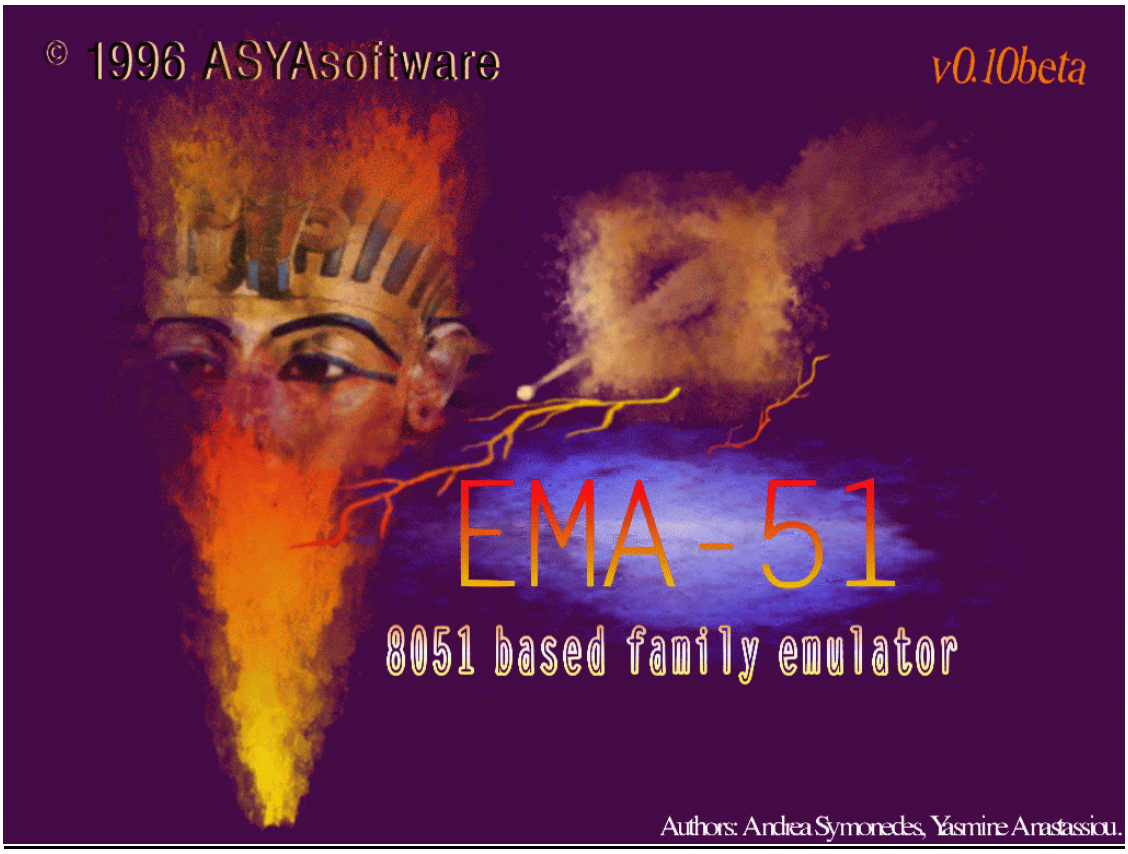
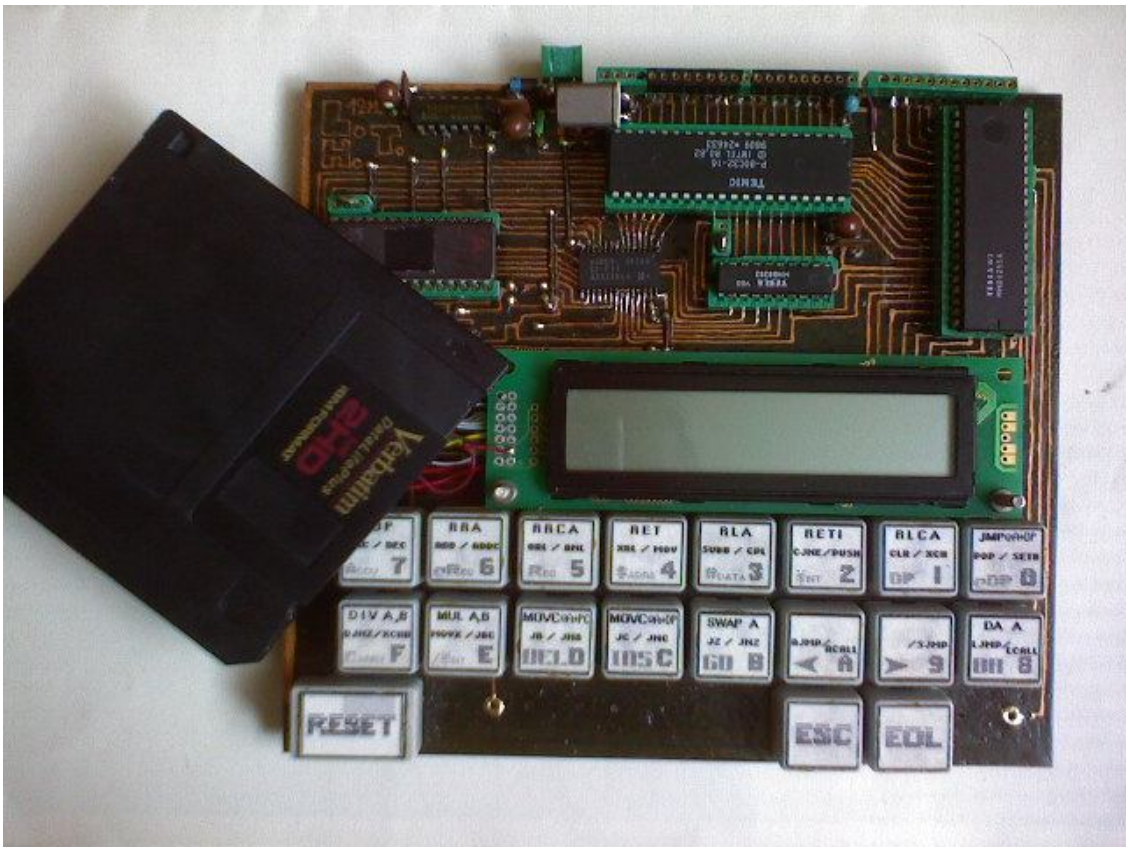

© 1996 ASYAsoftware
v0.10beta
EMA-51
8051 based family emulator
Authors: Andrea Symonedes, Yasmine Anastassiou.

Emulator EMA51 v0.10beta (1996)

(ASYAsoftware)

Je to HW emulátor jednoduchých aplikácii reálneho behu MCU rady-51. Pôvodne slúžil ako výukový mikropočítač pre školy elektrotechnického smeru a ako pomôcka pre začiatočníkov. Pracuje samostatne, nezávisle od počítača PC. Pôvodné zapojenie a prvý RAOS v0.10beta (Resident ASYAsoftware O.S.) monitor program, vytvoril ešte v roku 1995/6 dosť neznámy a dnes už (zrejme) neexistujúci grécky tím ASYAsoftware.

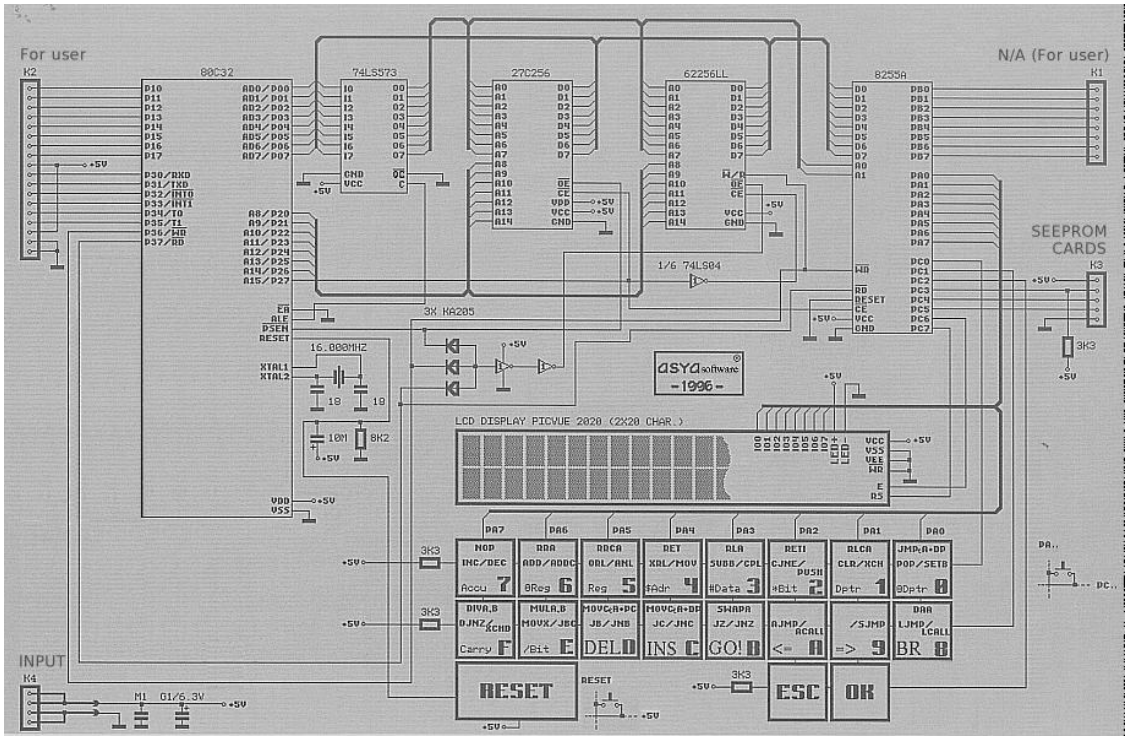
Popis zapojenia.



Prototyp EMA-51 s P80c32/16MHz/R.A.OS. v0.10b

Schéma je určená skúsenejším konštruktérom a uverejňujem ju tak, ako som ju naskenovanú získal. Doska je napájaná zo stabilizovaných 5V DC / 250mA. Základ tvorí štandardné zapojenie mikropočítača 80c32 s obvodom 74LS573 a externými pamaťami EPROM 27c256 (alebo FLASH 28c256), RAM 62256 a obvodom medzistyku 8255. Pamaťový priestor je teda rozdelený na 32 + 32kB (program / program+data). Mikropočítač môže byť ľubovolný z rady-51, musí mať však 256 bajtov internej RAM a možnosť externého adresovania, pretože RAOS monitor využíva pre svoju činnosť pouze RB0, A, PSW a DPTR. Všetko ostatné je k dispozícii pre užívateľa. A15 rozdeľuje

pomocou invertoru 74LS04 pamäťový priestor na: 0000H-7FFFH program, 8000H-FFFFH data + program, čiže do RAMky pristupujeme jak datovo, tak i programovo. Riadenie RAMky je riešené pomocou "&" hradla tvoreného 3-mi SI diódami a signál je vytvarovaný invertormi. Aktivácia RAMky je pri signáloch PSEN, WR alebo RD. HW skoky 6-stich prerušení (pre základný 80c32) sú presmerované na rovnaké adresy do RAM (0003H je na 8003H atd...). 8255 je adresovaný od adr: 0000H-0003H s opakujúcimi sa adresami pre PA, PB, PC, a STATUS registre ako ext. datová RAM. Plošný spoj neuvádzam z dôvodu ľubovolného výberu MCU. Dosku prototypu som navrhol a nakreslil ručne ako obojstrannú. Diery na plošnom spoji, prechádzajúce medzi stranami sú precínované drátikmi. Klávesnica je vyrobená z tlačidiel vyslúžilého ZX Spectrum. Ako display je použitý typ PICVUE 2020 - alfanumerický, 2 riadky po 20 znakov. Oživenie dosky: na ALE, PSEN a pinoch PA musia byť "obdĺžniky", na displayi treba mať dobrý kontrast. Ak je všetko zapojené správne, emulátor sa rozbehne okamžite, bez najmenších problémov.



Základné funkcie.

Po zobrazení loga ASYAsoftware naskočí na displayi niečo ako "okno" so 4rmi grafickými ikonkami. Kurzor, ktorý ovládame **šipkami**, a klávesami "**OK**" a "**ESC**"(menu **Setup** si popíšeme neskôr...) sa nachádza pod **prvou ikonou z ľava,** čo je:

1. HEXAEDITOR

Po **OK** zadáme adresu ext. pamäťového priestoru použitím kláves "**0 - F**", prístupná je pouze pamäť RAM:

```
ADRES=8000
```

Potvrdíme, a **blikajúci** kurzor naznačuje, že môžeme editovať klávesami "0 - F", vždy je však treba napísať celý bajt, aby zostal zobrazený blikajúci kurzor - v opačnom prípade kurzor zmizne a MCU čaká na prijatie druhej polovice bajtu, čo dosť uľahčuje orientáciu:

```
ADRES=8000
      7400F8D8FEFFFFFF
```

Ak potvrdíme **OK** opäť, kurzor sa zmení na podčiarknutý a teraz môžeme použiť klávesy "**INS**" pre vloženie bajtov FFH a "**DEL**" pre mazanie bajtov za pozíciou kurzoru. Oba príkazy fungujú od pozície kurzoru do konca rozsahu jednej stránky:

```
ADRES=8000
      7400F8FFFFFFD8FE
```

```
ADRES=8000
      7400F8FFFFFFFFFF
```

Pomocou HEXAEDITORu je možné ladiť aplikácie (ak si niekto trúfne písať program strojovo). Klávesou "**BR**" určíme break-point, na ktorý ak MCU narazí, zobrazí obsah A, PSW, SP,DPTR, R0 - R7 z banky RB0 a registra B. Pozor - bity RS0 a RS1 v PSW budú ako jediné vynulované z dôvodu prepnutia na RB0. Stav RB1, RB2, RB3, bitRAM ako aj ostatnú RAM na čipe MCU si môže užívateľ pozrieť v **CHIPEDITOR**e. Návrat do editora je cez "**ESC**". Ladený program spúšťame klávesou "**GO!**" - zadaním ardesy, odkiaľ sa má spustiť, a ak obsahuje nejakú cyklicky riešenú slučku, prerušíme ho jednoducho klávesou **RESET**:

```
BR-STOP POINT=8005
```

```
GO TO=8000
      7400F8FFFF120FF9
```

```
A PSW SP DPH,L R0 R1
00  20   7F  80  00  00  xx
```

HEXAEDITOR opustíme pomocou klávesy "**ESC**" návratom do základného okna. Pozor - ak sme mali určený break point, tak tento sa automaticky odstráni a výhodou je, že tiež zostane zapamätaná adresa poslednej „celej“ adresy. V základnom okne teraz posunieme kurzor na druhú ikonu z ľava, a s OK spustíme tzv.

2. CHIPEDITOR

Pomocou editoru si môžeme prezerať / editovať RAM na čipe MCU obdobne, ako v **HEXAEDITOR**e s tým, že sú vypustené funkcie "INS", "DEL", "GO" a "BR":

```
TO=00
```

```
RBANK
00     0002FB0107110001
```

```
BIT'S
20     FFFD76FF00FFFF1E
```

```
SRAM
30     FFFFFFFFFFFFFFFF
```

RB0 = 00H-07H(je v použití), RB1 = 08H-0FH, RB2 = 10H-17H, RB3 = 18H-1FH, bitRAM = 20H-2FH(128 bitov), ostatná RAM = 30H-FFH. Tu chcem upozorniť na SP, ktorý ukazuje vždy po zapnutí / RESETe emulátoru na adr: 7FH. SFR nie sú prístupné z dôvodu ľubovolného výberu MCU v tomto zapojení. Výhodou editoru je, že si užívateľ pred spustením určitej časti vyvíjanej aplikácie môže vopred navoliť hodnoty na určitých adresách, alebo si pozrieť zmeny, ktoré sa po skončení udiali. Tretia ikonka je tzv:

3. TEXTEDITOR

Editor automaticky "prekladá" zdrojový kód do textu, po zadaní adresy môžeme listovať šipkou vpravo po jednotlivých inštrukciách vpred. Keďže takýto preklad opačným spôsobom správne nepracuje, spôsobí šipka vľavo skok na inštrukciu, ktorej adresu sme zadali pri vstupe do tohoto editoru. Ostatné funkcie - "INS", "DEL", "GO" a "BR" tu plnia ten istý účel, ako v **HEXAEDITOR**e:

```
ADRES=8000
```

```
8000 MOV A,  #00
8002 MOV R0,  A
```

Chem ešte upozorniť, že niektoré inštrukcie, symboly alebo označenia sú na rozdiel od assembleru skrátené, alebo zmenené:

1. Bajt A5H (inštrukcia bez významu) je vypisovaný tak isto ako 00H - "NOP".
2. bitová premenná je označená symbolom "ψ".
3. Inštrukcia CJNE je písaná ako "CJ".
4. Registrový pár DPTR je písaný ako "DP".

Príklad zobrazenia: CLR bit 90; CJNE A #32, $00; MOV DPTR, #8400.

```
8003 CLR      ψ90
8005 CJ  A  #32,  $00
```

```
8005 CJ  A  #32,  $00
8008 MOV  DP,  #8400
```

Ostatné inštrukcie sa už všetky zhodujú s výrazmi podobne, ako pri písaní v assembleri. Stlačením OK prejdeme do režimu vkladania údajov z klávesnice. Ak hocikde pri písaní inštrukcie spravíme chybu napísaním nezmyslu, pomocou ESC riadok zrušíme. Ak napíšeme nezmyselnú inštrukciu, (zápis Accu do # dátovej premennej namiesto do adresy, alebo číslo registra väčšie, ako 07) editor vypíše chybové hlásenie: <error line>, či <Rx ??>. Príklady použitia editora pri písaní:

Chceme po sebe napísať inštrukcie <u>NOP</u>; <u>MOV</u> R4, #AB; <u>INC</u> R4.

1. RAMku 9000H-9007H naplníme kôli prehľadnosti v HEXAEDITORe bajtmi FFH. Otvoríme HEXAEDITOR na adrese 9000H, prepneme podčiarknutý kurzor, a 8x stlačíme klávesu INS.

```
ADRES=9000
      FFFFFFFFFFFFFFFF
```

2. Otvoríme TEXTEDITOR na adrese 9000H.

```
9000 MOV  R7,  A
9001 MOV  R7,  A
```

3. Stlačením OK prejdeme do editačného módu:

```
9000 MOV  R7,  A
     ^
```

Šipka "hore" naznačuje, že pre napísanie inštrukcie <u>NOP</u> stačí stlačiť klávesu pre "horný" príkaz - **NOP**(7)

```
9000 NOP
9001^
```

4. Teraz napíšeme inštrukciu, ktorá je na klávese ako "pravý" príkaz: <u>MOV</u> R4, #AB. Postupne stlačíme klávesy: →(9), **MOV**(4), **reg**(5), (**0**), (**4**), **OK**, **#data**(3), (**A**), (**B**), **OK**.

```
9000 NOP
9001→ MOV  R04,  #AB
```

```
9001 MOV  R04,  #AB
9003^
```

5. A pre napísanie inštrukcie INC R4, ktorá je ako "ľavý" príkaz stlačíme: ←(A), **INC**(7), **reg**(5), (**0**), (**4**), **OK**.

```
9001 MOV  R04,  #AB
9003←INC  R04
```

```
9003 INC  R04
9004^
```

Stlačením 3x ESC vyskočíme do základného okna a prejdeme do HEXAEDITORu, ktorý otvoríme na adrese 9000H. Zobrazí sa výpis:

```
ADRES=9000
      007CAB0CFFFFFFFF
```

Poslednou ikonkou na ploche základného okna je tzv:

4. DRAWER

Je to čosi ako adresár pripomínajúci "šuplík", ktorého otvorením získame okno za predpokladu, že máme prítomnú výmennú "naformátovanú" pamaťovú kartičku (RAOS monitor vyzve k jej formátovaniu). Tú jednoducho vytvoríme pomocou čipu sériovej EEPROM typu AT24cxxx. Ich kapacity sú 8, 16, 32 a 64kB. Vzhľadom na to, že aplikácie na MCU-51 sú niekoľko kbitové, tieto obvody bohate stačia, aby plnili výbornú funkciu akéhosi rozmerovo maličkého, lacného záznamového média s počtom zápisov 100k a dobou zachovania informácie okolo 20 - 40 rokov. Okno má 4 pozície. Prvá pozícia (z ľava) slúži pre návrat nazad, ďalšie tri sú pre uloženie ikon vytvorených aplikácií, alebo ďalších podadresárov. Otvorením ikony vytvoreného adresára sa otvorí vždy ďalšie okno, otvorením ikony uloženej aplikácie sa táto natiahne do RAM, a okamžite sa prevedie jej spustenie od adresy štartu (bližšie - setup). Návratom do základného okna, a stlačením klávesy "ESC" otvoríme menu:

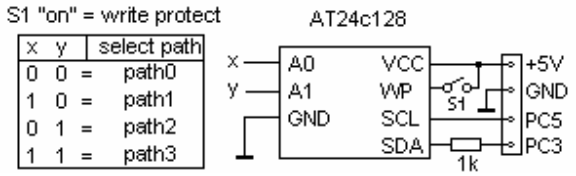
SETUP

Je to čosi ako správca súborov na výmenných kartičkách. Musím poznamenať, že setup menu nie je zrovna vychytané, ale základné funkcie čítanie / zápis / mazanie má a je úplne postačujúce. Doporučujem však viesť si záznamy o súboroch na výmenných čipoch. Činnosť v SETUP menu si ukážeme na príkladoch:

***1.* V RAMke od adr. 8000H po 8123H máme vytvorený program, ktorý spúšťame od adresy 8000H a chceme ho uložiť.**

***2.* Chceme si vytvoriť ďalší podadresár pre budúce použitie.**

Ako prvú vec vyberieme kartu, z ktorej / na ktorú chceme data ukladať. Zobrazené sú ako ikonky diskiet. Pamate je možné použiť až 4 - musia byť však hardwarovo naadresované pomocou adresných vstupov A0, A1. Príklad použitia pamäťovej karty - veľkosť 16kB:



Šipkami vyberieme DRAWER (alebo neskôr aj iný podadresár), ktorého ikona sa zobrazí v zátvorkách a do ktorého budeme ukladať aplikáciu (alebo ďalšie podadresáre). Stlačíme OK, pričom preblikne podčiarknutý kurzor pod **S**ETUP. Sme na pozícii prvej ikony v okne vybraného adresára. Zadaním bajtu v rozsahu 01H-B7H vyberieme najvhodnejšiu ikonu aplikácie (00H je tiež ikona, no nieje v použití = upozornenie RAOSmonitoru) pre náš vytvorený program. Stlačením OK zase preblikne kurzor a sme na pozícii druhej ikony. Zadaním bajtu v rozsahu B8H-FEH vyberieme vhodný symbol pre podadresár, ktorý chceme vytvoriť. Stlačíme OK a sme na tretej - poslednej pozícii. Tu neurobíme nič a následným stlačením OK sa objaví pod ikonami číselné menu SELECT:

```
SETUP:                ( X )
```

```
SETUP:   %   x        ( X )
```

```
SETUP:   %   x        ( X )
SELECT:[1] [2] [3]
```

("%"=aplikácia, "x"=podadresár, a "X"=DRAWER. Písmená tu nahradzujú grafické ikonky, ktoré LCD display normálne zobrazí)

Vyberieme [1] a zadáme stránky, na ktorých sa nami vytvorený program nachádza, čiže 80 -> 81 (uložený bude rozsah 8000H-81FFH) a adresu spustenia 8000 (ak zadáme adresu spustenia 0000H, tak program sa uloží ako dáta = nespustí sa). Potvrdením sa vrátime nazad do SELECT:

```
SETUP:   %   x        ( X )
Apl, at: 80 → 81,  go: 8000
```

Pre pozíciu [2] a [3] ponuka nieje,...

```
SETUP:  %  x        ( X )
CANT EDIT PARAMETER
```

...takže sa stlačením OK dostaneme do menu SAVE (ICON), kde vyberieme [1] - uloží sa nami vytvorený program a [2] - vytvorí sa podadresár:

```
SAVE(ICON): [1]  [2]  [3]
FORMAT:              [F]
```

Stlačením 2x ESC, SETUP opustíme do základného okna a overíme si uložené dáta v DRAWERi, ktorý sa otvorí a bude obsahovať ikonu našej aplikácie a ďalšieho podadresára. Stlačením OK na ikone aplikácie sa program natiahne do RAMky a beží. Na spodku okna obdržíme hlásenie: EXECUTIVE... Ikona podadresára otvorí iba prázdne okno.

***3.* Oba (vytvorený program aj podadresár) chceme odstrániť.**

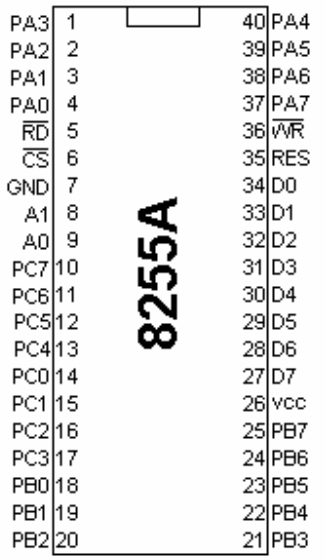
Vojdeme do SETUPu, vyberieme kartu, z ktorej / na ktorú chceme data ukladať a nalistujeme šipkami DRAWER. Vložíme na pozície ikon 1 a 2 bajt FFH (na predtým uloženej pozícii aplikácie sa nám zobrazí INFO o súbore: stránky "od - do" a adresa spustenia) prejdeme do menu SAVE(ICON) a vyberieme [1] a [2]. Ak potrebujeme naformátovať celú pamať, zvolíme [F]. Nevýhodou je, že program s rovnakou ikonou nie je možné prepísať, ale ho musíme najskor zmazať a až potom uložiť. Tiež adresár - pokiaľ obsahuje nejaké údaje - je ho možné zmazať až po ich odstránení. Predpokladám, že to ASYAsoftware spravili ako prevenciu voči nechcenému zmazaniu dát.

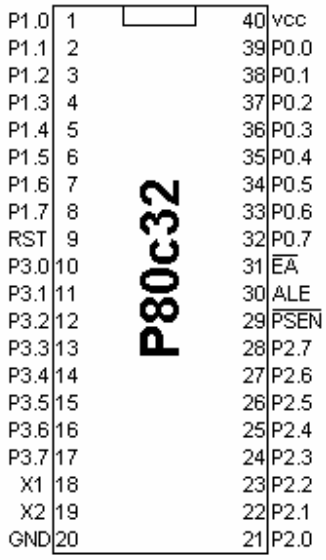
Je fakt, že písanie programu v zdrojáku a vypočítavanie skokov z hlavy sa v dnešnej dobe už dávno nenosí. Naproti tomu ponúkaná možnosť nasúkať do výmenných kariet emulátoru vlastné, už vyladené programy, či utility a použiť toto zariadenie miesto inej HW karty s pripojeným notebookom a hromadou prepojovacích káblov v prašnom, horúcom, či vlhkom prostredí na dostavovanie dát, alebo konštánt napr. na nejakom k tomu pripojenom zariadení určite stojí za zváženie. Veľkosť opísaného RAOS monitoru je cca 15kB.

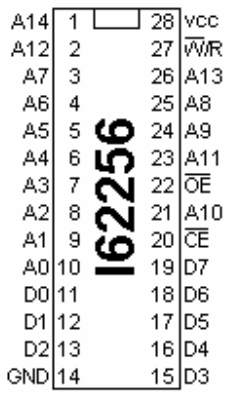
Súčiastky.

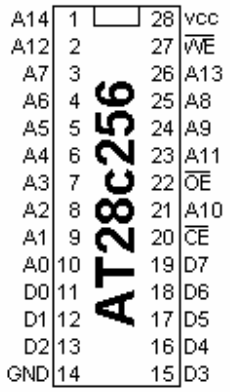
Na obrázku sú obvody k zapojeniu. Miesto EPROM je možné použiť pamäť FLASH.

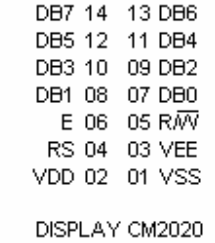
ODPORY:	KONDENZÁTORY:	POLOVODIČE:	
-4x 3k3	-2x 18p keram.	3x SI dioda	1x I27c256(28c256)
-1x 8k2	-1x 100n keram	1x P80c32	1x SN74LS573
KRYŠTÁL:	-1x 10M/6,3V tantal	1x I62256	1x SN74LS04
16.000MHz	-1x 100M/6,3V	1x I8255A	display CM 2020

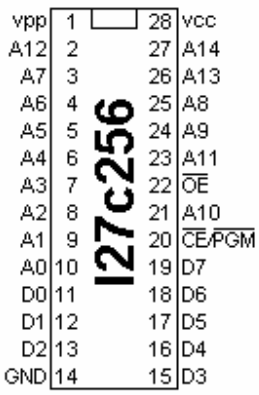

8255A
PA3 1 40 PA4
PA2 2 39 PA5
PA1 3 38 PA6
PA0 4 37 PA7
RD 5 36 WR
CS 6 35 RES
GND 7 34 D0
A1 8 33 D1
A0 9 32 D2
PC7 10 31 D3
PC6 11 30 D4
PC5 12 29 D5
PC4 13 28 D6
PC0 14 27 D7
PC1 15 26 vcc
PC2 16 25 PB7
PC3 17 24 PB6
PB0 18 23 PB5
PB1 19 22 PB4
PB2 20 21 PB3

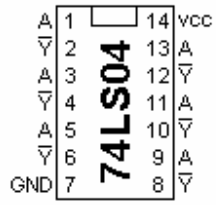

P80c32
P1.0 1 40 vcc
P1.1 2 39 P0.0
P1.2 3 38 P0.1
P1.3 4 37 P0.2
P1.4 5 36 P0.3
P1.5 6 35 P0.4
P1.6 7 34 P0.5
P1.7 8 33 P0.6
RST 9 32 P0.7
P3.0 10 31 EA
P3.1 11 30 ALE
P3.2 12 29 PSEN
P3.3 13 28 P2.7
P3.4 14 27 P2.6
P3.5 15 26 P2.5
P3.6 16 25 P2.4
P3.7 17 24 P2.3
X1 18 23 P2.2
X2 19 22 P2.1
GND 20 21 P2.0

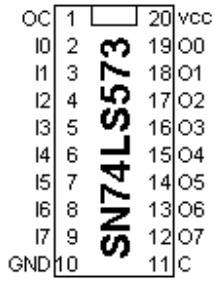

I62256
A14 1 28 vcc
A12 2 27 WR
A7 3 26 A13
A6 4 25 A8
A5 5 24 A9
A4 6 23 A11
A3 7 22 OE
A2 8 21 A10
A1 9 20 CE
A0 10 19 D7
D0 11 18 D6
D1 12 17 D5
D2 13 16 D4
GND 14 15 D3


AT28c256
A14 1 28 vcc
A12 2 27 WE
A7 3 26 A13
A6 4 25 A8
A5 5 24 A9
A4 6 23 A11
A3 7 22 OE
A2 8 21 A10
A1 9 20 CE
A0 10 19 D7
D0 11 18 D6
D1 12 17 D5
D2 13 16 D4
GND 14 15 D3


LEDK 16 15 LEDA
DB7 14 13 DB6
DB5 12 11 DB4
DB3 10 09 DB2
DB1 08 07 DB0
E 06 05 R/W
RS 04 03 VEE
VDD 02 01 VSS
DISPLAY CM2020


I27c256
vpp 1 28 vcc
A12 2 27 A14
A7 3 26 A13
A6 4 25 A8
A5 5 24 A9
A4 6 23 A11
A3 7 22 OE
A2 8 21 A10
A1 9 20 CE/PGM
A0 10 19 D7
D0 11 18 D6
D1 12 17 D5
D2 13 16 D4
GND 14 15 D3


74LS04
A 1 14 vcc
Y 2 13 A
A 3 12 Y
Y 4 11 A
A 5 10 Y
Y 6 9 A
GND 7 8 Y


SN74LS573
OC 1 20 vcc
I0 2 19 O0
I1 3 18 O1
I2 4 17 O2
I3 5 16 O3
I4 6 15 O4
I5 7 14 O5
I6 8 13 O6
I7 9 12 O7
GND 10 11 C