Univerza v Ljubljani
FAKULTETA ZA ELEKTROTEHNIKO

Sekvencer - ritem mašina

Seminarska naloga pri predmetu

ELEKTRONSKA VEZJA

Avtor: Dušan SLAVINEC
Štud. program/smer : UNI / elektronika
V Ljubljani, maj 2004

1. UVOD

Namen seminarske je bil seznanitev s snovanjem in problematiko digitalnih vezij ter izdelavo vezij naspošno.
Do sedaj še nisem imel nobenih izkušenj z načrtovanjem vezij, izdelavo tiskanih ploščic in praktičnih izkušenj z elektroniko. Seminarska naloga pri predmetu DIGITALNE STRUKTURE je bila dober razlog, da sem se tega lotil. Torej izdelati digitalno vezje. Pogoj je bil tudi, da naj ne bo uporabljen mikrokrmilnik.
Zamislil sem si preprosto ritem mašino oz. sekvencer.

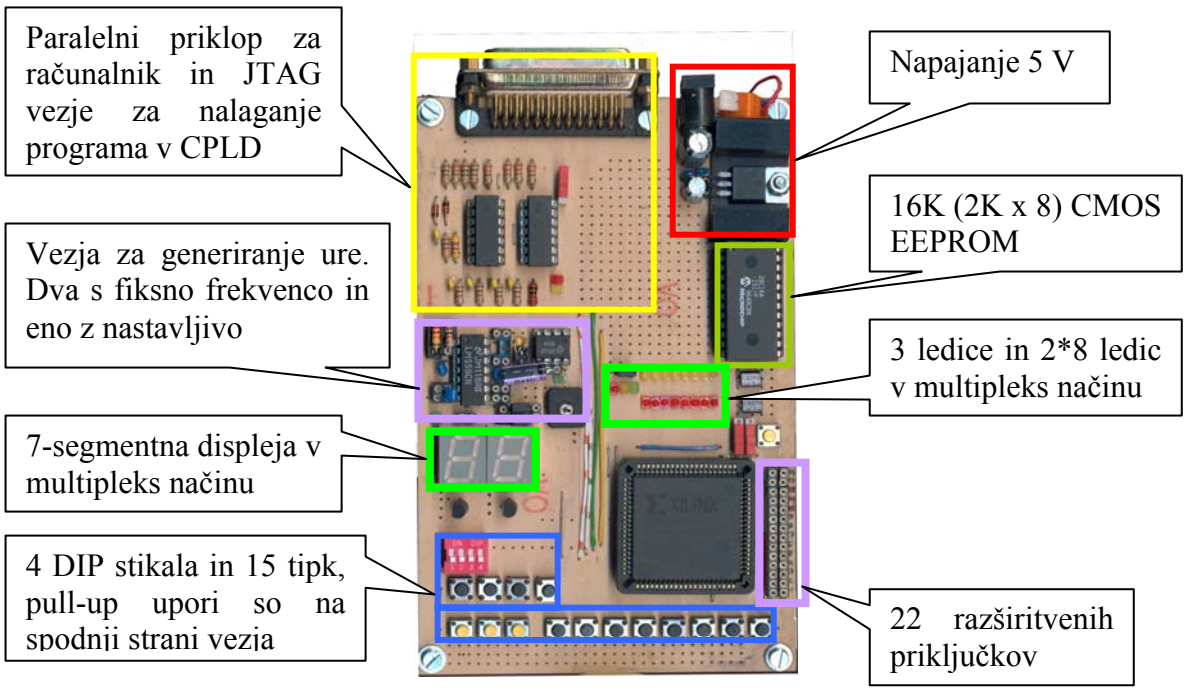
2. OPIS VEZJA

Ker sem si zamislil vezje kot večnamensko, uporabno tudi za kaj drugega, sem zato uporabil programabilni čip. Za srce vezja sem uporabil Xilinx XC9572-PC84 CPLD – programabilno vezje. Xilinx-a sem izbral ker:

- se ga da dobit pri nas,
- je v ohišju PLCC, ki ni zahtevno za spajkanje
- razvojno okolje Xilinx WebPack je dostopno na internetu

Ima 72 makrocelic in 69 uporabnih pinov.

Ostali sklopi predstavljajo:

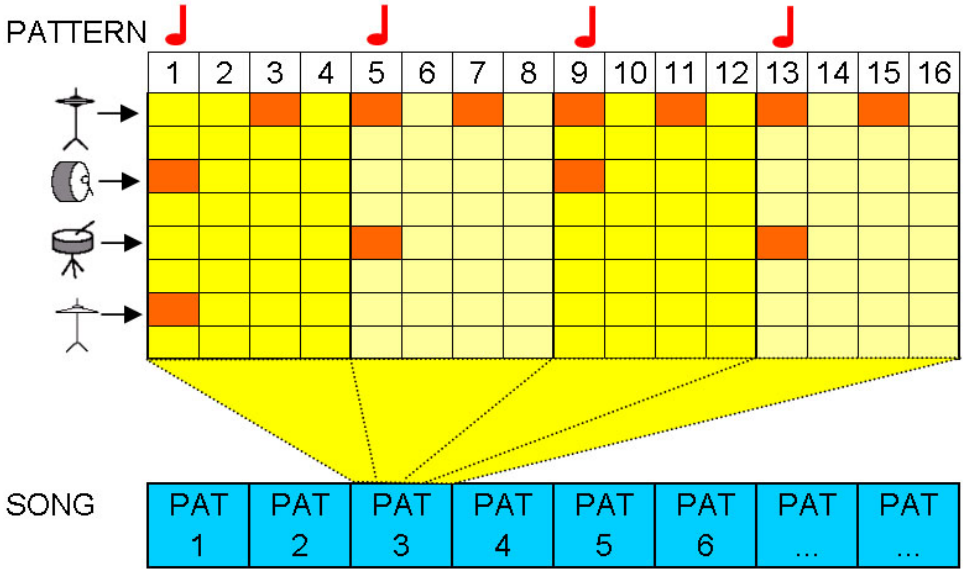


Vezje JTAG za nalaganje programa v CPLD je narejeno po načrtu na internet straneh proizvajalca.
Za generiranje ure sem uporabil LM556 s fiksno frekvenco (za multipleksiranje displejev) in LM567 za nastavljivo frekvenco (tempo). Upori in kondenzatorji se lahko hitro menjajo, tako da dobimo željeno frekvenco ure. EEPROM je priključen na razširitvene pine.

3. IZRAZI

V nadaljevanju bom uporabljal naslednje izraze – večinoma angleške, ki se uporabljajo :

PATTERN – je najmanjša enota, ki jo lahko predvajamo. Predstavlja 4 udarce – note četrtinke - 4/4 taktu in je razdeljen na 16 delov, not šestnajstink.



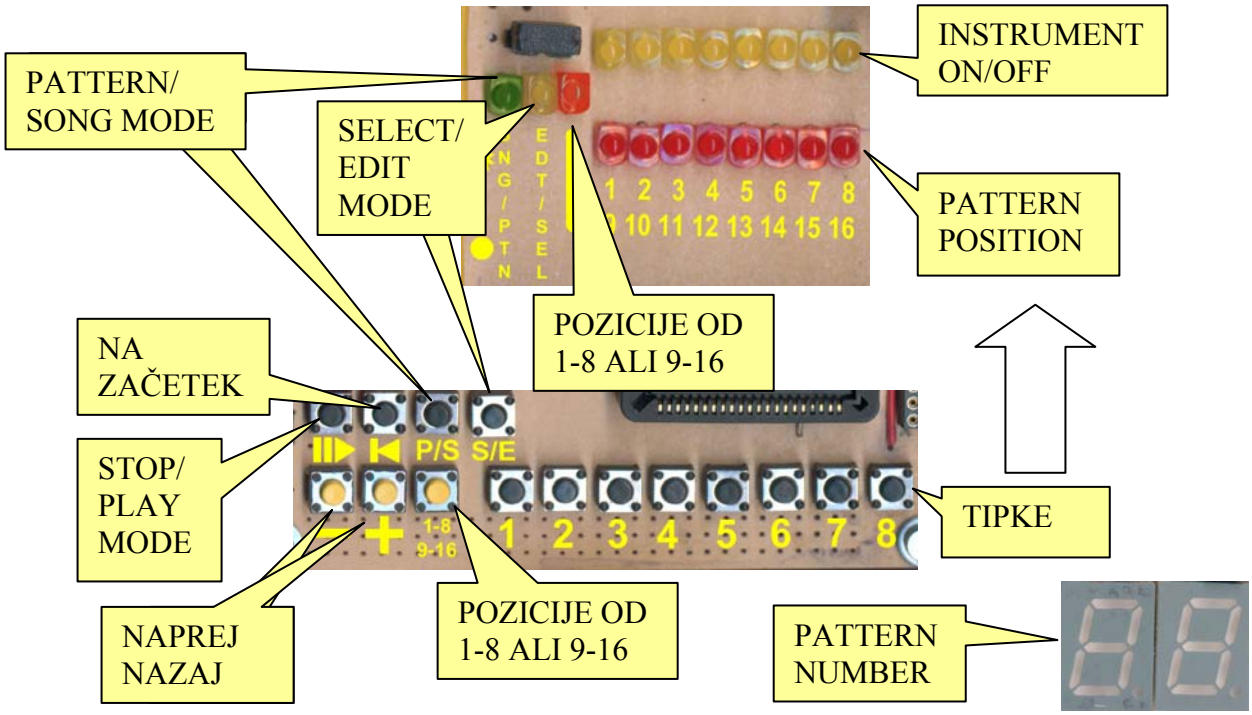
SONG – je sestavljen iz patternov, ki jih predvajamo zaporedno. V našem primeru imamo lahko največ 64 patternov.
STEP PROGRAMMING/EDITNIG – programiranje / urejanje po korakih, najmanjših enotah.
MODE – stanje, način v katerem deluje vezje.

4. DELOVANJE

Sekvencer omogoča koračno programiranje. Imamo 4/4 takt oz. programiramo note šesnajstinke. Ker ima EEPROM 8 pinov za podatke, imamo tako 8 stez, ki jih lahko uporabimo za proženje instrumentov – vezij za generiranje zvokov. Te prožilne signale dobimo na pinih B1-B8 razširitvenega konektorja.

TIPKE IN DISPLEJI

Sekvencer upravljamo s tipkami, delovanje pa opazujemo na ledicah:



Vezje deluje v več načinih med katerimi preklapljamo s tipkami STOP/PLAY, PATTERN/SONG in SELECT/EDIT. 7-segmentni displej prikazuje številko izbranega patterna. Zelena in rumena ledica kažeta način v katerem smo. Funkcije ostalih tipk so opisane v tabeli

TIPKE	STOP MODE				PLAY MODE
	PATTERN MODE		SONG MODE		
	SELECT MODE	INSTR. MODE	SELECT MODE	INSTR. MODE	
⏮	na začetek patterna		na začetek songa		
-	pattern nazaj	korak naprej	pattern nazaj	korak naprej	
+	pattern naprej	korak nazaj	pattern naprej	korak nazaj	
1-8/9-16	preklop med koraki 1-8 in 9-16		preklop med koraki 1-8 in 9-16		
1-8	izbira koraka od 1 do 8 ali 9 do 16	vklop/izklop note na trenutnem koraku	izbira koraka od 1 do 8 ali 9 do 16	vklop/izklop note na trenutnem koraku	

Z DIP stikalom 1 v položaju ON lahko v PLAY načinu pobrišemo pattern – izklopimo vse note. Z DIP stikalom 4 pa izberemo song A ali B.

Ob vklopu je vezje v STOP – PATTERN – SELECT načinu.

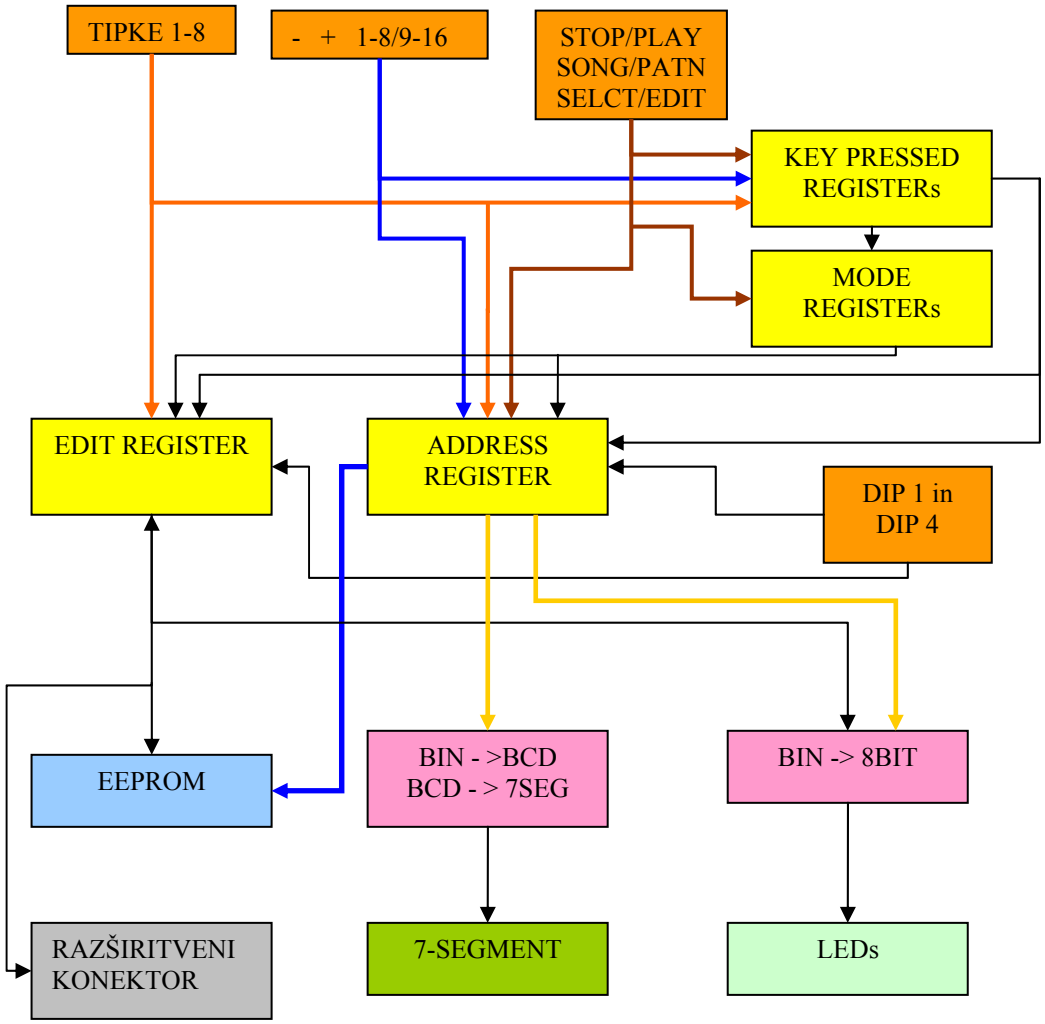
Vezje v delovanju porabi do 180mA toka.

5. OPIS PROGRAMA

Program je pisan v ABEL-u. Uporabljal sem programsko okolje XILINX WebPack 4.2.

Implementirani so naslednji sklopi:

- -EDIT REGISTER: register paralelnim dostopom, izmenjuje podatke z EEPROM-om. V tem registru editiramo note patterna.
- ADDRESS COUNTER: naslovni števec za EEPROM
- PREKODIRNIK BIN -> BCD: prekodira 5-bitno binarno število v BDC kodo z utežmi 8-4-2-1
- PREKODIRNIK BCD ->7-SEGMENT: prekodira 4-bitno BCD število v 7-segmentno kodo
- MODE REGISTRI: beležijo stanje, način delovanja
- KEY PRESSED REGISTRI: s temi registri ugotavljamo, kdaj je bila pritisnjena katera od skupin tipk



6. ZAKLJUČEK

Z delovanjem vezja sem zadovoljen. Izdelava in programiranje sta bili zelo dobri izkušnji. Ideje sem dobival sproti, tako da sem nenehno spreminjal in dopolnjeval vezje ter program. Pri risanju električne sheme vezja sem nihal med dvema programoma, saj je vsak imel določene prednosti oz. slabosti. Potem pa sem ugotovil, da bi z risanjem sheme porabil preveč časa, zato sem kar direktno risal PCB za tiskanino, saj je vezje dokaj preprosto. Shemo sem nato narisal kasneje.
Pri izdelavi in preverjanju vezja bi mi zelo prav prišel osciloskop. Doma sem imel na razpolago samo multimeter. Porabil sem kar nekaj ur, da sem ugotovil neustrezni signal za uro iz LM567. No, pomagal sem si s programi za PC-ja, ki delujejo kot osciloskop preko zvočne kartice. Nekaj napak je bilo tudi na sami tiskanini, saj je na bakru bila odrgnina, dovolj globoka, da so na več mestih bile linije prekinjene, kar pa ni bilo vidno na prvi pogled. Za testiranje programa bi mi zelo prav prišel simulator.
Če bi se še enkrat lotil izdelave bi:

- izbral bolj trdo ploščico
- JTAG del vezja bi izdelal posebej, na tiskanino bi dal samo priključke
- dodal 12,5 MHz ali 25MHz oscilator za VGA
- LED diode in 7SEG displej bi priključil preko ULN čipa, da ne bi preobremenjeval CPLD-ja
- namesto Xilinx XC9572 bi vzel raje Xilinx XC95108, ki ima isto ohišje, vendar več makrocelic

7. PRILOGE

- FITTER REPORT
- shema vezja
- program v ABEL-u
- podatki za LM556, LM567 in Xilinx XC9572, Microchip 28C16A EEPROM

FITTER REPORT

cpldfit: version E.37 Xilinx Inc. Fitter Report
Design Name: JAPANAC1 Date: 9-16-2002, 4:09AM
Device Used: XC9572-7-PC84
Fitting Status: Successful

********** Resource Summary **********

Macrocells Used	Product Terms Used	Registers Used	Pins Used	Function Block Inputs Used
62 /72 (86%)	306 /360 (85%)	25 /72 (34%)	62 /69 (89%)	126/144 (87%)

PIN RESOURCES:

Signal Type	Required	Mapped	Pin Type		Used	Remaining
Input	:18	18	I/O	:	60	3
Output	:26	26	GCK/IO	:	2	1
Bidirectional	:17	17	GTS/IO	:	0	2
GCK	:1	1	GSR/IO	:	0	1
GTS	:0	0				
GSR	:0	0				
	----	----				
Total	62	62				

MACROCELL RESOURCES:

Total Macrocells Available	72
Registered Macrocells	25
Non-registered Macrocell driving I/O	26

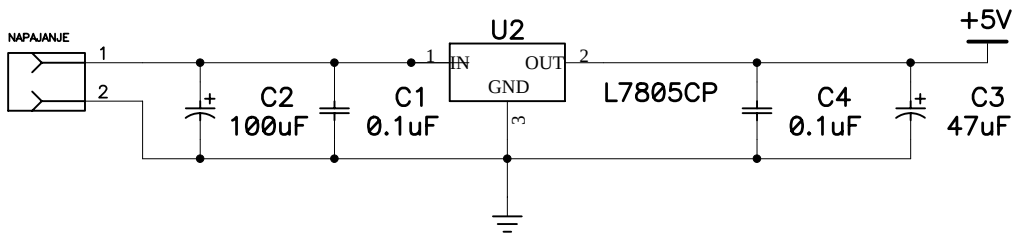
GLOBAL RESOURCES:

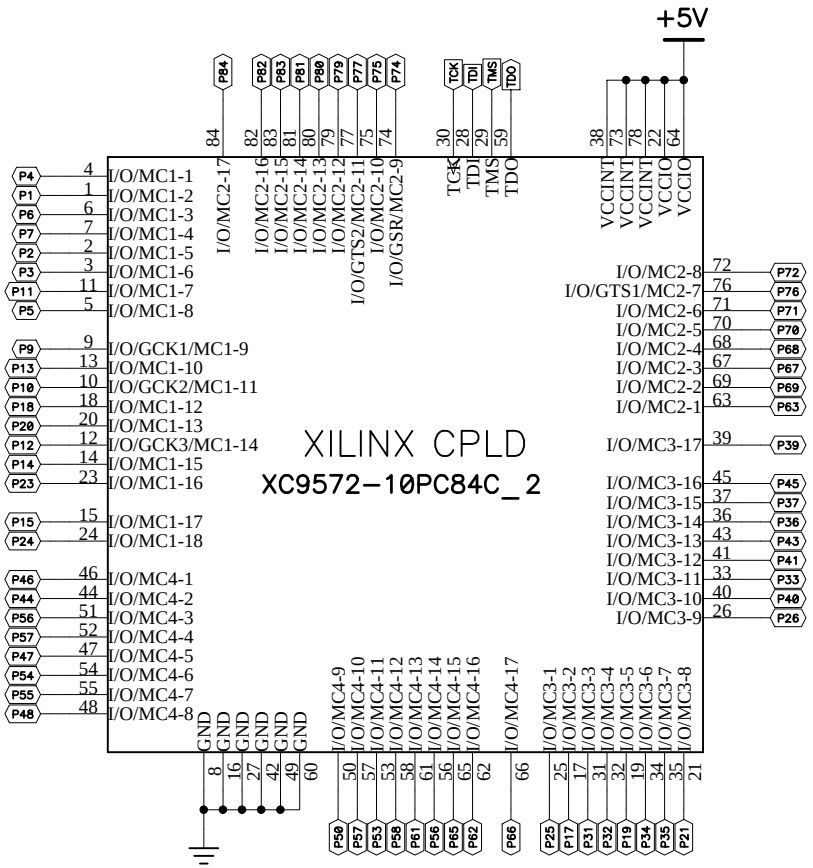
Signal 'clk_tempo' mapped onto global clock net GCK3.
Global output enable net(s) unused.
Global set/reset net(s) unused.

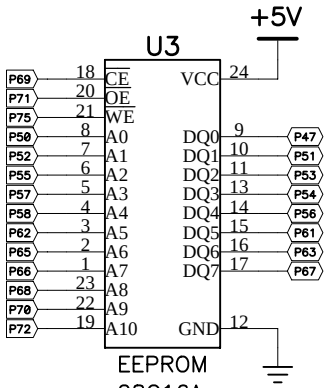
POWER DATA:

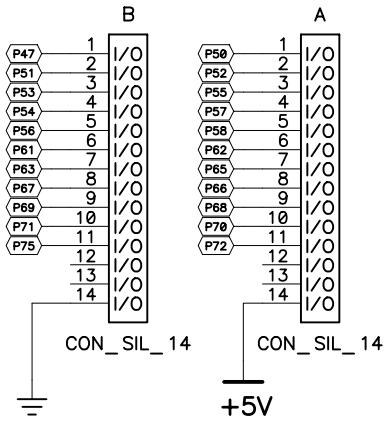
There are 0 macrocells in high performance mode (MCHP).
There are 62 macrocells in low power mode (MCLP).
There are a total of 62 macrocells used (MC).

End of Resource Summary

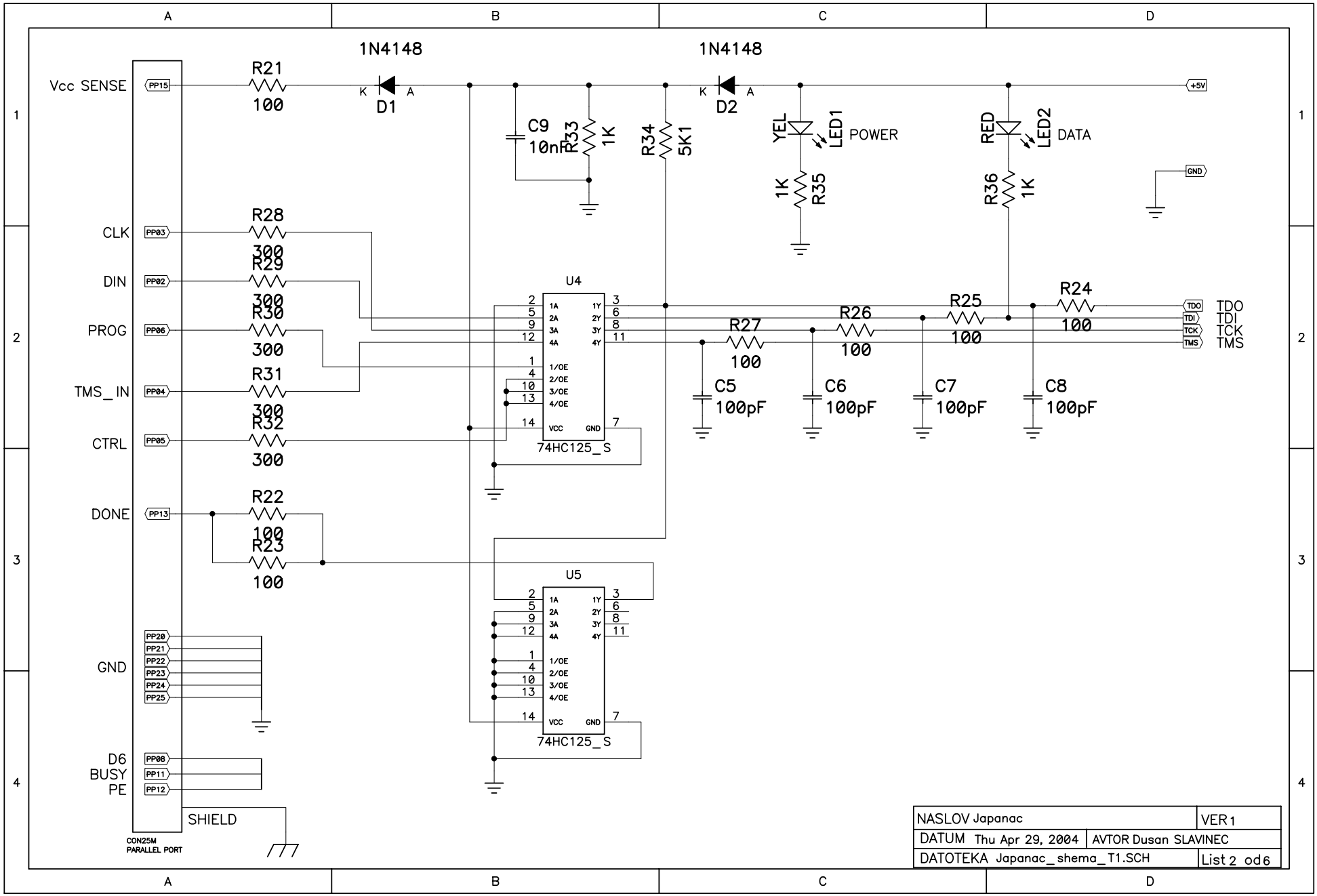


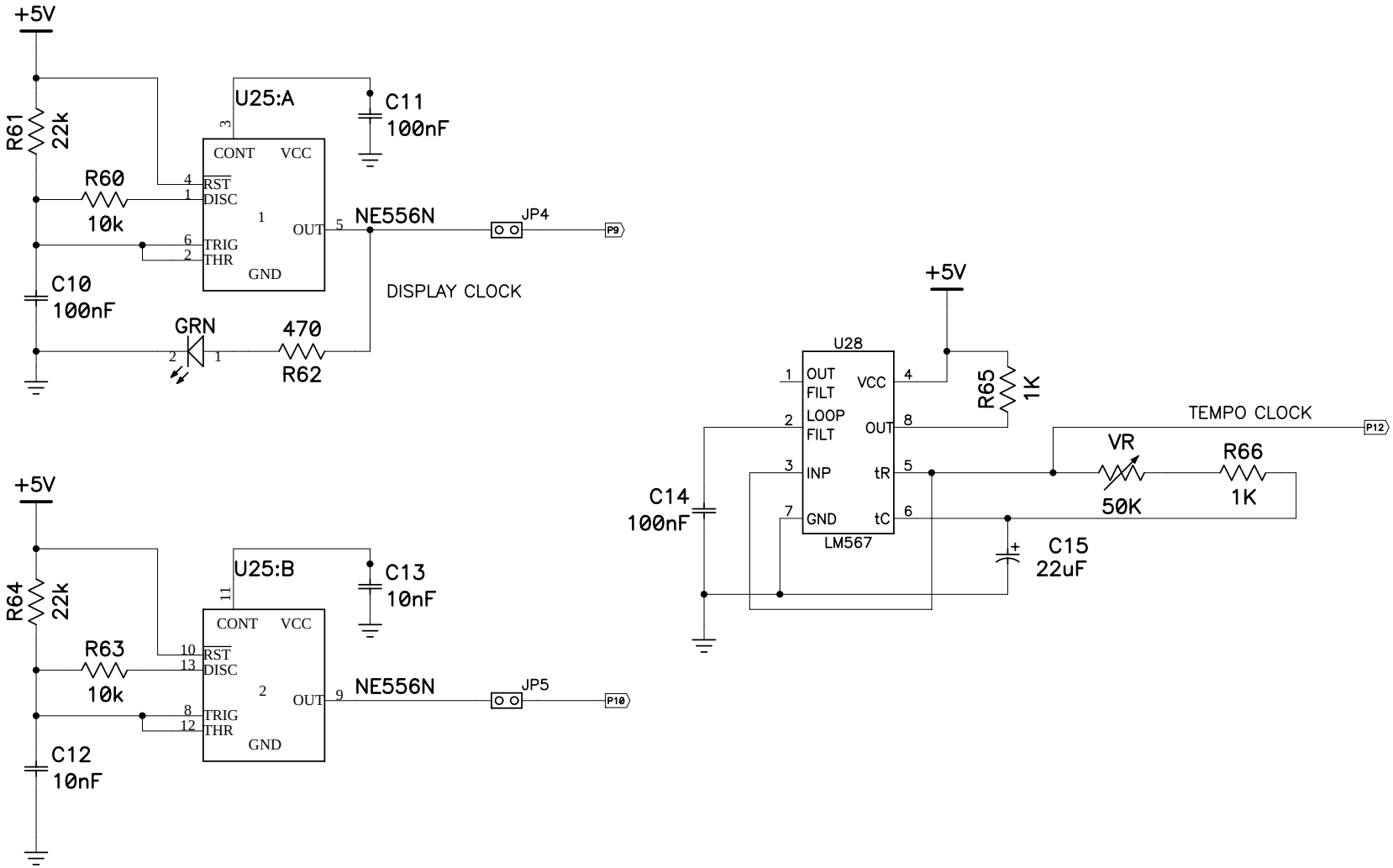




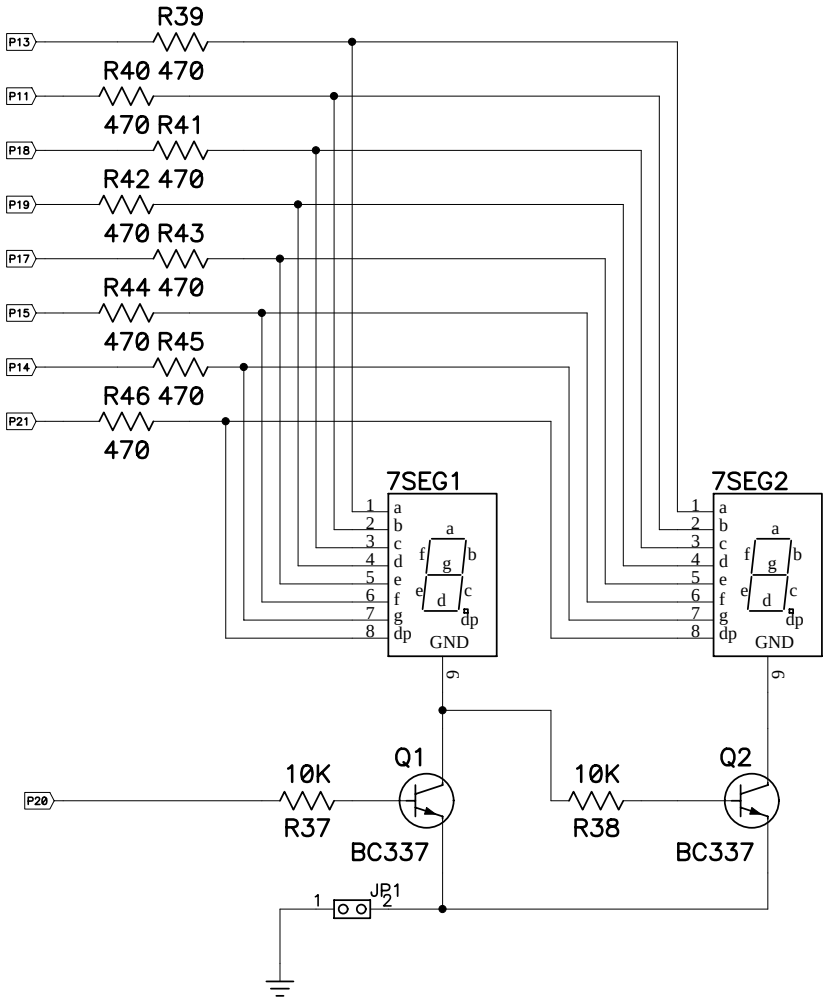


NASLOV Japanac		VER 1
DATUM Thu Apr 29, 2004	AVTOR Dusan SLAVINEC	
DATOTEKA Japanac_shema_T1.SCH		List 1 od 6

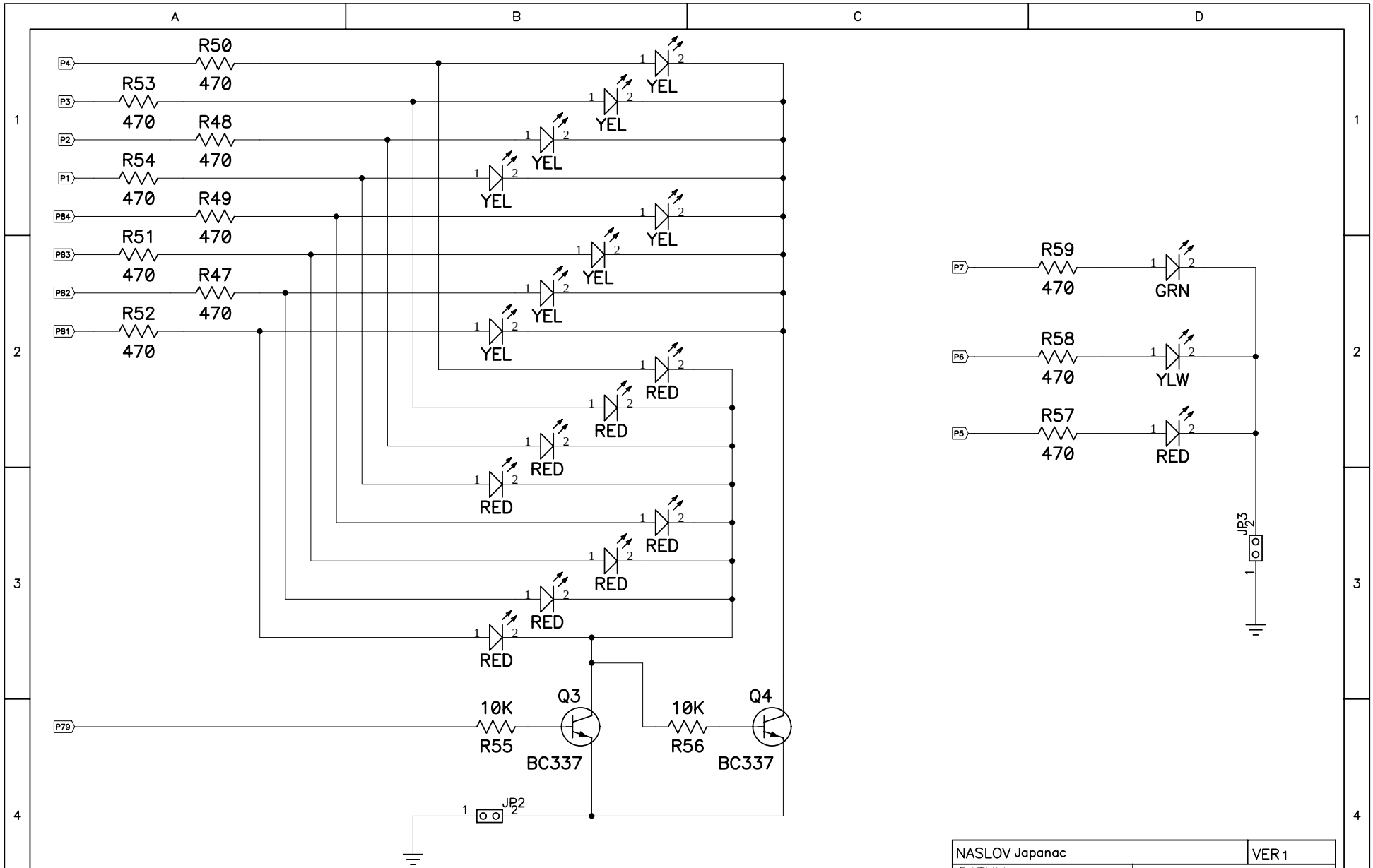

1N4148
1N4148
Vcc SENSE
PP15
R21
100
D1
D2
+5V
C9
10nF
R33
1K
R34
5K1
YEL
LED1
POWER
RED
LED2
DATA
1K
R35
R36
1K
GND
CLK
PP03
R28
300
DIN
PP02
R29
300
PROG
PP06
R30
300
TMS_IN
PP04
R31
300
CTRL
PP05
R32
300
U4
74HC125_S
R27
100
R26
100
R25
100
R24
100
TDO
TDI
TCK
TMS
C5
100pF
C6
100pF
C7
100pF
C8
100pF
DONE
PP13
R22
100
R23
100
U5
74HC125_S
GND
PP20
PP21
PP22
PP23
PP24
PP25
D6
PP08
BUSY
PP11
PE
PP12
SHIELD
CON25M
PARALLEL PORT
NASLOV Japanac
VER 1
DATUM Thu Apr 29, 2004
AVTOR Dusan SLAVINEC
DATOTEKA Japanac_shema_T1.SCH
List 2 od 6

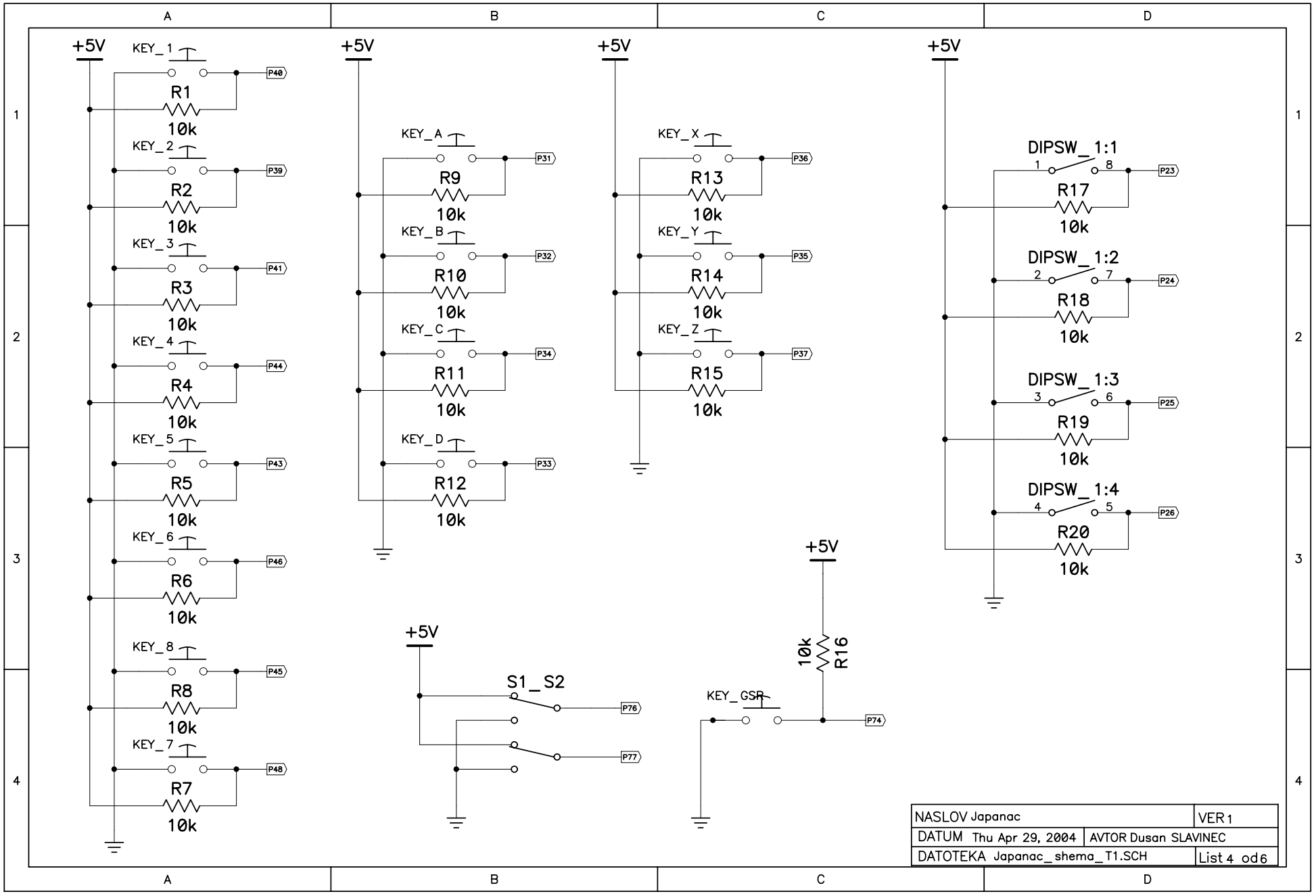


NASLOV Japanac		VER 1
DATUM Thu Apr 29, 2004	AVTOR Dusan SLAVINEC	
DATOTEKA Japanac_ shema_ T1.SCH		List 6 od 6



NASLOV Japanac		VER 1
DATUM Thu Apr 29, 2004	AVTOR Dusan SLAVINEC	
DATOTEKA Japanac_ shema_ T1.SCH		List 3 od 6


A
B
C
D
1
2
3
4
P4
P3
P2
P1
P84
P83
P82
P81
R50
R53
R48
R54
R49
R51
R47
R52
470
YEL
RED
P79
10K
R55
R56
Q3
Q4
BC337
JP2
P7
P6
P5
R59
R58
R57
GRN
YLW
JP3
NASLOV Japanac
VER 1
DATUM Thu Apr 29, 2004
AVTOR Dusan SLAVINEC
DATOTEKA Japanac_ shema_ T1.SCH
List 5 od 6


A
B
C
D
1
2
3
4
+5V
KEY_1
R1
10k
P40
KEY_2
R2
10k
P39
KEY_3
R3
10k
P41
KEY_4
R4
10k
P44
KEY_5
R5
10k
P43
KEY_6
R6
10k
P46
KEY_8
R8
10k
P45
KEY_7
R7
10k
P48
KEY_A
R9
10k
P31
KEY_B
R10
10k
P32
KEY_C
R11
10k
P34
KEY_D
R12
10k
P33
KEY_X
R13
10k
P36
KEY_Y
R14
10k
P35
KEY_Z
R15
10k
P37
DIPSW_1:1
R17
10k
P23
DIPSW_1:2
R18
10k
P24
DIPSW_1:3
R19
10k
P25
DIPSW_1:4
R20
10k
P26
S1_S2
P76
P77
R16
10k
KEY_GSR
P74
NASLOV Japanac
VER 1
DATUM Thu Apr 29, 2004
AVTOR Dusan SLAVINEC
DATOTEKA Japanac_shema_T1.SCH
List 4 od 6

```
1    MODULE JAPANAC1
2
3    "-INPUTS ------------------------------ INPUT SETS ------------------------
4      ![SWITCH_DIP_1..SWITCH_DIP_4] PIN;
5      ![KEY_DATA_1..KEY_DATA_8]     PIN;    KEYS_DATA = [KEY_DATA_1..KEY_DATA_8];
6      !KEY_DOWN           PIN;
7      !KEY_UP             PIN;
8      !KEY_SHIFT          PIN;
9
10
11     !KEY_STOP_PLAY      PIN;
12     !KEY_PATN_SONG      PIN;
13     !KEY_ADRS_INST      PIN;
14
15     !KEY_RESET_ADRS     PIN;
16
17
18     CLK_SYS             PIN;
19     CLK_TEMPO           PIN;
20
21   "- NODES ---------------------------------- NODE SET ---------------------
22     KEY_DATA_PRESSED      NODE ISTYPE 'REG';
23     KEY_ADRS_PRESSED      NODE ISTYPE 'REG';
24     KEY_MODE_PRESSED      NODE ISTYPE 'REG';
25     INT_LED1..INT_LED8    NODE ISTYPE 'COM';  A_LOW_LED = [INT_LED1..INT_LED8];
26
27     MODE_STOP_PLAY        NODE ISTYPE 'REG';
28     MODE_PATN_SONG        NODE ISTYPE 'REG';
29     MODE_ADRS_INST        NODE ISTYPE 'REG';
30
31     D3..D0                NODE ISTYPE 'COM';  INT_DISPLAY_7SEG=[D3,D2,D1,D0];
32     PRECOD11..PRECOD0     NODE ISTYPE 'COM';
33
34   "- OUTPUTS ------------------------------ OUTPUT SETS -------------------
35     ADR10                 PIN ISTYPE 'COM';
36     A9..A0                PIN ISTYPE 'REG'; ADDRES = [A9..A0];
37     PTN_DATA1..PTN_DATA8  PIN ISTYPE 'REG'; PATTERN_DATA=[PTN_DATA1..PTN_DATA8];
38     !READ                 PIN ISTYPE 'COM';
39     !WRITE                PIN ISTYPE 'COM';
40     !MEMORY_ENABLE        PIN ISTYPE 'COM';
41
42     LED1..LED8            PIN ISTYPE 'COM'; LEDS = [LED1..LED8];
43     LED_ROW_DSC           PIN ISTYPE 'COM';
44     RED_LED_ON            PIN ISTYPE 'COM';
45
46     LED_MODE_A            PIN ISTYPE 'COM';
47     LED_MODE_B            PIN ISTYPE 'COM';
48     LED_MODE_C            PIN ISTYPE 'COM';
49
50     OUT_7SEG_1..OUT_7SEG_7 PIN ISTYPE 'COM';OUT_DISPLAY_7SEGMENT =
51                                              [OUT_7SEG_1..OUT_7SEG_7];
52     OUT_7SEG_DSC          PIN ISTYPE 'COM';
53     OUT_7SEG_DOT          PIN ISTYPE 'COM';
54
55   "- SETS ------------------------------------------------------------------
56     A_COUNT   = [A9..A4];
57     A_SHIFT   =  A3;
58     A_LOW     = [A2..A0];
59     A_COMPLETE  = [A9..A0];
60     A_PATTERN = [A3..A0];
61
62     MODE_KEY_DOWN  = [KEY_STOP_PLAY, KEY_PATN_SONG, KEY_ADRS_INST];
63
64     PRECODER_AC=[PRECOD11..PRECOD8];
65     PRECODER_AB=[PRECOD7..PRECOD4];
66     PRECODER_AA=[PRECOD3..PRECOD0];
67
68     BCD_DATA=[0,0,A_COUNT];
69
70   "- CONSTANTS ----------------------------------------
71     STOP, PATN, ADRS, DSBL = 0,0,0,0;
72     PLAY, SONG, INST, ENBL = 1,1,1,1;
73     STOP_PLAY = [1,0,0];
74     PATN_SONG = [0,1,0];
75     ADRS_INST = [0,0,1];
76
77
78
79
```

```
"-------------------------------------------------------------------------
EQUATIONS
"%% CLOCK ASSIGMENTS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


  [KEY_DATA_PRESSED,KEY_ADRS_PRESSED,KEY_MODE_PRESSED].CLK = CLK_TEMPO;
  [ADDRES, PATTERN_DATA].CLK = CLK_TEMPO;
  [MODE_STOP_PLAY, MODE_PATN_SONG, MODE_ADRS_INST].CLK = CLK_TEMPO;

"%%
  KEY_DATA_PRESSED := KEY_DATA_1 # KEY_DATA_2 # KEY_DATA_3 # KEY_DATA_4
              # KEY_DATA_5 # KEY_DATA_6 # KEY_DATA_7 # KEY_DATA_8;
  KEY_ADRS_PRESSED := KEY_DOWN # KEY_UP # KEY_SHIFT;
  KEY_MODE_PRESSED := KEY_STOP_PLAY # KEY_PATN_SONG # KEY_ADRS_INST;


"%% MODE REGISTERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  WHEN ((MODE_KEY_DOWN == STOP_PLAY) & !KEY_MODE_PRESSED)
  THEN  MODE_STOP_PLAY := !MODE_STOP_PLAY.FB
  ELSE  MODE_STOP_PLAY :=  MODE_STOP_PLAY.FB;

  WHEN ((MODE_KEY_DOWN == PATN_SONG) & !KEY_MODE_PRESSED)
  THEN MODE_PATN_SONG := !MODE_PATN_SONG.FB
  ELSE MODE_PATN_SONG :=  MODE_PATN_SONG.FB;

  WHEN ((MODE_KEY_DOWN == ADRS_INST) & !KEY_MODE_PRESSED)
  THEN MODE_ADRS_INST := !MODE_ADRS_INST.FB
  ELSE MODE_ADRS_INST :=  MODE_ADRS_INST.FB;

"%%% EDIT REGISTER AND ADDRESS REGISTER %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  ADR10 = SWITCH_DIP_4;
  MEMORY_ENABLE = ENBL;
  RED_LED_ON = 0;

// enacbe za STOP nacin delovanja
  WHEN (MODE_STOP_PLAY == STOP) THEN
  {
    A_COUNT.AR = KEY_RESET_ADRS & (MODE_PATN_SONG == SONG);
    A_PATTERN.AR = KEY_RESET_ADRS;

    WHEN (MODE_ADRS_INST == ADRS) THEN
    {
      PATTERN_DATA := ( (KEY_ADRS_PRESSED # KEY_DATA_PRESSED )
                        & PATTERN_DATA.PIN) #
                      (!(KEY_ADRS_PRESSED # KEY_DATA_PRESSED )
                        & PATTERN_DATA.FB);
      READ = (KEY_ADRS_PRESSED # KEY_DATA_PRESSED );

           WHEN (KEYS_DATA == ^B10000000) THEN A_LOW := 0
      ELSE WHEN (KEYS_DATA == ^B01000000) THEN A_LOW := 1
      ELSE WHEN (KEYS_DATA == ^B00100000) THEN A_LOW := 2
      ELSE WHEN (KEYS_DATA == ^B00010000) THEN A_LOW := 3
      ELSE WHEN (KEYS_DATA == ^B00001000) THEN A_LOW := 4
      ELSE WHEN (KEYS_DATA == ^B00000100) THEN A_LOW := 5
      ELSE WHEN (KEYS_DATA == ^B00000010) THEN A_LOW := 6
      ELSE WHEN (KEYS_DATA == ^B00000001) THEN A_LOW := 7
      ELSE                                     A_LOW := A_LOW.FB;

      WHEN (KEY_UP & !KEY_DOWN & !KEY_ADRS_PRESSED)
        THEN A_COUNT := A_COUNT.FB+1
      ELSE WHEN (!KEY_UP & KEY_DOWN & !KEY_ADRS_PRESSED)
        THEN A_COUNT := A_COUNT.FB-1
      ELSE   A_COUNT := A_COUNT.FB;

      WHEN (KEY_SHIFT & !KEY_ADRS_PRESSED) THEN A_SHIFT := !A_SHIFT.FB
      ELSE                          A_SHIFT :=  A_SHIFT.FB;

    }
    ELSE WHEN (MODE_ADRS_INST == INST) THEN
    {
      [ WRITE , PATTERN_DATA.OE ]= KEY_DATA_PRESSED;
      READ = KEY_ADRS_PRESSED & !KEY_DATA_PRESSED ;

      PATTERN_DATA := ((( !KEY_DATA_PRESSED & !KEY_ADRS_PRESSED)
                      & (PATTERN_DATA.FB $ KEYS_DATA))
                    #   ( KEY_DATA_PRESSED & !KEY_ADRS_PRESSED
                      & PATTERN_DATA.FB)
                    #    (!KEY_DATA_PRESSED &  KEY_ADRS_PRESSED
                      & PATTERN_DATA.PIN));
```

```

      WHEN (MODE_PATN_SONG == PATN) THEN
      {
        WHEN      (KEY_UP & !KEY_DOWN & !KEY_ADRS_PRESSED)
          THEN A_PATTERN := A_PATTERN.FB+1
        ELSE WHEN (!KEY_UP & KEY_DOWN & !KEY_ADRS_PRESSED)
          THEN A_PATTERN := A_PATTERN.FB-1
        ELSE   A_PATTERN := A_PATTERN.FB;
        A_COUNT := A_COUNT.FB;
      }
      ELSE WHEN (MODE_PATN_SONG == SONG) THEN
      {
        WHEN      (KEY_UP & !KEY_DOWN & !KEY_ADRS_PRESSED)
          THEN A_COMPLETE := A_COMPLETE.FB+1
        ELSE WHEN (!KEY_UP & KEY_DOWN & !KEY_ADRS_PRESSED)
          THEN A_COMPLETE := A_COMPLETE.FB-1
        ELSE   A_COMPLETE := A_COMPLETE.FB;
      }
    }
  }
// enacbe za PLAY nacin delovanja
  ELSE WHEN (MODE_STOP_PLAY == PLAY) THEN
  {
    WHEN SWITCH_DIP_1==ENBL THEN
    {
      READ = DSBL; PATTERN_DATA.OE = ENBL;
      PATTERN_DATA := [0,0,0,0,0,0,0,0];
      WRITE = CLK_SYS;
    }
    ELSE
    {
      READ = !CLK_TEMPO;
      PATTERN_DATA.OE = DSBL;
      PATTERN_DATA := PATTERN_DATA.PIN;
    }

    WHEN (MODE_PATN_SONG == PATN) THEN
    {
      [A_SHIFT,A_LOW] := [A_SHIFT,A_LOW].FB +1;
      A_COUNT := A_COUNT.FB;
    }
    ELSE WHEN (MODE_PATN_SONG == SONG) THEN
      A_COMPLETE := A_COMPLETE.FB + 1;
  }

"%% LED DISPLAY %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
// prikaz podatkov na rumeni in rdeci vrstici ledic
  LED_ROW_DSC =  CLK_SYS; // ura za preklapljanje med vrsticama ledic

  LEDS = (!CLK_SYS & PATTERN_DATA.FB
          & ((MODE_STOP_PLAY==STOP) # (MODE_STOP_PLAY==PLAY) & !CLK_TEMPO))
      #  ( CLK_SYS & A_LOW_LED );

  LED_MODE_C = A3;
  LED_MODE_B = MODE_ADRS_INST;
  LED_MODE_A = MODE_PATN_SONG;

// prekodirna tabela za spodnjo-rdeco vrstico ledic, iz 3-bit v 8led
TRUTH_TABLE ( A_LOW -> A_LOW_LED)
            0 -> ^B10000000;
            1 -> ^B01000000;
            2 -> ^B00100000;
            3 -> ^B00010000;
            4 -> ^B00001000;
            5 -> ^B00000100;
            6 -> ^B00000010;
            7 -> ^B00000001;
```

```
"-------------------------------------------------------------------------
EQUATIONS
"%% 7SEGMENT DISPLAY %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  OUT_7SEG_DSC = CLK_SYS;"CLK_TEMPO;

"PREKODIRNIKI IZ 7-BIT V BCD
" 0 0 6 5 4 3 2 1 0
" | | | | | | | | |
" | [   AD   ]  | | | |
" | | | | | | | | |
" | | [   AC   ]  | | |
" | | | | | | | | |
" | | | [   AB   ]  | |
" | | | | | | | | |
" [   BA   ][   AA   ]  |
" | | | | | | | | |
" -------------------------
" [S][ DESETICE ][  ENICE  ]

"AD
" WHEN  (BCD_DATA[6..4]<5) THEN
"      PRECODER_AD=[0,BCD_DATA[6..4]]
"   ELSE
"      PRECODER_AD=([0,BCD_DATA[6..4]]+3);
"AC
" WHEN  ([PRECODER_AD[2..0],BCD_DATA[3]]<5) THEN
"      PRECODER_AC=[PRECODER_AD[2..0],BCD_DATA[3]]
"   ELSE
"      PRECODER_AC=([PRECODER_AD[2..0],BCD_DATA[3]]+3);

"AC
  WHEN  (BCD_DATA[5..3]<5) THEN
      PRECODER_AC=[0,BCD_DATA[5..3]]
    ELSE
      PRECODER_AC=([0,BCD_DATA[5..3]]+3);
"AB
  WHEN  ([PRECODER_AC[2..0],BCD_DATA[2]]<5) THEN
      PRECODER_AB=[PRECODER_AC[2..0],BCD_DATA[2]]
    ELSE
      PRECODER_AB=([PRECODER_AC[2..0],BCD_DATA[2]]+3);
"AA
  WHEN  ([PRECODER_AB[2..0],BCD_DATA[1]]<5) THEN
      PRECODER_AA=[PRECODER_AB[2..0],BCD_DATA[1]]
    ELSE
      PRECODER_AA=([PRECODER_AB[2..0],BCD_DATA[1]]+3);
"BA
" WHEN  ([0,PRECODER_AD[3],PRECODER_AC[3],PRECODER_AB[3]]<5) THEN
"      PRECODER_BA=[0,PRECODER_AD[3],PRECODER_AC[3],PRECODER_AB[3]]
"   ELSE
"      PRECODER_BA=([0,PRECODER_AD[3],PRECODER_AC[3],PRECODER_AB[3]]+3);

"----------
// izmenicni prikaz desetic in enic
  INT_DISPLAY_7SEG = !(CLK_SYS)
                   & ([0, PRECODER_AC[3], PRECODER_AB[3], PRECODER_AA[3]]) #
                     (CLK_SYS)
                   & ([PRECODER_AA[2..0], BCD_DATA[0]]);

// PREKODIRNA TABELA IZ bin-BCD V 7seg ZA 7-SEGMENTNA DISPLEJA
TRUTH_TABLE( INT_DISPLAY_7SEG -> OUT_DISPLAY_7SEGMENT)
                   0      ->  ^B1111110;"0
                   1      ->  ^B0110000;"1
                   2      ->  ^B1101101;"2
                   3      ->  ^B1111001;"3
                   4      ->  ^B0110011;"4
                   5      ->  ^B1011011;"5
                   6      ->  ^B1011111;"6
                   7      ->  ^B1110000;"7
                   8      ->  ^B1111111;"8
                   9      ->  ^B1111011;"9
//                  10      ->  ^B1110111;"A
//                  11      ->  ^B0011111;"b
//                  12      ->  ^B1001110;"C
//                  13      ->  ^B0111101;"d
//                  14      ->  ^B1001111;"E
//                  15      ->  ^B1000111;"F


END
```