   0061    SERBAD  04E5    SERDAT  0030    SEROK   0504
SERSTA  0031    STACK   FF00    STKMSG  0375    TEND    02AB
TESERR  0348    TESLOC  FFF0    TESTIT  02D4    TIMBAD  04CC
TIMER0  0010    TIMER1  0011    TIMER2  0012    TIMOK   04B7
TIMSTA  0013    TINT    0116    TINT05  014B    TPPI    0151
TPPI05  017F    TRAM05  0064    TRAM10  006A    TRAM13  006F
TRAM15  00E0    TRAM20  00E6    TRAM30  00F8    TRAM35  00FE
TRAM45  0110    TRUE    00FF    TTIM10  01D0    TTIMER  0185
TUSAR7  020D    TUSART  01E8    VAL0    00FF    VAL1    0092
VAL2    0096    VAL3    009E    VALA    0055    VALB    00AA
VALC    005A


No  Fatal error(s)
```

MICROCALCULATORUL L/B 881

NICOARA PAULIAN
LIVIU IONESCU
ION RUSOVICI
GHEORGHE CHITA

In acest număr, paralel cu descrierea comenzilor, este dat in intregime (in format hex) monitorul 881/Mon, versiunea 2.4. După cum am mai arătat, prima parte a monitorului reprezintă o colecţie de subrutine apelabile din orice program utilizator; datorită importanţei deosebite a acestei părţi, vom publica şi listingul sursă. Listingul se adresează in special celor ce intenţionează să dezvolte programe in asamblare pentru L/B881, ajutindu-i să inţeleagă funcţionarea subrutinelor respective. Veţi găsi continuarea lui in numerele viitoare.

1.3. Comanda "L"

Format: L titlu(,aaaa)<cr>

Comanda L este utilizată pentru incărcarea de date sau programe in memorie de pe caseta magnetică. Comanda se desfăşoară astfel:

- pe STATUS se afişează mesajul "System busy";
- microcalculatorul aşteaptă recepţionarea inregistrării cu titlul specificat in comandă;
- dacă din bandă se recepţionează un alt titlu, numele acestuia va fi tipărit pe STATUS impreună cu zona de memorie pe care o ocupă, microcalculatorul aşteptind in continuare titlul indicat iniţial; se poate părăsi această stare prin apăsarea tastei CANCEL (CTRL X), controlul fiind dat interpretorului de comenzi al monitorului;
- dacă recepţionează titlul dorit, monitorul va activa un indicator clipitor pe STATUS şi va proceda la transferul datelor ce vin de pe bandă incepind cu adresa de inceput care a fost specificată la salvarea iniţială; dacă a fost specificat şi parametrul opţional (,aaaa), atunci adresa de inceput din memorie va fi chiar aaaa. In situaţia in care noua adresă de inceput este aleasă in aşa fel incit blocul de date depăşeşte adresa maximă de memorie (FFFF), operaţiunea va inceta imediat cu afişarea mesajului "Errors in file xxxx".
- simultan cu transferul datelor in memorie se face şi controlul CRC (CYCLIC REDUNDANCY CHECK).
- in final se va afişa pe STATUS un mesaj ce indică dacă incărcarea s-a efectuat cu sau fără erori.

Comanda L<cr> (fără parametru) va determina afişarea pe STATUS a titlurilor şi adreselor inregistrărilor de pe bandă in ordinea recepţionării lor.

1.4. Comanda "M"

a. Format: Mssss<cr>

Comanda M cu un parametru va afişa zona de memorie dintre adresa ssss şi adresa ssss+256. Fiecare linie de pe ecran va incepe cu adresa primei locaţii de memorie afişată urmată de alte 16 locaţii de memorie. După terminarea afişării, cursorul va fi poziţionat pe ecran in dreptul locaţiei specificate prin ssss, iar pe STATUS se va tipări adresa efectivă pe care se află cursorul.

In acest moment se pot vizualiza datele din zona respectivă de memorie, se pot schimba cu altele noi (format HEX), se pot introduce programe intregi, avind la dispoziţie setul de comenzi al cursorului şi comenzile N (pentru afişarea următoarei pagini de 256 octeţi) şi P (pentru afişarea paginii anterioare de 256 octeţi). Comanda poate fi incheiată prin apăsarea tastei RETURN.

Dacă la manevrarea cursorului sau la introducerea datelor se depăşesc limitele paginii afişate, monitorul va intra automat in zona următoare sau anterioară, in funcţie de sensul manevrării cursorului.

Dacă in timpul introducerii datelor a fost efectuată o greşeală (caractere non-HEX), pe STATUS va apare un mesaj de eroare, iar locaţia respectivă din memorie va rămine nealterată. Cu această ocazie, intreaga pagină din ecran este reafişată, putindu-se astfel urmări modul in care au fost făcute modificările din memorie.

b. Format: Mssss,eeee<cr>

Comanda M urmată de doi parametri realizează afişarea in formatul prezentat in alineatul precedent a datelor cuprinse in memorie, intre adresele ssss şi eeee. Această comandă este utilă pentru un "vidaj de memorie" pe o imprimantă. Este

```
881/Mon (C) 1985 Lixco Software MACRO-80 3.36  17-Mar-80        PAGE  1
                        name    ('MONRUT')
                        title   881/Mon (C) 1985 Lixco Software
                        subttl  User Low Level Monitor Routines

                ;
                ;       Created:        01 Mar 1985
                ;       Last revision:  03 Mar 1985


                ;   "881 - Monitor routines" contains the main routines
                ; used in all 881 monitors and resident operating systems.


                        extrn   succes,main13,inits
                        extrn   ermsg,null,out05,pagrut,ird4,ird5,iwr5,msg3

0000'                   aseg

0000                    
0000  C3 0000*  start:  jmp     inits   ; Monitor cold entry point (reset)
0003            return:
0003  C3 0000*          jmp     succes  ; Successful extension return point
0006            error:
0006  C3 0000*          jmp     main13  ; Erroneous extension return point

                ;getnm  Takes ASCII parameters from Display, converts them to
                ; hex and pushes them into stack. Default param's are 0.
                ;   Input: B = 8 bit data, param's number,
                ;          DE = 16 bit addr, pointer to first digit on Disp.
                ;   Output: CY = 1 hex error or to many param's,
                ;           Top of stack = param's, in reverse order,
                ;           DE = 16 bit addr, pointer on Disp after ';'.
                ;   Destroys: all.

0009            getnm:
0009  0E 00             mvi     c,0
000B            gnm05:
000B  0C                inr     c
000C  C5                push    b
000D  CD 00B8           call    gethx
0010  C1                pop     b
0011  E3                xthl
0012  E5                push    h
0013  DA 001F           jc      gnm10
0016  FE 3B             cpi     term
0018  CA 0027           jz      gnm15
001B  05                dcr     b
001C  C2 000B           jnz     gnm05
001F            gnm10:
001F  E1                pop     h
0020  E3                xthl
0021  0D                dcr     c
0022  C2 001F           jnz     gnm10
0025  37                stc
0026  C9                ret
0027            gnm15:
0027  05                dcr     b
0028  C8                rz
0029  21 0000           lxi     h,0
002C  E3                xthl
002D  E5                push    h
002E  C3 0027           jmp     gnm15
```

```
881/Mon (C) 1985 Lixco Software MACRO-80 3.36  17-Mar-80        PAGE  1-1
User Low Level Monitor Routines

                ;sb2    Two bytes subtraction routine (HL = HL - DE).
                ;   Input: HL, DE = 16 bit integers.
                ;   Output: HL = result.
                ;   Destroys: AF, HL.

0031            sb2:
0031  7D                mov     a,l
0032  93                sub     e
0033  6F                mov     l,a
0034  7C                mov     a,h
0035  9A                sbb     d
0036  67                mov     h,a
0037  C9                ret

                        Restart entry point (rst 7).

0038            rstart:
0038  C3 FF20           jmp     rvect

003B            hexerr:
003B  E5                push    h
                ;erorms Outputs Bell and prints "Error" on Status.
                ;   Destroys: AF, HL.

003C            erorms:
003C  CD 007D           call    bell
003F  21 0000*          lxi     h,ermsg
0042  C3 0048           jmp     strist

                ;clsta  Clears Status line.
                ;   Destroys: HL, AF.

0045            clsta:
0045  21 0000*          lxi     h,null
                ;strist Prints an ASCII string on Status line and clears the
                ; rest of line.
                ;   Input: HL = 16 bit addr, string's begining (terminator
                ;          is null).
                ;   Destroys: HL, AF.

0048            strist:
0048  D5                push    d
0049  11 F840           lxi     d,row1+64
004C            str05:
004C  7E                mov     a,m
004D  B7                ora     a
004E  CA 0057           jz      str10
0051  12                stax    d
0052  23                inx     h
0053  13                inx     d
0054  C3 004C           jmp     str05
0057            str10:
0057  3E 78             mvi     a,120
0059  EB                xchg
005A            str15:
005A  36 20             mvi     m,
005C  23                inx     h
005D  BD                cmp     l
005E  C2 005A           jnz     str15
0061  D1                pop     d
0062  C9                ret
```

```
0000 C3 FC 03 C3 55 09 80 CA 05 0E 00 0C C5 CD B8 00
0010 C1 E3 E5 DA 1F 00 FE 3B CA 27 00 05 C2 0B 00 E1
0020 E3 0D C2 1F 00 37 C9 05 C8 21 00 00 E3 E5 C3 27
0030 00 7D 93 6F 7C 9A 67 C9 C3 20 FF E5 CD 7D 00 21
0040 A0 05 C3 48 00 21 F7 05 D5 11 40 F8 7E B7 CA 57
0050 00 12 23 13 C3 4C 00 3E 78 EB 36 20 23 BD C2 5A
0060 00 D1 C9 D6 30 D8 FE 17 3F D8 FE 0A 3F D0 D6 07
0070 FE 0A C9 E6 0F C6 30 FE 3A D8 C6 07 C9 C5 01 18
0080 05 78 CD 90 03 C1 C9 F5 DB 31 E6 01 CA 88 00 F1
0090 D3 30 C9 0D C2 A1 00 0E 0A 3A 76 F8 EE 80 32 76
00A0 F8 7C BA C0 7D BB C9 3E 3F BB 3E 3B D8 1A CD 1B
00B0 02 FE 20 C0 13 C3 A7 00 21 00 00 44 CD A7 00 13
00C0 FE 2C C8 FE 3B C8 CD 63 00 D8 29 29 29 29 4F 09
00D0 C3 BC 00 F5 C5 D5 E5 47 2A 35 FF FE 20 D2 18 01
00E0 FE 1F CA 1D 01 FE 1E CA 48 01 FE 08 CA 6C 01 FE
00F0 0A CA 65 01 FE 0D CA 5C 01 FE 0C CC AC 01 00 10

0100 CC 7D 00 FE 0F CA 0E 01 FE 0E C2 13 01 2F E6 80
0110 32 37 FF E1 D1 C1 F1 C9 3A 37 FF 80 77 23 11 80
0120 FE CD A1 00 DA 56 01 E5 01 80 F8 11 C0 F8 21 C0
0130 05 1A 02 13 03 2B 7C B5 C2 31 01 01 40 20 21 40
0140 FE 70 23 0D C2 41 01 E1 11 C0 FF 19 11 80 F8 CD
0150 A1 00 D2 56 01 EB 22 35 FF C3 13 01 3E C0 A5 6F
0160 C3 56 01 10 01 11 40 00 19 C3 1E 01 2B C3 4C 01
0170 E5 F5 2A 33 FF FE 20 D2 87 01 FE 1F CA 88 01 FE
0180 08 CA 96 01 F1 E1 C9 77 23 3E 40 BD C2 90 01 2B
0190 22 33 FF C3 84 01 7D B7 CA 90 01 C3 8F 01 21 40
01A0 F8 2D 36 20 C2 A1 01 22 33 FF EB C9 F5 C5 D5 E5
01B0 21 80 F8 01 00 06 36 20 23 0B 79 B0 C2 B6 01 21
01C0 80 F8 C3 56 01 FF 3E 20 C3 CD 01 3E 0D CD D3 00
01D0 FE 0A C2 CE 03 CD 4A FF C3 FB 03 E5 2A 35 FF 7E
01E0 32 13 FF 2A 33 FF 7E 32 17 FF 3A 0B FF 32 44 FF
01F0 21 1B FF 7E E6 20 CA F3 01 7E E6 DF 77 3A 32 FF

0200 E6 7F F5 AF 32 44 FF 3A 17 FF 2A 33 FF 77 3A 13
0210 FF 2A 35 FF 77 F1 E1 C9 CD DB 01 FE 61 D8 FE 7B
0220 D0 D6 20 C9 4F 0F 0F 0F 0F CD 73 00 47 79 CD 73
0230 00 4F C9 7A CD 38 02 7B C5 CD 24 02 78 CD CD 01
0240 79 C1 C3 CD 01 21 00 00 DA 4E 02 2A 35 FF 09 22
0250 35 FF C9 1A 02 CD A1 00 03 13 C2 53 02 C9 7A CD
0260 63 02 7B CD 24 02 70 23 71 23 C9 3E B1 D3 13 06
0270 CF 3E F1 32 3A FF 21 F8 05 22 11 FF 21 08 06 CD
0280 BE 02 21 1B FF 3A 3A FF E6 01 C8 7E E6 20 CA 85
0290 02 7E E6 DF 77 3A 32 FF FE 18 C2 85 02 3E C7 D3
02A0 00 DB 01 F6 30 D3 01 3E BF D3 13 37 C9 06 DF 3E
02B0 BF D3 13 3E 75 D3 12 3E 13 D3 12 21 50 06 22 15
02C0 FF 3E C3 32 10 FF 32 14 FF 3E C3 D3 00 DB 01 A0
02D0 D3 01 21 E4 05 01 60 F8 7E B7 C8 02 23 03 C3 D8
02E0 02 3A 3A FF E6 02 CA E1 02 AF 32 3A FF 3A 3D FF
02F0 C3 1B 03 AF 06 10 21 FE 02 E5 C5 CD 1F 03 2A 38

0300 FF CD 04 03 7C CD 09 03 7D F5 3A 3A FF E6 02 C2
0310 0A 03 F6 02 32 3A FF F1 32 3D FF C5 E5 06 08 F5
0320 2A 38 FF 17 7D 17 6F 7C 17 67 D2 34 03 EE 10 67
0330 7D EE 21 6F 22 38 FF F1 07 05 C2 1F 03 E1 C1 C9
0340 0A 03 FE FF C8 BC CA 4E 03 03 03 C3 40 03 0A 6F
0350 03 0A 67 C1 E9 7D F5 7C 21 78 F8 CD 63 02 36 3A
0360 23 F1 CD 63 02 36 3A 23 36 30 23 36 30 C9 22 48
0370 FF C9 3E 7F D3 13 7D D3 11 7C D3 11 C9 21 87 00
0380 22 4B FF 7C B5 3E C9 CA 8C 03 3E C3 32 4A FF C9
0390 F5 3E 3F D3 13 79 D3 10 78 D3 10 F1 32 45 FF C9
03A0 4D 43 08 47 55 05 52 C3 09 56 04 0B 53 86 0A 4C
03B0 05 0B 46 12 05 58 47 FF FF FF FF FF FF FF FF FF
03C0 FF FF FF FF FF FF FF FF FF FF FF FF FF FF CD 4A
03D0 FF FE 0D C0 3E 0A C3 CD 01 1A CD 63 00 29 29 29
03E0 29 85 6F 7C CE 00 67 13 0D C2 D9 03 C9 3A 32 FF
03F0 FE 18 CA DB 01 FE 20 CC DB 01 37 C9 3E CE D3 31

0400 3E 37 D3 31 3E 83 32 1B FF D3 63 3E C3 32 04 FF
0410 32 47 FF 32 20 FF 21 3D 05 22 48 FF 21 56 04 22
0420 21 FF 21 00 FF 22 26 FF F9 21 46 FF 3E C9 BE CA
0430 3C 04 77 32 94 FF 21 00 00 CD 55 03 21 7F 06 22
0440 05 FF 3E C9 32 4A FF 32 23 FF CD AC 01 21 BE 05
0450 CD 48 00 C3 AA 04 F3 22 2A FF E1 22 28 FF F5 21
0460 02 00 39 22 26 FF F1 31 32 FF F5 C5 D5 31 00 FF
0470 2A 3E FF 3A 42 FF 77 2A 40 FF 3A 43 FF 77 EB 2A
0480 28 FF 2B CD A1 00 EB CA 9C 04 2A 3E FF CD A1 00
0490 CA 9C 04 21 A8 05 CD 48 00 C3 AA 04 EB 22 28 FF
04A0 EB 21 B8 05 CD 48 00 CD 3A 0A CD 9E 01 21 00 F8
04B0 22 33 FF AF 32 37 FF 3E BF D3 13 3E 16 D3 00 3E
04C0 FF D3 01 3E FD D3 01 FB CD 94 FF CD 7D 00 3E 01
04D0 32 0B FF CD DB 01 47 CD 9E 01 CD 45 00 78 FE 0D
04E0 CA F6 04 FE 0C CC AC 01 CD 70 01 CD DB 01 FE 18
04F0 CC 9E 01 C3 DE 04 3E 3B CD 70 01 3E 02 32 0B FF

0500 CD A7 00 DA CE 04 13 D5 67 01 A0 03 CD 40 03 C3
0510 3D 05 CD A7 00 13 01 55 09 C5 F5 CD B8 00 C1 DA
0520 3C 05 78 FE 42 CA 72 03 FE 4F CA 80 03 FE 58 CA
0530 6E 03 FE 50 CA 7D 03 FE 43 CA 55 03 D1 D1 1B 3E
0540 40 BB DA 4F 05 1A EE 80 12 FE BB 13 C2 3F 05 CD
0550 3C 00 C3 FB 04 06 03 CD 09 00 DA 3D 05 E1 22 40
0560 FF 7E 32 43 FF 36 FF E1 22 3E FF 7E 32 42 FF 36
0570 FF E1 7D B4 CA 7A 05 22 28 FF F3 31 2C FF D1 C1
0580 F1 2A 26 FF F9 2A 28 FF E5 2A 2A FF FB C9 41 20
0590 46 42 20 43 44 20 45 48 20 4C 50 43 20 53 50 20
05A0 A0 C5 F2 F2 EF F2 A0 00 4D 6F 6E 69 74 6F 72 20
05B0 63 6F 6E 74 72 6F 6C 00 42 72 65 61 6B 00 4C 69
05C0 78 63 6F 20 38 38 31 2F 4D 6F 6E 20 56 32 2E 34
05D0 00 4E 6F 20 45 72 72 6F 72 73 20 69 6E 20 46 69
05E0 6C 65 3A 00 A0 D3 F9 F3 F4 E5 ED A0 E2 F5 F3 F9
05F0 A0 00 53 61 76 65 64 00 F5 3E 40 D3 12 3E 03 D3

0600 12 3E 64 D3 00 F1 FB C9 FB F5 E5 CD 75 06 23 7E
0610 1F 77 3C C2 34 06 2B 3A 60 FF 96 CA 25 06 21 C5
0620 01 86 C2 34 06 21 3B 06 22 15 FF AF 32 3A FF 3E
0630 08 32 3B FF E1 3E 65 D3 00 F1 C9 FB F5 E5 CD 75
0640 06 3A 3B FF 3D C2 31 06 7E 23 77 3E 02 C3 2C 06
0650 FB F5 E5 CD 75 06 21 75 13 DA 5F 06 21 88 06 7D
0660 D3 12 7C D3 12 3A 3B FF 3D C2 31 06 3A 3D FF 32
0670 3C FF C3 2B 06 21 3C FF DB 62 1F 7E 1F 77 C9 FB
0680 F5 C5 D5 E5 21 07 FF 35 C2 AA 06 36 0F 3A 44 FF
0690 47 E6 01 CA 9D 06 2A 33 FF 7E EE 80 77 78 E6 02
06A0 CA AA 06 2A 35 FF 7E EE 80 77 21 45 FF 7E B7 CA
06B0 BA 06 35 C2 BA 06 3E 3F D3 13 21 FB 06 E5 21 0F
06C0 FF 35 C0 36 33 06 02 11 39 30 21 7F F8 34 7B BE
06D0 D0 72 2B 34 3E 35 BE D0 72 2B 2B 05 C2 CD 06 2B
06E0 3E 32 BE CA EF 06 23 34 7B BE D0 72 2B 34 C9 23
06F0 34 3E 34 BE C0 72 2B 72 C3 23 FF 21 60 00 DB FF

0700 EE E9 25 77 11 FE 07 21 D3 07 01 08 00 7B D3 60
0710 07 5F DB 61 FE FF C2 2D 07 09 15 C2 0D 07 21 1B
0720 FF 3E 70 A6 77 AF 32 03 FF C3 CC 07 23 0F DA 2C
0730 07 11 54 07 D5 56 3E 7F D3 60 DB 61 E6 03 FE 02
0740 DA 49 07 C0 0E 38 09 56 C9 7A FE 40 D8 CD 1B 02
0750 D6 40 57 E1 3A 1B FF 47 E6 40 CA 62 07 7A CD 1B
0760 02 57 78 E6 80 CA 85 07 3E 08 A0 C2 7E 07 3A 03
0770 FF C6 40 32 03 FF D2 CC 07 04 78 C3 C9 07 78 E6
0780 10 C2 CC 07 05 3E 80 B0 47 32 1B FF 7A 32 1F FF
0790 FE 1B C2 9C 07 E1 D1 C1 F1 C3 56 04 FE 1C C2 A7
07A0 07 78 EE 40 C3 C9 07 FE 1D C2 B2 07 78 EE 10 C3
07B0 C9 07 32 32 FF 21 F7 05 0E 64 3E 08 AE 2B 0D C2
07C0 BC 07 B7 C2 CC 07 3E A0 B0 32 1B FF 3E 61 D3 00
07D0 C3 13 01 78 7A 73 61 77 71 32 31 76 63 66 64 72
07E0 65 34 33 6E 62 68 67 79 74 36 35 2C 6D 6B 6A 69
07F0 75 38 37 2F 2E 3B 6C 70 6F 30 39 09 5C 7F 5B 5D

0800 40 3A 3D 20 FF 0D 0A 1F 08 1E 5E 58 5A 53 41 57
0810 51 22 21 56 43 46 44 52 45 24 23 4E 42 48 47 59
0820 54 26 25 3C 4D 4B 4A 49 55 28 27 3F 3E 2B 4C 50
0830 4F 30 29 09 7C 5F 7B 7D 60 2A 2D 20 FF 0D 0A 1F
0840 08 1E 7E 06 03 CD 09 00 DA 3D 05 C1 E1 D1 78 B1
0850 CA 59 08 CD 53 02 C3 52 09 7C B5 CA 7E 08 CD CB
0860 01 CD AC 01 7B E6 F0 5F CD AC 09 CD ED 03 D2 52
0870 09 1B EB CD A1 00 EB 13 DA 68 08 C3 52 09 21 4A
0880 FF 7E 32 5F FF 36 C9 CD AC 01 EB 7D E6 F0 5F 54
0890 D5 E5 21 00 F9 22 35 FF 0E 10 CD AC 09 0D C2 9A
08A0 08 E1 E5 7D 93 47 E6 0F 4F 07 81 C6 05 4F 78 07
08B0 07 47 E6 C0 81 4F 78 E6 03 47 21 00 F9 09 22 35
08C0 FF D1 0E FF C5 21 5A F8 CD 5E 02 C1 CD 18 02 F5
08D0 CD 45 00 F1 FE 0D CA 4B 09 FE 4E CA 3E 09 FE 50
08E0 CA 90 09 CD D3 00 FE 1F CA 17 09 FE 1E CA A1 09
08F0 FE 0A CA 96 09 FE 08 CA 62 09 CD 63 00 21 46 09

0900 DA 3B 00 0C 0D CA 0F 09 07 07 07 07 06 0F 21 06
0910 F0 F5 1A A0 E1 B4 12 79 EE FF 4F CA C4 08 3E 1F
0920 CD D3 00 13 7B E6 0F C2 C4 08 C5 01 10 00 CD 45
0930 02 C1 3E FD BC C2 C4 08 21 10 00 C3 88 09 21 00
0940 01 D1 19 C3 8B 08 EB D1 C3 90 08 D1 3A 5F FF 32
0950 4A FF CD 45 00 D1 CD A7 00 13 FE 3B C2 56 09 C3
0960 FB 04 79 EE FF 4F C2 C4 08 3E 08 CD D3 00 1B 7B
0970 E6 0F FE 0F C2 C4 08 C5 01 F0 FF CD 45 02 C1 3E
0980 F8 BC C2 C4 08 21 F0 FF EB E3 19 EB E1 C3 90 08
0990 21 00 FF C3 41 09 21 10 00 19 EB 2A 35 FF C3 32
09A0 09 21 F0 FF 19 EB 2A 35 FF C3 7F 09 CD 33 02 CD
09B0 C6 01 06 10 1A 13 CD 38 02 CD C6 01 05 C2 B4 09
09C0 C3 CB 01 CD 3A 0A 0E 00 79 0F 0F B7 17 17 17 5F
09D0 16 00 21 84 F9 19 22 35 FF CD 18 02 F5 CD 45 00
09E0 F1 FE 0D CA 11 0A FE 08 CA 08 0A FE 1F CA FE 09
09F0 47 CD 63 00 21 C8 09 DA 3B 00 78 CD D3 00 0C 79

0A00 FE 18 CA C6 09 C3 C8 09 0D F2 C8 09 0E 17 C3 C8
0A10 09 11 84 F9 21 31 FF 06 06 E5 21 00 00 0E 04 CD
0A20 D9 03 EB E3 72 2B 73 2B D1 13 13 13 13 05 C2 19
0A30 0A CD 63 0A CD CB 01 C3 55 09 CD AC 01 CD CB 01
0A40 21 04 F9 22 35 FF 11 8E 05 0E 06 06 03 1A CD CD
0A50 01 13 05 C2 4D 0A 06 05 CD C6 01 05 C2 58 0A 0D
0A60 C2 4B 0A CD CB 01 21 84 F9 22 35 FF 21 31 FF 0E
0A70 06 56 2B 5E 2B CD 33 02 06 04 CD C6 01 05 C2 7A
0A80 0A 0D C2 71 0A C9 CD AD 02 3E FF 47 CD 09 03 05
0A90 C2 8C 0A 3E 16 CD 09 03 06 10 CD A7 00 13 FE 2A
0AA0 C2 A5 0A 3E AC CD 09 03 05 CA FF 0A E6 7F FE 2C
0AB0 C2 9A 0A 06 02 CD 09 00 DA FE 0A D1 E1 CD 04 03
0AC0 EB CD 04 03 E5 21 00 00 22 38 FF E1 EB 0E 0A 7E
0AD0 CD 09 03 CD 93 00 23 DA CF 0A CD F3 02 AF 32 4D
0AE0 FF 21 F2 05 3E D1 3E C9 32 94 FF CD 48 00 21 4D
0AF0 FF CD D5 02 CD 9D 02 CD 7D 00 C3 55 09 D1 CD 9D

0B00 02 C3 3D 05 3E AF 32 5F FF CD 6B 02 DA 52 09 01
0B10 10 00 D5 21 4C FF CD E1 02 23 32 92 FF E6 7F 77
0B20 0D CA E2 0B FE 2C CA 35 0B CD A7 00 13 BE CA 16
0B30 0B 47 C3 16 0B CD A7 00 13 FE 3B CA 43 0B BE CA
0B40 43 0B 47 36 00 4F D5 CD E1 02 67 CD E1 02 6F CD
0B50 E1 02 57 CD E1 02 5F E5 21 00 00 22 38 FF 22 95
0B60 FF E1 3A 5F FF B7 C2 96 0B B0 C2 CB 0B 79 FE 2C
0B70 C2 8B 0B 3A 92 FF B7 FA E2 0B EB CD 31 00 E3 EB
0B80 CD B8 00 D1 DA FD 0A EB 19 EB 3E F1 F1 3E C3 32
0B90 94 FF E5 C3 9B 0B 47 AF 32 92 FF 0E 01 CD E1 02
0BA0 05 04 C2 A6 0B 77 CD 93 00 23 DA 9D 0B CD E1 02
0BB0 CD E1 02 2A 38 FF 7D B4 3A 92 FF 21 D4 05 C2 E9
0BC0 0B 21 D1 05 B7 F2 E5 0A C3 9D 02 D5 E5 21 4D FF
0BD0 CD 48 00 21 52 F8 D1 CD 5E 02 36 2D 23 D1 CD 5E
0BE0 02 D1 D1 CD 6F 02 C3 0C 0B B7 F2 E5 0A E3 EB 42
0BF0 4B 1B CD 53 02 E1 C3 E6 0A FF FF FF FF FF FF 1A
```

util deci de reţinut că in timp ce comanda M cu un parametru afişează doar pe CRT, comanda M cu doi parametrii utilizează şi vectorul OVECT.

c. Format: Mssss,eeee,dddd<cr>

Comanda M cu trei parametri va produce mutarea datelor din memorie de la adresa ssss pina la adresa eeee inclusiv, la adresa de destinaţie dddd. Conţinutul cimpului sursă va rămine neschimbat.

ATENTIE ! Dacă adresa dddd este situată intre adresele ssss şi eeee, monitorul va executa mutarea pină ce zona sursă va fi epuizată, dar incepind cu adresa dddd datele vor fi eronate. Această proprietate poate fi folosită pentru a executa un "Fill memory". De exemplu, dacă la adresa 4000 se introduce data FF şi se execută comanda

M4000,7FFF,4001<cr>

toată zona de memorie cuprinsă intre 4000 şi 8000 inclusiv va conţine octeţi de FF.

. Comanda "R"

Format: R<cr>

Comanda R oferă posibilitatea examinării şi modificării conţinutului registrelor procesorului. Pe ecran se va afişa un tabel ca in exemplul de mai jos:

A F	B C	D E	H L	PC	SP
07F3	0010	41FF	4100	01C5	FF00

Folosind tastele de orientare a cursorului, se poate ajunge şi modifica conţinutul oricăruia din registrele procesorului, singurele date valide ce pot fi introduse fiind caractere HEX. In cazul unei greşeli, un mesaj de eroare va fi afişat pe STATUS, fără insă să altereze valoarea registrului respectiv. Ieşirea din comandă se face prin apăsarea tastei RETURN.

Dacă inainte de execuţia comenzii R s-a executat comanda FP, registrele vor fi tipărite şi prin OVECT (adică pe imprimantă), atit inainte cit şi după eventualele corecţii aplicate.

De subliniat faptul că registrele afişate reprezintă de fapt o reflectare a acestora in niste locaţii de memorie; reactualizarea locaţiilor respective nu se face decit in urma execuţiei unei instrucţiuni RST 7 sau a unei comenzi de revenire forţată in monitor, cu ESCAPE (ctrl [).

(CONTINUARE IN NUMARUL VIITOR)

MICROCALCULATORUL

L/IB 881

NICOARA PAULIAN
LIVIU IONESCU
ION RUSOVICI
GHEORGHE CHITA

1.6. Comanda "S"

Format: Stitlu,ssss,eeee<cr>

Comanda S asigură salvarea unei zone de memorie pe caseta magnetică. Casetofonul trebuie să fie conectat şi pornit pe poziţia ÎNREGISTRARE. Comanda este executată imediat, monitorul afişind mesajul "System busy" pe STATUS. Datele sint transferate cu o viteză de aprox. 2000 Bd, inceputul fiecărei inregistrări conţinind un preambul de 256 octeţi de OFFH, care se poate recunoaşte uşor după tonul constant de 1000Hz.

Parametrii comenzii sint:

- titlu, care reprezintă numele dat de utilizator inregistrării respective, nume după care va fi recunoscută ulterior la incărcarea in memorie. Lungimea maximă a titlului este de 16 caractere alfanumerice. De semnalat că titlurile cu litere mari şi mici sint echivalente.

- ssss şi eeee sint adresele de inceput, respectiv de sfirşit ale zonei de memorie ce va fi salvată.

Preambulul de 1000Hz este transmis indiferent dacă există sau nu o eroare in lista parametrilor. Comanda va calcula şi CRC-ul, care va fi depus pe bandă la sfîrşitul inregistrării. În timpul inregistrării, pe STATUS va apare un indicator clipitor, care simbolizează faptul că datele sint in curs de transferare. La sfîrşitul inregistrării se va tipări mesajul "Saved", iar controlul va fi dat interpretorului de comenzi al monitorului.

1.7. Comanda "V"

Format: V<cr>

Comanda V permite verificarea corectitudinii unei inregistrări. Verificarea se face asupra datelor inregistrate pe casetă, neflind necesară existenţa lor şi in memorie (nu se face nici un transfer sau comparaţie cu memoria). Comanda se execută imediat, se afişează "System busy", iar microcalculatorul aşteaptă inceputul unei inregistrări (preambulul). Dacă acesta se recepţionează corect, monitorul va incepe verificarea inregistrării şi va activa indicatorul clipitor de pe STATUS, după care va fi afişat un mesaj ce reflectă dacă există sau nu erori pe banda magnetică. Verificarea are loc prin CRC.

1.8. Comanda "X"

Format: X<cr>

Comanda X este concepută ca o extensie la lista de comenzi acceptate de monitor. Executarea ei constă dintr-un salt necondiţionat la adresa XVECT, unde incepe extensia, adresă setată anterior prin comanda FX. ATENTIE! Acţionarea butonului de RESET va reseta şi vectorul XVECT. In cazul utilizării unei extensii se recomandă reintrarea in monitor prin vectorii RETURN (in caz de succes) sau ERROR. Totodată trebuie conservate stiva şi poziţionarea sistemului de intreruperi al monitorului.

Adresele pentru vectori:
RETURN = 0003; ERROR = 0006.

2. Observaţii

Organizarea monitorului a fost făcută astfel incit o serie de subrutine cu caracter general să poată fi utilizate de programele externe. Mai jos sint scoase in evidenţă citeva particularităţi ale unor subrutine esenţiale; vom reveni mai pe larg in numărul viitor.

Rutina OUTPUT interpretează următoarele coduri:

- 09h (tab) provoacă deplasarea cursorului pină la următoarea poziţie de tabulare, memorată in tabelul de la adresa OFF66h (max. 16 octeţi). Această tabelă se iniţializează cu valori particulare la fiecare intrare intr-un program (ex. editor, asamblor), deci utilizatorul este dator să prevadă in programul său o astfel de iniţializare. De exemplu, editorul din cadrul sistemului de operare extins 881/Sys iniţializează tabulatorii din 8 in 8.
- ODh (return) este automat insoţit de un line feed (OAh).
- OCh (form feed) realizează atit ştergerea ecranului, cit şi saltul la pagină nouă pe imprimantă (dacă vectorul OVECT este setat in acest sens).

(CONTINUARE ÎN NUMĂRUL VIITOR)

```
881/Mon (C) 1985 Lixco Software MACRO-80 3.36  17-Mar-80      PAGE   1-2
User Low Level Monitor Routines
                        ;aschex Converts ASCII character to hex digit.
                        ;   Input: A = 8 bit data, ASCII char.
                        ;   Output: CY = 1 hex error,
                        ;           A = 8 bit hex digit.
                        ;   Destroys: AF.

0063                    aschex:
0063  D6 30                     sui     '0'
0065  D8                       rc
0066  FE 17                    cpi     17h
0068  3F                       cmc
0069  D8                       rc
006A  FE 0A                    cpi     0Ah
006C  3F                       cmc
006D  D0                       rnc
006E  D6 07                    sui     7
0070  FE 0A                    cpi     0Ah
0072  C9                       ret
                        ;hexasc Converts a hex digit to corresponding ASCII cha
                        ;   Input: A = 8 bit hex digit.
                        ;   Output: A = 8 bit data, ASCII char.
                        ;   Destroys: AF.

0073                    hexasc:
0073  E6 0F                    ani     0Fh
0075  C6 30                    adi     '0'
0077  FE 3A                    cpi     '9'+1
0079  D8                       rc
007A  C6 07                    adi     7
007C  C9                       ret
                        ;bell   Sends a 2.6 KHz signal to a loudspeaker.
                        ;   Destroys: AF.

007D                    bell:
007D  C5                       push    b
007E  01 0518                  lxi     b,val00
0081  78                       mov     a,b
0082  CD 0390                 call    beep
0085  C1                       pop     b
0086  C9                       ret

                        ;serdrv Soft driver for serial port.
                        ;   Input: A = 8 bit data to be send.

0087                    serdrv:
0087  F5                       push    psw
0088                    serdr5:
0088  DB 31                    in      sersta
008A  E6 01                    ani     1
008C  CA 0088                  jz      serdr5
008F  F1                       pop     psw
0090  D3 30                    out     serdat
0092  C9                       ret

0093                    blink:
0093  0D                       dcr     c
0094  C2 00A1                  jnz     hilo
0097  0E 0A                    mvi     c,10
0099  3A F876                  lda     row1+118
009C  EE 80                    xri     80h
009E  32 F876                  sta     row1+118
```

```
881/Mon (C) 1985 Lixco Software MACRO-80 3.36  17-Mar-80      PAGE   1-3
User Low Level Monitor Routines
                        ;hilo   Two bytes comparison (HL & DE).
                        ;   Output: CY = 1 DE>HL
                        ;           CY = 0 DE<=HL
                        ;           Z = 1 DE=HL.
                        ;   Destroys: AF.

00A1                    hilo:
00A1  7C                       mov     a,h
00A2  BA                       cmp     d
00A3  C0                       rnz
00A4  7D                       mov     a,l
00A5  BB                       cmp     e
00A6  C9                       ret
                        ;getch  Gets a char. from Disp and converts it to capitals.
                        ;   Input: DE = pointer on Display.
                        ;   Output: CY = 1 if end of Display line,
                        ;           A = 8 bit char. or ';' if CY = 1.
                        ;   Destroys: AF, DE.

00A7                    getch:
00A7  3E 3F                    mvi     a,63
00A9  BB                       cmp     e
00AA  3E 3B                    mvi     a,term
00AC  D8                       rc
00AD  1A                       ldax    d
00AE  CD 021B                  call    conv
00B1  FE 20                    cpi     ' '
00B3  C0                       rnz
00B4  13                       inx     d
00B5  C3 00A7                  jmp     getch
                        ;gethx  Takes a parameter from Display line.
                        ;   Input: DE = pointer on param's first digit.
                        ;   Output: CY = 1 hex error,
                        ;           HL = 16 bit data, parameter,
                        ;           DE = 16 bit addr, pointer on Display after
                        ;             separator,
                        ;           A = 8 bit data, param's separator.
                        ;   Destroys: all.

00B8                    gethx:
00B8  21 0000                  lxi     h,0
00BB  44                       mov     b,h
00BC                    geth05:
00BC  CD 00A7                  call    getch
00BF  13                       inx     d
00C0  FE 2C                    cpi     ','
00C2  C8                       rz
00C3  FE 3B                    cpi     term
00C5  C8                       rz
00C6  CD 0063                  call    aschex
00C9  D8                       rc
00CA  29                       dad     h
00CB  29                       dad     h
00CC  29                       dad     h
00CD  29                       dad     h
00CE  4F                       mov     c,a
00CF  09                       dad     b
00D0  C3 00BC                  jmp     geth05

                        ;locrt  Handles CRT-controller.
                        ; Writes a char. in Main area at cursor CRT position.
```

```
                        ; Interprets following control codes:
                        ;   * arrows (up, down, right and left).
                        ;   * FF - Form Feed, clear screen code.
                        ;   * Bell.
                        ;   * SI and SO for video reverse and normal video.
                        ; Other ASCII codes lower then 20h are ignored.
                        ;   Input: A = 8 bit char.

0003                    locrt:
0003 F5                         push    psw
0004 C5                         push    b
0005 D5                         push    d
0006 E5                         push    h
0007 47                         mov     b,a
0008 2A FF35                    lhld    mcurs
000B FE 20                      cpi     ' '
000D D2 0118                    jnc     print
00E0 FE 1F                      cpi     rar
00E2 CA 011D                    jz      htcmd
00E5 FE 1E                      cpi     uar
00E7 CA 0148                    jz      vtcmd
00EA FE 08                      cpi     bs
00EC CA 016C                    jz      bscmd
00EF FE 0A                      cpi     lf
00F1 CA 0165                    jz      lfcmd
00F4 FE 0D                      cpi     cr
00F6 CA 015C                    jz      crcmd
00F9 FE 0C                      cpi     ff
00FB CC 01AC                    cz      clsc
00FE FE 07                      cpi     bel
0100 CC 007D                    cz      bell
0103 FE 0F                      cpi     si
0105 CA 010E                    jz      revon+1
0108 FE 0E                      cpi     so
010A C2 0113                    jnz     exit
010D                    revon:
010D 2F                         cma
010E E6 80                      ani     80h
0110 32 FF37                    sta     revsw
0113                    exit:
0113 E1                         pop     h
0114 D1                         pop     d
0115 C1                         pop     b
0116 F1                         pop     psw
0117 C9                         ret
0118                    print:
0118 3A FF37                    lda     revsw
011B 80                         add     b
011C 77                         mov     m,a
011D                    htcmd:
011D 23                         inx     h
011E                    pr10:
011E 11 FE80                    lxi     d,0FE80h
0121 CD 00A1                    call    hilo
0124 DA 0156                    jc      pr40
0127 E5                         push    h
0128 01 F880                    lxi     b,row1+128
012B 11 F8C0                    lxi     d,row1+192
012E 21 05C0                    lxi     h,1600-128
0131                    pr20:
0131 1A                         ldax    d
0132 02                         stax    b
0133 13                         inx     d
```



```
0134 03                         inx     b
0135 2B                         dcx     h
0136 7C                         mov     a,h
0137 B5                         ora     l
0138 C2 0131                    jnz     pr20
013B 01 2040                    lxi     b,2040h
013E 21 FE40                    lxi     h,row1+1600
0141                    pr30:
0141 70                         mov     m,b
0142 23                         inx     h
0143 0D                         dcr     c
0144 C2 0141                    jnz     pr30
0147 E1                         pop     h
0148                    vtcmd:
0148 11 FFC0                    lxi     d,-64
014B 19                         dad     d
014C                    pr35:
014C 11 F880                    lxi     d,row1+128
014F CD 00A1                    call    hilo
0152 D2 0156                    jnc     pr40
0155 EB                         xchg
0156                    pr40:
0156 22 FF35                    shld    mcurs
0159 C3 0113                    jmp     exit
015C                    crcmd:
015C 3E C0                      mvi     a,0C0h
015E A5                         ana     l
015F 6F                         mov     l,a
0160 C3 0156                    jmp     pr40
0163 1001               mic:    dw      1001h
0165                    lfcmd:
0165 11 0040                    lxi     d,64
0168 19                         dad     d
0169 C3 011E                    jmp     pr10
016C                    bscmd:
016C 2B                         dcx     h
016D C3 014C                    jmp     pr35
                        ;disp   Writes a char. in Display line, at cursor crt. position
                        ; Interprets folowing control codes:
                        ;   * arrows (left and right).
                        ;   Input: A = 8 bit char.
0170                    disp:
0170 E5                         push    h
0171 F5                         push    psw
0172 2A FF33                    lhld    dcurs
0175 FE 20                      cpi     ' '
0177 D2 0187                    jnc     dprint
017A FE 1F                      cpi     rar
017C CA 0188                    jz      htdcmd
017F FE 08                      cpi     bs
0181 CA 0196                    jz      bsdcmd
0184                    dexit:
0184 F1                         pop     psw
0185 E1                         pop     h
0186 C9                         ret
0187                    dprint:
0187 77                         mov     m,a
0188                    htdcmd:
0188 23                         inx     h
0189 3E 40                      mvi     a,64
018B BD                         cmp     l
018C C2 0190                    jnz     dpr10
```



```
018F 2B                         dcx     h
0190                    dpr10:
0190 22 FF33                    shld    dcurs
0193 C3 0184                    jmp     dexit
0196                    bsdcmd:
0196 7D                         mov     a,l
0197 B7                         ora     a
0198 CA 0190                    jz      dpr10
019B C3 018F                    jmp     dpr10-1

                        ;cldis  Clears Display line and resets DCURS pointer.
                        ;   Destroys: HL, DE, F.

019E                    cldis:
019E 21 F840                    lxi     h,row1+64
01A1                    cldis1:
01A1 2D                         dcr     l
01A2 36 20                      mvi     m,' '
01A4 C2 01A1                    jnz     cldis1
01A7 22 FF33                    shld    dcurs
01AA EB                         xchg
01AB C9                         ret

                        ;clsc   Clears screen (Main area) and resets MCURS pointer home

01AC                    clsc:
01AC F5                         push    psw
01AD C5                         push    b
01AE D5                         push    d
01AF E5                         push    h
01B0 21 F880                    lxi     h,row1+128
01B3 01 0600                    lxi     b,1664-128
01B6                    cls05:
01B6 36 20                      mvi     m,' '
01B8 23                         inx     h
01B9 0B                         dcx     b
01BA 79                         mov     a,c
01BB B0                         ora     b
01BC C2 01B6                    jnz     cls05
01BF 21 F880                    lxi     h,row1+128
01C2 C3 0156                    jmp     pr40
01C5 FF                 uic:    db      0FFh

                        ;spout  Sends a blank to console via output routine.
                        ;   Destroys: AF.

01C6                    spout:
01C6 3E 20                      mvi     a,' '
01C8 C3 01CD                    jmp     output

                        ;crout  Sends a CR to console via output routine.
                        ;   Destroys: AF.

01CB                    crout:
01CB 3E 0D                      mvi     a,cr

                        ;output Sends a char. from A to locrt and ovect. Doubles CR
                        ; with LF. Interprets TAB code. Pages listing on printer.
                        ;   Input: A = 8 bit char. to send.
                        ;   Destroys: F (and A if CR).

01CD                    output:
01CD CD 00D3                    call    locrt
01D0 FE 0A                      cpi     lf
```



```
01D2 C2 0000*                   jnz     out05
01D5 CD FF4A                    call    ovect
01D8 C3 0000*                   jmp     pagrut

                        ;input  Takes byte from keyboard buffer, if kstat bit 5 is set.
                        ;   Handles both cursors.
                        ;   Output: A = 8 bit data from keyboard.
                        ;   Destroys: AF.

01DB                    input:
01DB E5                         push    h
01DC 2A FF35                    lhld    mcurs
01DF 7E                         mov     a,m
01E0 32 FF13                    sta     cschar
01E3 2A FF33                    lhld    dcurs
01E6 7E                         mov     a,m
01E7 32 FF17                    sta     dschar
01EA 3A FF0B                    lda     cursw
01ED 32 FF44                    sta     cursw2
01F0 21 FF1B                    lxi     h,kstat
01F3                    in10:
01F3 7E                         mov     a,m
01F4 E6 20                      ani     20h
01F6 CA 01F3                    jz      in10
01F9 7E                         mov     a,m
01FA E6 DF                      ani     0DFh
01FC 77                         mov     m,a
01FD 3A FF32                    lda     kbuff
0200 E6 7F                      ani     7Fh
0202 F5                         push    psw
0203 AF                         xra     a
0204 32 FF44                    sta     cursw2
0207 3A FF17                    lda     dschar
020A 2A FF33                    lhld    dcurs
020D 77                         mov     m,a
020E 3A FF13                    lda     cschar
0211 2A FF35                    lhld    mcurs
0214 77                         mov     m,a
0215 F1                         pop     psw
0216 E1                         pop     h
0217 C9                         ret

                        ;inputc Reads a char. via input routine and converts it to
                        ; capitals.
                        ;   Destroys: AF.

0218                    inputc:
0218 CD 01DB                    call    input

                        ;conv   Converts to capitals the character from A register.
                        ;   Destroys: AF.

021B                    conv:
021B FE 61                      cpi     'a'
021D D8                         rc
021E FE 7B                      cpi     'z'+1
0220 D0                         rnc
0221 D6 20                      sui     20h
0223 C9                         ret

                        ;convns Dispatches byte from A in two ASCII char's.
                        ;   Input: A = 8 bit, two hex digits.
                        ;   Output: BC = 16 bit ASCII corresponding two bytes.
                        ;   Destroys: AF, BC.
```

MICROCALCULATORUL L/B 881

NICOARA PAULIAN
LIVIU IONESCU
ION RUSOVICI
GHEORGHE CHITA

3. SISTEMUL DE ÎNTRERUPERI

Microprocesorul 8080 are implementată posibilitatea lucrului pe intreruperi prin prezenţa unui pin avind această funcţiune (INT), ceea ce ii permite să gestioneze un singur nivel de întreruperi fără logică externă. În general, microcalculatoarele sint prevăzute să lucreze pe intreruperi sub o formă sau alta; in cazul lui L/B881 intreruperile sint deosebit de importante intrucit au condus la optimizarea hardware a construcţiei sale. Datorită faptului că un singur nivel de intrerupere era insuficient, a fost inclus in sistem un controlor programabil (Intel 8259) ce ii asigură o deosebită flexibilitate.

În continuare vom explica modul in care L/B881 foloseşte sistemul de intreruperi pentru a veni in ajutorul utilizatorului in dezvoltarea programelor sale. De la inceput trebuie subliniat faptul că L/B881 funcţionează in permanenţă sub un mediu "intrerupt", fapt ce are unele implicaţii in dezvoltarea programelor.

3.1. Aspecte hardware

Din punct de vedere hardware, sistemul de intreruperi al microcalculatorului L/B881 se compune dintr-un controlor de intreruperi 8259 şi liniile INT/INTA ale grupului procesor 8080/8228. Mecanismul de funcţionare este următorul: in momentul in care oricare din liniile IR0-IR7 ale circuitului 8259 devine activă (trece in unu), aceasta activează linia INT a procesorului; dacă procesorul are intreruperile nemascate (bistabilul intern INTE este in poziţia activă), el termină instrucţiunea in curs şi recunoaşte intreruperea prin activarea liniei INTA către 8259 care pune pe magistrala de date instrucţiunea CALL (0CDh). Ca urmare, procesorul execută incă două cicluri de citire (activind linia INTA de două ori) la care 8259 depune pe magistrală vectorul intreruperii respective, după cum a fost programat la iniţializare. Pentru simplificare, in descrierea de mai sus au fost omise sistemele de determinare a priorităţii şi a măştii de intreruperi; detalii suplimentare pot fi găsite in foaia de catalog a circuitului 8259 sau intr-una din numeroasele lucrări apărute pe această temă.

Cele 8 nivele de intreruperi sint alocate după cum urmează:

- IR0 liber;
- IR1 claviatură, ceas de timp real, cursoare;
- IR2, IR3 USART;
- IR4, IR5 sistem casetă;
- IR6, IR7 liber;

Nivelele libere sint scoase la conector şi pot fi utilizate in aplicaţii ce necesită intreruperi; similar, IR2/IR3 care sint scoase la conector şi au prevăzute două jumpere pe placa de interfaţă a conectorului KB: in cazul in care nu se lucrează cu USART-ul pe intreruperi, ele pot fi folosite in alte scopuri. IR1 este insă dedicat monitorului: pe el este cuplat un semnal de 50 Hz (20 ms) ce provine din circuitele de sincronizare pe cadre TV. După cum vom arăta in continuare, monitorul foloseşte aceste intreruperi pentru rezolvarea claviaturii, a ceasului de timp real şi a cursoarelor. Nivelele IR4 şi IR5 sint dedicate casetei, dar IR5 care este legat la ieşirea OUT2 a divizorului programabil 8253 poate fi folosit şi in alte scopuri. IR5/OUT2 formează un instrument puternic de lucru in stabilirea unor intervale de timp exacte in cadrul unor programe, ca urmare a posibilităţii de intrerupere a procesorului de către divizorul rogramabil.

3.2. Aspecte software

Monitorul 881/Mon (ca şi 881/Sys) iniţializează controlorul de intreruperi alocind o tabelă de vectori in RAM incepind de la adresa 0FF00h. Toate nivelele sint mascate, mai puţin IR1. Vectorii sint dispuşi la un interval de patru octeţi, după cum urmează:

0FF00 = IR0	0FF04 = IR1
0FF08 = IR2	0FF0C = IR3
0FF10 = IR4	0FF14 = IR5
0FF18 = IR6	0FF1C = IR6

Atenţie la schimbarea vectorilor de intreruperi: ei nu ocupă decit trei octeţi (JMP ADDR), al patrulea este folosit de monitor ca locaţie de variabilă şi nu trebuie distrus!

(CONTINUARE ÎN NUMĂRUL VIITOR)

```
881/Mon (C) 1985 Lixco Software MACRO-80 3.36  17-Mar-80      PAGE    1-8
User Low Level Monitor Routines

0224                            cnvnm:
0224  4F                                mov     c,a
0225  0F                                rrc
0226  0F                                rrc
0227  0F                                rrc
0228  0F                                rrc
0229  CD 0073                           call    hexasc
022C  47                                mov     b,a
022D  79                                mov     a,c
022E  CD 0073                           call    hexasc
0231  4F                                mov     c,a
0232  C9                                ret

                                ;wdout  Sends two bytes (ASCII packed) to console via output
                                ; routine.
                                ;   Input: DE = 16 bit data.
                                ;   Destroys: AF.

0233                            wdout:
0233  7A                                mov     a,d
0234  CD 0238                           call    nmout
0237  7B                                mov     a,e

                                ;nmout  Sends byte (ASCII packed) to console via output routine
                                ;   Input: A = 8 bit data.
                                ;   Destroys: AF.

0238                            nmout:
0238  C5                                push    b
0239  CD 0224                           call    cnvnm
023C  78                                mov     a,b
023D  CD 01CD                           call    output
0240  79                                mov     a,c
0241  C1                                pop     b
0242  C3 01CD                           jmp     output

                                ;corr   Corrects MCURS position.
                                ;   Input: CY = 1 and BC = cursor's absolute addr. in screen,
                                ;          CY = 0 and BC = positive or negative offset for
                                ;          current position.
                                ;   Output: HL = 16 bit addr, cursor's new position.
                                ;   Destroys: AF, HL.

0245                            corr:
0245  21 0000                           lxi     h,0
0248  DA 024E                           jc      corr05
024B  2A FF35                           lhld    mcurs
024E                            corr05:
024E  09                                dad     b
024F  22 FF35                           shld    mcurs
0252  C9                                ret

                                ;mvsr   Moves memory to right, between given addr's.
                                ;   Input: DE = 16 bit addr, begin of source area,
                                ;          HL = 16 bit addr, end of source area,
                                ;          BC = 16 bit addr, begin of destination area.
                                ;   Destroys: AF, BC, DE.

0253                            mvsr:
0253  1A                                ldax    d
0254  02                                stax    b
0255  CD 00A1                           call    hilo
0258  03                                inx     b
```

```
881/Mon (C) 1985 Lixco Software MACRO-80 3.36  17-Mar-80      PAGE    1-9
User Low Level Monitor Routines

0259  13                                inx     d
025A  C2 0253                           jnz     mvsr
025D  C9                                ret

                                ;wdsta  Converts in ASCII four hex digits and stores them in
                                ; memory at HL addr. (possible a screen addr.)
                                ;   Input: DE = 16 bit data, hex digits,
                                ;          HL = 16 bit addr, pointer in memory.
                                ;   Destroys: AF, BC, HL.

025E                            wdsta:
025E  7A                                mov     a,d
025F  CD 0263                           call    twoset
0262  7B                                mov     a,e

                                ;twoset Converts in ASCII two hex digits and stores them in
                                ; memory at HL addr. (possible a screen addr.)
                                ;   Input: A = 8 bit data, hex digits,
                                ;          HL = 16 bit addr, pointer in memory.
                                ;   Destroys: AF, BC, HL.

0263                            twoset:
0263  CD 0224                           call    cnvnm
0266  70                                mov     m,b
0267  23                                inx     h
0268  71                                mov     m,c
0269  23                                inx     h
026A  C9                                ret

                                ;crinit Inits cassette device for read operation. Routine first
                                ; hunts a SYNC code and then passes to Working Mode.
                                ;   Output: CY = 1 if "Cancel" from kbrd. in Hunt Mode.
                                ;   Destroys: AF, BC, HL.

026B                            crinit:
026B  3E B1                             mvi     a,cwt2
026D  D3 13                             out     timsta
026F  06 CF                             mvi     b,0CFh
0271  3E F1                             mvi     a,0F1h
0273  32 FF3A                           sta     cstat
0276  21 0000*                          lxi     h,ird4
0279  22 FF11                           shld    ir4+1
027C  21 0000*                          lxi     h,ird5
027F  CD 028E                           call    cinit
0282  21 FF1B                           lxi     h,kstat
0285                            crin05:
0285  3A FF3A                           lda     cstat
0288  E6 01                             ani     1
028A  C8                                rz
028B  7E                                mov     a,m
028C  E6 20                             ani     20h
028E  CA 0285                           jz      crin05
0291  7E                                mov     a,m
0292  E6 DF                             ani     0DFh
0294  77                                mov     m,a
0295  3A FF32                           lda     kbuff
0298  FE 18                             cpi     can
029A  C2 0285                           jnz     crin05

                                ;casend Disables cassette device at end of transfer operation.
                                ;   Destroys: AF.

029D                            casend:
029D  3E C7                             mvi     a,icw4
```

```
029F  D3 00                   out     intct0
02A1  DB 01                   in      intct1
02A3  F6 30                   ori     30h
02A5  D3 01                   out     intct1
02A7  3E BF                   mvi     a,cwt20
02A9  D3 13                   out     timsta
02AB  37                      stc
02AC  C9                      ret

                      ;cwinit Inits cassette device for write operation.
                      ;   Destroys: AF, BC, HL.

02AD                  cwinit:
02AD  06 DF                   mvi     b,0DFh
02AF  3E BF                   mvi     a,cwt20
02B1  D3 13                   out     timsta
02B3  3E 75                   mvi     a,val21l
02B5  D3 12                   out     timer2
02B7  3E 13                   mvi     a,val21h
02B9  D3 12                   out     timer2
02BB  21 0000*                lxi     h,iwr5
02BE                  cinit:
02BE  22 FF15                 shld    ir5+1
02C1  3E C3                   mvi     a,0C3h
02C3  32 FF10                 sta     ir4
02C6  32 FF14                 sta     ir5
02C9  3E C3                   mvi     a,icw3
02CB  D3 00                   out     intct0
02CD  DB 01                   in      intct1
02CF  A0                      ana     b
02D0  D3 01                   out     intct1

                      ;waitms Prints "System busy" in the middle of Status line.
                      ;   Destroys: AF, BC, HL.

02D2                  waitms:
02D2  21 0000*                lxi     h,msg3

                      ;nwstri Prints an ASCII string in the middle of Status line.
                      ;   Input: HL = 16 bit addr, string's begining (terminator
                      ;          is null).
                      ;   Destroys: AF, BC, HL.

02D5                  nwstri:
02D5  01 F860                 lxi     b,row1+96
02D8                  nwstr1:
02D8  7E                      mov     a,m
02D9  B7                      ora     a
02DA  C8                      rz
02DB  02                      stax    b
02DC  23                      inx     h
02DD  03                      inx     b
02DE  C3 02D8                 jmp     nwstr1

                      ;casin  Takes a byte from cassette buffer if cstat bit 1 is set
                      ; Shifts bits through CRC.
                      ;   Output: A = 8 bit data from tape.
                      ;   Destroys: AF.

02E1                  casin:
02E1  3A FF3A                 lda     cstat
02E4  E6 02                   ani     2
02E6  CA 02E1                 jz      casin
02E9  AF                      xra     a
```



```
02EA  32 FF3A                 sta     cstat
02ED  3A FF3D                 lda     byte
02F0  C3 031B                 jmp     crcrut

                      ;cwrend Shifts through CRC two null bytes and writes CRC
                      ; code on magnetic tape.
                      ;   Destroys: AF, BC, HL.

02F3                  cwrend:
02F3  AF                      xra     a
02F4  06 10                   mvi     b,16
02F6  21 02FE                 lxi     h,cwrd5
02F9  E5                      push    h
02FA  C5                      push    b
02FB  CD 031F                 call    crc03
02FE                  cwrd5:
02FE  2A FF38                 lhld    crc
0301  CD 0304                 call    wdcas

                      ;wdcas  Sends two bytes to magnetic tape and shifts them
                      ; through CRC.
                      ;   Input: HL = 16 bit data.
                      ;   Destroys: AF.

0304                  wdcas:
0304  7C                      mov     a,h
0305  CD 0309                 call    casout
0308  7D                      mov     a,l

                      ;casout Sends a byte from A to magnetic tape and shifts it
                      ; through CRC.
                      ;   Input: A = 8 bit data.
                      ;   Destroys: AF.

0309                  casout:
0309  F5                      push    psw
030A  3A FF3A                 lda     cstat
030D  E6 02                   ani     2
030F  C2 030A                 jnz     casout+1
0312  F6 02                   ori     2
0314  32 FF3A                 sta     cstat
0317  F1                      pop     psw
0318  32 FF3D                 sta     byte

                      ;crcrut Cyclic Redundancy Check routine. Shifts byte from A
                      ; through CRC. Result (CRC word) is in RAM location "CRC"
                      ;   Input: A = 8 bit data.
                      ;   Output: RAM loc."CRC" = new rest.
                      ;   Destroys: AF.

031B                  crcrut:
031B  C5                      push    b
031C  E5                      push    h
031D  06 08                   mvi     b,8
031F                  crc03:
031F  F5                      push    psw
0320  2A FF38                 lhld    crc
0323  17                      ral
0324  7D                      mov     a,l
0325  17                      ral
0326  6F                      mov     l,a
0327  7C                      mov     a,h
0328  17                      ral
0329  67                      mov     h,a
```



```
032A  D2 0334                 jnc     crc05
032D  EE 10                   xri     pgen
032F  67                      mov     h,a
0330  7D                      mov     a,l
0331  EE 21                   xri     pgen1
0333  6F                      mov     l,a
0334                  crc05:
0334  22 FF38                 shld    crc
0337  F1                      pop     psw
0338  07                      rlc
0339  05                      dcr     b
033A  C2 031F                 jnz     crc03
033D  E1                      pop     h
033E  C1                      pop     b
033F  C9                      ret

                      ;onkey  Handles a jump table. Every entry is a key and an addr.
                      ; Terminator is 0FFh. If key not found, returns.
                      ;   Input: BC = 16 bit addr. of table,
                      ;          H = 8 bit data, key.
                      ;   Destroys: AF, BC, HL.

0340                  onkey:
0340  0A                      ldax    b
0341  03                      inx     b
0342  FE FF                   cpi     0FFh
0344  C8                      rz
0345  BC                      cmp     h
0346  CA 034E                 jz      onky05
0349  03                      inx     b
034A  03                      inx     b
034B  C3 0340                 jmp     onkey
034E                  onky05:
034E  0A                      ldax    b
034F  6F                      mov     l,a
0350  03                      inx     b
0351  0A                      ldax    b
0352  67                      mov     h,a
0353  C1                      pop     b
0354  E9                      pchl

                      ;cset,extset,brset,pset,oset
                      ; Sets real-time clock, xvect, baud-rate.
                      ; Sets ovect to serial output driver or an user addr.
                      ; Resets ovect if addr. is null.
                      ;   Input: HL = 16 bit data.
                      ;   Destroys: AF or nothing (AF, BC, HL at cset).

0355                  cset:
0355  7D                      mov     a,l
0356  F5                      push    psw
0357  7C                      mov     a,h
0358  21 F878                 lxi     h,row1+120
035B  CD 0263                 call    twoset
035E  36 3A                   mvi     m,':'
0360  23                      inx     h
0361  F1                      pop     psw
0362  CD 0263                 call    twoset
0365  36 3A                   mvi     m,':'
0367  23                      inx     h
0368  36 30                   mvi     m,'0'
036A  23                      inx     h
036B  36 30                   mvi     m,'0'
036D  C9                      ret
```



```
036E                  extset:
036E  22 FF48                 shld    xvect+1
0371  C9                      ret
0372                  brset:
0372  3E 7F                   mvi     a,cwt1
0374  D3 13                   out     timsta
0376  7D                      mov     a,l
0377  D3 11                   out     timer1
0379  7C                      mov     a,h
037A  D3 11                   out     timer1
037C  C9                      ret
037D                  pset:
037D  21 0087                 lxi     h,serdrv
0380                  oset:
0380  22 FF4B                 shld    ovect+1
0383  7C                      mov     a,h
0384  B5                      ora     l
0385  3E C9                   mvi     a,0C9h
0387  CA 038C                 jz      oset1
038A  3E C3                   mvi     a,0C3h
038C                  oset1:
038C  32 FF4A                 sta     ovect
038F  C9                      ret

                      ;beep   Sends an acoustic signal to the loudspeaker.
                      ;   Do not waits emission to complete.
                      ;   Input: BC = 16 bit BCD integer,value that will devide
                      ;          the internal 1375 KHz frequency to form
                      ;          the output signal,
                      ;          A = 8 bit integer, multiple of 20 msec, sound's
                      ;          lenght. If A = 0, infinite sound, (until Reset
                      ;          or another call beep).

0390                  beep:
0390  F5                      push    psw
0391  3E 3F                   mvi     a,cwt0
0393  D3 13                   out     timsta
0395  79                      mov     a,c
0396  D3 10                   out     timer0
0398  78                      mov     a,b
0399  D3 10                   out     timer0
039B  F1                      pop     psw
039C  32 FF45                 sta     cntbip
039F  C9                      ret


                      ;   Monitor equates:

0031                  sersta  equ     31h
0030                  serdat  equ     30h
007F                  cwt1    equ     7Fh
000D                  cr      equ     0Dh
000A                  lf      equ     0Ah
0007                  bel     equ     7
0008                  bs      equ     8
001E                  uar     equ     1Eh
000C                  ff      equ     'L'-40h
0018                  can     equ     'X'-40h
000E                  so      equ     0Eh
000F                  si      equ     0Fh
0013                  timsta  equ     13h
0010                  timer0  equ     10h
0011                  timer1  equ     11h
0012                  timer2  equ     12h
```

MICROCALCULATORUL L/B 8081

NICOARA PAULIAN
LIVIU IONESCU
ION RUSOVICI
GHEORGHE CHITA

Nivelul 1 de intrerupere funcţionează permanent şi deserveşte o rutină care rezolvă problemele de codare a claviaturii, clipirea cursoarelor şi incrementarea ceasului de timp real. Una din cele mai importante implicaţii ale acestui fapt este aceea că secvenţa de oprire a unui program utilizator cu instrucţiunea HLT nu funcţionează corect (aşa cum sint date exemple in unele lucrări de programarea microprocesorului 8080):

```
    ...
    instrucţiuni
    ...
    hlt
```

De fapt, instrucţiunea HLT trebuie interpretată mai corect drept "wait for interrupt" (aşteaptă intreruperea) decit o oprire totală a procesorului. In manualul Intel se specifică că procesorul 8080 iese din starea de HALT doar la RESET sau la activarea pinului INT (cu condiţia ca bistabilul INTE să fie activ). Deci la orice intrerupere recunoscută, procesorul iese din starea de HALT, după executarea intreruperii şi trece la execuţia instrucţiunii imediat următoare lui HLT, care in exemplul de mai sus nu este specificată, ducind la rezultate imprevizibile.

Rezolvarea corectă a situaţiei de mai sus se face cu instrucţiunea RST7, care este tratată de către monitor ca un punct de breakpoint. La intilnirea ei, procesorul execută un CALL la adresa 38H unde se află un vector spre rutina din monitor ce execută salvarea stării curente a procesorului; urmează apoi revenirea in monitor şi afişarea mesajului "Monitor control".

De asemeni instrucţiunea DI (disable interrupts) trebuie folosită cu multă grijă. Este important de reţinut faptul că la fiecare 20 msec. apare o intrerupere şi că un DI pe o perioadă mai lungă duce la pierderea corectitudinii ceasului. Programele utilizator nu trebuie să "omoare" decit pentru perioade scurte sistemul de intreruperi (circa 500 instrucţiuni).

O altă consecinţă demnă de remarcat este aceea că nu se pot obţine intirzieri prin bucle software decit dacă acestea nu sint critice. Cauza este evidentă: lungimea execuţiei unei rutine pe nivelul 1 de intrerupere este foarte variabilă, funcţie de diverşi parametrii cum ar fi valoarea momentană a variabilelor ceasului de timp real, faptul că este sau nu o tastă apăsată, momentul aprinderii sau stingerii cursorului (sau cursoarelor) etc.

Ca urmare, prin natura asincronă a apariţiei intreruperilor faţă de programul rulat nu se poate conta pe valori exacte de timp ale buclelor software. Pentru perioade de timp ce depăşesc sute de milisecunde, problema se rezolvă insă foarte elegant chiar cu ajutorul intreruperilor. Să presupunem că dorim să obţinem o intirziere de 160 msec.:

```
delay:
    mvi  b,8  ; 8 * 20 ms = 160 ms
del10:
    hlt       ;Aşteaptă intrerup.
    dcr  b    ;A venit
    jnz  del10;Dacă contorul nu e
    ret       ; zero, mai aşteaptă
```

Rutina de mai sus este foarte bună pentru timpi relativ lungi. In cazul in care dorim să obţinem intirzieri mai scurte, se poate folosi nivelul de intrerupere 5 (aşa cum vom arăta mai jos).

Datorită flexibilităţii sistemului de intreruperi al microcalculatorului L/B881, există posibilitatea execuţiei unei rutine suplimentare pe nivelul 1. Dacă este necesară execuţia unei părţi a programului utilizator la fiecare 20 msec., se poate reinitializa in acest scop vectorul nivelului 1. Sint două cazuri: cind rutina utilizatorului trebuie executată inainte de rutina monitorului sau după aceasta. In primul caz soluţia este deosebit de simplă, una din posibilităţi fiind:

```
inir1:    Iniţializarea nivelului
          ; 1 de intrerupere
    lhld ir1+1;Se salvează vecto-
    shld temp ; rul original
    lxi  h,nir;Se pune noua adre-
    shld ir1+1; să
    ...       ;Alte instrucţiuni
    ...       ; ale programului.

nir:      ;Aici este noul punct de
    push h    ; intrare a intreru-
    push d    ; perilor, nivel 1
    push b    ;Salvarea registre-
    push psw  ; lor
    ...       ;Corpul rutinei de
    ...       ; pe intrerupere
    pop  psw
    pop  b    ;Se restaurează re-
    pop  d    ; gistrele
    lhld temp ;Se readuce valoarea
    xthl      ; originală ce se
    ret       ; schimbă cu cea a
          ;registrului HL din stivă
temp:
          ds   2
ir1       equ  0FF04h
```

La ieşirea din programul utilizator, trebuie refăcut sistemul de intreruperi astfel:

```
exitir:
    lhld temp
    shld ir1+1
    ...
```

In al doilea caz, soluţia este pină la un punct similară, in sensul că se modifică la fel vectorul spre rutina din monitor şi se execută apoi următoarea secvenţă:

```
nir:
    xthl ;Se salvează adresa de
    shld temp1; intoarcere
    lxi  h,user;Se pune in loc
    xthl      ; noua adresă
    push h
    lhld temp ;Se recuperează
    xthl      ; adresa rutinei
    ret       ; normale ce se pune in
    ...       ; stivă şi apoi se revine
    ...       ;Alte instrucţiuni
user:
    push h
    push psw;Se salvează regiştrii
    ...
    ...     ;Corpul rutinei
    ...
    pop  psw;Se restaurează
    lhld temp1;Se recuperează
    xthl      ; adresa de intoar-
    ret       ; cere şi se revine
              ; din intrerupere
temp:
          ds   2
temp1:
          ds   2
```

Deşi metodele de mai sus nu sint singurele posibile, ele asigură o flexibilă manevrare a nivelului 1 de intrerupere. Trebuie reţinut faptul că rutina din monitor comandă circuitul 8259 şi in funcţie de contextul programului utilizatorul va folosi una sau alta din metode. In primul caz, utilizatorul va trebui să aibă in vedere comandarea circuitului 8259 in cazul in care rutina respectivă trebuie să poată fi intreruptă la rindul ei de un alt nivel de prioritate mai ridicată. In al doilea caz orice nouă intrerupere va intrerupe execuţia rutinei in curs (chiar provenind din nivelul unu dacă rutina utilizatorului este suficient de lungă), datorită faptului că rutina din monitor "rearmează" atit controlorul de intreruperi 8259 (printr-un SEOI - Specific End Of Interrupt), cit şi procesorul (EI).

Nivelul 4 de intrerupere este folosit de monitor in conjuncţie cu nivelul 5 pentru a asigura funcţia de bază a citirii datelor de pe casetele magnetice. La fiecare front crescător provenit din bandă se generează o intrerupere pe nivelul 4 care execută o rutină de iniţializare a timerului 2 cu o valoare standard de polare a intrării de casetă; practic, sistemul este analog cu un monostabil de precizie obţinut prin software. In momentul in care timpul prescris s-a scurs, timerul 2 generează o intrerupere pe nivelul 5 care execută o rutină ce asigură citirea corectă a valorii bitului din intrare, făcind şi asamblarea in octeţi.

La scrierea datelor, se foloseşte doar nivelul 5 care execută o rutină ce iniţializează timerul 2 in funcţie de valoarea bitului ce trebuie transmis. Ieşirea semnalului audio este chiar ieşirea OUT2 a timerului 2.

Nivelul de intrerupere 4 nu poate fi utilizat in alte scopuri; din contră insă, nivelul 5 prezintă facilităţi deosebit de interesante pentru programatori, din care vom scoate citeva in evidenţă in cele ce urmează.

Mai devreme arătam că nu pot fi obţinute valori de temporizări foarte scurte şi exacte prin utilizarea nivelului 1 de intrerupere. Datorită faptului că nivelul 5 este generat de ieşirea OUT2 a timerului 2, se poate programa timerul cu orice valoare de timp (rezoluţia fiind chiar un tact de procesor, adică 727 nanosecunde) şi aştepta intreruperea respectivă. In acest caz insă, este foarte important ca nivelul de intrerupere 5 să fie cel mai prioritar, altfel nivelul 1 va altera valorile de timp aşteptate. Iată un exemplu:

```
iniic:              ;Iniţializări
    lxi  h,usr5ir;Adresa rutinei
    shld ir5+1    ; utilizatorului
    mvi  a,prior ;Noua prioritate
    out  intct0
    in   intct1  ;Se modifică mas-
    ani  11011111b ; ca (nivelul
    out  intct1  ; 5 activ)
    mvi  a,true  ;Iniţializează şi
    sta  test    ; semaforul
    mvi  a,vallow;Iniţializare
    out  timer2  ; timer2
    mvi  valhig
    out  timer2
wait:     ;Aici testează dacă s-a
    lxi  h,test  ; scurs timpul
loop:
    mov  a,m
    ora  a       ;Gata?
    jnz  loop    ;Nu, mai aşteaptă
    ...          ;Alte instrucţi-
    ...          ; uni
stopir:
    in   intct1  ;Repoziţionarea
    ori  100000b ; măştii iniţiale
    out  intct1
    mvi  a,oldpr ;... şi a priori-
    out  intct0  ; tăţii normale
    ...          ;Alte instrucţi-
    ...          ; uni
usr5ir:   ;Rutina ce se execută pe
    push psw     ; intrerupere
    xra  a       ;Sterge semaforul
    sta  test
    mvi  a,seoi5 ;Se reiniţiali-
    out  intct0  ; zează 8259
    pop  psw
    ret          ; şi se revine
test:
         ds   1
intct0   equ  0
intct1   equ  1
timer2   equ  12h
seoi5    equ  65h
prior    equ  0C4h
oldpr    equ  0C7h
ir5      equ  0FF14h
true     equ  0FFh
vallow   ;Valori definite de
valhig   ; utilizator funcţie de
         ; durata temporizării
```

In exemplul de mai sus nu a fost iniţializat modul de lucru al timerului 2, care se poate face fie in modul BCD zecimal, fie in modul hexazecimal (in primul mod factorul de divizare maxim este 9999, in timp ce in al doilea este 65535).

Nivelul 5 de intrerupere poate fi folosit şi in scopul declanşării execuţiei unor rutine particulare programului utilizator ce necesită un timing foarte precis (cum ar fi simularea unor USART-uri prin software).

Deşi dificil de implementat (şi mai ales greu de depanat datorită naturii asincrone faţă de programul principal), rutinele pe intreruperi oferă satisfacţii deosebite programatorului, odată puse la punct.

(CONTINUARE IN NUMARUL VIITOR)

```
0010                    pgen    equ     10h
0021                    pgen1   equ     21h
0000                    intct0  equ     0
0001                    intct1  equ     1
F800                    row1    equ     0F800h
00C3                    icw3    equ     0C3h
00C7                    icw4    equ     0C7h
FF00                    ram     equ     0FF00h
FF00                    stack   equ     ram
003B                    term    equ     ';'
003F                    cwt0    equ     3Fh
0518                    val00   equ     518h
00B1                    cwt2    equ     0B1h
00BF                    cwt20   equ     0BFh
0075                    val21l  equ     75h
0013                    val21h  equ     13h

                        ;   Monitor variables:
                        ; ******* "taboo" sequence for compatibility *******

                        ;       org     ram

FF00                    ir0     equ     0FF00h
                        ;       ds      3       ;
FF03                    kstap   equ     0FF03h
                        ;       ds      1       ;
FF04                    ir1     equ     0FF04h
                        ;       ds      3       ;
FF07                    cont    equ     0FF07h
                        ;       ds      1       ;
FF08                    ir2     equ     0FF08h
                        ;       ds      3       ;
FF0B                    cursw   equ     0FF0Bh
                        ;       ds      1       ;
FF0C                    ir3     equ     0FF0Ch
                        ;       ds      3       ;
FF0F                    count1  equ     0FF0Fh
                        ;       ds      1       ;
FF10                    ir4     equ     0FF10h
                        ;       ds      3       ;
FF13                    cschar  equ     0FF13h
                        ;       ds      1       ;
FF14                    ir5     equ     0FF14h
                        ;       ds      3       ;
FF17                    dschar  equ     0FF17h
                        ;       ds      1       ;
FF18                    ir6     equ     0FF18h
                        ;       ds      3       ;
FF1B                    kstat   equ     0FF1Bh
                        ;       ds      1       ;
FF1C                    ir7     equ     0FF1Ch
                        ;       ds      3       ;
FF1F                    chabuf  equ     0FF1Fh
                        ;       ds      1       ;
                                                ;
FF20                    rvect   equ     0FF20h  ;
                        ;       ds      3       ;
FF23                    dvect   equ     0FF23h
                        ;       ds      3       ;
                                                ;
FF26                    spsave  equ     0FF26h
                        ;       ds      2       ;
FF28                    pcsave  equ     0FF28h
                        ;       ds      2       ;
```



```
FF2A                    lsave   equ     0FF2Ah
                        ;       ds      2       ;
FF2C                    esave   equ     0FF2Ch
                        ;       ds      2       ;
FF2E                    csave   equ     0FF2Eh
                        ;       ds      2       ;
FF30                    fsave   equ     0FF30h
                        ;       ds      2       ;
                                                ;
                                                ;
FF32                    kbuff   equ     0FF32h
                        ;       ds      1       ;
FF33                    dcurs   equ     0FF33h
                        ;       ds      2       ;
FF35                    mcurs   equ     0FF35h
                        ;       ds      2       ;
FF37                    revsw   equ     0FF37h
                        ;       ds      1       ;
FF38                    crc     equ     0FF38h
                        ;       ds      2       ;
FF3A                    cstat   equ     0FF3Ah
                        ;       ds      1       ;
FF3B                    cntr    equ     0FF3Bh
                        ;       ds      1       ;
FF3C                    sreg    equ     0FF3Ch
                        ;       ds      1       ;
FF3D                    byte    equ     0FF3Dh
                        ;       ds      1       ;
                                                ;
FF3E                    badr1   equ     0FF3Eh
                        ;       ds      2       ;
FF40                    badr2   equ     0FF40h
                        ;       ds      2       ;
FF42                    biadr1  equ     0FF42h
                        ;       ds      1       ;
FF43                    biadr2  equ     0FF43h
                        ;       ds      1       ;
                                                ;
FF44                    cursw2  equ     0FF44h
                        ;       ds      1       ;
FF45                    cntbip  equ     0FF45h
                        ;       ds      1       ;
FF46                    rflag   equ     0FF46h
                        ;       ds      1       ;
FF47                    xvect   equ     0FF47h
                        ;       ds      3       ;
FF4A                    ovect   equ     0FF4Ah
                        ;       ds      3       ;
FF4D                    cbuff   equ     0FF4Dh
                        ;       ds      16      ;
FF5D                    tempwd  equ     0FF5Dh
                        ;       ds      2       ;
FF5F                    tempbd  equ     0FF5Fh
                        ;       ds      1       ;
                        ;***************
FF60                    csync   equ     0FF60h
                        ;       ds      1

FF61                    rowpag  equ     0FF61h
                        ;       ds      2       ; No. of rows in page & free rows at eop
FF63                    flag    equ     0FF63h
                        ;       ds      1       ; Paging on screen too (if 0)
FF64                    lfcntr  equ     0FF64h
                        ;       ds      1       ; Lines counter in page
```



```
FF65                    tabind  equ     0FF65h
                        ;       ds      1       ; Char's counter in line
FF66                    tabtab  equ     0FF66h
                        ;       ds      17      ; Tab's table
FF77                    flagnp  equ     0FF77h
                        ;       ds      1       ; No page number (if 0)
FF78                    pageno  equ     0FF78h
                        ;       ds      3+1     ; 3 bytes for page number & 00h

FF7C                    begtxt  equ     0FF7Ch
                        ;       ds      2       ; Header's begin
FF7E                    codbeg  equ     0FF7Eh
                        ;       ds      14      ; Assembler variables
FF80                    codmax  equ     codbeg+2
FF82                    symbeg  equ     codmax+2
FF84                    symbmx  equ     symbeg+2
FF86                    global  equ     symbmx+2

FF8C                    begin   equ     0FF8Ch
                        ;       ds      2       ; Text begin, after first marker
FF8E                    max     equ     0FF8Eh
                        ;       ds      2       ; Text superior limit
FF90                    count   equ     0FF90h
                        ;       ds      2       ; Text final location (second marker)

FF92                    bflag   equ     0FF92h
                        ;       ds      2
FF94                    rsvect  equ     0FF94h
                        ;       ds      3

                        ; ********* end of "taboo" sequence **********
FF97                    free    equ     0FF97h

                                end
```


Macros:

Symbols:

```
ASCHEX  0063    BADR1   FF3E    BADR2   FF40    BEEP    0390
BEGIN   FF8C    BEGTXT  FF7C    BEL     0007    BELL    007D
BFLAG   FF92    BIADR1  FF42    BIADR2  FF43    BLINK   0093
BRSET   0372    BS      0008    BSCMD   016C    BSDCMD  0196
BYTE    FF3D    CAN     0018    CASEND  029D    CASIN   02E1
CASOUT  0309    CBUFF   FF4D    CHABUF  FF1F    CINIT   02BE
CLDIS   019E    CLDIS1  01A1    CLS05   01B6    CLSC    01AC
CLSTA   0045    CNTBIP  FF45    CNTR    FF3B    CNVNM   0224
CODBEG  FF7E    CODMAX  FF80    CONT    FF07    CONV    021B
CORR    0245    CORR05  024E    COUNT   FF90    COUNT1  FF0F
CR      000D    CRC     FF38    CRC03   031F    CRC05   0334
CRCMD   015C    CRCRUT  031B    CRIN05  0285    CRINIT  026B
CROUT   01CB    CSAVE   FF2E    CSCHAR  FF13    CSET    0355
CSTAT   FF3A    CSYNC   FF60    CURSW   FF0B    CURSW2  FF44
CWINIT  02AD    CWRD5   02FE    CWREND  02F3    CWT0    003F
CWT1    007F    CWT2    00B1    CWT20   00BF    DCURS   FF33
DEXIT   0184    DISP    0170    DPR10   0190    DPRINT  0187
DSCHAR  FF17    DVECT   FF23    ERMSG   0040*   ERORMS  003C
ERROR   0006    ESAVE   FF2C    EXIT    0113    EXTSET  036E
FF      000C    FLAG    FF63    FLAGNP  FF77    FREE    FF97
FSAVE   FF30    GETCH   00A7    GETH05  00BC    GETHX   00B8
GETNM   0009    GLOBAL  FF86    GNM05   000B    GNM10   001F
GNM15   0027    HEXASC  0073    HEXERR  003B    HILO    00A1
HTCMD   011D    HTDCMD  0188    ICW3    00C3    ICW4    00C7
IN10    01F3    INITS   0001*   INPUT   01DB    INPUTC  0218
INTCT0  0000    INTCT1  0001    IR0     FF00    IR1     FF04
IR2     FF08    IR3     FF0C    IR4     FF10    IR5     FF14
IR6     FF18    IR7     FF1C    IRD4    0277*   IRD5    027D*
IWR5    02BC*   KBUFF   FF32    KSTAP   FF03    KSTAT   FF1B
LF      000A    LFCMD   0165    LFCNTR  FF64    LOCRT   00D3
LSAVE   FF2A    MAIN13  0007*   MAX     FF8E    MCURS   FF35
MIC     0163    MSG3    02D3*   MVSR    0253    NMOUT   0238
NULL    0046*   NWSTR1  02D8    NWSTRI  02D5    ONKEY   0340
ONKY05  034E    OSET    0380    OSET1   038C    OUT05   01D3*
OUTPUT  01CD    OVECT   FF4A    PAGENO  FF78    PAGRUT  01D9*
PCSAVE  FF28    PGEN    0010    PGEN1   0021    PR10    011E
PR20    0131    PR30    0141    PR35    014C    PR40    0156
PRINT   0118    PSET    037D    RAM     FF00    RETURN  0003
REVON   010D    REVSW   FF37    RFLAG   FF46    ROW1    F800
ROWPAG  FF61    RSTART  0038    RSVECT  FF94    RVECT   FF20
SB2     0031    SERDAT  0030    SERDR5  0088    SERDRV  0087
SERSTA  0031    SI      000F    SO      000E    SPOUT   01C6
SPSAVE  FF26    SREG    FF3C    STACK   FF00    START   0000
STR05   004C    STR10   0057    STR15   005A    STRIST  0048
SUCCES  0004*   SYMBEG  FF82    SYMBMX  FF84    TABIND  FF65
TABTAB  FF66    TEMPBD  FF5F    TEMPWD  FF5D    TERM    003B
TIMER0  0010    TIMER1  0011    TIMER2  0012    TIMSTA  0013
TWOSET  0263    UAR     001E    UIC     01C5    VAL00   0518
VAL21H  0013    VAL21L  0075    VTCMD   0148    WAITMS  02D2
WDCAS   0304    WDOUT   0233    WDSTA   025E    XVECT   FF47
```

No Fatal error(s)

MICROCALCULATORUL L/B 881

NICOARA PAULIAN
LIVIU IONESCU
ION RUSOVICI
GHEORGHE CHITA

Monitorul foloseşte exclusiv o zonă din memoria RAM situată in ultima pagină (C000-FFFF). In această zonă, utilizatorul are la dispoziţie spaţiul dintre adresele C000-F7FF. Restul este folosit de sistem şi monitor astfel:

- F800-FE7F memoria ecranului;
- FE80-FF00 128 octeţi pentru stivă;
- FF00-FFFF tabela de salturi pentru intreruperi, buffere şi variabile ale monitorului.

Tot in această zonă se găsesc şi locaţiile ceasului de timp real. Ele pot fi citite sau modificate prin software şi se găsesc la următoarele adrese (in format ASCII):

- F878 zeci ore;
- F879 unităţi ore;
- F87B zeci minute;
- F87C unităţi minute;
- F87E zeci secunde;
- F87F unităţi secunde.

Pentru activarea cursoarelor in zonele denumite STATUS şi MAIN (primele 2 două rinduri şi respectiv celelalte 24 ale ecranului), se incarcă locaţia CURSW (adr. FF0B) după cum urmează:

00 = ambele cursoare stinse;
01 = STATUS aprins, MAIN stins;
02 = STATUS stins, MAIN aprins;
03 = ambele cursoare aprinse.

De remarcat ca orice lansare de program cu comanda G va activa cursorul pe MAIN.

Claviatura are citeva coduri cu semnificaţie aparte:

- Caps: pentru setarea şi resetarea pe litere mari se foloseşte codul CTRL \.
- Autorepeat: pentru setarea şi resetarea repetării automate a unui caracter se foloseşte codul CTRL].
- Home: pentru ştergerea ecranului şi poziţionarea cursorului in stinga sus se foloseşte codul CTRL L.
- Videorevers: pentru afişarea pe CRT in video inversat se utilizează codul CTRL N, iar pentru revenire la normal, codul CTRL O.

In monitor, deşi USART-ul nu este utilizat direct, el este iniţializat cu 2 biţi de stop, unul de start, 8 biţi de date transmişi fără paritate şi tactul de Bd.rate * 16. Dacă intr-un program de utilizator este necesară o altă iniţializare pentru USART, atunci acesta trebuie resetat software cu secvenţa:

```
        mvi     a,40h
        out     sersta
sersta  equ     31h     ; SERSTA este
        ; adresa registrului de stare
```

după care se poate face noua iniţializare.

Interfaţa paralela PPI 1 este utilizată pentru intrarea de claviatură. Totuşi, portul C este practic liber pentru utilizator şi este programat de către monitor la iniţializare ca PC0-PC3 input şi PC4-PC7 output. In afară de PC0 care este folosit ca intrare pentru interfaţa de casetă, toate celelalte sint libere. In cazul in care utilizatorul va folosi oricare din aceste porturi, este esenţial ca iniţializarea lui PPI 1 să rămină neschimbată (pentru buna funcţionare a claviaturii).

Asignarea timerelor (din cadrul LSI 8253) este următoarea:

TIMER0 - Bell
TIMER1 - USART Bd.rate
TIMER2 - Timing caseta

Toate timerele pot fi folosite de către utilizator fără nici un fel de restricţie specială.

Utilizarea interrupt controllerului este ceva mai delicată (vezi in numărul trecut). In principiu se poate reconfigura in orice poziţie de prioritate avind insa in vedere ca nivelul 1 să nu fie mascat, deoarece pe el funcţionează claviatura, cursorul, ceasul şi bell-ul.

Pe pagina alăturată este dat sumarul comenzilor şi subrutinelor monitorului.

BIBLIOTECA DE PROGRAME A MICROCALCULATORULUI L/B881

Un calculator "trăieşte" prin baza software pe care o posedă: dacă aceasta este redusă sau inexistentă, practic acel calculator este complet inutil. Pină aici am prezentat modul de realizare hardware a calculatorului şi monitorul minimal (sau "firmware"-ul cum i se mai spune). Am insistat asupra acestuia din urmă datorită importanţei pe care o are in dezvoltarea de programe de aplicaţie.

In cele ce urmează vom face o scurtă trecere in revistă a principalelor programe care au fost scrise sau adaptate pe L/B881.

Sistemul de operare 881/Sys

881/Sys este un superset al monitorului 881/Mon, care include in componenţa sa, pe lingă un monitor extins, compatibil cu 881/Mon, un editor de texte şi un asamblor pentru mnemonicele lui 8080. El este o necesitate pentru cei ce dezvoltă programe in limbajul de asamblare. Editorul de texte, orientat pe ecran, permite inserarea/ştergerea de caractere singulare, rinduri intregi sau zone de text. Conţine şi macrofacilităţi de genul "find string" şi "find and substitute" (căutări de şiruri şi inlocuiri), mutări şi copieri de zone de text. Permite o editare comodă a textelor tip document sau a programelor sursă.

Asamblorul, parte integrantă a lui 881/Sys, permite utilizarea de etichete mnemonice, expresii complexe, operanzi ASCII, hex sau zecimali. Poate fi folosit cu tabele de simboli externi pentru asamblarea unor programe sursă foarte lungi (compuse din mai multe bucăţi).

881/Sys există numai in versiune de ROM, in locul monitorului 881/Mon şi ocupă 8 Kocteţi.

DDT

DDT (Dynamic Debugging Tool = Instrument pentru depanare dinamică) este un program destinat testării şi depanării altor programe. DDT permite trasarea execuţiei unui program, dezasamblarea lui, afişarea pe display a unei zone de memorie sub formă de caractere tipăribile şi modificarea ei intr-un mod analog cu cel al comenzii "m" a monitorului. (De altfel, DDT funcţionează ca o extensie a acestuia).

DDT este destinat in special depanării programelor utilizator pentru care există text sursă in memorie editat cu ajutorul editorului rezident. El foloseşte parametrii acestor programe (adresele de inceput şi de sfirşit ale codului obiect generat la asamblare şi adresa tabelei de simboluri), preluaţi din textul sursă, pentru trasarea sau dezasamblarea programelor. La trasare şi dezasamblare adresele care există in tabela de simboluri sint inlocuite cu simbolul corespunzător, uşurind astfel inţelegerea listingurilor generate.

DDT poate fi folosit şi pentru trasarea sau dezasamblarea unor programe pentru care nu există text sursă in memorie. Listele rezultate de la trasare sau dezasamblare pot fi scoase şi pe un dispozitiv periferic auxiliar (de exemplu, o imprimantă).

Programul există atit in versiune de ROM cit şi de casetă.

881/BASIC

Limbajul BASIC se bucură de o largă răspindire printre utilizatorii de microcalculatoare ca urmare a simplităţii sale, a modului interactiv de lucru şi a puternicelor sale facilităţi aritmetice. 881/Basic este un interpretor scris special pentru acest microcalculator, folosindu-i din plin resursele.

Pe lingă funcţiile standard ale Basic-ului (ce includ şi puternice facilităţi aritmetice, trigonometrice şi logice), 881/Basic conţine şi numeroase funcţii grafice (PSET, PRESET, LINE, CIRCLE, DRAW) şi sonore (PLAY); de semnalat posibilitatea de lucru multitasking (PLAY in background şi/sau cu instrucţiunea ON TIMER).

Deosebit este şi editorul "full-screen" care ajută enorm la introducerea in memorie şi depanarea programelor; editorul este compatibil cu comenzile editorului din 881/Sys.

881/Basic este compatibil cu Microsoft Basic, asigurind astfel portabilitatea programelor Basic scrise pe Commodore 64, Apple II, TRS-80, şi familia M18/118-Felix CUB (MBASIC sau GW-BASIC sub CP/M). Programul are aprox. 15 Kocteţi şi există doar in versiune de casetă.

CIP

CIP (Cassette Interchange Program) este un program utilitar de casetă. Asigură copierea oricărui tip de fişier in scopul creerii de back-up-uri. Permite protejarea şi/sau atribuirea statutului de autolansare a fişerelor scrise pe casete magnetice. Utilizarea sa este extrem de simplă şi comodă.

CommPack

CommPack este un program destinat radioamatorilor pasionaţi de comunicaţiile de date prin eter. Programul poate transmite şi recepţiona in următoarele moduri de lucru:

* Morse: recepţie automată intre 40 şi 350 semne/minut;
  emisie intre 40 şi 350 semne/minut in trepte de cite 10 s/m;
* Baudot: emisie/recepţie pe vitezele de 45,45; 50; 75; 100; 110 bauds;
* ASCII: emisie/recepţie pe vitezele de 45,45; 50; 75; 100; 110 bauds folosind paritate pară, impară, sau fără paritate (7 biţi sau 8 biţi ASCII).

Facilităţile pe care le include sint foarte numeroase: buffer FIFO pentru claviatură, buffer pentru recepţie, 10 memorii de lungime variabilă, posibilitatea de inversare a semnalului intrare/ieşire, split-screen, sistem sofisticat de lucru cu memoriile (folosind o stivă internă), utilizarea de variabile in memorii prin care se pot defini numele, RST-ul sau indicativul staţiilor corespondente, posibilitatea asignării unui header de inceput de transmisiune şi multe altele.

CommPack are aproximativ 15 Kocteţi şi se lansează de la adresa 8000 hex de pe casetă.

Morse Tutor

Este un program destinat celor ce doresc să inveţe sau să se antreneze in recepţia codului Morse; practic, este un "profesor de telegrafie" ce transmite foarte corect, la viteză cerută, mesajele existente in memoria calculatorului. Viteza poate fi cuprinsă intre 10 şi 400 semne pe minut (standard Paris). Mesajele se generează cu ajutorul editorului din 881/Sys, programul funcţionind ca o extensie a acestuia.

Jocuri

Ca pentru orice calculator, şi pentru L/B881 au fost scrise sau adaptate numeroase jocuri de diverse tipuri (inteligenţă artificială sau gen "arcade"). De semnalat dintre acestea programul lui Viorel Darie de jucat şah ATOM 64 şi cel al lui Dan Teodosiu OTHELLO (ce simulează jocul "Reversi"); din a doua categorie fac parte Penetrator, Frog, Rally, Invaders ş.a.

L/B881 a apărut şi s-a dezvoltat iniţial ca urmare a entuziasmului citorva pasionaţi ai informaticii; ulterior li s-au adăugat din ce in ce mai mulţi care au avut fiecare in parte o contribuţie mai mică sau mai mare la imbogăţirea bibliotecii software, ceea ce a dus la lărgirea domeniului de aplicabilitate a microcalculatorului.

Au fost trecute aici in revistă doar o parte din programele de a căror existenţă am aflat; pe lingă autorii acestor rinduri, se cuvin menţionate şi numele lui Sandru Nichita şi Ovidiu Băloiu care au contribuit la imbogăţirea bibliotecii software a lui L/B881; probabil insă că sint mulţi alţii printre cei ce şi-au construit acest microcalculator ce au realizări remarcabile in domeniul programării, şi pe care ii invităm să le publice in paginile revistei.

*** Citire din linia DISPLAY.

00A7 getch citeşte in A de la adresa din DE, ignoră spaţii, converteşte la litere mari. CY=1 la sfirşitul liniei, altfel DE=adresa urmatorului caracter.
  Distruge AF,DE.

00B8 gethx citeşte in HL un parametru hex de la adresa din DE şi iese cu separatorul in A şi adresa de după acesta in DE. CY=1 hex error.
  Distruge totul.

0009 getnm citeşte in stivă 1-8 parametri hex funcţie de B, de la adresa din DE, luind ca nuli parametrii lipsă. CY=1 hex error sau prea mulţi parametri.
  Distruge totul.

*** Scriere in ASCII in memorie (eventual pe ecran).

0263 twoset scrie A in ASCII (2 caractere) la adresa din HL.
  Distruge AF,BC,HL.

025E wdsta scrie DE in ASCII (4 caractere) la adresa din HL.
  Distruge AF,BC,HL.

*** Subrutine de uz general.

00A1 hilo compară DE şi HL. CY=1 DE > HL
                             CY=0 DE <= HL
                             Z=1   DE = HL
  Distruge AF.

0031 sb2 efectuează HL=HL-DE.
  Distruge AF,HL.

0253 mvsr transferă zona de la (DE,HL) la zona BC.
  Distruge AF,BC,DE.

0340 onkey face saltul la o tabelă de adrese din BC prin cheia din H. Dacă nu există cheia, revine.
  Distruge AF,BC,HL.

0087 serdrv scrie caracterul din A la portul serie.

*** Control al traductorului acustic.

007D bell trimite un semnal de 2,6 KHz.
  Distruge AF.

0390 beep trimite un semnal cu lungimea multiplu de 20 mS dată in A şi f=1375 KHz/(valoarea din BC) in BCD. Nu aşteaptă să se termine operaţia.

*** Conversii.

021B conv converteşte caracterul din A in litere mari.
  Distruge AF.

0224 cnvnm transformă A in ASCII (2 caractere) in BC.
  Distruge AF,BC.

0063 aschex converteşte A intr-o cifră hex. CY=1 hex error.
  Distruge AF.

0073 hexasc converteşte cifra hex din A in caracter ASCII.
  Distruge AF.

*** Controlul vectorilor de sistem.

0355 cset setează ceasul la valoarea BCD din HL.
  Distruge AF,BC,HL.

036E extset setează XVECT la adresa din HL.

0372 brset setează factorul de divizare pentru Baud-rate la valoarea BCD din HL.
  Distruge A.

0380 oset setează vectorul OVECT la HL, sau il resetează.
  Distruge AF.

037D pset setează vectorul OVECT la driver-ul serie.
  Distruge AF.

*** Control caseta audio.

026B crinit iniţializează citirea şi asteaptă SYNC. La CTRL-X de la claviatură, iese cu CY=1.
  Distruge AF,BC,HL.

02AD cwinit iniţializează scrierea.
  Distruge AF,BC,HL.

029D casend dezactivează caseta.
  Distruge AF.

02E1 casin citeşte in A şi shiftează prin CRC.
  Distruge AF.

0309 casout scrie A şi shiftează prin CRC.
  Distruge AF.

0304 wdcas scrie H şi L shiftind cei doi octeţi prin CRC.
  Distruge AF.

02F3 cwrend scrie CRC-ul pe bandă.
  Distruge AF,BC,HL.

031B crcrut shiftează A prin CRC.
  Distruge AF.

SUMAR AL COMENZILOR MONITORULUI GRUPATE PE FUNCTII

*** Comenzi de transfer cu caseta audio.

L titlu(,adr) caută şi incarcă fişierul cu titlul specificat la adresa indicată sau la adresa de unde a fost salvat.

S titlu,adr1,adr2 salvează zona de memorie intr-un fişier cu titlul specificat.

V verifică dacă primul fişier intilnit este corect.

*** Comenzi de lucru cu memoria şi registrele.

M adr afişează pe CRT o pagină de memorie şi aşteaptă corecţii in aceasta.

M adr1,adr2 videază o zonă de memorie atit pe CRT cit şi pe periferiCul conectat prin OVECT.

M adr1,adr2,adr3 transferă zona de memorie dintre primele adrese la zona care incepe cu adr3.

R afişează registrele şi aşteaptă corecţii.

*** Comanda de lansare in execuţie.

G adr1(,adr2(,adr3)) transferă controlul programului de la adr1 sau continuă programul intrerupt anterior dacă adr1 nu apare sau este 0. Celelalte adrese sint eventuale adrese de Breakpoint.

*** Comenzi de control.

FC hhmm setează ceasul la valoarea dată.

FB nnnn setează coeficientul de divizare pentru Baud-rate-ul USART-ului.

FX adr setează vectorul de extensie monitor.

FO adr setează/resetează vectorul suplimentar de ieşire OVECT.

FP setează vectorul OVECT la driver-ul serie USART.

*** Caractere speciale de control.

ESC (CTRL [, 1Bh) generează o intrerupere in program.

FS (CTRL \, 1Ch) bistabil soft pentru CAPS.

GS (CTRL], 1Dh) bistabil soft pentru autorepeat.

SUMAR AL SUBRUTINELOR GRUPATE PE FAMILII

(Pentru detalii se va consulta textul sursă)

*** Citire de la claviatură.

01DB input citeşte in A un caracter.
  Distruge AF.

0218 inputc citeşte in A şi converteşte la litere mari.
  Distruge AF.

*** Scriere in zona MAIN a ecranului.

01CD output scrie A pe ecran şi la OVECT, la CR face şi LF.
  Distruge F (şi A la CR).

01C6 spout scrie un spaţiu pe ecran şi la OVECT.
  Distruge AF.

01CB crout scrie CR, LF pe ecran şi la OVECT.
  Distruge AF.

00D3 locrt scrie A numai pe ecran.

FF4A ovect scrie A numai la OVECT.

01AC clsc sterge zona MAIN şi pune cursorul in stinga sus.

0245 corr mută cursorul la adresa din BC pentru CY=1 sau adună la poziţia sa valoarea din BC la CY=0.
  Distruge AF,HL.

0238 nmout scrie A in ASCII (2 caractere) pe ecran şi OVECT.
  Distruge AF.

0233 wdout scrie DE in ASCII (4 caractere) pe ecran şi OVECT.
  Distruge AF.

*** Scriere in linia DISPLAY a ecranului.

0170 disp scrie A in linia DISPLAY.

019E cldis şterge linia DISPLAY.
  Distruge F,DE,HL.

*** Scriere in linia STATUS a ecranului.

0048 strist scrie textul de la HL şi şterge restul liniei.
  Distruge AF,HL.

02D5 nwstri scrie textul de la HL in a doua parte a liniei.
  Distruge AF,BC,HL.

0045 clsta şterge linia STATUS.
  Distruge AF,HL.

003C erorms afişează "Error" pe STATUS şi face Bell.
  Distruge AF,HL.

02D2 waitms afişează "System busy" pe STATUS, la mijloc.
  Distruge AF,BC,HL.

INFORMATICĂ

DRIVER GRAFIC

CĂTĂLIN BRATU, YO4QF

Autorul programului prezentat în coloanele revistei noastre, CĂTĂLIN BRATU, este elev în clasa a X-a la Liceul „Mircea cel Bătrîn" din Constanţa. Rezultatele foarte bune la învăţătură, performanţele obţinute pe linia creaţiei tehnico-ştiinţifice cu realizări din domeniul informaticii — două titluri naţionale la campionatul republican organizat de Federaţia Română de Radioamatorism — îl recomandă pe tînărul autor şi vă invită, stimaţi cititori, la cunoaşterea unei interesante realizări.

Programul următor realizează copierea ecranului pe orice imprimantă compatibilă EPSON (ex. ROBOTRON K6313, K6314 ş.a.).

Codul obiect poate fi rulat pe un microcalculator L/B881, şi este autorelocabil la orice adresă, după cum vom vedea în continuare. Programul a fost scris în limbajul C şi compilat cu compilatorul Aztec C folosind un L/B881 prevăzut cu unitate de disc flexibil rulînd sub sistemul de operare CP/M.

Codul listat mai jos se va introduce în memorie de la adresa 4000, după care se va verifica corectitudinea introducerii. După ce totul este în regulă, se lanseaza programul cu comanda g4000<CR>. Dacă codul a fost introdus corect, pe linia STATUS va apărea mesajul: "Introduceţi adresa de relocare:". Se va introduce adresa de rulare a programului, alegerea fiind dictată, evident, de existenţa altor programe în memorie în acel moment. De exemplu, dacă doriţi ca acesta să ruleze de la adresa 8000, veţi introduce "8000<CR>". Din acest moment este activată comanda <CTRL/P> care va copia literalmente conţinutul ecranului (inclusiv caracterele grafice) pe coala imprimantei. De reţinut că programul este activat prin testarea anumitor condiţii generate de către nivelul 1 de întreruperi şi astfel poate fi invocat indiferent de programul rulat în microcalculator. Cît timp imprimanta va copia ecranul, sistemul va fi blocat.

In continuare sînt date atît listingul sursă al modulului principal scris în C, cît şi codul obiect; acesta din urmă va fi introdus în microcalculator folosind comenzile monitorului rezident. Programul poate fi apoi salvat pe casetă pentru utilizare ulterioară.

```
/*      Print Screen function
 *      (C) 1987 MicroLines Software
 */

#define SERDRV  0x87

extern char chgen[];
char nl[] = { 27, 'J', 9, 0xD, 0 }; /* string for newline */
char initg[] = { 27, '*', 4, 128, 1, 0 }; /* string to init graphics */
char *ram;
int byte, x, i, row;

print()
{
 int y = 26;
 char *point = (char *)0xF800;
 register char *p;

 ram = chgen;
 oset(SERDRV);  /* switch printer on serial port */
 brset(0x18);   /* set baud rate at 4800 */

 while (y--) {
        for (row=0; row<3; row++) {
                x=64;
                ostr("          ");     /* paper border */
                ostr(initg);    /* switch printer to graphics mode */
                while (x--) {
                        byte = *(point + 63 - x);
                        /* get current byte from screen */
                        for (i=0, p=ram+18*(byte & 0x7F)+6*row; i<6; i++) {
                                (byte & 0x80) ? ovect((*p++) ^ 7) : ovect(*p++);
                                }
                        }
                ostr(nl);       /* write newline */
                }
                point += 64;
        }
 oset(0);       /* switch printer off */
}


ostr(xx)        /* output a string to printer */
 register char *xx;
{
 while(*xx)
        ovect(*xx++);
```

```
4000 3E 01 32 0B FF CD 9E 01 21 0D 41 CD 4B 00 CD DB
4010 01 FE 0D CA 1C 40 CD 70 01 C3 0E 40 06 01 CD 09
4020 00 DA 05 40 C1 3E 02 32 0B FF CD 45 00 CD 9E 01
4030 21 6D 40 3E 50 B7 CA 53 40 5E 23 56 23 C5 EB 01
4040 2D 41 09 C1 F5 2B 7E 81 77 23 7E 88 77 F1 EB 3D
4050 C3 35 40 11 2D 41 21 F3 0B E5 09 44 4D E1 19 7E
4060 02 CD A1 00 0B 2B C2 5F 40 60 69 23 E9 05 00 0A
4070 00 0D 00 14 00 17 00 1B 00 1E 00 27 00 33 00 3B
4080 00 4A 00 4D 00 50 00 53 00 57 00 D6 09 F1 09 F4
4090 09 FB 09 03 0A 1A 0A 20 0A 23 0A 26 0A 2A 0A 2E
40A0 0A 35 0A 38 0A 3E 0A 41 0A 45 0A 49 0A 4D 0A 51
40B0 0A 55 0A 5B 0A 6A 0A 6D 0A 74 0A 7A 0A 7D 0A 83
40C0 0A 87 0A 8D 0A 93 0A 99 0A 9F 0A A2 0A A6 0A AA
40D0 0A B1 0A B4 0A B7 0A BD 0A C0 0A CF 0A D3 0A D7
40E0 0A E4 0A E8 0A EB 0A EE 0A F2 0A F6 0A 0C 0B 13
40F0 0B 23 0B 33 0B 40 0B 44 0B 79 0B 87 0B 8A 0B 8F
4100 0B 94 0B 9E 0B A3 0B AE 0B D9 0B DE 0B 49 6E 74
4110 72 6F 64 75 63 65 74 69 20 61 64 72 65 73 61 20
4120 64 65 20 72 65 6C 6F 63 61 72 65 3A 00 2A 05 FF
4130 22 6A 00 3E C3 32 69 00 21 12 00 22 05 FF FF 22
4140 6C 00 21 1F 00 E5 2A 6C 00 C3 69 00 E5 D5 C5 F5
4150 DB 01 32 70 00 3E FF D3 01 3A 1B FF E6 20 CA 55
4160 00 3A 32 FF FE 10 C2 55 00 3A 1B FF E6 DF 32 1B
4170 FF 21 00 00 39 22 6E 00 31 D4 00 CD D4 09 2A 6E
4180 00 F9 3A 70 00 D3 01 F1 C1 D1 E1 C9 D1 E1 E5 D5
4190 7D CD 4A FF B7 C9 00 00 00 00 00 00 00 00 00 00
41A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
41B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
41C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
41D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
41E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
41F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4210 00 00 00 07 07 07 00 00 00 00 00 00 00 00 00 00
4220 00 00 00 00 00 00 00 00 07 07 07 00 00 00 00 00
4230 00 00 00 00 00 00 00 07 07 07 07 07 07 00 00 00
4240 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07
4250 07 07 00 00 00 00 00 00 00 00 00 00 07 07 07 00 00
4260 00 07 07 07 00 00 00 00 00 00 00 00 00 00 00 00 00
4270 07 07 07 07 07 07 00 00 00 00 00 00 00 00 00 07
4280 07 07 07 07 07 07 07 07 00 00 00 00 00 00 00 00 00
4290 00 00 00 00 00 00 00 00 00 00 00 07 07 07 00 00 00
42A0 00 00 00 07 07 07 00 00 00 00 00 00 00 07 07 07 00
42B0 00 00 00 00 00 00 00 00 00 07 07 07 00 00 00 07 07
42C0 07 00 00 00 00 00 00 00 07 07 07 07 07 07 00 00 00
42D0 07 07 07 00 00 00 00 00 00 00 00 00 00 00 00 00 07
42E0 07 07 07 07 07 00 00 00 00 00 00 07 07 07 00 00
42F0 00 07 07 07 07 07 07 00 00 00 00 00 00 00 00 00 00
4300 07 07 07 07 07 07 07 07 07 07 00 00 00 00 00 00 07
4310 07 07 07 07 07 07 07 07 07 07 07 07 00 00 00 00 00
4320 00 00 00 00 00 00 00 00 00 00 00 00 00 00 07 07 07
4330 00 00 00 07 07 07 00 00 00 00 00 00 00 00 00 00 07
4340 07 07 00 00 00 00 00 00 00 07 07 07 00 00 00 00 00
4350 00 07 07 07 00 00 00 07 07 07 07 07 07 00 00 00
```

```
4360 00 00 00 07 07 07 00 00 00 00 00 00 00 00 00 07
4370 07 07 00 00 00 07 07 07 00 00 00 07 07 07 00 00
4380 00 07 07 07 00 00 00 07 07 07 00 00 00 00 00 00
4390 07 07 07 07 07 07 00 00 00 07 07 07 00 00 00 07
43A0 07 07 07 07 07 07 07 07 00 00 00 07 07 07 00 00
43B0 00 00 00 00 00 00 00 00 00 00 00 07 07 07 07 07
43C0 00 00 00 07 07 07 00 00 00 00 00 00 07 07 07 07
43D0 07 07 00 00 00 00 00 00 07 07 07 00 00 00 07 07
43E0 07 07 07 07 00 00 00 07 07 07 07 07 07 00 00 00
43F0 07 07 07 07 07 07 00 00 00 00 00 00 00 00 00 07
4400 07 07 07 07 07 07 07 07 00 00 00 07 07 07 00 00
4410 00 07 07 07 07 07 07 07 07 07 00 00 00 00 00 00
4420 07 07 07 07 07 07 07 07 07 07 07 07 00 00 00 07
4430 07 07 07 07 07 07 07 07 07 07 07 07 07 07 00 00
4440 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4450 00 00 00 00 00 00 03 00 00 00 00 00 07 00 00 00
4460 00 00 02 00 00 00 00 03 00 03 00 00 00 04 00 04
4470 00 00 00 00 00 00 00 00 00 03 00 03 00 00 05 07
4480 05 07 05 00 00 06 00 06 00 00 00 01 03 01 01 00
4490 04 02 07 02 01 00 04 04 06 04 00 00 03 03 00 00
44A0 01 00 00 01 02 04 00 00 04 00 00 06 06 00 01 02
44B0 01 00 00 00 05 02 05 00 01 00 04 02 02 04 02 00
44C0 00 00 01 02 00 00 00 04 00 00 00 00 00 00 00 00
44D0 00 00 00 01 02 00 00 00 07 00 00 00 00 00 00 04
44E0 02 00 00 00 00 00 02 01 00 00 00 00 00 07 00
44F0 00 00 02 04 00 00 00 00 01 00 00 00 05 02 07 02
4500 05 00 00 00 04 00 00 00 00 00 01 00 00 00 02 02
4510 07 02 02 00 00 00 04 00 00 00 00 00 00 00 00 00
4520 00 00 00 00 00 00 00 01 06 00 00 00 00 00 00 00
4530 00 00 02 02 02 02 02 00 00 00 00 00 00 00 00 00
4540 00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00
4550 00 00 00 00 01 00 00 01 02 04 00 00 04 00 00 00
4560 00 00 01 02 02 02 01 00 07 01 02 04 07 00 04 02
4570 02 02 04 00 00 01 03 00 00 00 00 00 07 00 00 00
4580 00 02 06 02 00 00 01 02 02 02 01 00 00 00 01 02
4590 04 00 02 06 02 02 02 00 01 02 02 02 01 00 00 00
45A0 02 02 05 00 04 02 02 02 04 00 00 00 01 03 00 00
45B0 03 05 01 07 01 00 00 00 00 06 00 00 03 02 02 02
45C0 02 00 04 04 04 04 03 00 04 02 02 02 04 00 00 00
45D0 01 02 00 00 01 06 02 02 01 00 04 02 02 02 04 00
45E0 02 02 02 02 03 00 00 00 01 02 04 00 02 04 00 00
45F0 00 00 01 02 02 02 01 00 05 02 02 02 05 00 04 02
4600 02 02 04 00 01 02 02 02 01 00 04 02 02 03 04 00
4610 00 02 04 00 00 00 00 00 00 00 00 00 00 00 04 00
4620 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00
4630 04 00 00 00 00 01 06 00 00 00 00 00 01 02 00 00
4640 02 05 00 00 00 00 00 00 04 02 00 00 00 00 00 00
4650 00 00 05 05 05 05 05 00 00 00 00 00 00 00 00 02
4660 01 00 00 00 00 00 00 05 02 00 00 02 04 00 00 00
4670 01 02 02 02 01 00 00 00 03 04 00 00 00 00 02 00
4680 00 00 01 02 02 02 01 00 07 00 07 03 06 00 04 02
4690 02 02 02 00 00 01 02 01 00 00 07 01 01 01 07 00
46A0 06 00 00 00 06 00 02 03 02 02 01 00 00 07 02 02
```

```
46B0 05 00 02 06 02 02 04 00 01 02 02 02 01 00 07 00
46C0 00 00 00 00 04 02 02 02 04 00 02 03 02 02 01 00
46D0 00 07 00 00 07 00 02 06 02 02 04 00 03 02 02 02
46E0 02 00 07 02 02 02 00 00 06 02 02 02 02 00 03 02
46F0 02 02 02 00 07 02 02 02 00 00 06 00 00 00 00 00
4700 01 02 02 02 02 00 07 00 00 01 01 00 04 02 02 02
4710 06 00 03 00 00 00 03 00 07 02 02 02 07 00 06 00
4720 00 00 06 00 00 02 03 02 00 00 00 00 00 07 00 00 00
4730 00 02 06 02 00 00 00 00 00 00 00 03 00 00 00 00 00
4740 07 00 04 02 02 02 04 00 03 00 00 01 02 00 07 02
4750 05 00 00 00 06 00 00 04 02 00 03 00 00 00 00 00
4760 07 00 00 00 00 00 06 02 02 02 02 00 03 01 00 01
4770 03 00 07 00 06 00 07 00 06 00 00 00 06 00 03 00
4780 00 00 03 00 07 04 02 01 07 00 06 00 00 00 06 00
4790 01 02 02 02 01 00 07 00 00 00 07 00 04 02 02 02
47A0 04 00 03 02 02 02 01 00 07 02 02 02 04 00 06 00
47B0 00 00 00 00 01 02 02 02 01 00 07 00 01 00 07 00
47C0 04 02 02 04 02 00 03 02 02 02 01 00 07 02 03 02
47D0 04 00 06 00 00 04 02 00 01 02 02 02 01 00 04 02
47E0 02 02 01 00 04 02 02 02 04 00 02 02 03 02 02 00
47F0 00 00 07 00 00 00 00 00 06 00 00 00 03 00 00 00
4800 03 00 07 00 00 00 07 00 04 02 02 02 04 00 03 00
4810 00 00 03 00 04 03 00 03 04 00 00 00 06 00 00 00
4820 03 00 00 00 03 00 07 00 03 00 07 00 06 04 00 04
4830 06 00 03 00 00 00 03 00 00 05 02 05 00 00 06 00
4840 00 00 06 00 03 00 00 00 03 00 00 04 03 04 00 00
4850 00 00 06 00 00 00 02 02 02 02 03 00 00 01 02 04
4860 00 00 06 02 02 02 02 00 03 03 02 02 02 00 07 07
4870 00 00 00 00 06 06 02 02 02 00 01 00 00 00 00 00
4880 00 04 02 01 00 00 00 00 00 00 04 00 02 02 02 03
4890 03 00 00 00 00 07 07 00 02 02 02 06 06 00 00 00
48A0 00 00 00 00 01 02 04 02 01 00 00 00 00 00 00 00
48B0 00 00 00 00 00 00 00 00 00 00 00 00 00 02 02 02 02
48C0 02 00 00 02 01 00 00 00 00 00 00 00 04 00 00 00 00
48D0 00 00 00 00 00 00 00 00 00 00 00 00 00 05 05 05 03 00
48E0 04 02 02 02 06 00 03 00 00 00 00 00 07 04 04 04
48F0 03 00 06 02 02 02 04 00 00 00 00 00 00 00 03 04
4900 04 04 00 00 04 02 02 02 00 00 00 00 00 00 03 00
4910 03 04 04 04 07 00 04 02 02 02 06 00 00 00 00 00
4920 00 00 03 05 05 05 03 00 04 02 02 02 00 00 00 00
4930 01 02 00 00 00 02 07 02 00 00 00 00 06 00 00 00
4940 00 00 00 00 00 00 00 03 04 04 04 07 00 00 04 05 05
4950 06 00 03 00 00 00 00 00 07 04 04 04 03 00 06 00
4960 00 00 06 00 00 00 02 00 00 00 00 00 04 07 00 00 00
4970 00 02 06 02 00 00 00 00 00 00 02 00 00 00 00 04 07
4980 00 00 02 01 01 06 00 00 03 00 00 00 00 00 07 01
4990 02 04 00 00 06 00 04 02 00 00 00 02 03 00 00 00
49A0 00 00 07 00 00 00 00 02 06 02 00 00 00 00 00 00
49B0 00 00 07 04 03 04 03 00 06 00 06 00 06 00 00 00
49C0 00 00 00 00 07 04 04 04 03 00 06 00 00 00 06 00
49D0 00 00 00 00 00 00 03 04 04 04 03 00 04 02 02 02
49E0 04 00 00 00 00 00 00 00 00 07 04 04 04 03 00 07 04
49F0 04 04 00 00 00 00 00 00 00 00 00 03 04 04 04 07 00
```

```
4A00 00 04 04 04 07 00 00 00 00 00 00 00 07 02 04 04
4A10 00 00 06 00 00 00 00 00 00 00 00 00 00 00 02 05
4A20 05 05 00 00 02 02 02 02 04 00 00 00 03 00 00 00
4A30 00 04 07 04 00 00 00 00 04 02 00 00 00 00 00 00
4A40 00 00 07 00 00 00 07 00 04 02 02 02 04 00 00 00
4A50 00 00 00 00 07 00 00 00 07 00 00 04 02 04 00 00
4A60 00 00 00 00 00 00 07 00 01 00 07 00 04 02 04 02
4A70 04 00 00 00 00 00 00 00 04 02 01 02 04 00 02 04
4A80 00 04 02 00 00 00 00 00 00 00 07 00 00 00 07 00
4A90 00 04 05 05 06 00 00 00 00 00 00 00 04 04 05 06
4AA0 04 00 02 06 02 02 02 00 00 00 01 02 00 00 00 02
4AB0 05 00 00 00 00 00 04 02 00 00 00 00 03 00 00 00
4AC0 00 00 05 00 00 00 00 00 06 00 00 00 00 02 01 00
4AD0 00 00 00 00 05 02 00 00 00 02 04 00 00 00 00 00
4AE0 00 00 00 00 01 02 02 02 04 00 00 00 00 00 00 02
4AF0 05 02 05 02 05 05 02 05 02 05 02 02 05 02 05 02
```

```
4B00 05 CD 65 0B FC FF 21 1A 00 EB 21 06 00 39 73 23
4B10 72 21 00 F8 EB 21 04 00 39 73 23 72 21 D4 00 22
4B20 20 4D 21 87 F0 E5 CD 5B 0B D1 21 18 00 E5 CD 51
4B30 0B D1 21 06 00 39 E5 7E 23 66 6F 2B EB E1 73 23
4B40 72 EB 23 7C B5 CA 0D 0B 21 00 00 22 28 4D C3 2C
4B50 0A 2A 28 4D 23 22 28 4D 2B 2A 28 4D 11 03 00 EB
4B60 CD A9 0B CA F7 0A 21 40 00 22 26 4D 21 16 0B E5
4B70 CD 21 0B D1 21 4B 0B E5 CD 21 0B D1 2A 26 4D 2B
4B80 22 26 4D 23 7C B5 CA EC 0A 21 04 00 39 5E 23 56
4B90 21 3F 00 19 EB 2A 26 4D CD C3 0B 5E 16 00 EB 22
4BA0 24 4D 21 00 00 22 22 4D 2A 28 4D 11 06 00 CD CB
4BB0 0B E5 2A 24 4D 11 7F 00 CD E3 0B 11 12 00 CD CB
4BC0 0B D1 19 EB 2A 20 4D 19 44 4D C3 A8 0A 2A 22 4D
4BD0 23 22 22 4D 2B 2A 22 4D 11 06 00 EB CD A9 0B CA
4BE0 E9 0A 2A 24 4D 11 80 00 CD E3 0B CA DB 0A 60 69
4BF0 23 44 4D 2B 5E 16 00 21 07 00 CD EB 0B E5 CD 5F
```

```
4C00 00 D1 C3 E6 0A 60 69 23 44 4D 2B 5E 16 00 D5 CD
4C10 5F 00 D1 C3 A0 0A C3 4F 0A 21 46 0B E5 CD 21 0B
4C20 D1 C3 24 0A 21 40 00 EB 21 04 00 39 E5 7E 23 66
4C30 6F 19 EB E1 73 23 72 C3 05 0A 21 00 00 E5 CD 5B
4C40 0B D1 C9 20 20 20 20 20 20 20 20 20 20 00 CD 65
4C50 0B 00 00 21 08 00 39 4E 23 46 60 69 7E B7 CA 45
4C60 0B 60 69 23 44 4D 2B 5E 16 00 D5 CD 5F 00 D1 C3
4C70 2D 0B C9 1B 4A 09 0D 00 1B 2A 04 80 01 00 D1 E1
4C80 E5 D5 CD 72 F3 7C B5 C9 D1 E1 E5 D5 CD 80 F3 7C
4C90 B5 C9 E1 C5 5E 23 56 23 44 4D 21 00 00 39 EB 39
4CA0 F9 D5 60 69 CD 82 0B EB E1 F9 C1 EB 7C B5 C9 E9
4CB0 7C B5 CA A4 0B C3 95 0B 7D 93 C2 95 0B 7C 92 CA
4CC0 A4 0B 21 00 00 AF C9 7D 93 C2 A4 0B 7C 92 CA 95
4CD0 0B 21 01 00 B5 C9 EB 7C AA FA BB 0B 7D 93 7C 9A
4CE0 3E 00 CE 00 6F 26 00 C9 7C 07 E6 01 6F 26 00 C9
4CF0 7B 95 6F 7A 9C 67 B5 C9 C5 44 4D 21 00 00 3E 10
4D00 29 EB 29 EB D2 DB 0B 09 3D C2 D3 0B C1 7D B4 C9
4D10 7C A2 67 7D A3 6F B4 C9 7C AA 67 7D AB 6F B4 C9
```