TREBAR A.

This boots drive 0 only.

~~777~~ 172004 :

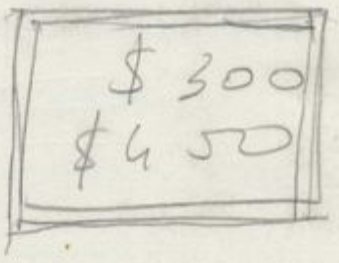
```
1000/012701     MOV  #176500,R1
1002/176500
1004/012702     MOV  #176504,R2
1006/176504
1010/010100     MOV  R1,R0
1012/005212     INC  (R2)
1014/105712 C:  TSTB (R2)
1016/100376     BPL  C
1020/006300     ASL  R0
1022/001004     BNE  B
1024/005012     CLR  (R2)
1026/005200     INC  R0
1030/005761     TST  2(R1)
1032/000002
1034/042700 B:  BIC  #20,R0
1036/000020
1040/010062     MOV  R0,2(R2)
1042/000002
1044/001363     BNE  C
1046/005003     CLR  R3
1050/105711 D:  TSTB (R1)
1052/100376     BPL  D
1054/116123     MOVB 2(R1),(R3)+
1056/000002
1060/022703     CMP  #1000,R3
1062/001000
1064/101371     BHI  D
1066/005007     CLR  PC         (HALT)
```

F650 | F1746 F611 | F2806

5. 6. | 7 [910] 11 12 13 14

1300
PREVENTIVE MAINTENANCE

Y2.47 = 1f
2.27
2.17



M 3000 ⊔ 04 4⊔ BE75F

TERMINAL

M 3000 ␣ 04 ␣␣ BE75F

CIKLIČNE KODE

$$V = (\underbrace{c_1, c_2, \dots, c_k}_{\text{kontrolni biti}}, \underbrace{a_1, a_2, \dots a_m}_{\text{informacijski biti}})$$

Informacija izražena v polinomski obliki

$a_1, a_2, a_3, \dots, a_m$ so informacijski biti

$$P(x) = a_1 + a_2 x + \dots + a_m x^{m-1} \quad (1)$$

Opomba: Čeprav moderni kodeki prenašajo 9 bitne podatke vzporedno si predstavljamo kot da se prenašajo serijsko.

GENERATORSKI POLINOM

karakterizira ciklično kodo in določa zmožnost ciklične kode za detektiranje in korigiranje napake.

Predpostavimo da je podatkovni polinom $P(x)$ pred kodiranjem

$$P(x) = a_1 + a_2 x + \dots + a_m x^{m-1} \quad (2)$$

Generatorski polinom $G(x)$ je

$$G(x) = 1 + b_1 x + \dots + b_{k-1} x^{k-1} + x^k \quad (3)$$

Rezidualni polinom $R(x)$ dobimo z deljenjem produkta $P(x) \cdot x^k$ z $G(x)$

$$x^k \cdot P(x) = Q(x) \cdot G(x) + R(x) \quad (4)$$

$$R(x) = c_1 + c_2 x + \dots + c_k x^{k-1} \quad (5)$$

$Q(x)$ je kvocientni polinom.

Reziduali polinom je kontrolni polinom in $c_1, c_2, c_3 \dots c_k$ so kontrolni biti. Kodiran podatkovni polinom $F(x)$ je izražen kot

$$F(x) = x^k \cdot P(x) + R(x) \quad (6)$$

DETEKCIJA NAPAK

Glede na enačbe (4) in (5) postane kodiran podatkovni polinom $F(x)$

$$\begin{aligned} F(x) &= x^k \cdot P(x) + R(x) \\ &= Q(x) \cdot G(x) + R(x) + R(x) \pmod 2 \\ &= Q(x) \cdot G(x) \end{aligned} \quad (7)$$

Enačba (7) dokazuje, da je podatkovni polinom $F(x)$ deljiv z generatorskim polinomom $G(x)$.

Polinom napake $E(x)$ predstavlja vpliv šuma na (sprejem) podatkov.
Če je sprejet podatkovni polinom $F'(x)$ potem je relacija med $F(x)$, $F'(x)$ in $E(x)$ naslednja:

$$F'(x) = F(x) + E(x) \quad (8)$$

$F'(x)$ se deli z $G(x)$ na sprejemni strani. Če ni napake (to je če je $E(x)=0$) potem enačba (8) postane:

$$F'(x) = F(x).$$

Z drugimi besedami deljenje se lahko izvede brez ostanka.
Ostanek $R'(x)$ indicira, da $F'(x)$ vsebuje napako. Relacija je naslednja:

$$F'(x) = G(x) \cdot Q'(x) + R'(x) \quad (9)$$

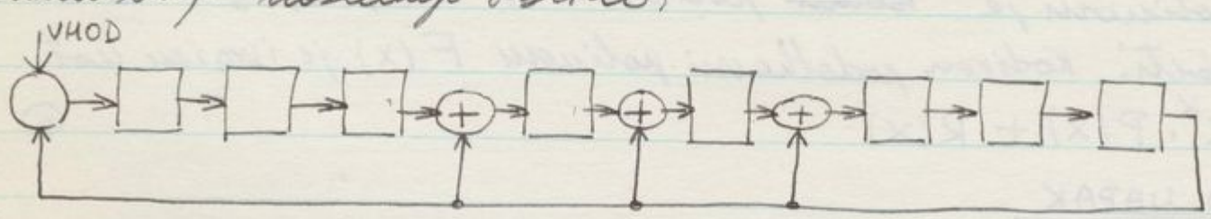
$R'(x)$ je ostanek, ki ga dobimo z deljenjem $F'(x)$ z $G(x)$. To isti ostanek lahko dobimo tudi z deljenjem $F(x)+E(x)$ z $G(x)$

$$E(x) = G(x) \cdot Q''(x) + R'(x) \quad (10)$$

$R'(x)$ je SINDROM in po uporabljamo za iskanje napačnih bitov.

Uporaba ciklični kod pri magnetnih trakovih

Kontrolni znaki ki bazirajo na principu cikličnih kod se uporabljajo kot CRC znaki na magnetnem traku za 800 bpi načinu kot ECC - CRC znaki in pomožni CRC znaki pri 6250 bpi načinu.
Če je generatorski polinom $G(x) = 1 + x^3 + x^4 + x^5 + x^8$ ima delilno vezje naslednjo obliko:



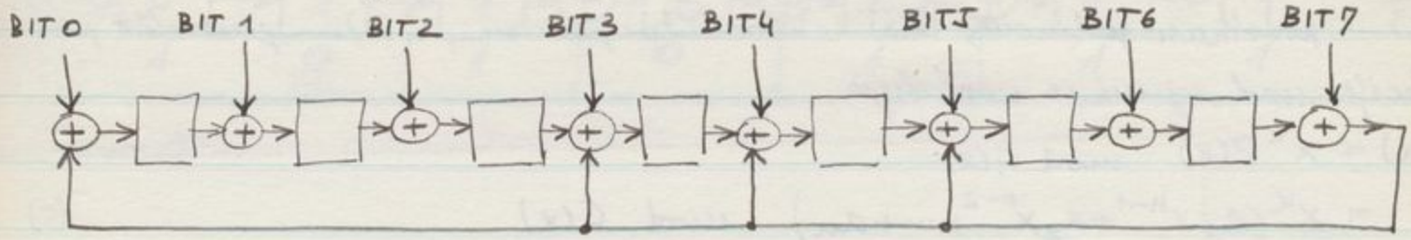
Če je n bitov informacije, ki vstopa v delilnik $a_1, a_2, \ldots, a_n$ lahko to izrazimo s polinomom.

$$P(x) = a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_n \quad (11)$$

~~kot n naki podatki vstopajo po:~~

Organizacija delilnega vezja, kjer podatki vstopajo v bytih, kot

je to v primeru magnetnih trakov vidimo na naslednji sliki:



Ta primer kaže, da 8 bitov vstopa hkrati. Polinom $P'(x)$ ki pripada podatkovnemu polinomu z n-bytom vhodnih podatkov je izražen z:

$$P'(x) = D_1(x)\, x^{n-1} + D_2(x) \cdot x^{n-2} + \cdots + D_n(x)$$

$D_K(x)$ .... je 8 bitni vhodni podatek za byte K, kjer je $1 < k < n$

biti 0 ~ 7 pripadajo koeficientom $x^0 \sim x^7$ za vsak posamezen $D_K(x)$

Napimer podatkovni polinom, ki ima 5 bytov podatkov dobimo na sledeč način:

ST. BITA / BYTE	0 1 2 3 4 5 6 7	POLINOM
BYTE 1	1 0 0 1 1 0 0 1	$D_1(x) = 1 + x^3 + x^4 + x^7$
BYTE 2	0 0 1 0 1 1 1 0	$D_2(x) = x^2 + x^4 + x^5 + x^6$
BYTE 3	1 1 1 1 0 0 0 1	$D_3(x) = 1 + x + x^2 + x^3 + x^7$
BYTE 4	0 0 0 1 1 0 0 0	$D_4(x) = x^3 + x^4$
BYTE 5	1 0 1 1 0 1 0 1	$D_5(x) = 1 + x^2 + x^3 + x^5 + x^7$

Podatkovni polinom $P'(x)$ se izračuna na naslednji način:

$$\begin{aligned} P'(x) &= D_1(x)x^4 + D_2(x)x^3 + D_3(x)x^2 + D_4(x)x + D_5(x) = \\ &= (1 + x^3 + x^4 + x^7)x^4 + (x^2 + x^4 + x^5 + x^6)x^3 + \\ &+ (1 + x + x^2 + x^3 + x^7)x^2 + (x^3 + x^4)x + \\ &+ (1 + x^2 + x^3 + x^5 + x^7) = \\ &= x^{11} + x^9 + x^8 + x^7 + x^5 + 1 \end{aligned} \qquad (3)$$

Ostanek, ki ga dobimo po deljenju $P'(x)$ z generatorskim polinomom $G(x)$ je kontrolni znak.

Kontrolne znake dobimo v obliki matričnih izrazov. Predpostavimo da so podatkovni znaki $D_1, D_2, \ldots, D_n$ in da je kontrolni znak C. Vsak znak je izražen kot polinom in vsak bit v znaku ustreza koeficientu polinoma.

Kontrolno kodo $R(x)$ za bit K v nizu bitov dobimo na sledeč način:

$$x^k P(x) = Q(x) G(x) + R(x) \qquad (4)$$

$P(x)$ ... podatkovni polinom, $Q(x)$ .... kvocientni polinom, $G(x)$ --- generator polino

Relacija med njimi je naslednja:

$$R(x) = x^k P(x) \mod G(x)$$
$$= x^k (a_1 x^{n-1} + a_2 x^{n-2} + \cdots + a_n) \mod G(x) \qquad (5)$$

To se dogaja pri paralelnem procesiranju naslednega byta. Ci vstavimo C za kontrolno ~~znak~~ kodo in 1 za kontrolni znak ($k=1$ ker ima kontrolni znak samo en byte) in podatkovni znaki $D_1, D_2, \ldots, D_n$ za informacijske bite

$$C = X (x^{n-1} D_1 + x^{n-2} D_2 + \cdots + D_n) \mod G(x)$$
$$= (x^n D_1 + x^{n-1} D_2 + \cdots + X D_n) \mod G(x) \qquad (6)$$

To ponazarja 1 bitno ciklično operacijo delilnega vezja. Ostanek v shift registru se množi z X in deli z generatorskim polinomom. Ostanek vstane v shift registru. Operacijo lahko izrazimo na sledeč način:

$$A = TB \qquad (7)$$

B in A je vrednost shift registra: B pred shiftanjem, A po shiftanju.

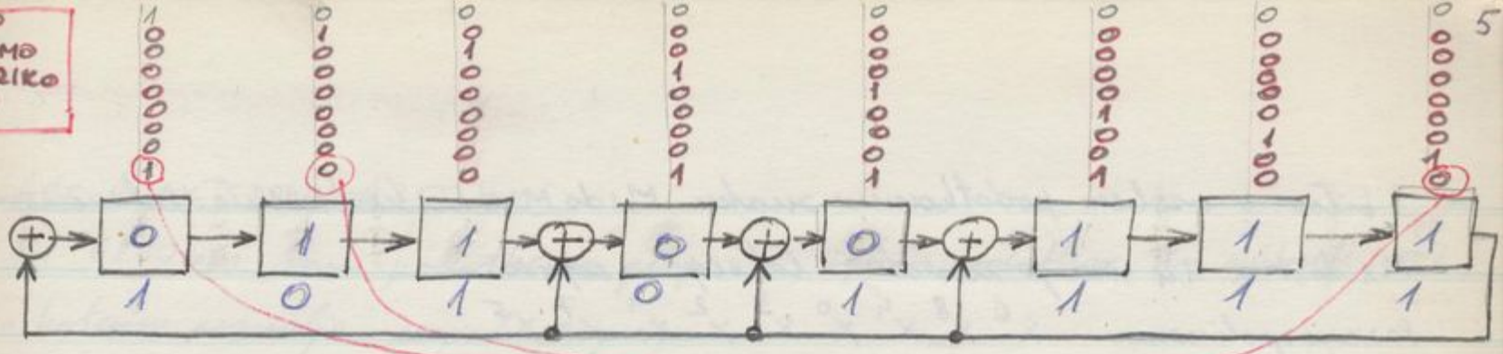
T je matrika generatorskega polinoma in se imenuje:

asociated matrix ali timing matrix.

Napimer za generatorski polinom $G(x) = 1 + x^3 + x^4 + x^5 + x^8$ ima

$$T = \begin{bmatrix} 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \end{bmatrix} \qquad (8)$$

KAKO DOBIMO MATRIKO T



Predpostavimo, da register vsebuje 01100111

$$A = TB = \begin{bmatrix} 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \qquad (9)$$

Rezultat enobitnega shifta je 10101111.

Če uporabljamo to metodo potem postane enačba (6):

$$C = T^n D_1 + T^{n-1} D_2 + \cdots + T D_n \qquad (10)$$

C predstavlja ciklični kontrolni znak in $T^j D_K$ dobimo s shiftanjem podatkovnega znaka K, j krat. D_K je 1 byte katerega matrika je:

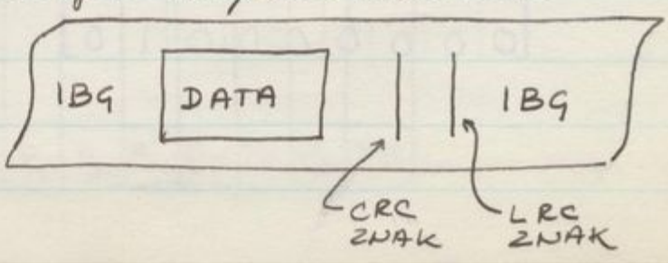
$$D_K = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ \vdots \\ d_n \end{bmatrix} \qquad \begin{array}{l} d_1 \text{ ustreza koeficientu } x^0 \\ d_2 \text{ ——— // ——— } x^1 \\ \vdots \\ d_n \text{ ——— // ——— } x^{n-1} \end{array} \qquad (11)$$

V enačbi (12) dobimo ciklični shift s tem ko pristevamo podatke ostanku, nov ostanek dobimo na sledeč način:

$$C = T(T \cdots (TD_1 + D_2) D_3) + \cdots + D_{n-1}) + D_n \qquad (12)$$

PRIMER UPORABE CIKLIČNE KODE

Pri 8006pi CRC znak detektira in korigira napake. Znak se zapiše v 4-to kolono za podatkovnimi znaki.



- 9 bitov v vrstnem podotkanem znaku M_1 do M_n so koeficienti polinomov M_1 do M_n in imajo naslednjo lokacijo v zapisu:

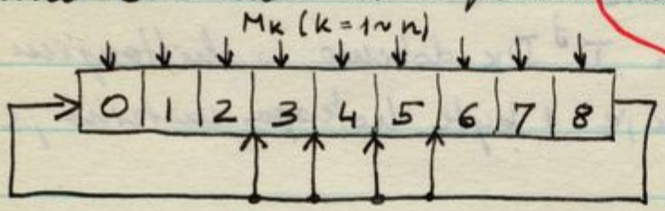
Pozicija polinoma	x^6	x^8	x^4	x^0	x^3	x^2	x^1	x^7	x^5
Številka sledi	1	2	3	4	5	6	7	8	9

- 9 bitov v CRC znaku C, predstavlja koeficiente polinoma C z isto oznako sledi.

- C se izračuna iz podotkanega polinoma M_1 do M_n z uporabo generatorskega polinoma $G(x) = 1+x^3+x^4+x^5+x^6+x^9$ sledi na naslednjo relacijo

$$C = (x^n M_1 + X^{n-1} M_2 + \cdots + X^2 M_{n-1} + X M_n) +$$
$$+ (1+X+x^2+x^4+x^6+x^7+x^3) \text{ Mod } G(x).$$

Rezultat te enačbe je CRC znak. Če je število podotkanih znakov vodo imna CRC znak liho paritetno in obratno.



- Tej enačbi pripada naslednja matrika:

$$C = T^n M_1 + T^{n-1} M_2 + \cdots + T^2 M_{n-1} + T M_n + K$$

$$T = \begin{bmatrix} 0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0&0&1 \\ 0&0&0&0&1&0&0&0&1 \\ 0&0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0 \end{bmatrix}, \quad K = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

ECC ZNAK PRI 6250 bpi

Pri 6250 bpi ima podatkovna grupa 7 bytov podatkov in en byte ECC s katerim popravlja napake v grupi.

- 8 bitov v vrstem znaku D1 do D7 so koeficienti polinoma D1 do D7 in zaradi tega sledijo naslednji sledi na traku:

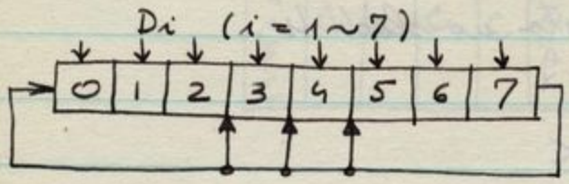
Pozicija v polinomu	x^1	x^4	x^7	–	x^3	x^6	x^0	x^2	x^5
Številka sledi	1	2	3	4	5	6	7	8	9

Sled 4 vsebuje paritetni bit P vsakega po vrsti ECC znaka.

- 8 bitov v ECC znaku E so koeficienti v polinomu E in ~~imajo~~ zasedajo iste sledi. Sled 4 nosi liho pariteto za ECC znake.

- E se izračuna iz podatkovnega polinoma D1 do D7 z uporabo generatorskega polinoma $G(x) = 1 + x^3 + x^4 + x^5 + x^8$ glede na naslednjo relacijo:

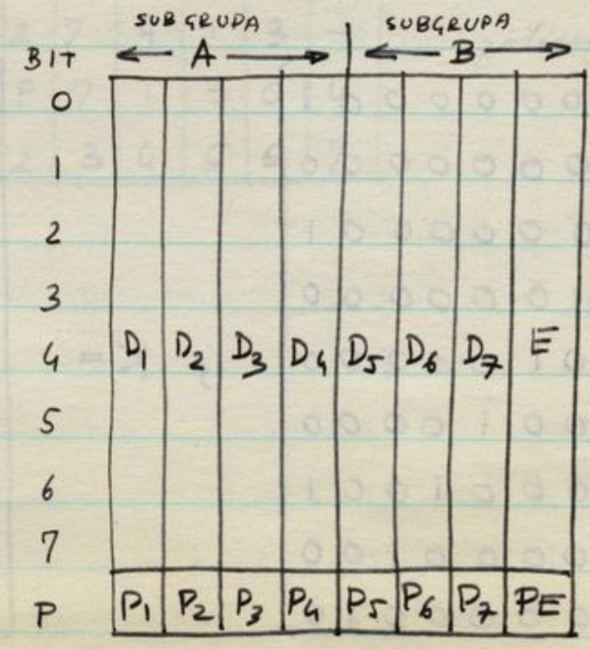
$$E = (x^7 D_1 + x^6 D_2 + x^5 D_3 + x^4 D_4 + x^3 D_5 + x^2 D_6 + x D_7) \bmod G(x)$$



- Tej enačbi pripada naslednja matrika:

$$E = T^7 D_1 + T^6 D_2 + T^5 D_3 + D^4 D_4 + T^3 D_5 + T^2 D_6 + T D_7$$

$$T = \begin{vmatrix} 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \end{vmatrix}$$



POMOŽNI CRC ZNAK

Pri 6250 bpi je na pozicijo 7 vstavljen 9 biti pomožni CRC znak za residualno podatkovno grupo v vsakem bloku in služi za detektiranje napak.

- 9 bitov v vsakem podatkovnem znaku M1 do M_n so koeficienti polinoma M1 do M_n in imajo naslednje pozicije:

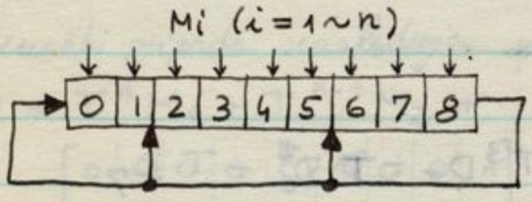
Pozicija v polinomu	X^0	X^4	X^6	X^3	X^1	X^5	X^7	X^2	X^8
Štev. sledi	1	2	3	4	5	6	7	8	9

- 9 bitov v pomožnem CRC znaku N so koeficienti polinoma N ~~v zadnjem znaku~~ in pripadajo sledi na traku.
- N se izračuna iz podatkovnega polinoma M1 do M_n z uporabo generatorskega polinoma $G(X) = 1 + X + X^6 + X^7 + X^8$ glede na naslednjo relacijo.

$$N = (X^n M_1 + X^{n-1} M_2 + \cdots + X M_n) + (1 + X + X^6 + X^7 + X^8) \quad \text{Mod } G(X)$$

- Pomožni CRC znak ima liho pariteto. Če ima N sodo pariteto, se postavi v liho pariteto z invertiranjem bita na sledi 4
- Matrična enačba je sledeča:

$$N = T^n M_1 + T^{n-1} M_2 + \cdots + T M_n + K$$



$$T = \begin{bmatrix} 0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&1 \\ 0&0&1&0&0&0&0&0&0 \\ 0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&1&0 \end{bmatrix}, \quad K = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

CRC ZNAK PRI 6250 bpi

CRC znak je zapisan 5 do 6 krat na koncu vsakega bloka za detekcijo napak

- CRC znak ima 9 bitov in je formiran iz naslednjih podatkov:
  * VSI PODATKOVNI ZNAKI
  * PADDING ZNAK (residualna podatkovna grupa in CRC podatkovna grupa)
  * POMOŽNI CRC znaki (brez ECC znakov)
- Koeficienti v podatkovnem polinomu, razporejeni na traku in generatorski polinom bi se uporabljali za formiranje CRC znaka pri 6250 bpi so identični tistim za 800 bpi.
- CRC znak ima liho pariteto. Če je vsota podatkovnih znakov, padding znaka v residualni podatkovni grupi in pomožnega CRC znaka liha se zahteva padding znak s samimi 0 biti v prvi byte CRC podatkovne grupe. Če je vsota soda se zapiše CRC znak. Na to način moremo do za izračunani znaki liki. Identični CRC znaki se zahtevajo od 5–6 byta.

CRC podatkovna grupa

PADDING OR CRC					RESIDUALNI ZNAK	
C	C	C	C	C	E	
R	R	R	R	R	C	
C	C	C	C	C	C	

	X^0	X^1	X^2	X^3	X^4	X^5	X^6	X^7	X^8
800 bpi CRC	P	0	1	2	3	4	5	6	7
6250 bpi ECC	0	5	6	2	7	4	1	3	—
—"— Aux. CRC	5	2	6	P	7	1	3	0	4
—"— CRC	P	0	1	2	3	4	5	6	7

Relacijo med pozicijo bita in stopnjo v polinomu

KOREKCIJA NAPAK PRI 800 Bpi

Pri 9 sledvem 800bpi načinu, lahko detektiramo napake na eni sledi s kombinacijo ciklične kode in VRC funkcije.

PRINCIP KOREKCIJE NAPAK

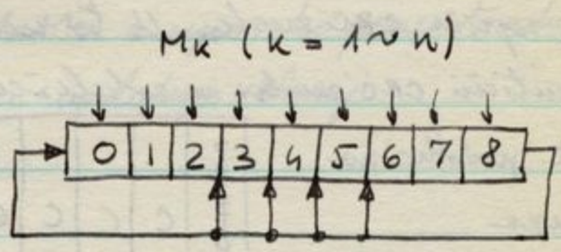
Polinom C za CRC znak se izračuna iz naslednje enačbe či so koeficienti podatkovnega polinoma M_1 do M_n med write operacijo:

$$C = T^n M_1 + T^{n-1} M_2 + \cdots + T M_n + K \qquad (1)$$

T je matrika, ki pripada generatorskemu polinomu

$G(x) = 1 + x^3 + x^4 + x^5 + x^6 + x^9$ in K je konstanta

$$T = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&1 \\ 0&0&0&1&0&0&0&0&0&1 \\ 0&0&0&0&1&0&0&0&0&1 \\ 0&0&0&0&0&1&0&0&1&1 \\ 0&0&0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \end{bmatrix}, \quad K = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$



CRC znak se določi ko se prečitajo podatki (n+1). Či ni nobene napake pri čitanju, postane enačba 1 sledeča:

$$K = T^n M_1 + T^{n-1} M_2 + \cdots + T M_n + C \qquad (2)$$

Či ni nobene napake potem v CRC registru ostane vrednost

1 1 1 0 1 0 1 1 1.

Predpostavimo da je v podatkih napaka M'_1 do M'_n in C'. Vsebina CRC registra (S_1) je sledeča:

$$S_1 = T^n M'_1 + T^{n-1} M'_2 + \cdots + T M'_n + C' \qquad (3)$$

Če je $S_1 = K$ smatramo da ni bilo napake.

Če dobimo napako zaradi šuma E_1 do E_n in E_c postane enačba 1:

$$M'_k = M_k + E_k \quad (k = 1 \sim n \text{ in } C; \text{ v primeru c, } C' = C + E_c) \qquad (4)$$

Enačba 3 postane:

$$S_1 = T^n M_1' + T^{n-1} M_2' + \cdots + T M_n' + C' =$$
$$= T^n (M_1 + E_1) + T^{n-1}(M_2 + E_2) + \cdots + T(M_n + E_n) + (C + E_C) =$$
$$= T^n M_1 + T^{n-1} M_2 + \cdots + T M_n + C +$$
$$+ T^n E_1 + T^{n-1} E_2 + \cdots + T E_n + E_C \qquad (5)$$

Glede na enačbo (2), lahko modificirano enačbo (5) na sledeč način:

$$S_1 + K = T^n E_1 + T^{n-1} E_2 + \cdots + T E_n + E_C \qquad (6)$$

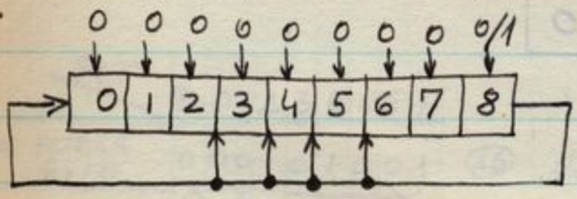
Če je napaka le na eni sledi (naprimer na sledi bi ustreza stopnji x^2, E_K ($K = 1 \sim n, c$) lahko to izrazimo na sledeč način:

$$E_K = \begin{bmatrix} 0 \\ 0 \\ 1/0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} \text{pozicija ustreza } x^0 \\ x^1 \\ x^2 \\ x^3 \\ x^4 \\ x^5 \\ x^6 \\ x^7 \\ x^8 \end{matrix} \qquad K = 1 \sim n, c$$

V tem primeru se VRC napaka detektira v prečitanih podatkih.

Vsak znak se posebej preveri (VRC check) ko se podatki čitajo. Če ~~ni nobene napake je vrednost~~ ali je napaka se ustavi v shift register vrednost 000000001. Struktura tega registra je ista kot je vezje za generiranje CRC. Če ni nobene napake je vrednost, ki jo vsebuje 0000 0000 To shift register se imenuje ERROR PATTERN register. Vhodni vektor v to register je P_K.

Vhod P_K ($K = 1 \sim n, c$)



EP REGISTER

$$P_K = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0/1 \end{bmatrix}$$

končni rezultat (S_2) v EP registru si lahko izrazimo sledeč način:

$$S_2 = T^n P_1 + T^{n-1} P_2 + \cdots + T P_n + P_C \qquad (7)$$

Če je VRC napaka, se v EP register ustavi 1 na mestu ki ustreza X^8. Če lahko določimo do je v izrazu X^i napaka potem vnesemo E_K namesto P_K, da dobimo $S_1 + K$ vektor iz enačbe (6) v EP registru. Ker ustavimo v EP register dejansko X^8 in ne X^i premaknemo vrednost na desno za $8-i$. Relacija med E_K in P_K je izražena:

$$T^{8-i} E_K = P_K \; (k = 1 \sim n, c) \qquad (8)$$

Iz enačbe (8), (6) in (7) dobimo:

$$\begin{aligned} S_2 &= T^n P_1 + T^{n-1} P_2 + \cdots + T P_n + P_C = \\ &= T^{8-i} (T^n E_1 + T^{n-1} E_2 + \cdots + T E_n + E_c) \\ &= T^{8-i} (S_1 + K) \end{aligned} \qquad (9)$$

Rezultat (S_2) se zadrži v EP registru, rezultat (S_1) CRC registra pa se prišteje po mod 2 z (K) 111010111. Če dobimo zaporedje po shiftanju vrednosti shift registra imamo slednjo katero je napaka v odvisnosti od št. premikov.

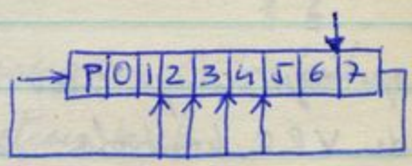
METODA ZA KOREKCIJO NAPAKE

Primer:

BIT / BYTE	P	0	1	2	3	4	5	6	7
BYTE 1	1	1	0	0	0	0	0	1	0
2	0	0	1	0	1	0	1	1	1
3	0	0	1	0	1	0	0	1	0
4	0	0	0	1	1	1	1	1	0
5	0	1	0	1	0	1	1	1	0

GENERIRANJE CRC MED OPERACIJO VPISOVANJA

BIT / BYTE	P	0	1	2	3	4	5	6	7	
	0	0	0	0	0	0	0	0	0	① začetna vrednost CRC registra
1 BYTE	1	1	0	0	0	0	0	1	0	② prvi podatkovni byte
	1	1	0	0	0	0	0	1	0	③ = ① ⊻ ②
	0	1	1	0	0	0	0	0	1	④ CRC reg se pomakne v desno za 1 bit (ROTATE POMIK glej zlika spodaj)
2. BYTE	0	0	1	0	1	0	1	1	1	⑤ drugi byte podatkov
	0	1	0	0	1	0	1	1	0	⑥ = ④ ⊻ ⑤
	0	0	1	0	0	1	0	1	1	⑦ CRC REG se pomakne za en bit v desno
3. BYTE	0	0	1	0	1	0	0	1	0	⑧ tretji byte podatkov
	0	0	0	0	1	1	0	0	1	⑨ = ⑦ ⊻ ⑧
	1	0	0	0	0	1	1	0	0	⑩ CRC REG SE POMAKNE ZA en bit v desno
			1	1	1	1				⑪ Če je P = 1, se biti 2 do 5 invertirajo (Razlog! glej zlika)
4. BYTE	0	0	0	1	1	1	1	1	0	⑫ četrti byte podatkov
	1	0	0	0	0	1	1	1	0	⑬ = ⑩ ⊻ ⑪ ⊻ ⑫
	0	1	0	0	0	0	1	1	1	⑭ pomik za en bit v desno
5. BYTE	0	1	0	1	0	1	1	1	0	⑮ peti podatkovni byte
	0	0	0	1	0	1	0	0	1	⑯ = ⑭ ⊻ ⑮
	1	0	0	0	1	0	1	0	0	⑰ pomik CRC registra za en bit v desno
			1	1	1	1				⑱ Če je P = 1, se biti 2 do 5 invertirajo
	1	1	1	0	1	0	1	1	1	⑲ konstanta ko generiramo CRC K
CRC	0	1	1	1	1	1	1	1	1	⑳ = ⑰ ⊻ ⑱ ⊻ ⑲



PROCESIRANJE MED ČITANJEM NAPREJ: CRC na tretira enako kot zgoraj, le da namesto kot podatek. Če ni napake ostane v registru vrednost 11101 111.

BIT	P	0	1	2	3	4	5	6	7	
DO PETEGA BYTA	0	0	0	1	0	1	0	0	1	⑯ Vsebina CRC registra, ko smo procesirali do petega byta.
	1	0	0	0	1	0	1	0	0	⑰ pomik za en bit v desno
			1	1	1	1				⑱ Če je P = 1 se biti 2 – 5 invertirajo
CRC BYTE	0	1	1	1	1	1	1	1	1	⑲ Preštejemo CRC byte
ZADNJA VRED. V CRC REG.	1	1	1	0	1	0	1	1	1	⑳ = ⑰ ⊻ ⑱ ⊻ ⑲

Če ni nobene napake je vsebina CRC registra vedno 11101 0 111.

Predpostavimo, da smo podatke sprejeli nepravilno naprimer bit 4 v tretjem bytu se prehlopi od 0 na 1 zarodi drop in napake. četrti bit pelepa bytov in CRC byta se prehlopi iz 1 na 0 zarodi drop out napake:

drop in napaka: če gre 0→1

drop out napaka: če gre 1→0

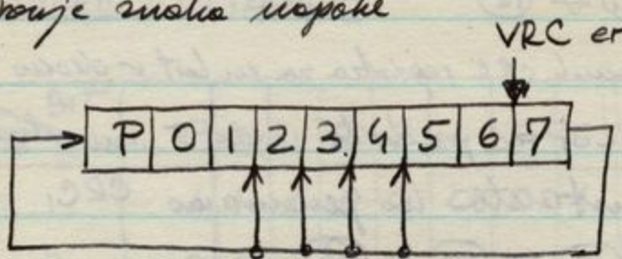
PREČITANI PODATKI	ŠT. BITA P	0	1	2	3	4	5	6	7	VCR NAPAKA
BYTE 1	1	1	0	0	0	0	0	1	0	
2	0	0	1	0	1	0	1	1	1	
3	0	0	1	0	1	(1)	0	1	0	x
4	0	0	0	1	1	1	1	1	0	
5	0	1	0	1	0	(0)	1	1	0	x
CRC byte	0	1	1	1	1	(0)	1	1	1	x

TRACK P TRACK 0 ...

CRC vrne sodo pariteto če je št podotknutih bitov liho

Pri operaciji čitanja, ne v CRC register prevzame nova vrednost istočasno pa se izvede VRC kontrola. Zaok napake (error pattern) se vpiše v EP register. Logika EP registra je podobna tisti iz CRC registra.

Vezje za generiranje zaoka napake EP register



Zadnja vrednost v CRC registru je 11011010 in ne 11010111, kar indicira napako. Zadnja vrednost v EP registru je 10111010 če je VRC napaka. To vrednost se imenuje ERROR PATTERN.

PREČITANI PODATKI	CRC REGISTER POZICIJA BITOV P	0	1	2	3	4	5	6	7	VRC NAPAKA	
	0	0	0	0	0	0	0	0	0		začetna vrednost CRC registra
BYTE 1	1	1	0	0	0	0	0	1	0		Xri prečitani byte
	1	1	0	0	0	0	0	1	0		∀
	0	1	1	0	0	0	0	0	1		shift desno za 1 bit
BYTE 2	0	0	1	0	1	0	1	1	1		drugi prečitani byte
	0	1	0	0	1	0	1	1	0		∀
	0	0	1	0	0	1	0	1	1		shift desno za 1 byte bit
BYTE 3	0	0	1	0	1	(1)	0	1	0	X	tretji prečitani byte
	0	0	0	0	1	0	0	0	1		∀
	1	0	0	0	0	1	0	0	0		Pomik desno za 1 bit
				1	1	1	1				Če je P=1 so biti 2 do 5 invertirani.
BYTE 4	0	0	0	1	1	1	1	1	0		četrti precitani byte
	1	0	0	0	0	1	0	1	0		∀
	0	1	0	0	0	0	1	0	1		shift desno za 1 by bit
BYTE 5	0	1	0	1	0	(0)	1	1	0	X	peti prečitani byte
	0	0	0	1	0	0	0	1	1		∀
	1	0	0	0	1	0	0	0	1		shift v desno za 1 bit
				1	1	1	1				Če je P=1 so invertirajo biti od 2 do 5
CRC byte	0	1	1	1	1	(0)	1	1	1	X	VRC napaka na ECC byte
	1	1	1	0	1	1	0	1	0		∀

BYTE	EP REGISTER (BIT POZICIJA) P	0	1	2	3	4	5	6	7	VRC	
	0	0	0	0	0	0	0	0	0		začetna vrednost EP registra
1									0		1. BYTE BREZ NAPAKE
	0	0	0	0	0	0	0	0	0		∀
	0	0	0	0	0	0	0	0	0		SHIFT DESNO ZA 1 BIT
2									0		2. BYTE BREZ NAPAKE
	0	0	0	0	0	0	0	0	0		∀
	0	0	0	0	0	0	0	0	0		SHIFT DESNO ZA 1 BIT
3									1	X	VCR NAPAKA 3. BYTE
	0	0	0	0	0	0	0	0	1		∀
	1	0	0	0	0	0	0	0	0		SHIFT DESNO ZA 1 BIT
				1	1	1	1				ČEJE P=1 SE INVERTIRA BITE 2 do 5
4									0		NI NAPAKE NA 4. BYTU
	1	0	0	1	1	1	1	0	0		∀
	0	1	0	0	1	1	1	1	0		EP REG SHIFT DESNO ZA 1 BIT
5									1	X	VRC NAPAKA 5. BYTE
	0	1	0	0	1	1	1	1	1		∀
	1	0	1	0	0	1	1	1	1		POMIK DESNO ZA 1 BIT
				1	1	1	1				ČE JE P=1 SE BITI 2 do 5 INVERTIRAJO
CRC									1	X	VRC NAPAKA V CRC BYTU
	1	0	1	1	1	0	0	1	0		∀ (zadnja vrednost v EP registru)

Detekcija napačne sledi (sledi z napako)

Na koteri sledi je napaka detektirana iz zadnje vrednosti CRC in EP reg. Error znak v EP registru ostane nespremenjen, vsi biti iz CRC registra razen bitov 2 in 4 pa se invertirajo. Potem primerjamo vsebino EP in CRC registrov. Če se vsebini ne ujemata se vsebina v CRC registru shifta in zopet primerja.

	P	0	1	2	3	4	5	6	7	
ZADNJA VREDNOST V EP REGISTRU	1	0	1	1	1	0	0	1	0	
ZADNJA VREDNOST V CRC REGISTRU	1	1	1	0	1	1	0	1	0	
	1	1	1		1		1	1	1	
	0	0	0	0	0	1	1	0	1	≠
POMIK V DESNO	1	0	0	0	0	0	1	1	0	
				1	1	1	1			
	1	0	0	1	1	1	0	1	0	≠
POMIK V DESNO	0	1	0	0	1	1	1	0	1	
POMIK V DESNO	1	0	1	0	0	1	1	1	0	
				1	1	1	1			
	1	0	1	1	1	0	0	1	0	=

Kontrolna pri generaciji CRC je K = 11101011

kontrolo moramo odšteti od CRC vrednosti.

Vse bite razen bitov 2 in 4 invertiramo

PRVIČ PRIMERJAMO Z EP NE SOVPADA

ČE JE P=1 SE BITI 2 do 5 invertirajo

DRUGA PRIMERJAVA SE NE UJEMA

TRETJA PRIMERJAVA SE NE UJEMA

ČE JE P=1 INVERTIRAMO BITE OD 2 do 5

ČETRTA PRIMERJAVA SE UJEMA

Operacijo lahko ponavljamo do 9 krat. Če vsebina še vedno ni ustrezna po izvajanju 9-te primerjave, pomeni, da imamo več napak. Sled na kateri je napaka se identificira iz števila primerjav.

Spodnja tabela kaže primerjavo med št sledi in št primerjav.

ŠT PRIMERJAV PREDNO JE PRIŠLO DO SOVPADANJA	1	2	3	4	5	6	7	8	9
ERROR BIT NR. PRI FORWARD READ	7	6	5	4	3	2	1	0	P
ERROR BIT NR. PRI BACKWARD READ	P	0	1	2	3	4	5	6	7

Ker se v zgornjem primeru ujemata obeh registrov v 4-ti primerjavi pomeni da je napaka na sledi št 4.

PRENOS INFORMACIJE O NAPAKI

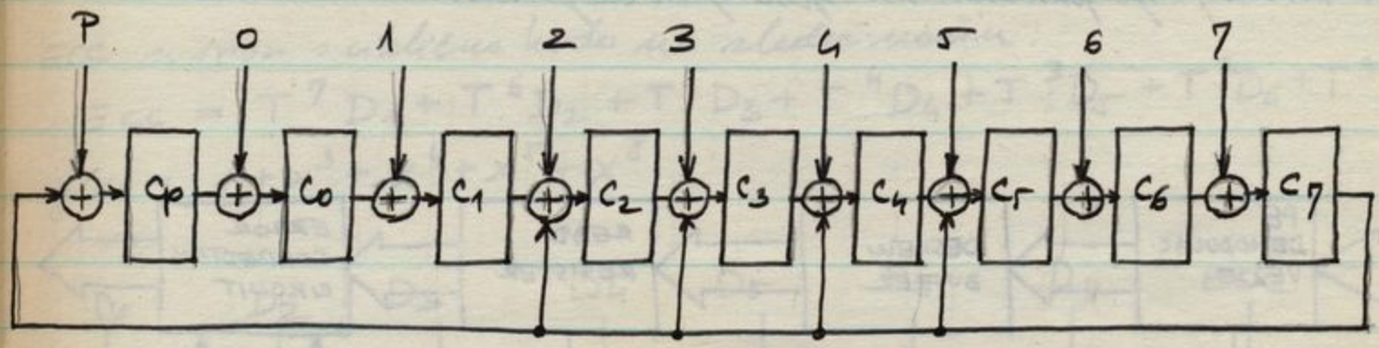
Napako detektira kontroler, podatke o napaki pa posilje program kot sense informacijo.

KOREKCIJA NAPAKE

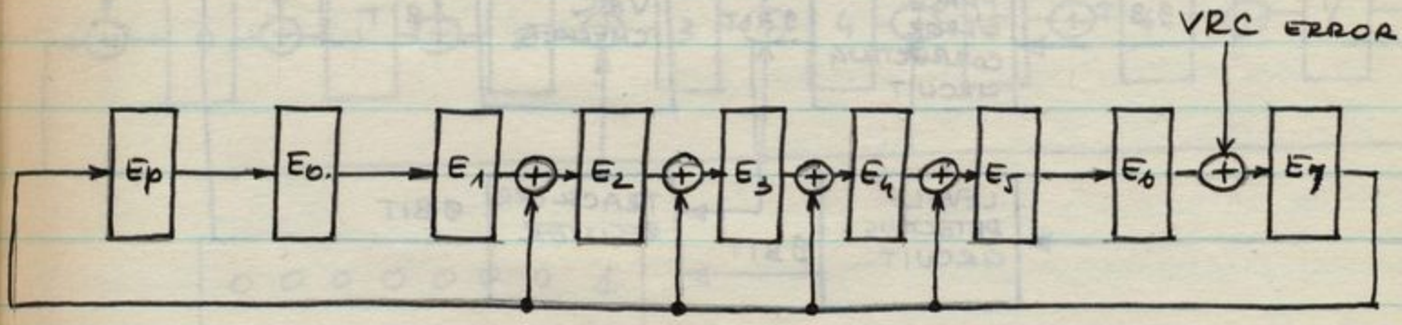
Ponovno čitanje avtomatično popravi napako na spodnjem podatoplu.

- READ ali READ BACKWARD ukaz detektira CRC napako.
- SENSE komanda zbere informacije o napakah na traku.
- BACK SPACE ali SPACE ukaz prenje (reviund) trak za 1 blok.
- Request track in Error komanda prenese 1 byte informacije na traku v MTC. V MTC ni informacija o napaki shrani v ET register
- Read ali READ BACKWARD komanda ni povezuje na REQUEST TRACK in ERROR komando. Če je detektira VRC napako v MTC, se prikodejo čiti v ET registru invertirajo in podatki o napaki se prenesejo v komol.

Pri pravilni korekciji postane vrednost v CRC registru 11101011



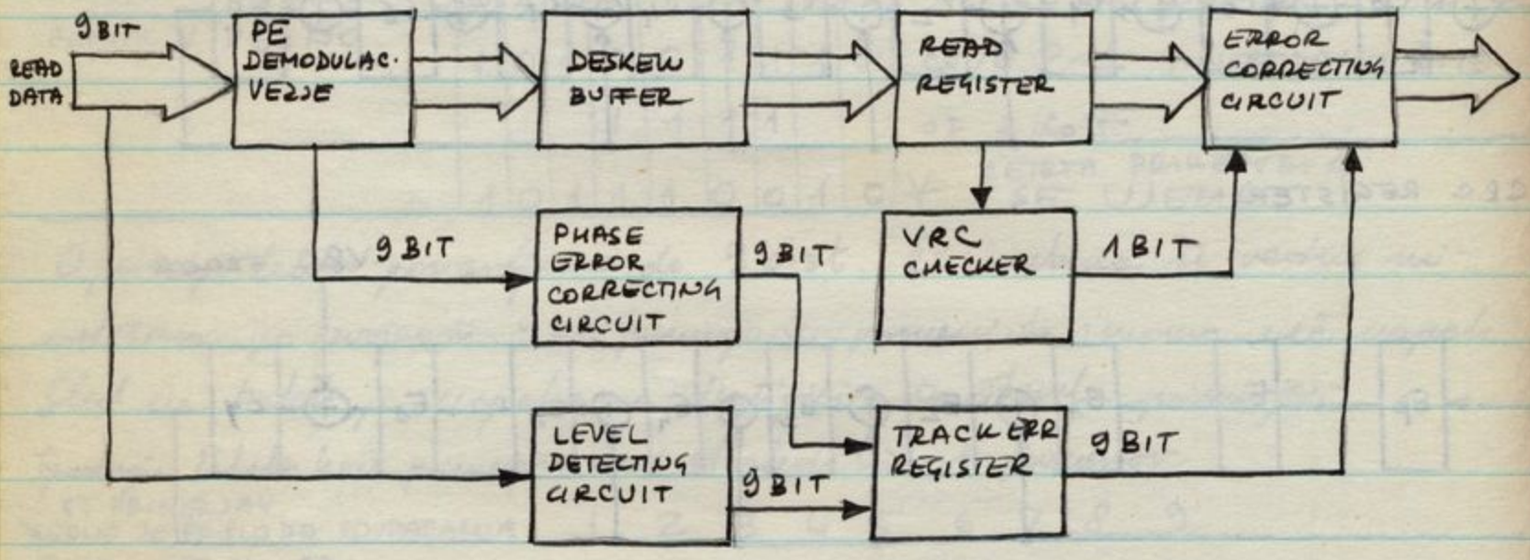
CRC REGISTER



KOREKCIJA NAPAK PRI 1600 BPI

Korekcija napak pri tem načinu uporablja kombinirani način za določanje sledi v katerih je napaka in po pariteti bit namesto ciklične kode. Korekcijo lahko izvedemo le za eno sled

Pri PE metodi se magnetni poli invertirajo enkrat ali dvakrat za vsak bit. V kratkem: vedno dobimo read signal iz vsake sledi med read operacijo. Če je signal manjši kot je standardni nivo ali pa če read pulz ni v pravi poziciji, potem je sled pri tolikih pojavih napačna (napaka na trackn). Vertikalno morajo imeti podatki liho pariteto. Če je napaka samo na eni sledi in če detektiramo byte z napačno pariteto se bit na tej sledi invertira in podatek o napaki se pošlje v glavni pomnilnik. Postopek korekcije je prikazan na spodnjem diagramu.



KOREKCIJA NAPAK PRI 6250 BPI

Pri tem načinu lahko korigiramo napake na dveh sledeh hkrati.

BIT	SUBGRUPA A				SUBGRUPA B			
0								
1								
2								
3	D_1	D_2	D_3	D_4	D_5	D_6	D_7	ECC
4								
5								
6								
7								
P	P_1	P_2	P_3	P_4	P_5	P_6	P_7	PE

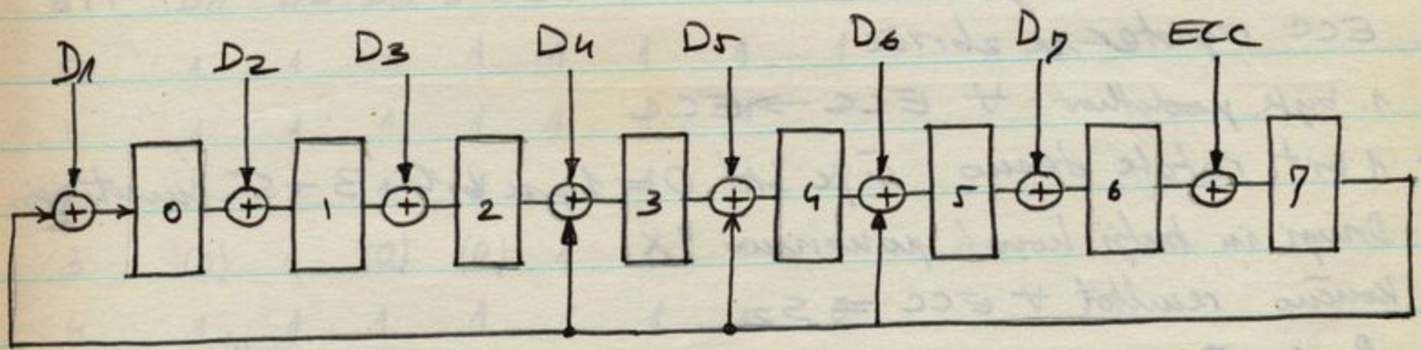
Podatkovna grupa ima 7 podatkovnih bytov in ECC byte

Vsak byte ima liho pariteto.

ECC se izvaja s ciklično kodo na sledeč način:

$$Ecc = T^7 D_1 + T^6 D_2 + T^5 D_3 + T^4 D_4 + T^3 D_5 + T^2 D_6 + T^1 D_7 \quad (1)$$

$$g(x) = 1 + x^3 + x^4 + x^5 + x^8$$



$$T = \begin{bmatrix} 0&0&0&0&0&0&0&1 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&0&0&1 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \end{bmatrix}$$

GENERACIJA ECC MED VPISOVANJEM

- ECC register se briše
- 1 BYTE ⊕ ECC → ECC
- 1 BIT shift (rotate) v desno. Če je BIT 0=1 se biti 3–5 inventirajo.
- ECC BYTE dobimo s ponavljanjem zgornjih dveh korakov 7X.
- Dodan je paritetni bit (~~za vsak stolpec~~) ki tako da je vsota vseh bytov in ECC byta liha (to je PE)

PROCESIRANJE MED ČITANJEM NAPREJ

S_1 (Syndrom 1) predstavlja rezultat paritetne kontrole v vsakem bytu.
S_2 (Syndrome 2) je rezultat ECC kontrole. Če ni napake lahko S_1 in S_2 izrazimo na sledeč način

$$S_1 = 0$$

$$S_2 = T^7 D_1 + T^6 D_2 + T^5 D_3 + T^4 D_4 + T^3 D_5 + T^2 D_6 + T D_7 + ECC = 0 \quad (2)$$

S_1 se generira na isti način kot ECC. ECC byte se vrne kot byte 8.
Procedura je naslednja:

- ECC register se zbriše
- 1. byte podatkov ⊕ ECC → ECC
- 1 bit rotate desno. Če je bit 0=1, se biti 3–5 inventirajo.
- Drugi in tretji korak ponovimo 7X
- končno rezultat ⊕ ECC = S_2

~~Če~~ Predpostavimo, da pride do napake na i bitu in j bitu sledi. Te napake se detektirajo s pomočjo napake, kot dropout ali kot neveljavno kodo. Če E_i in E_j predstavljajo vektor napake na i-tem bitu in j-tem bitu potem lahko postavimo naslednje enačbe.

$$S_1 = E_i + E_j \quad (3)$$

$$S_2 = T^i E_i + T^j E_j \quad \text{pri tem je } 0 \leq i < j \leq 7 \quad (4)$$

Iz enačb (3) in (4) dobimo:

$$E_i = S_1 + E_j \pmod 2 \quad (5)$$

$$E_j = \frac{1}{I + T^{j-i}} \left(S_1 + T^{-i} S_2 \right) \quad \text{glej stran 23} \quad (6)$$

I... identitetna matrika

če je $M_{j-i} = \frac{1}{I + T^{j-i}}$ in $S_3 = S_1 + T^{-i} S_2$ potem

potem enačbo (6) $\boxed{E_j = M_{j-i} S_3}$ (7)

Če no podane vrednosti i in j potem dobimo Ei in Ej z S_1 in S_2.

Postopek je naslednji:

- S_1 se premakne v levo za i krat in potem se izvede XOR z S_1, da dobimo S_3
- Izberemo M_{j-i}, ki pripada vrednosti $j-i$
- Ei dobimo z enačbo (7)
- Ej dobimo z enačbo (6)

Pri čitanju naprej

$S_3 = S_1 + T^{7-i} B S_2$ $B S_2$; S_2 pri čitanju naprej.

S_3 dobimo s premikanjem BS_1 v desno za 7 krat. Nadaljuje operacije na identičen način za čitanje naprej.

PRIMER ZA KOREKCIJO NAPAK

BIT	D1	D2	D3	D4	D5	D6	D7	ECC
0	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	0
3	(0)	1	(0)	(0)	1	1	1	1
4	1	1	1	1	1	1	1	0
5	1	1	1	1	1	1	1	0
6	1	1	(0)	1	1	(0)	1	0
7	1	1	1	1	1	1	1	1
PARITY ERROR	1	1	1	1	1	1	1	1
PARITY ERROR	X			X		X		

Ker je S1 paritetni vektor, ga lahko izrazimo na sledeč način:

$S_1 =$ [0, 0, 1, 0, 1, 0, 0, 1]

zaporedje kot je pušice: napake so pri D1, D4 in D6

S_2 določimo iz izraza:

$$S_2 = T^7D_1 + T^6D_2 + T^5D_3 + T^4D_4 + T^3D_5 + T^2D_6 + TD_7 + ECC$$

	0	1	2	3	4	5	6	7	
BIT									
initial	0	0	0	0	0	0	0	0	
BYTE 1	1	1	1	0	1	1	1	1	
	1	1	1	0	1	1	1	1	+
	1	1	1	1	0	1	1	1	→
				1	1	1			
BYTE 2	1	1	1	1	1	1	1	1	
	0	0	0	1	0	1	0	0	+
	0	0	0	0	1	0	1	0	→
BYTE 3	1	1	1	0	1	1	0	1	
	1	1	1	0	0	1	1	1	+
	1	1	1	1	0	0	1	1	→
				1	1	1			
BYTE 4	1	1	1	0	1	1	1	1	
	0	0	0	0	0	0	0	0	+
	0	0	0	0	0	0	0	0	→
BYTE 5	1	1	1	1	1	1	1	1	
	1	1	1	1	1	1	1	1	+
	1	1	1	1	1	1	1	1	→
				1	1	1			
BYTE 6	1	1	1	1	1	1	0	1	
	0	0	0	1	1	1	1	0	+
	0	0	0	0	1	1	1	1	→
BYTE 7	1	1	1	1	1	1	1	1	
	1	1	1	1	0	0	0	0	+
	0	1	1	1	1	0	0	0	→
ECC	1	1	0	1	0	0	0	1	
S2 =	1	0	1	0	1	0	0	1	+

$$S_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

bit 0

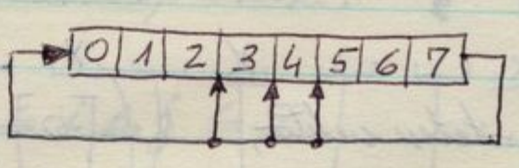
bit 7

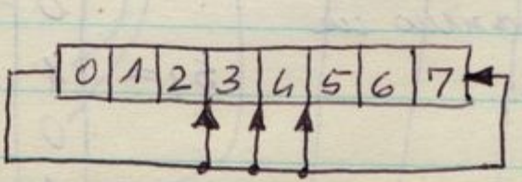
za S_1 in S_2

Če so vse vrednosti enake 0 se smatra, da ni napake v podatkovnem polju.

Če predpostavljamo, da je $i=3$ in $j=6$, potem S_2 premaknemo v LEVO z ECC vezjem za i krat (3 krat)

	0	1	2	3	4	5	6	7	
$S_2 =$	1	0	1	0	1	0	0	1	
$T^{-1}S_2 =$	0	1	1	0	1	0	1	1	
$T^{-2}S_2 =$	1	1	0	0	1	1	1	0	
$T^{-3}S_2 =$	1	0	0	1	0	1	0	1	





BIT	0	1	2	3	4	5	6	7
$S_2 =$	1	0	1	0	1	0	0	1
				→	1	1	1	
	1	0	1	1	0	1	0	1 ✓
$T^{-1}S_2$	0	1	1	0	1	0	1	1
$T^{-2}S_2$	1	1	0	1	0	1	1	0
				→	1	1	1	
	1	1	0	0	1	0	1	0 ✓
$T^{-3}S_2$	1	0	0	1	0	1	0	1

$$S_3 = S_1 + T^{-3}S_2 = \begin{bmatrix}0\\0\\1\\0\\1\\0\\0\\1\end{bmatrix} + \begin{bmatrix}1\\0\\0\\1\\0\\1\\0\\1\end{bmatrix} = \begin{bmatrix}1\\0\\1\\1\\1\\1\\0\\0\end{bmatrix}$$

$$S_1 = E_i + E_j \longrightarrow E_i = S_1 + E_j \quad (\text{mod } 2)$$

$$S_2 = T^i E_i + T^j E_j$$

$$S_2 = T^i (S_1 + E_j) + T^j E_j$$

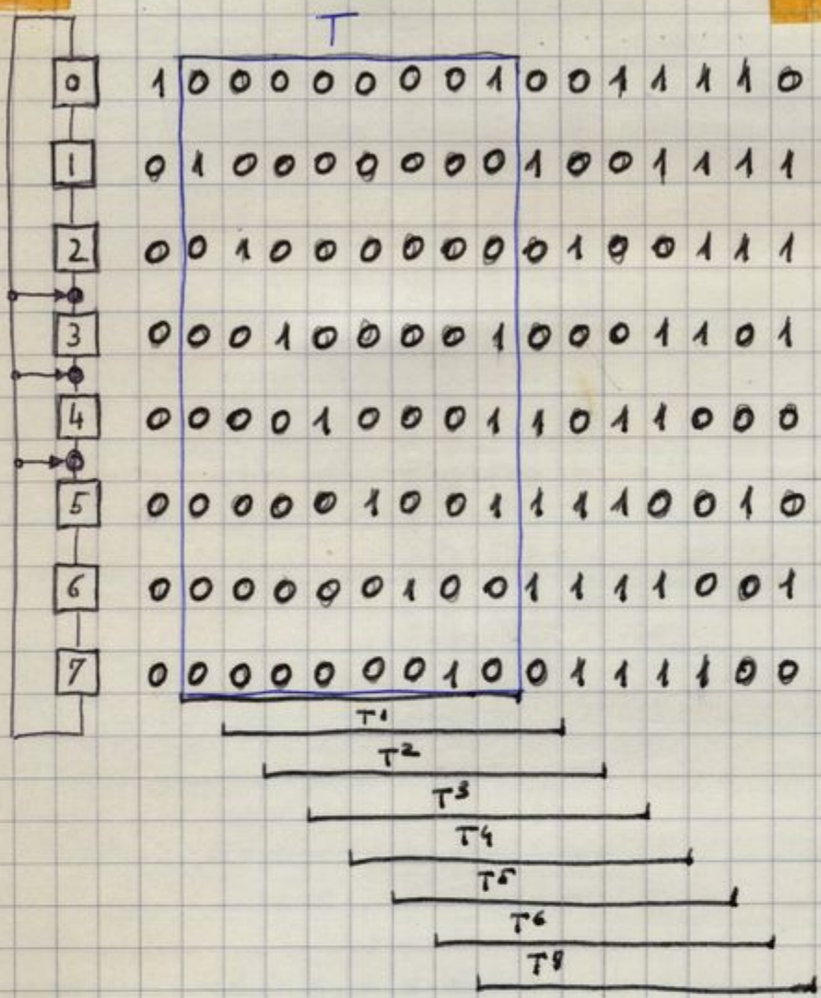
$$S_2 = T^i S_1 + T^i E_j + T^j E_j$$

$$S_2 + T^i S_1 = (T^i + T^j) E_j$$

$$E_j = \frac{1}{T^i + T^j} (S_2 + T^i S_1) \quad \Big| \cdot \frac{T^{-i}}{T^{-i}}$$

$$E_j = \frac{1}{I + T^{j-i}} (T^{-i} S_2 + S_1)$$

$$E_j = \frac{1}{I + T^{j-i}} (S_1 + T^{-i} \cdot S_2)$$



Kako dobimo matrike
$T, T^1, T^2, \ldots T^8$

Grupa matrik $M_{j-i} = \dfrac{1}{I + T^{j-i}}$

$$M_1 = \begin{bmatrix} 0&1&1&1&1&1&1&1 \\ 0&0&1&1&1&1&1&1 \\ 0&0&0&1&1&1&1&1 \\ 1&1&1&1&0&0&0&0 \\ 0&0&0&0&0&1&1&1 \\ 1&1&1&1&1&1&0&0 \\ 1&1&1&1&1&1&1&0 \\ 1&1&1&1&1&1&1&1 \end{bmatrix}, M_2 = \begin{bmatrix} 0&0&1&0&1&0&1&0 \\ 0&0&0&1&0&1&0&1 \\ 0&0&0&0&1&0&1&0 \\ 1&0&1&0&1&1&1&1 \\ 1&1&1&1&1&1&0&1 \\ 0&1&0&1&0&1&0&0 \\ 1&0&1&0&1&0&1&0 \\ 0&1&0&1&0&1&0&1 \end{bmatrix}, M_3 = \begin{bmatrix} 1&1&0&0&1&0&0&1 \\ 0&1&1&0&0&1&0&0 \\ 1&0&1&1&0&0&1&0 \\ 1&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&1&0&0&1&0&0&1 \\ 0&0&1&0&0&1&0&0 \\ 1&0&0&1&0&0&1&0 \end{bmatrix}$$

$$M_4 = \begin{bmatrix} 0&1&0&1&1&1&0&1 \\ 1&0&1&0&1&1&1&0 \\ 0&1&0&1&0&1&1&1 \\ 0&1&1&1&0&0&1&0 \\ 0&1&1&0&0&1&1&0 \\ 1&1&1&0&1&1&1&0 \\ 0&1&1&1&0&1&1&1 \\ 1&0&1&1&1&0&1&1 \end{bmatrix}, M_5 = \begin{bmatrix} 1&0&0&0&1&0&0&0 \\ 1&1&0&0&0&1&0&0 \\ 0&1&1&0&0&0&1&0 \\ 0&0&1&1&1&0&0&1 \\ 1&0&0&1&0&1&0&0 \\ 0&1&0&0&0&0&1&0 \\ 0&0&1&0&0&0&0&1 \\ 0&0&0&1&0&0&0&0 \end{bmatrix}, M_6 = \begin{bmatrix} 0&0&1&1&1&1&1&0 \\ 1&0&0&1&1&1&1&1 \\ 1&1&0&0&1&1&1&1 \\ 0&1&0&1&1&0&0&1 \\ 1&0&0&1&0&0&1&0 \\ 1&1&1&1&0&1&1&1 \\ 1&1&1&1&1&0&1&1 \\ 0&1&1&1&1&1&0&1 \end{bmatrix}$$

$M_7 =$

1	0	0	1	1	1	0	0
0	1	0	0	1	1	1	0
1	0	1	0	0	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	0	1	1
1	1	1	0	0	0	0	1
0	1	1	1	0	0	0	0
0	0	1	1	1	0	0	0

Opomba: Vrednost za i in j dobimo iz track pointerja

Kadar je $i = 3$ in $j = 6$ se E_6 izraži na sledeči način:

$$E_6 = M_3 \cdot S_3 = \begin{bmatrix} 1&1&0&0&1&0&0&1 \\ 0&1&1&0&0&1&0&0 \\ 1&0&1&1&0&0&1&0 \\ 1&0&0&1&0&0&0&0 \\ 0&0&0&0&0&0&0&1 \\ 0&1&0&0&1&0&0&1 \\ 0&0&1&0&0&1&0&0 \\ 1&0&0&1&0&0&1&0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

↑ # sledi

Ustrezno
ECC
D7
D6
D5
D4
D3
D2
D1

$$E_3 = E_6 + S_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

↑ # sledi

Ustrezno
ECC
D7
D6
D5
D4
D3
D2
D1

Vrednosti vektorja ustrezajo mestom napake ~~na~~ v kolonah ECC, D_7, D_6, D_5, ..., D_1. Tisti byte, ki ima enico je napačen. Sled št 3 ima napake v 1., 3. in 4. byte. Sled št 6 ima napake v 3. in 6. byte

Error track pointer – definicija napake na sledi

Če hočemo izvajati korekcijo na dveh sledeh moramo vedeti katera sled ima napako. Error track pointer se setira pod naslednjimi pogoji:

- PHASE ERROR — Fazna ~~d~~ razlika med vrhovi impulzov
- SKEW ERROR — Prevelik SKEW na sledi
- INVALID CODE — Prisotnost kode, ki ne spada v 5/4 kodo v času čitanja
- TIME SENSE "OFF" — Ča ni peak impulza potem, ko je sel nivo za čitanje v low.

Povezava med ECC dimenzijo in bitom

Definiramo jo na naslednji način:

STOPNJA POLINOMA	1	X	X^2	X^3	X^4	X^5	X^6	X^7
ECC dimenzija	0	1	2	3	4	5	6	7
Binarno bit	0	5	6	2	7	4	1	3
Stevilka sledi	7	1	8	5	2	9	6	3

Drugi primeri:

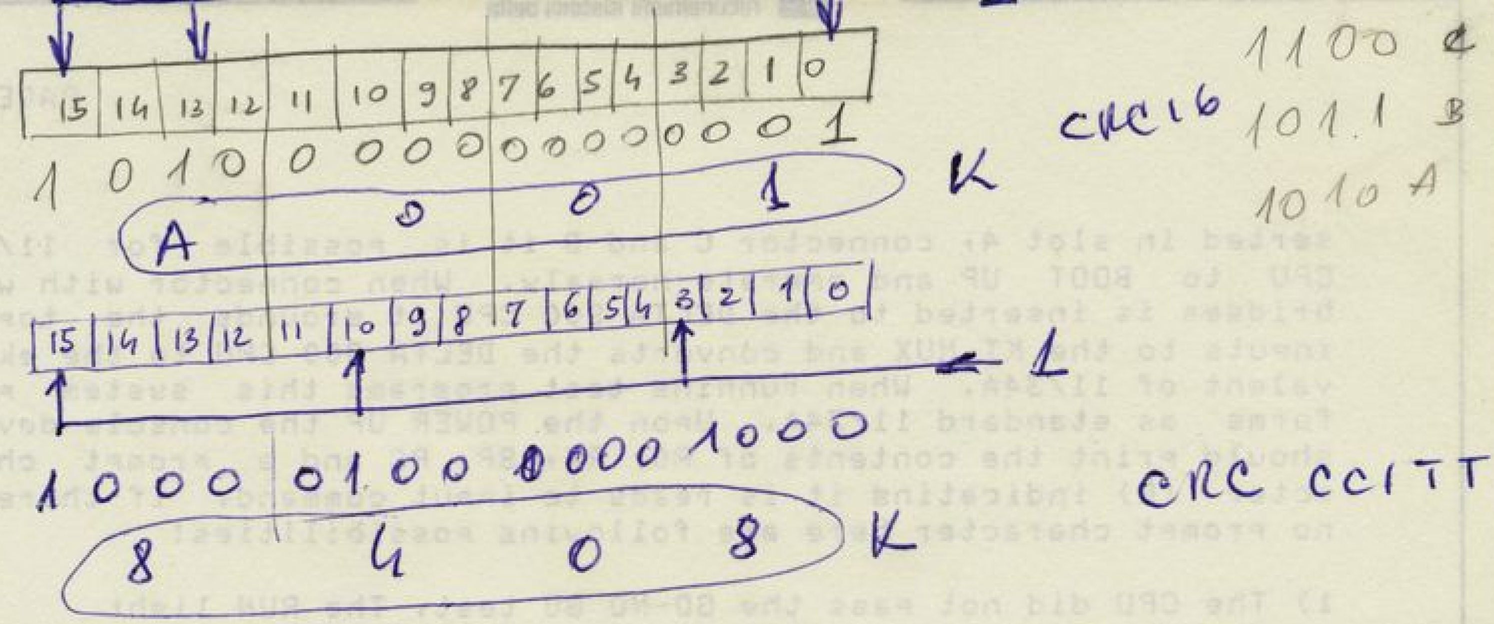
Napaka na eni sledi: ECC lahko detektira in korigira napako na eni sledi brez track pointerja. Detekcija in korekcija se izvaja s pomikanjem S_2 na enak način kot pri NRZI. Sled z napako detektiramo s primerjavo med S1 in z izračunano S1, potem pa se biti, ki pripadajo S1 invertirajo.

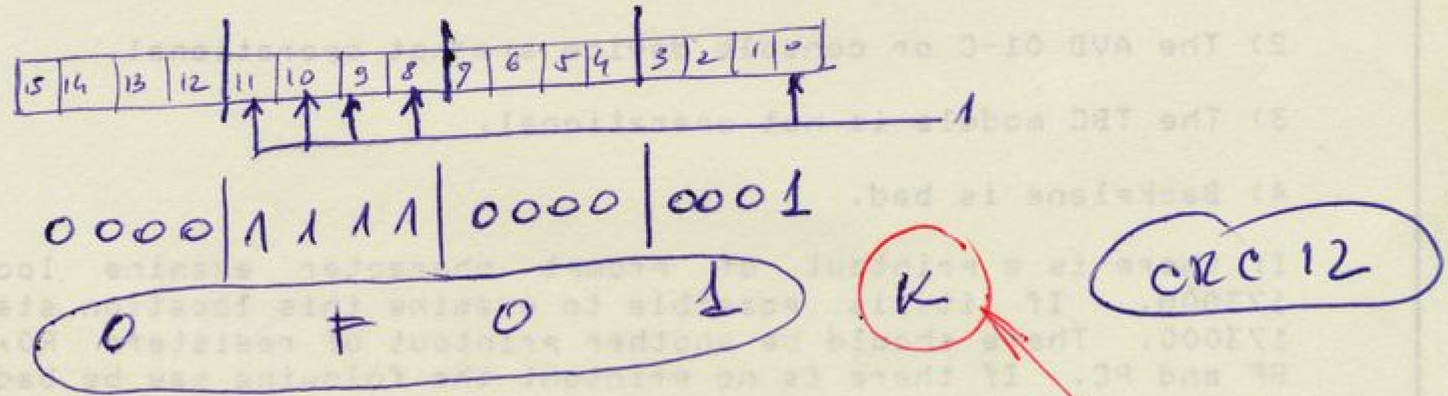
Napaka na P sledi če je napaka samo na tisti sledi, ki nosi zapis pozitete ($S_2 = 0$), takoj pozitetni bit ki ustreza S1 invertiramo.

V primeru napake na dveh sledeh in če error track pointer kaže na pozitetno sled potem napako korigiramo z izračunano S1 in S_2 po naslednjih enačbah

$$S_1 = E_i + E_p$$

$$S_2 = T^i E_i$$





Algoritem za računanje CRC vrednosti za poljubne polinome

Rutina je počasna ker obdeluje bit za bitom

```
CRC = 0
FOR I = 1 TO N
IF (DATA (I) .EXOR. (CRC .AND. 1)) = 1 THEN FEED=K
ELSE FEED = 0
CRC = CRC * 2
CRC = CRC .EXOR. FEED
NEXT
```

serted in slot 4, connector C and D it is possible for 11/34A CPU to BOOT UP and operate normaly. When connector with wire bridges is inserted to the DELTA 800 CPU it grounds the top 4 inputs to the KT-MUX and converts the DELTA 800 CPU to the ekvivalent of 11/34A. When running test programs this system performs as standard 11/34A. Upon the POWER UP the console device should print the contents of R0, R4, SP, PC and a prompt character (@) indicating it is ready to input command. If there is no prompt character here are following possibilities:

1) The CPU did not pass the GO-NO GO test. The RUN light should be ON. If the RUN light is OFF the detection of abnormal error is indicated (double bus error...).

2) The AVD 01-C or console device are not operational.

3) The TBC module is not operational.

4) Backplane is bad.

If there is a printout of prompt character examine location 173000. If it is possible to examine this location start at 173000. There should be another printout of registers R0, R4, SP and PC. If there is no printout the folowing may be bad:

1) AVD 01-C
2) TBC
3) D800
4) backplane

If there is a prompt then you can add RUP, MPC and MPE256. Now with the power on there should be a register and propt printout. Verify that the operations as deposit and examine of memory locations is possible. Use commands described earlyer in this manual. If it is not possible to check the memory locations there might be a fault in one of the modules:

1) MPE256
2) RUP
3) MPC
4) backplane

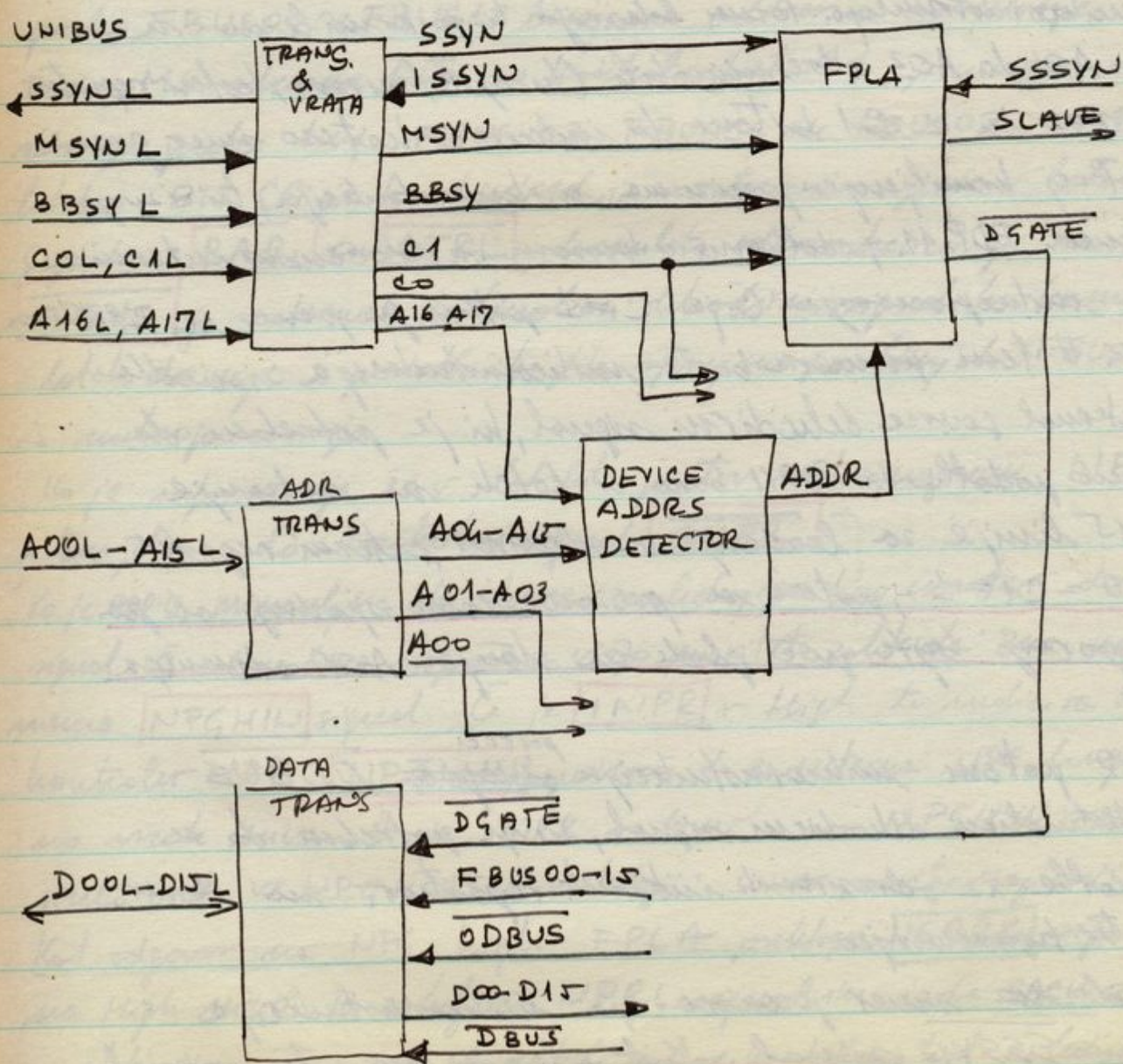
Remove the MPE256 and insert another one to the second slot of the DD 11-MK backplane. If the prompt does not appear at the power up there is probably a fault at one of the modules:

1) RUP
2) MPC
3) backplane
4) MPE256
5) D800
6) TBC

PDP 11 INTERFACE

① PRENOSI PRI KATERIH JE KONTROLER SLAVE

2 namenom da prenašamo že postavljene (določene; set-up) bremede v kontroler, oli pa, da nadziramo status kontrolerja, PDP11 (procesor) inicira prenos v katerem je on bus master. PDP11 procesor postavi določeno adreso (odreso registra v kontrolerju) na adresni bus, postavi signale COL in C1L na ustrezne nivoje in postavi BBSYL in MSYNL.
Kontroler komparira odresni bus A04 — A17 — to je tisti del odrese, ki je skupen vsem registrom kontrolerja, z odreso, ki je programirano z jumperjem. Ko detektira enakost gre ADDR → HIGH



V odgovor na aktivni BBSYL in na SSYNL signal ~~pošlje~~ povzroči ADDR high do dela FPLA SLAVE signal z visokim nivojem, kar povzroči interrupt mikroprograma. Pri $\overline{\text{SLAVE}}$ low signalu se sprejemi latch-i bus transceiverjev enablirani in sledijo podatkom na PDP11 podatkovnem busu. To omogoča prenos podatkov iz PDP 11 procesorja v kontrolerjeve registre.

$\overline{\text{C1=0}}$ indicira prenos v smeri proti PDP11 procesorju $\overline{\text{DGATE}}$ gre v LOW pri sovpadanju ~~signalov~~ ADDR → High, BBSYL in MSYNL. To ~~pomeni~~ da gredo podatki iz data transceiver latchov na PDP11 & data bus.

Ko je SLAVE high, ko vsaki mikroinstrukcija, ki se dissklira mikroprogramske interrupt funkcije, sledi JUMP v interrupt service mikroinstrukcijo. Točno lokacija ki jo kreiramo se določa z AO1 do AO3 odresnimi biti (ki specificirajo, kateri register je adresiran in z C1 bitom ki indicira v katero smer gre prenos. Brej kreiranje prevzame mikroinstrukcija, ki izvaja prenos med PDP 11 podatkovnim busom in določenim internim registrom mikroprocesorja. Če je C=1 potem gre prenos iz PDP11 bus v register. V tem primeru bo to mikroinstrukcija izvedla $\overline{\text{DBUS}}$ external source dekodirni signal, ki je potreben da prevzamemo podatke iz PDP11 bus v latch na notranje D00 do D15 linije za loadanje v odresiran interni register. Če je C0 = C1 = 1, potem se prevzame samo spodnji oli pa samo zgornji byte počl glede na stanje A00 odresnega bita.

Če je C1=0 potem mikroinstrukcija ~~omogoča~~ pošlje $\overline{\text{ODBUS}}$ external destination dekodirni signal, ki je potreben da se pošlje podatke iz odresiranih internih registrov na driver latch data transceiverjev.

Ne glede na smer prenosa pa mikroinstrukcija pošlje $\overline{\text{SSYN}}$ signal. Kot odgovor na to signal

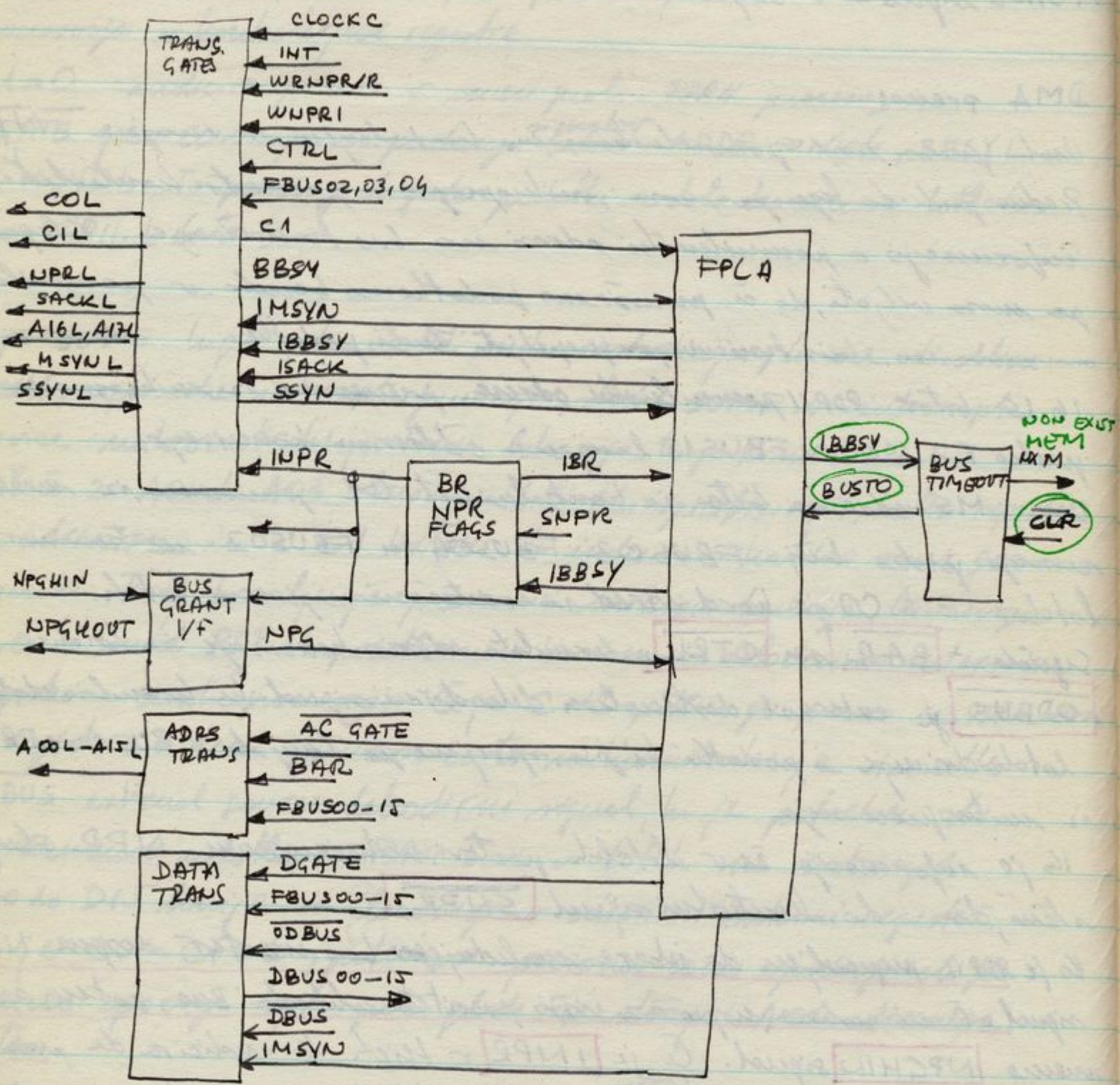
na FPLA do ISSYN v High in zaključi SLAVE (Hilyp) signal.
ISSYN high postori PDP 11 SSYNL signal v aktivno stanje.
Signala ISSYN in $\overline{\text{DGATE}}$ (za prenos v smeri procesorja) sta v aktivnem stanju dokler PDP11 procesor ne zaključi aktivnega MSYNL signala v odgovor na aktivni SSYNL signal.

DMA prenos

Vsak prenos med pomnilnikom in krmilnikom se izvaja z NPR
Prvo pride do tega pa mora mikroprogram prenesti krmilnikovo in informacijo o pomnilniški adresi na bus krmilnika. Poleg tega pa mora določiti, da ali prenašamo podatkovno besedo v pomnilnik ali moramo na bus krmilnika pripeljati tudi podatke
16 LS bitov PDP11 pomnilniške adrese prenesemo iz mikroprocesorja preko FBUS00 ~ FBUS15 linij na address bus krmilnika.
Dva MS odresna bita in kontrolni bit C1 prenesemo iz mikroprocesorja preko linij FBUS03, FBUS04 in FBUS02 na krmilnikov latch. Bit C0 je hardwired in se ne more vpisat v latch.
Signala BAR in $\overline{\text{CTRL}}$ krmilita address krmilnika in latch-e.
$\overline{\text{ODBUS}}$ je external destination decoding signal ki krmili odpiranje latch krmilnika s podatki ki jih sprejme po linijah FBUS00 ~ FBUS15 iz mikroprocesorja.
Ko je informacija že v latchih postori mikroprogram NPR flag s tem, da da kontrolni signal $\overline{\text{SNPR}}$
Ko je PDP11 pripravljen da izbere naslednjega bus master dvigne NPGH signal. Če ni nobene zahteve z višjo prioriteto zahteva Bus, potem sprejmemo NPGHIN signal. Če je INPR v High to indicira da jima kontroler NPR. NPGHIN povzroči da se interna NPG linija postavi na visok nivo. Če je INPR v Low se NPGHIN signal vrne preko vrat v NPGHOUT in naslednji lower priority naprave.
Kot odgovor na NPG high FPLA pošlje ISACK signal na High nivo. To zaključi NPRL signal in izda SACK signal z aktivnim nivojem da potrdi izbor kontrolerja kot naslednji bus master

Kot odgovor na SACKL signal PDP11 seključi HPGH signal
2. ISACK v high FPLA preklopi IBBSY signal na vzoh
mimo lokor hitro pre BBSY v low, kar indicira do je
prejšnji bus master prepustil kontrolo.

PDP 11 BUS.



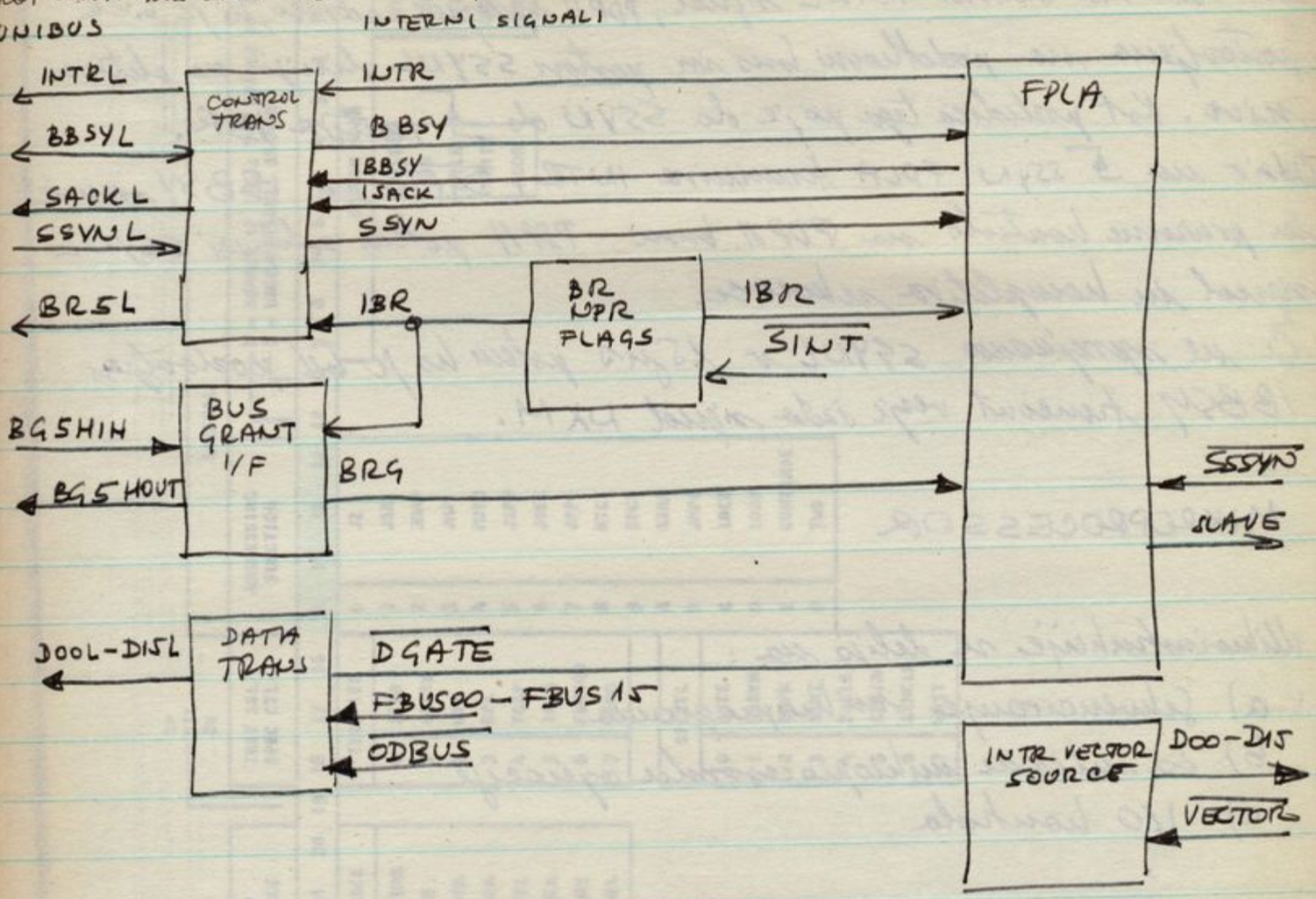
Ko se IBBSY preklopi na high, gre BBSYL v low in BBSY (zopet) v high.
Pri IBBSY in BIBSY high in SSYN low (kar indicira da je prejšnji bus slave kompletoval prejšnjo tranzakcijo, kontroler FPLA preklopi ACGATE v low, IMSYN v high in konča ISACK signal.
Pri ACGATE low se informacija, kije bila prej v control in address transceiver lotchih prenese na unibus in v PDP spomin. kar je indicirano s C1 signalom.
DGATE signal gre v low in stem sprosti podatke na unibus.
Signal IMSYN (v high) gre MSYNL na unibus s primerno zakasnitvijo.
Kot odgovor na MSYNL pomnilnik sprejme podatke kijih da kontroler, slipa do na podatkovna vodila podatke kijih sprejme kontroler. in potem postavi SSYNL v aktivno nivo. SSYNL postavi SSYN v high
Kadar gredo podatki iz PDP 11 kontrolerja, se podatki iz PDP 11 buna pomnejo v sprejemnih lotch. Ta lotch je enabliran z IMSYN signalom. Za to smer prenosa je WNPRI postavljen v low s prihodom signala SSYNL signala iz PDP 11 pomnilnika. To ~~izda~~ prekine mikroprocesor in ko je to interrupt servisiran se bendix s sprejetimi podatki prenese iz sprejemnih lotchov v mikroprocesor. Med servisiranjem interrupta se ob zaporedju INT in WRNPRIR signalov iz mikroprocesorja, sprosti CLOCKC in resetira WNPRI. INT indicira, da se je zaključil interrupt servis. WRNPRIR je sinhronizirana verzija WNPRI.
Za vsak prenos v določeno smer, se resetira INPR flag z ↑ SSYN.
Ko je SSYN v high FPLA konca IMSYN, ACGATE in DGATE in IBBSY signale po tem vrstnem redu.
Ko FPLA preklopi IBBSY v high se aktivira Bus timeout funkcija. To funkcija omogoča zaključek prenosa in postavitev error bita če kontroler odreagira neobstoječo pomnilniško lokacijo. Če pride do tega nimamo SSYNL odziva. Če je IBBSY v high za cca

15 µs potem bus timeout funkcija setira BUSTO in NXM. Kot odziv na BUSTO (high) signal pa FPLA zahtjuči vsa spremenjena signale na da si čakalo na SSYNL. Ko IBBSY preklopi na low se BUSTO resetira. Non existent memory flag signal NXM ostane setiran dokler ga ne resetira CLR signal iz mikroprocesorja.

BUS REQUEST / INTERRUPT SEKVENCA

Med to sekvenco kontroler izvede interrupt pdp 11 procesorja na postavi interrupt vektor odreso na PDP 11 podatkovni bus. To odresa pride na D00 do D15 linije vmikrun. Ti podatki predstavljajo kot posledica VECTOR zunanjega izvora dekodiranega signala iz mikroprocesorja. To vektor je nastavljen z jumperji.
Bus request sekvenca se začne ko mikroprocesor izda SINT signal. Ta signal pride iz mikroprocesorskega kontrolnega delovnega setira IBR flop. Ta postavi PDP 11 BR5L linije v aktivni nivo. Standardna postavitev jumperjev na kontrolerji konfigurira kontroler, da komunicira preko level 5 bus request linije. Vendar se lahko to spremeni z jumperjem.
Ko je PDP11 pripravljen, da se odzove na request na BR5L liniji, postavi BG5H signal v visok nivo. Če ni nobene napake na višji prioriteti bi bi zahtevala transakcijo na busu se sprejme BG5HIN signal. Če imamo IBR high, kar indicira da imo kontroler bus request (pending), BG5HIN signal povzroči, da se notranja BRG linija preklopi na high nivo. (Če je IBR low, se BG5HIN signal pošlje na BG5HOUT linijo k naslednji napravi z nižjo prioriteto. Neomočene BG4HIN linije se povežejo direktno na pripadajoče BG4HOUT linije)
Kot odziv na BRG signal da FRA SACK signal z nivojem HIGH To konča aktivni BR5L signal in postavi PDP11 SACKL na aktivni

nivo s čemer pripove na znanje izbiro kontrolerja kot naslednji bus master.
Kot odziv na aktivni SACK signal, PDP 11 terminira BG5H signal.



Ko je SACK v high, FPLA prekloplji IBBSY na visok nivo, kolikorkrat se BBSY preklopi v low, kar indicira, da je prejšnji bus master prekustil kontrolo na bus-u. Ko se IBBSY preklopi v high se PDP 11 BBSYL linija zopet postavi v nizek nivo. Sedaj je kontroler Bus master.
Če je IBR, ISACK in IBBSY v high, kontrolni FPLA prevzame SLAVE signal v mikroprocesor, s čemer indicira da je prevzel kontrolo nad unibusom. Mikroprogram mora sedaj odgovoriti s tem, da load-a PDP 11 interrupt vektorsko adreso na bus driverje latch in da odda $\overline{SSSYN}$ signal.
Bus driver latch se prevzamejo podatki iz internega vodila FBUS00 - 15 kot odziv na $\overline{ODBUS}$ signal iz mikroprocesorjevega destination dekodirja. Signal $\overline{SSSYN}$ pride iz povratnega dekodirja ki ga ima mikroprocesor.
Kot odziv na $\overline{SSSYN}$ signal, kontrolni FPLA terminira SLAVE signal, preklopi $\overline{DGATE}$ signal na nizek nivo, terminira ISACK in postavi INTR signal na visok nivo. $\overline{DGATE}$ signal bo povrnil do ~~prej~~ interrupt vektor na unibus.

INTR high signal po postoru unibus interrupt linijo na aktivni nivo. Kot odziv na aktivni INTRL signal, PDP11 sprejme odresco, ki je bila postavljena na podatkovni bus in postavi SSYNL linijo na aktivni nivo. Kot posledica tega poje do SSYN ob ↓ resetira IBR.
Odziv na ↓ SSYN FPLA terminira INTR, DGATE in IBBSY signale in prevzame kontrolo na PDP11 busu. PDP11 potem zaključi SSYNL signal in kompletira sekvenco.
Če se sprejemno SSYNL v 15 μs potem ko je bil postavljen IBBSY timeout vezje izda signal NXM.

MIKROPROCESSOR

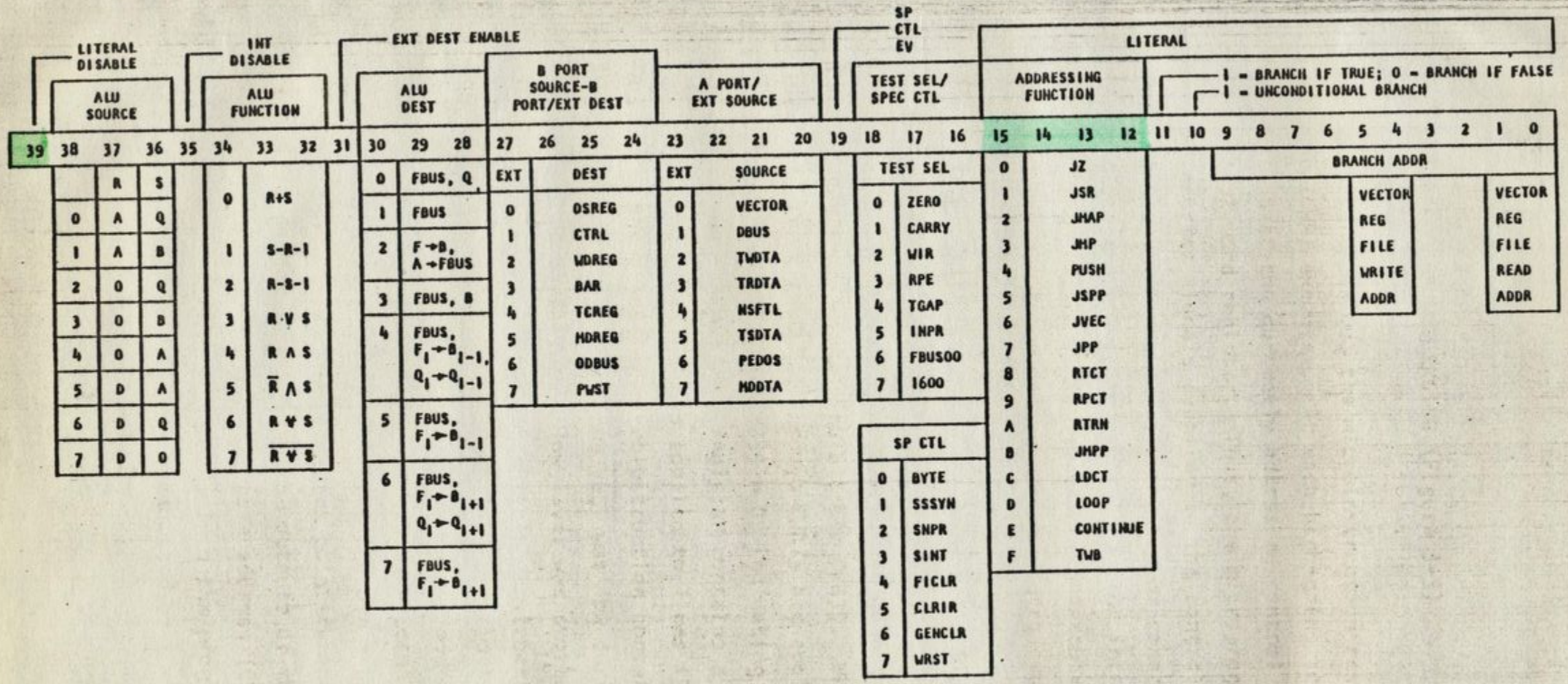
Mikroinstrukcije se delijo na:
a) Sekvenciranje mikroprograma
b) Za interne mikroprocesorske operacije
c) I/O kontrola

SEKVENCIRANJE MIKROPROGRAMA

CWR = CONTROL WORD REGISTER

Mikroinstrukcija, ki se naloži v CWR se izvršuje v času CLOCK A
V CWR se mikroinstrukcija vpiše ob ↓ ClockA, toko je mikroprogramska sekvenca določena z adresnim signalom ki je v mikroprogramski pomnilnik preko CSA00 – CSA09 linij.
Adresa lahko pride na mikroprogramski pomnilnik iz dveh izvorov: 1) mikroprogramski kontrolni modul 2910
2) interrupt address vrata in mux.
Kadar se ne izvaja interrupt, se odreso mikroprogramskega pomnilnika določa z mikroprogramskim kontrolnim modulom.

2910 izvaja eno izmed 16 odresnih funkcij glede na vhodno kodo, ki jo dobi preko linij I0 do I3 iz sekvenčnega kodnega mux-a

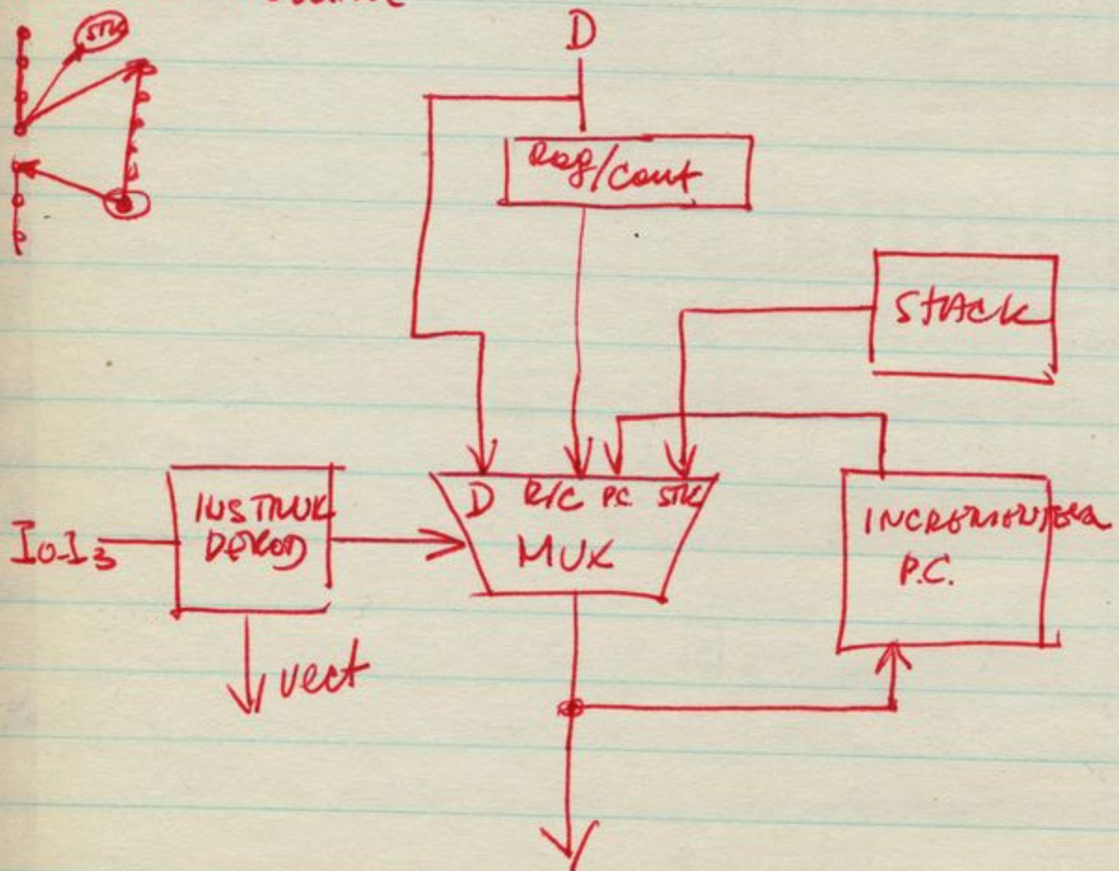

LITERAL DISABLE
ALU SOURCE
INT DISABLE
ALU FUNCTION
EXT DEST ENABLE
ALU DEST
B PORT SOURCE-B PORT/EXT DEST
A PORT/ EXT SOURCE
SP CTL EV
TEST SEL/ SPEC CTL
ADDRESSING FUNCTION
LITERAL
1 = BRANCH IF TRUE; 0 = BRANCH IF FALSE
1 = UNCONDITIONAL BRANCH
39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
BRANCH ADDR
VECTOR REG FILE WRITE ADDR
VECTOR REG FILE READ ADDR
R S
0 A Q
1 A B
2 0 Q
3 0 B
4 0 A
5 D A
6 D Q
7 D 0
0 R+S
1 S-R-1
2 R-S-1
3 R ∨ S
4 R ∧ S
5 R̄ ∧ S
6 R ∀ S
7 R ∀ S
0 FBUS, Q
1 FBUS
2 F→B, A→FBUS
3 FBUS, B
4 FBUS, F_I→B_I-1, Q_I→Q_I-1
5 FBUS, F_I→B_I-1
6 FBUS, F_I→B_I+1 Q_I→Q_I+1
7 FBUS, F_I→B_I+1
EXT DEST
0 OSREG
1 CTRL
2 WDREG
3 BAR
4 TCREG
5 MDREG
6 ODBUS
7 PWST
EXT SOURCE
0 VECTOR
1 DBUS
2 TWDTA
3 TRDTA
4 NSFTL
5 TSDTA
6 PEDOS
7 MDDTA
TEST SEL
0 ZERO
1 CARRY
2 WIR
3 RPE
4 TGAP
5 INPR
6 FBUS00
7 1600
SP CTL
0 BYTE
1 SSSYN
2 SNPR
3 SINT
4 FICLR
5 CLRIR
6 GENCLR
7 WRST
0 JZ
1 JSR
2 JMAP
3 JMP
4 PUSH
5 JSPP
6 JVEC
7 JPP
8 RTCT
9 RPCT
A RTRN
B JMPP
C LDCT
D LOOP
E CONTINUE
F TWB

krmilni;

Vhod v to mux je CWR literal-inhibit Bit CWR 39 = 1, podatkovni vhod pa za biti CWR12 ~ CWR15 ki skozi MUX ustrezajo na I0 do I3 vhode 2910.

Kadar je CWR 39 = 0 to pomeni da polje CWR0 – CWR15 vsebuje konstanto, kodo z eno samo izjemo pripeljemo na I3 do I0 vhodo (1110) CONTINUE. Izjema nastopi, če je bilo predhodna mikroinstrukcija JUMP to DIRECT. V tem primeru je $\overline{VECT}$ low med prejšnjo mikroinstrukcijsko periodo, tako da je to low level shranjen v $\overline{VECT}$ flip flopu ob startu tekoče mikroinstrukcijske periode. Ko je $\overline{VECT}$ shranjen pride na vhode I3 do I0 vrednost (1010) CRTN če je CWR 39 = 0. Kadar izhodi mikroprogramskega kontrolnega modula krmilija control store address (CSA) potem vsak $\frac{1}{4}$ clock A vpiše CSA + 1 v programski števec. Toko ne toliko časa dokler je izbran programski števec kot izvor za izhod mikroprogramskega kontrolnega modula mikroinstrukcije izvajajo zaporedno. Z momentom da uvedemo branch lahko izberemo za izhod D, R/C, ali STK

condition return



D označuje direktni vhod linij D0–D9. V primeru jump to D/VECT adresne funkcije takrat pripeljemo CWR 00 – CWR09 na linije D0–D9. Pri jump-to-D/VECT adresni funkciji povzroči $\overline{VECT}$ low signal, da se vsebina vektorjevega registra na katerega lokacijo kažejo biti CWR00 – CWR 01 ki jih pripeljemo na vhode D0 do D3 namesto CWR00 do CWR03. Na ta način lahko skočimo na katerokoli izmed 16 lokacij v tabeli katere adresa je specificirana s CWR04 do CWR 09 biti.

R/C je trenutna vsebina register/counterja. Pri adresni kodi 1100 se register counter napolni iz direktnih vhodnih linij D0 do D9. Pri drugih adresno funkcijskih kodah se testira zero status in se uporablja za določanje adresne funkcije. V teh primerih če R/C ≠ 0 se dekrementira.

STK označuje lokacijo STACK-a na katerega trenutno kaže stack pointer. Kadar se izvaja BRANCH, lahko PC vrednost postavimo v stack in jo uporabljamo kot povratno adreso. PUSH operacija inkrementira stack pointer in vpiše PC v na novo adresirano lokacijo. STK lahko izberemo kot izvor adrese z ali pa brez POP operacije. POP operacija dekrementira STK potem ko je bilo izvedeno adresiranje. Continue funkcija ima kodo 1110, ki jo pripeljemo na I3 do I0 linije takrat ko je CWR35 = 0 in če predhodna instrukcija ne vsebuje jump to D/VECT adresnega načina, potem koda 1010 izbere pogojni skok (return).
Za večino adresnih funkcijskih kod se izvršuje jump če je CWR10 = 1 ali pa če je zadoščeno testnemu pogoju, ki ga specificirajo biti CWR 16 – CWR18. Biti CWR 16 – 18 se uporabljajo za izbiro 8 pogojnih bitov ki so v registru, ki se (obnovijo) loada ob vzponu ↑ clock A signala.
Če je CWR11 = 1 potem se dovoli skok, če izbran pogojni bit ustreza stanju "true". Če je CWR 11 = 0 potem se dovoli skok če izbran pogojni bit ustreza stanju FALSE. Testiranje pogoja ob koncu mikroinstrukcijske periode (n-1) se uporablja za omogočanje jump-a ob koncu mikroinstrukcijske periode n.
Vsebina lokacije v vektor File registru se uporablja

za to da se lahko izbere slush na eno izmed 16 lokacij v tabeli. Vektor register pill lokacije dobijo informacije iz mikblov na FBUS00 — FBUS03 iz mikroprocesorja pod kontrolo mikroprocesorske I/O funkcije. V fili se opisuje ob $\overline{\text{OSREQ}}$ signalu na odresno CWR04 in CWR05, 06 starta vsake mikroinstrukcijske periode CLOCKA obnovi sinhronizacijski register na trenutno stanje za naslednje signale ki pomenijo interrupt popoje:
$\overline{\text{SLAVE}}$, $\overline{\text{WNPRI}}$, $\overline{\text{INIT}}$, $\overline{\text{TRDY}}$ in $\overline{\text{RIR}}$. Pomen teh signalov je naslednji:

LOW – AKTIVNO STANJE
ZE POGOJ ZA INTERRUPT
VRSTNI RED PO PRIORITETI

		INT. ADRS HEX
		MSB LSB C1 A03 A02 A01
$\overline{\text{SLAVE}}$	Če je $\overline{\text{IBBSY}}$ v high, PDP11 procesor deluje kot bus master in odresira kontrolerjev register.	3E X
	Če je $\overline{\text{IBBSY}}$ low potem je kontrolor ~~portal~~ bus master med request/interrupt sekvenco	3FC
$\overline{\text{WNPRI}}$	Indicira, da smo sprejeli SLAVE SYNC iz PDP11 spomina med NPR sekvenco v kateri smo podatke prenesli iz PDP11 spomina v kontroler	3F3
$\overline{\text{INIT}}$	Indicira, da je PDP11 INITL signal aktiven	3F4
$\overline{\text{TRDY}}$	Tape drive je v stanju NOT READY in ni smo mu poslali komando za prenos	3F5
$\overline{\text{RIR}}$	Read input ready.	3F6

Izhodi sinhronizacijskega registra so odvisni v prioriteti encoder, ki je enabliran kadar je CWR35=0. Ko je prioriteti encoder enabliran in sprejema enega ali več aktivnih interrupt signalov, potem dobimo na izhodu signala INT in $\overline{\text{INT}}$. INT signal izzabliva izhodna vrata mikroprocesorskega kontrolnega modula 2910, $\overline{\text{INT}}$ signal pa enablira interrupt vrata in multiplekser, ki doje interrupt odresno na CSA00 do CSA03.

Tisti interrupt signal, ki ima najvišjo prioriteto določa izbrano interrupt odresno. Kadar je sinhronizirana verzija $\overline{\text{SLAVE}}$ signala aktivna, zunanji izbira interrupt odrese tudi od stanja $\overline{\text{IBBSY}}$ signala ki identificira transakcijo v kateri je PDP11 bus master in kjer ji kontrolor bus slave. Ta popoj forsira interrupt odresno

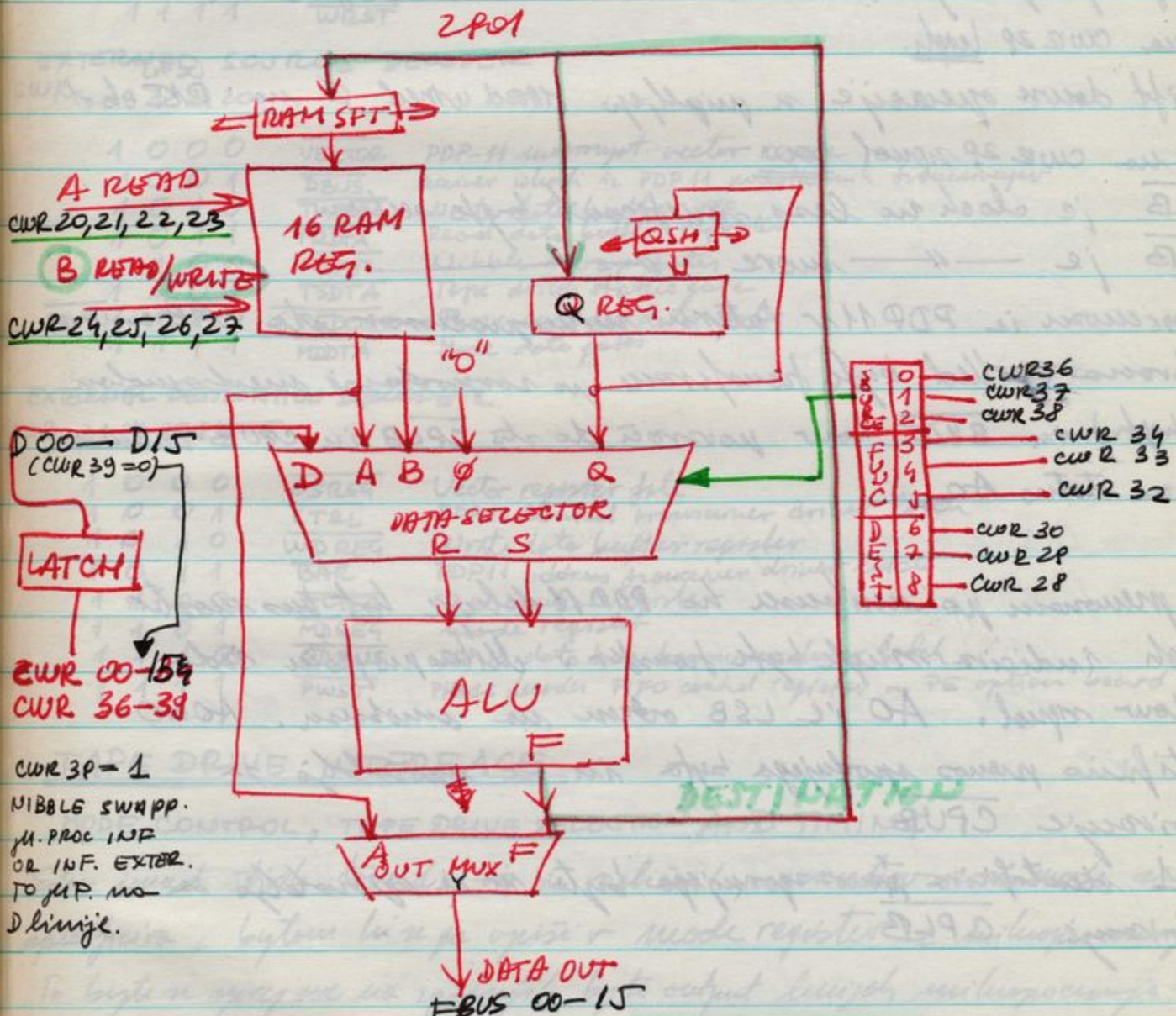
MSB LSB

na 3EX, kjer je X = C1, A3, A2, A0

smer prenosa — odresa kontrolerjevega registra

Kadar sta SLAVE in IBBSY oba low to pomeni da je kontroler bus master med request/interrupt selekcijo, di kadar je SLAVE high in ni je edinen kateri izmed low priority interrupt signalov, potem je interrupt odreso 3FY. V tem primeru je Y določen z A0 A1 A2 izhodi iz interrupt enkoderja.

INTERNE OPERACIJE MIKROPROCESORJA



Nibble swapp pomeni da least significant nibble prihajočimo na MSB nibble pozicijo na Dbusu vsak ostal nibble gre skozi pa je prihajočemu na eno nibble pozicijo nižje.

DBUS BIT	D15 D14 D13 D12	D11 D10 D9 D8	D7 D6 D5 D4	D3 D2 D1 D0
FBUS BIT	03 02 01 00	15 14 13 12	11 10 09 08	07 06 05 04

CWR			SELECTED SOURCE	
38	37	36	R	S
0	0	0	A	Q
0	0	1	A	B
0	1	0	0	Q
0	1	1	0	B
1	0	0	0	A
1	0	1	D	A
1	1	0	D	Q
1	1	1	D	0

CWR			FUNCTION
34	33	32	
0	0	0	R+S
0	0	1	S-R-1
0	1	0	R-S-1
0	1	1	R∨S
1	0	0	R&S
1	0	1	$\overline{R}$&S
1	1	0	R⊕S
1	1	1	$\overline{R+S}$

CWR			DESTINATION
30	29	28	
0	0	0	Y→FBUS, Q
0	0	1	Y→FBUS
0	1	0	Y→B, A→FBUS
0	1	1	Y→FBUS, B
1	0	0	Y→FBUS; $Y_i \to B_{i-1}$ $Q_i \to Q_{i-1}$
1	0	1	Y→FBUS; $Y_i \to B_{i-1}$
1	1	0	Y→FBUS, $Y_i \to B_{i+1}$ $Q_i \to Q_{i+1}$
1	1	1	Y→FBUS, $Y_i \to B_{i+1}$

Carry input = log "0".

Za shift up operacije z enobitno end around carry iz $\overset{RIT}{Q_{15}}$ na $\overset{RO}{Q_0}$ bot odziv na CWR 28 high.

Za shift down operacije z pipeliji hkrati vpred 0 na $\overset{Q\,IN}{RIT}$ bot odziv na CWR 28 nivoj low

$\overline{CPLB}$ je clock za less significant byte section

$\overline{CPUB}$ je —"— more —"—

OB premosn iz PDP11 v interni mikroprocesor sta $\overline{CPLB}$ in $\overline{CPUB}$ sinhrona. Med byte transferom pa razpoznavanje med vsebino CO (high) in $\overline{BYTE}$ low povzroči do sta $\overline{CPLB}$ in $\overline{CPUB}$ vzbudimo glede na status A0.

Med premosn po unibusu ko PDP11 delaje bot bus master CO high indicira single byte transfer. Mikroprogram vzdo $\overline{BYTE}$ low signal. A0 je LSB odsek na unibusu. A0=0 identificira prenos spodnjega byta in se uporablja za inhibiranje $\overline{CPUB}$

A0 = 1 identificira prenos zgornjega byta in se uporablja za inhibiranje $\overline{CPLB}$

I/O DEKODERJI

Imamo 3 dekoderje ki dajejo signale za izvajanje kontrolne funkcije dipa izbirajo izvajanje izvore in ponore.

External control functions dekoder:

CWR 19 18 17 16 — LOW ACTIVE

1 0 0 0	$\overline{\text{BYTE}}$	
1 0 0 1	$\overline{\text{SSYN}}$	
1 0 1 0	$\overline{\text{SUPR}}$	
1 0 1 1	$\overline{\text{SINT}}$	
1 1 0 0	$\overline{\text{FICLR}}$	
1 1 0 1	$\overline{\text{CLRIR}}$	
1 1 1 0	$\overline{\text{GENCLR}}$	
1 1 1 1	$\overline{\text{WRST}}$	

EXTERNAL SOURCE DECODER

CWR 39 22 21 20

1 0 0 0	$\overline{\text{VECTOR}}$	PDP-11 interrupt-vector source
1 0 0 1	$\overline{\text{DBUS}}$	Receiver izhodi iz PDP 11 podatkovnih transceiverjev
1 0 1 0	$\overline{\text{TWDTA}}$	Write data buffer reg.
1 0 1 1	$\overline{\text{TRDTA}}$	Read data buffer register
1 1 0 0	$\overline{\text{NSFTL}}$	Nibble swap gates
1 1 0 1	$\overline{\text{TSDTA}}$	Tape drive status gate
1 1 1 0	$\overline{\text{PEDOS}}$	Phase encoded data gates
1 1 1 1	$\overline{\text{MDDTA}}$	Mode data gates

EXTERNAL DESTINATION DEKODER

CWR 31 26 25 24

1 0 0 0	$\overline{\text{OSREG}}$	Vector register file
1 0 0 1	$\overline{\text{CTRL}}$	PDP 11 Control transceiver driver latch
1 0 1 0	$\overline{\text{WD REG}}$	Write data buffer register
1 0 1 1	$\overline{\text{BAR}}$	PDP11 address transceiver driver latch
1 1 0 0	$\overline{\text{FCREG}}$	Tape control register
1 1 0 1	$\overline{\text{MDREG}}$	mode register
1 1 1 0	$\overline{\text{ODBUS}}$	PDP 11 data bus transceiver driver latch
1 1 1 1	$\overline{\text{PWST}}$	PHase decoder FIFO control register on PE option board.

TAPE DRIVE INTERFACE

MODE CONTROL, TAPE DRIVE SELECTION AND TIMING

Eden izmed štirih tape drive-ov in različnih parametrov za mode se specificira z bytom ki se zapiše v mode register z mikroprocesorjem. To byte se vpiše na zgornjih byte output linijah mikroprocesorja. FBUS 08 ÷ FBUS 15 in se vpiše v mode register kot odziv na ~~MOB~~ $\overline{\text{MD REG}}$ signal iz mikroprocesorja

FBUS 15 SC1 } ti biti kontrolirajo PDP11 CO in C1 linije indicirajo smer
FBUS 14 SCO } prenosa podatkov, kodor kontroler igra vlogo bus master naprave.

FBUS13 DE5 Kadar je aktiven NR2I način ti biti določajo gostoto zapisa
FBUS12 DE8 na traku

DE8	DE5			
0	0	7 sledi	200 BPI	
0	1	7 sledi	556 BPI	
1	0	7 sledi	800 BPI	
1	1	9 sledi	800 BPI	ali 7 track NIBBLE način.

FBUS11 O/EPS O = odd parity, 1 = even parity
FBUS10 HID za 1600BPI phase-encoded/800 BPI, NR2I drive
HID = 1 specificira 1600BPI phase-encoded način.
FBUS09 SE1 } Tape drive selection bits
FBUS08 SE0 }

(5)

Trije PROMi dajejo signale, ki kontrolirajo način, tape drive selecijo in timing. Štirje izmed petih odresnih bitov ki ostopajo v PROMs so: SE0, SE1, DE5 in DE8 (to so biti iz mode registra. Peti odresni bit ki je vhod v PROM je (PE) = HID + $\overline{NR2S}$, kjer je HID bit iz mode registra, NR2S pa je kvocijent iz PE kartice.

Frekvenca 48 CLK signala je določena s petimi PROM izhodi na ÷2 flip flop in oscilator/rate multiplexor, ki določa frekvenco 48 CLK signala. To signal, ki se uporablja na Phase encoded option kartici, mora biti 48 ~~bit~~ bit rate, kadar je phase-encoded mode aktiven. 48CLK signal daje rate vhod v programski delilnik ki daje signale ~~TACL~~ TGAPCLK in WRSTBCLK signale.

TGAPCLK se uporablja za dolbcijo GAP na traku.
Znaša 4X bit rate za NR2I način
8X —"— PE način

WRSTBCLK je clock vhod na write strobe logiko in znaša
2X bit rate za NR2I način
4X —"— PE —"—

Konstante za deljenje daje 8 signalov iz C16 proma. Oba signala TGAPCLK in WRSTBCLK sta odvisna tako od programiranega delilnega faktorja, kot od 48CLK razmerja.

Bit rate je produkt hitrosti traku in gostote posnetka. Vsak izmed 4 tape drive enot deluje s fiksno hitrostjo. Promi so programirani da kadar izberemo določen tape drive in določeno gostoto zapisa dobimo bit rate ki je produkt hitrosti izbranega drive in izbrane gostote.

Druga karakteristika vsakega tape driva je format zapisa 7 sledi ali 9 sledi. Eden izmed Promov daje 7TRK signal ki identificira format zapisa ki ga ima posamezni drive. Ta signal se uporablja kot bit v TAPE DRIVE STATUS WORD, ki je lahko naslovljeno z mikroprocesorjem. Drugi izhodi iz PROM-a (ki služijo za izbiro načina) se uporabljajo kot biti v MODE WORD-u, ki jo lahko čita mikroprocesor. Ta beseda gre na μProc. input data bus linije D00–D11 kot odgovor na $\overline{\text{MDDTA}}$ signal iz mikroprocesorja.

BITI NA D-BUSU	BIT	
D11	$\overline{\text{SEL1}}$	Izbran je tape drive 1
D10	$\overline{\text{SEL2}}$	—— " —— 2
D09	$\overline{\text{SEL3}}$	—— " —— 3
D08	$\overline{\text{SEL0}}$	—— " —— 0
D07	NRZS	NRZI način izbire signala iz izbranega tape drive-a preko PE kartice
D06	IBM PK	za 9 sledni format, specificira da je zadnji byte vrste računalniške besede vpisan ali prečitan 1. znak na traku spodnji byte pa v drugi znak na traku
D05	NIBMODE	Za 7 tračni format, specificira da je rač. beseda razstavljena na 4 nibble ki se vpišejo na zaporedna mesta na traku ali pa se čitajo z njih.
D04	(DATA LOST)	Med čitanjem traku smo izgubili podatke nad non proc. requestom ker ga nismo pravočasno servisirali (ta signal dobimo ob tape read funkciji)
D03	1600	izbran je 1600 bpi PE način
D02	SPD2	to je most significant bit hitrosti traku
D01	SPD1	(MSB – 1) tape speed
D00	SPD0	LSB bit hitrosti traku

SEL0 do SEL3 signali gredo na driverje ki kontrolirajo izhodne signale za 4 tape drive enote ($\overline{\text{SEL0}}$ do $\overline{\text{SEL3}}$). Normalno so SEL0 do 3

zaporedno dekodirane SE1 do SE0 vrednosti 00, 01, 10, 11. Prom lahko programiramo tako, da dobimo dvopočno relacijo med SE1 - SE0 vrednostmi in SEL0 do SEL3 signali. V tem primeru se izbira logičnega tape drive (SE1 - SE0) razlikuje od izbire fizične tape drive enote, ki jo dajejo SEL0 do SEL3 signali. Imamo matriko jumperjev, s katero lahko priključimo $\overline{SEL0}$ do $\overline{SEL3}$ signale na ($\overline{SEL0*}$) do ($\overline{SEL*}$) tako da ti signali predstavljajo logično izbiro kolov.

Izhodni signal iz PROM-a PE ENB gre na PE kartico in specificira PE ~~tea~~ način delovanja.
HI DEN gre na driver ki kontrolira $\overline{PE}$/NRZ linijo na tape drive.
Izbira 1600 BPI PE / 800 BPI NRZI drive.

65

Write strobe logika se aktivira ko gre WOR signal v high, kar pomeni da je znak pripravljen v izhodnem registru FIFO za zapis znaka. To je konjunkcija OUTPUT ready signalov FIFO registra.
Či je WOR high v času ⎍ WRSTBCLK, potem dobimo ARCLK pulz, ki mu sledi WRSTB pulz za vsaka dva WRSTBCLK impulza.
ARCLK se uporablja za premikanje znaka iz izhodnega registra FIFO vezja v register ki krmili podatkovne linije tape drive enote
WRSTB potem stroba znak v tape drive enoto.
(Za PE metodo imamo prenos 2 znakov in med vsako periodo pride do dveh WRSTB impulzov). (Pravi podatki se prenesejo v prvi polovici v drugi polovici pa se prenese komplement. To generira fazno kodirano obliko. Z namenom, da dobimo dvojno prenosno hitrost je vzmeje za WRSTBCLK izbrano 4X bit rate za fazno kodirano način ki je bil omenjen prej)

Ko pošljemo zadnji znak zapisa na tape drive je WOR low v naslednjem ⎍ WRSTBCLK. To prepreči Write strobe logiko

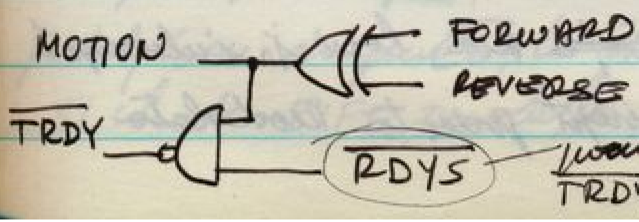
da bi dala WRSTB puls. Če je PEENP low, kar indicira, da je NRZI način aktiven, potem ~~write ojačevalnik resetira impulz~~ UVRS to je impulz za resetiranje write ojačevalnika.

Kodor je enabliran z DGIN iz tape kontrol registra tedaj ~~de~~logika ki detektira GAP akumulira seštevek TGAPCLK impulzov, ki so se pojavili med zaporednimi RDSTB impulzi. Pri NZRI načinu GAP v dolžini 3 bitov omogoča seštevek do 12, kar povzroči da ~~se~~ naslednji TGAPCLK setira TGAP. TGAP potem ostane resetiran dokler mikroprocesor ne resetira DGIN. Logika za detekcijo gap-a deluje na identični način tudi v primeru PE samo da se v tem primeru impulzi pojavijo 8X bit rate to pomeni da se seštevek 12 akumulira v ~~času~~ ~~dveh~~ četrtini polbitne periode.

TAPE CONTROL IN STATUSNI INTERFACE

Kontrolne linije v tape drive grejo preko tape control registra in driverjev. TAPE CONTROL REGISTER [6] se loada iz linij FBUS00 ~ FBUS07 kot odziv na $\overline{TCREG}$ signal iz mikroprocesorja

	CONTROL BIT	DRIVER OUTPUT	
FBUS 07	D/PS		data/parity selekcija. Določa izvor paritete za znak na traku med vpisovanjem na trak. Če je D/PS=0 se izbere izhod iz paritetnega generatorja. Če je D/PS=1 se izbere 9. podatkovni bit iz write buffer registra.
FBUS06	DGIN		Data gate signal enablira tape read funkcijo na PE kartici in gap detekcijo
FBUS05		$\overline{OVERWR}$	Overwrite signal na tape drive, ta signal povzroči da tape drive izklopi tok za vpisovanje, kot odziv na WRITE AMPLIFIRE reset signal da prepreči na brisanje naslednjega zapisa pri editiranju.
FBUS04	WRITE	$\overline{WREN}$	Write pogoj za PE opcijsko kartico za write operacijo. $\overline{WREN}$ vklopi vpisovalni tok.
FBUS 03		$\overline{OFF LINE}$	povzroči da gre izbrani tape drive v offline
FBUS02		$\overline{REW}$	Povzroči da se izbrani drive previje v loadpoint pozicijo.
FBUS01		$\overline{SREV}$	Reverse motion komanda
FBUS00	FORWARD	$\overline{SFWD}$	FORWARD pogoj na PE kartico za premik v naprej. $\overline{SFWD}$ je forward motion komanda na izbrani tape drive



$\overline{RDYS}$ — pomeni da izbrani tape drive ni ready
$\overline{TRDY}$ low se uporablja, da da INTERRUPT mikroprograma
[illegible]

Za čitanje statusa tape drive enote, mikroprocesor postavi $\overline{TSDA}$ signal v LOW. To omogoči tape drive status invert vrata, ki dajo status tape drive enote na linije D00 do D07 in D10.

D BUS BIT	TAPE DRIVE STATUS BIT	
D10	($\overline{EOT}$)	To je izhod iz EOT flip flopa. Ta flip flop se setira v odvisnosti od ustreznega $\overline{EOTS}$ signala iz tape drive enote. Ker mora ta je bil detektiran odbojni trak na traku, ko je bil EOT postavljen s clock impulzom ostane setiran dokler se ne pride CLR kot odziv na GENCLR komando.
D07	$\overline{NXM}$	Non existent memory flip flop v PDP11 interface enoti.
D06	$\overline{ONLNS}$	On line signal iz izbranega tape drive
D05	$\overline{BOTS}$	To je beginning of tape signal iz izbranega drive.
D04	$\overline{7TRK}$	to je 7 track signal iz izbrane drive enote
D03	$\overline{SPDS}$	Speed status signal
D02	$\overline{FPRTS}$	File protect status
D01	$\overline{REWS}$	Rewind status from selected tape drive
D00	$\overline{RDYS}$	Ready signal iz ~~—— " ——~~

Mikroprocesor lahko prečita phase-encode signal ali po end of file informacijo iz PE kartice. Ta informacija se vpiše na podatkovne linije D00 do D11 kot odziv na $\overline{PEDOS}$ signal iz mikroprocesorja.

D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
PEMULDO	PE ALLDO	$\overline{PE\ EOF}$	PEDOP	PEDO0	PEDO1	PEDO2	PEDO3	PEDO4	PEDO5	PEDO 6	PEDO7
multi channel dropout	L not used	end of file	dropouts of individual channels								

TAPE READ FUNCTION

$\overline{PEOPT}$ signal se sprejme iz PE opcijske kartice. Ta postavi PEOPT v high in onemogoči tape drive read data vrata. To dopušča read data strobe, read strobe in NRZ1 izbirne signale, da se sprejmejo na RDB0-RDB7, RDBP in NRZS linije okrog PE kartice.

Vsak READ STROBE IMPULZ na RDSTB liniji vpisuje podatke na RDPB in RDB0 — RDB7 linijah v ~~READ DATA BUF REG~~ 5 in setira read input ready flag (RIR) na low. ~~RIR~~ RIR low da interrupt mikroprocesorju. Rutina, ki servisira interrupt prečita Read data

buffer register in resetira $\overline{DIR}$ signal na High. To funkcijo izvedejo mikroprocesorski signali $\overline{TRDTA}$ in $\overline{CLRDIR}$.
$\overline{TRDTA}$ postavi read data buffer register na linije D00 do D08 in postavi hard wired 0 na linije D09~D15.

TAPE WRITE FUNCTION

Med opisovanjem na trak se vsak znak prenese iz mikroprocesorja preko FBUS-a v write data buffer register s signalom $\overline{WD REG}$ iz mikroprocesorja. Write data buffer register ima spomino za 9 bitov (FBUS 00 ~ FBUS 08). 8 least significant bits grejo v PARITY CHECKER in v FIFO. Če je D/PS signal iz kontrolnega registra v LOW se kot deveti bit vzame bit iz parity generatorja in se go vodi v FIFO. Kateri izhod iz parity generatorja vzamemo zavisi od signala O/EPS iz MODE REGISTRA. LIH če je O/EPS LOW, SOD če je O/EPS HIGH. Če je D/PS high potem je deveti bit izhod iz write data buffer registra za opis v FIFO. Poleg izbire ~~bita~~ izvora za 9 bit lahko mikroprocesor kontrolira tudi polariteto le tega. Ko je mikroprocesorjev signal $\overline{TWDTA}$ high (neaktiven) je izhodni 8 bit invertiran, če je LOW pa neinvertiran. To zmožnost se uporablja za generacijo all zero preamble in postamble znaka.

Devet bitov, ki jih lahko pripeljemo na FIFO se lahko prečita v mikroprocesor preko WRITE DATA TO D-BUS vrat. Ta vrata enablira low $\overline{TWDTA}$ signal. Mikroprocesor podatke po tej poti za računanje CRC vrednosti.

Kadar je vstopni register v FIFO prost nam FIFO da WIR signal. Mikroprocesor potem izda $\overline{WRSI}$ komando ki je potrebna za loadanje FIFO enote. $\overline{WRSI}$ signal je dolg 1/4 clock periode se podaljša na dolžino 1 clock periode z vpisom v register, ki se ga updatira z vsakim ⎍ prehodom $\overline{clock}$ signala. Ta podaljšani impulz, ki se pojavi na WSI liniji se pripelje na SI vhode FIFO modula za opis v FIFO vhodni register

STR A2
(2)

Ko je znak vpisan v FIFO propagira proti izhodu. Ko znak pride v izhodni register FIFO enote se WOR signal preklopi v high. WOR se uporablja v Mode control in timing logiki za enabliranje prenosa znaka iz izhodnega registra FIFO enote v write output register in strobira znak v TAPE DRIVE enoto. Prenos se izvede kot odgovor na AR CLK impulz iz mode control in timing logike. Temu sledi WRSTB impulz ki se po pošlje v $\overline{\text{WRSK}}$ linijo TAPE DRIVE enote.

Ko se AR CLK preklopi v high, se WOR signal resetira na LOW. Ko se AR CLK vrne na LOW nivo naslednji znak, či je prisoten propagira v izhodni register FIFO enote in WOR se spet preklopi v high. Dokler so v FIFO registru na voljo znaki se ti s primerno hitrostjo prenašajo v write output register in se strobirajo v tračno enoto s hitrostjo prenosa ki ustreza hitrosti traku in gostoti posnetka.

Ko je WOR low v času, ko bi moralo priti do vpisa naslednjega znaka v izhodni register, se write operacija konča. V tem primeru za NRZI način ~~write operation reset WARS~~ sprejmemo WARS impulz za resetiranje write ojačevalnika. WARS (reset) impulz pride iz mode control in timing logike v času ko bi moral sprejeti WRSTB impulz. To impulz krmili $\overline{\text{WARS}}$ linijo na tape drive.

Mikroprocesor inicializira WRITE character FIFO v smislu $\overline{\text{FICLR}}$ komande. $\overline{\text{FICLR}}$ signal, ki je podaljšan s clock impulzom pripeljemo na WRITE CHARACTER FIFO na $\overline{\text{FIFOCLR}}$ linijo. FIFOCLR zbriše vse informacije v FIFO in postavi WIR signal na high nivo in WOR signal na LOW nivo. WOR ostane v LOW dokler se nov znak ne vpiše v FIFO in propagira do izhoda.

Podaljšan signal je tudi $\overline{\text{GENCLR}}$ (glej na CLR in $\overline{\text{RERST}}$ linij) Ob vklopu R-C vezje da signale na $\overline{\text{CLR}}$ / $\overline{\text{RERST}}$ in $\overline{\text{FIFOCLR}}$ linije za inicializiranje kontrolerja.

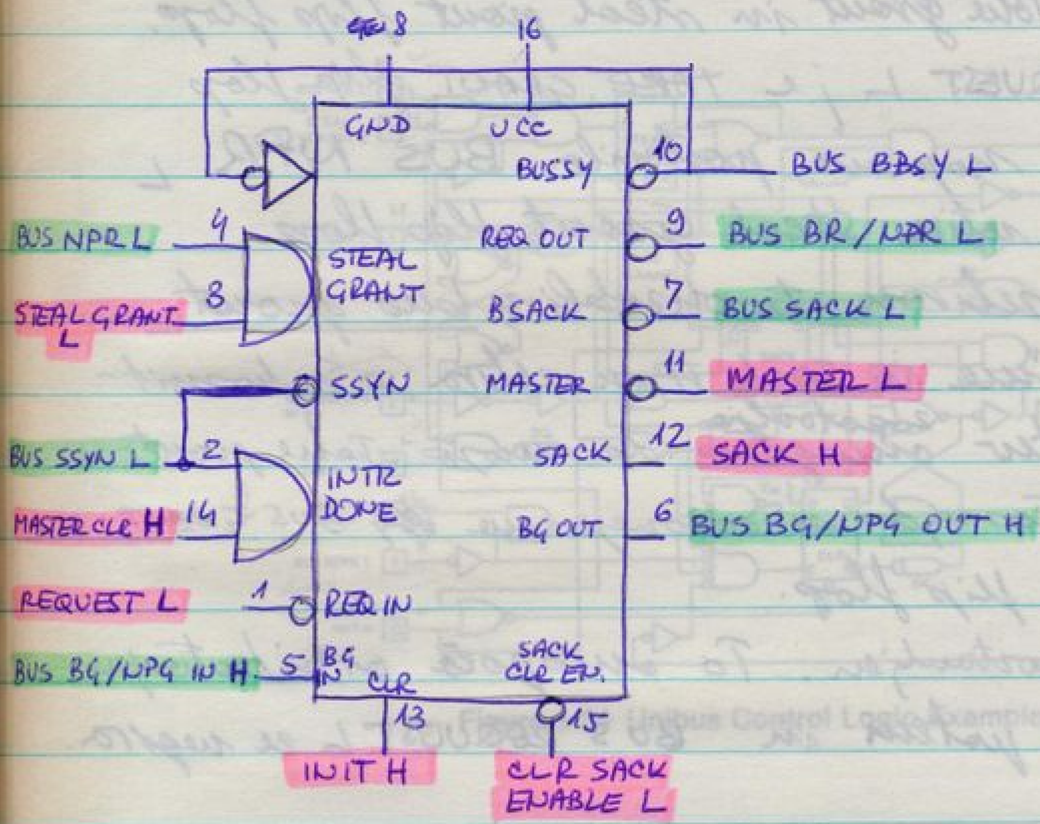
UNIBUS CONTROL LOGIC (8647 unibus protokol čip)

1) Signali priključeni direktno na unibus linije:

BUS NPR L
BUS REQUEST L
BUS GRANT IN H
BUS GRANT OUT H
BUS SACK L
BUS BBSY L
BUS SSYN L

2) Signali priključeni glede na notranje vezja:

REQUEST L
MASTER CLR H
STEAL GRANT L
MASTER L
INIT H
CLR SACK ENB L
SACK . H



[1]

Request L generira master naprava. S tem, ko ga postavi v L se sproži PRIORITY TRANSFER sekvenca. Ko je enkrat aktiven, moramo REQUEST L negirati in zopet postaviti predno začnemo naslednji cikel. (MASTER CLEAR H [14] ne sme biti na GROUND, nasprotno ne sme biti že v stanju bus master in ne sme dopustiti BUS BBSY L). Pri resetiranem SACK flip flopu se postavi BUS REQUEST L.

Naprava ne more sprožiti sekvence za prioritetni

prenos ča je že bus master, ker morata biti BUS BBSY L in BUS SACK L negirano predno postavimo BUS SACK L. Naprava negira BUS REQUEST L hodor postavlja BUS SACK L.
BUS SACK L mora biti postavljen istočasno ali pa preden je BR ali NPR negiran.
Stanje BUS BBSY L določa ali je zaprt BUS GRANT IN H prepušča naslednji vezje, ali blokira.
Ča je BUS BBSY L v LOW napravo zaprejme grant in ga blokira toliko da ne pride do drugi napravi. V tem primeru naprava želi postati naslednji bus master. Ča je BUS BBSY L v High grant pošlje v naslednjo napravo na vsem prostetnem nivoju.
BUS GRANT IN H prači tako grant in steal grant flip flop.
Ča je postavljen BUS REQUEST L je TAKE GRANT flip flop setiran, ča pa je druga naprava postavila BUS NPR L in STEAL GRANT L je setira Steal Grant flip flop.
Katerikoli flip flop je setiran to dirobliro bus grant driver; Grant se zaprejme in SACK FF se setira z zakasnitvijo 0ns. To zakasnitev zagotavlja omogoča da izvaja TAKE grant in steal grant dovolj časa, da se odzovejo na BUS GRANT IN predno se zgodi SACK flip flop.
[7] BUS SACK L je postavljen. To omogoča arbitratorju da negira grant 75 ns potem in BUS REQUEST L se negira.

BBSY flip flop se setira ko so izpolnjeni pogoji za clock input in ko so BUS GRANT IN H, BUS SSYN L in BUS BBSY L vsi negirani.

[10] [11] BUS BBSY L in MASTER L so postavljeni. BUS BBSY L ohranja bus dokler master ne izvede podatkovnega ali interrupt prenosa. MASTER L je signal, ki ga lahko uporablja master da zgodi prenos podatkov ali za interrupt sekvenco, uporablja se lahko za enoblokovanje

primernih podatkovnih linij in za bus interrupt.
Kadar naprava, ki je zahtevala prenos le tega zaključi, postavi MASTER CLR H [14]. Ko sta postavljena oba MASTER CLR H in BUS SSYN L (kar pomeni da se je prenos zaključil), se BBSY flip flop resetira. To negira MASTER L in po 80ns (Delay D4) negira BUS BBSY L.

CLR SACK ENABLE je dvičojno ozemljen.

INIT H direktno briše BBSY in SACK flip flop če je BUS GRANT negiran.

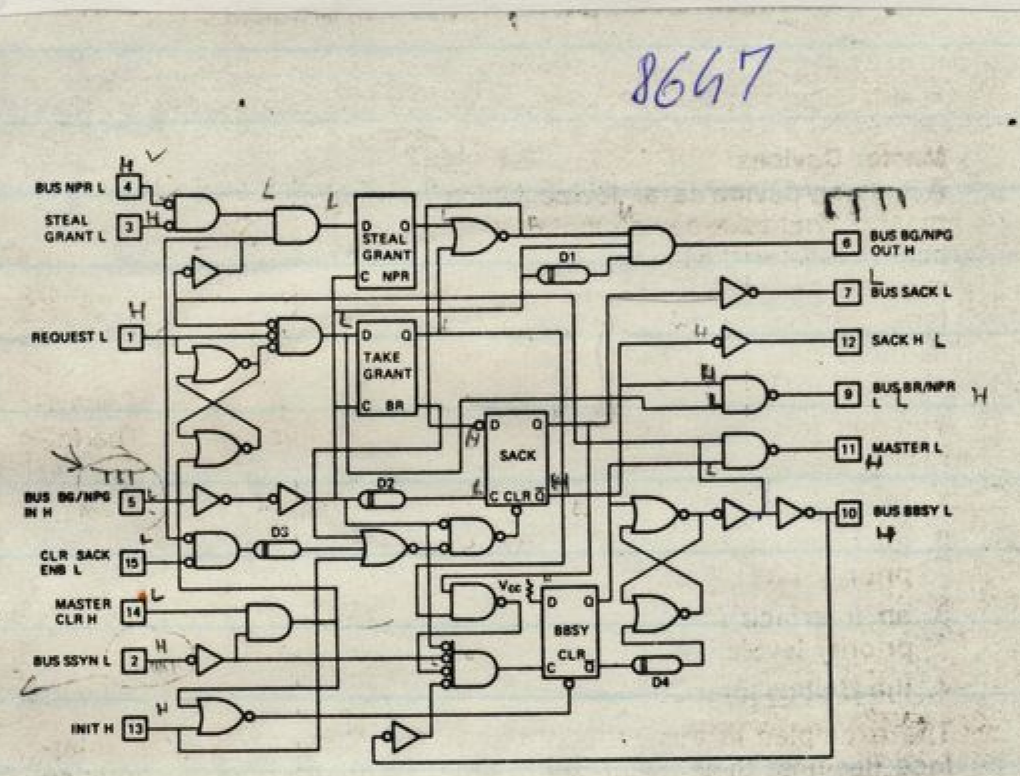


Figure 1-34 Unibus Control Logic Example

BR

Bus request sproži naprava kadar ji je potreben interrupt. Interrupt enable je bit v kontrolnem in statusnem registru naprave. Kadar je INT ENB H in INT H se postavi REQUEST L. To sproži na bus priority transfer sekvenco. Ko naprava sprejme grant, se postavi MASTER L in prične se interrupt sekvenca. Master negira REG SELECT L in IN L in disablira D 00-01 in D10-15

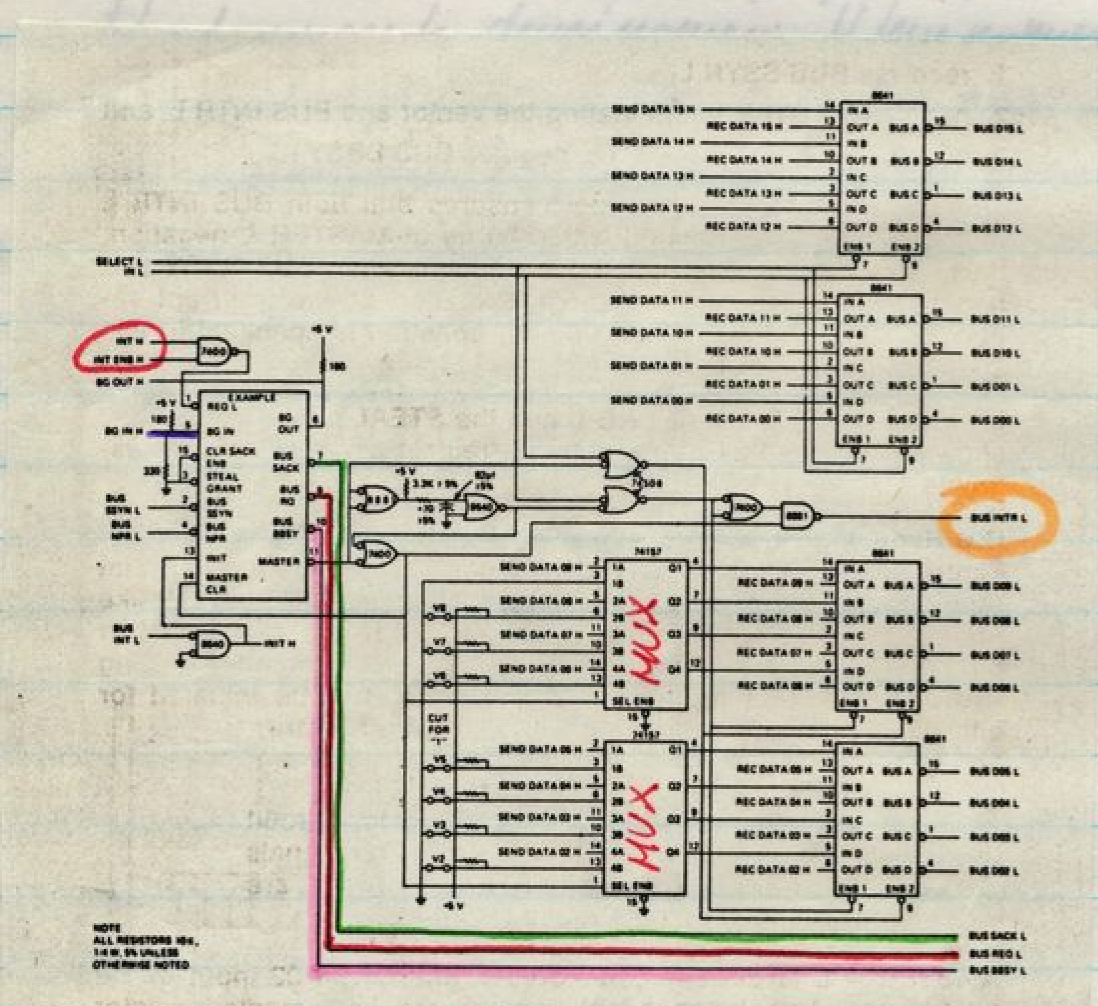


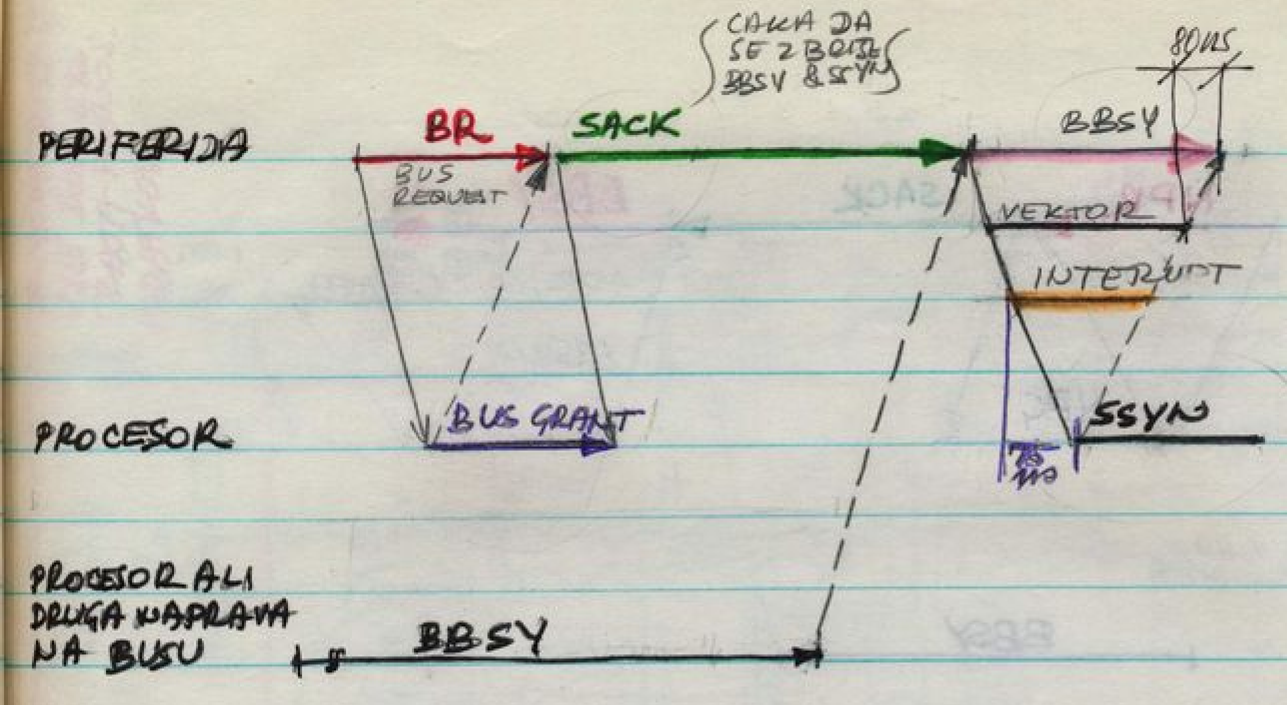
Figure 1-35 Bus Request (BR) Device—One Vector Circuit

1 2 3 4

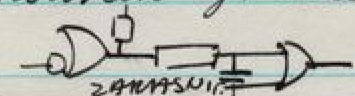
SEL vhod v MUX   0 na izhod gre A
                  1 — " — B

Postavitev MASTER L

1) enablira en vhod v BUS INTR. driver 8881
2) Aktivira SEL vhod v mux } VEKTOR NA BUS
3) ~~Ak~~ Aktivira ENB 2 linije transceiverjev ki pogonijo D<02:09>
4) Po zakasnitvi enablira vhod v BUS INTERRUPT INTR driver.



BUS INTR L u postavi potem, ko so transceiverji enoblirani. Interrupt vektor se postavi pred BUS INTRL

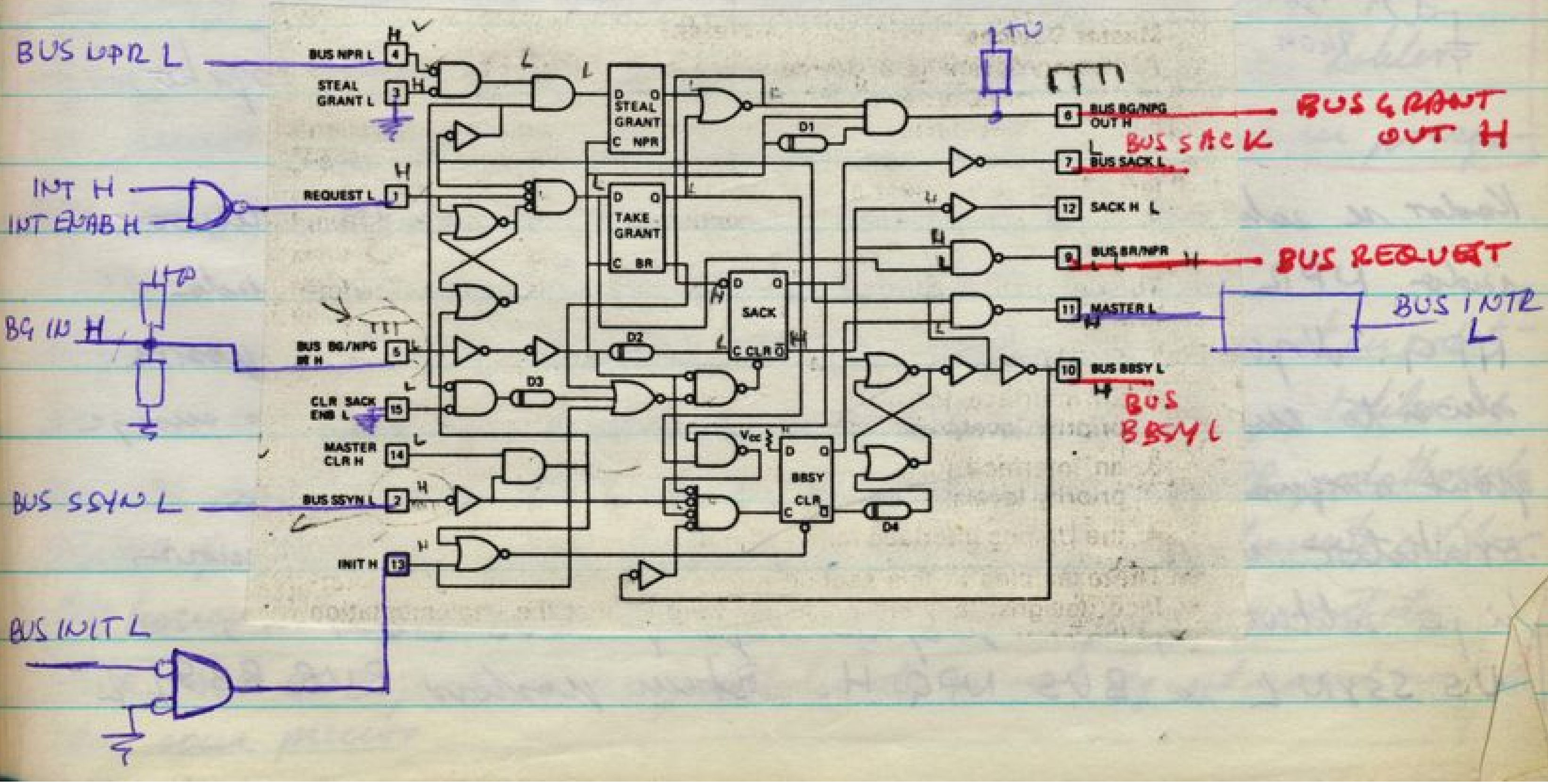


Ko processor sprejme BUS INTR L shrani vektor z D linij in pošlje BUS SSYN L

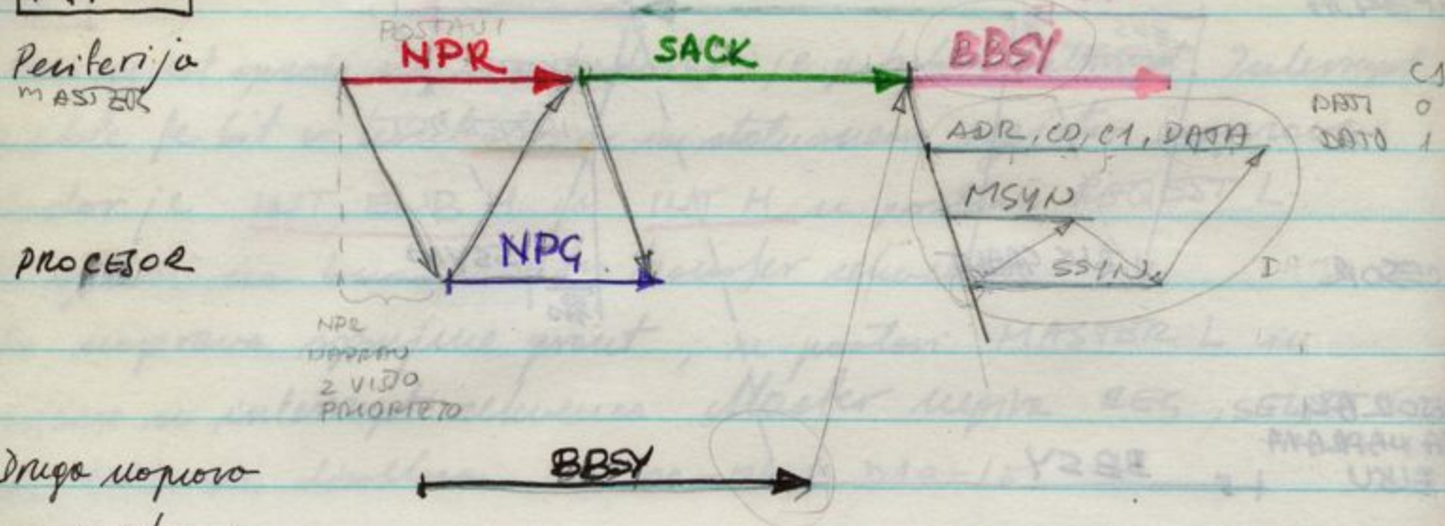
Naprava sedaj izvede:

1) Sprejme BUS SSYN L
2) Negira MASTER L terminira vektor in BUS INTR L in po
3) 80 ns negira BUS BBSY L

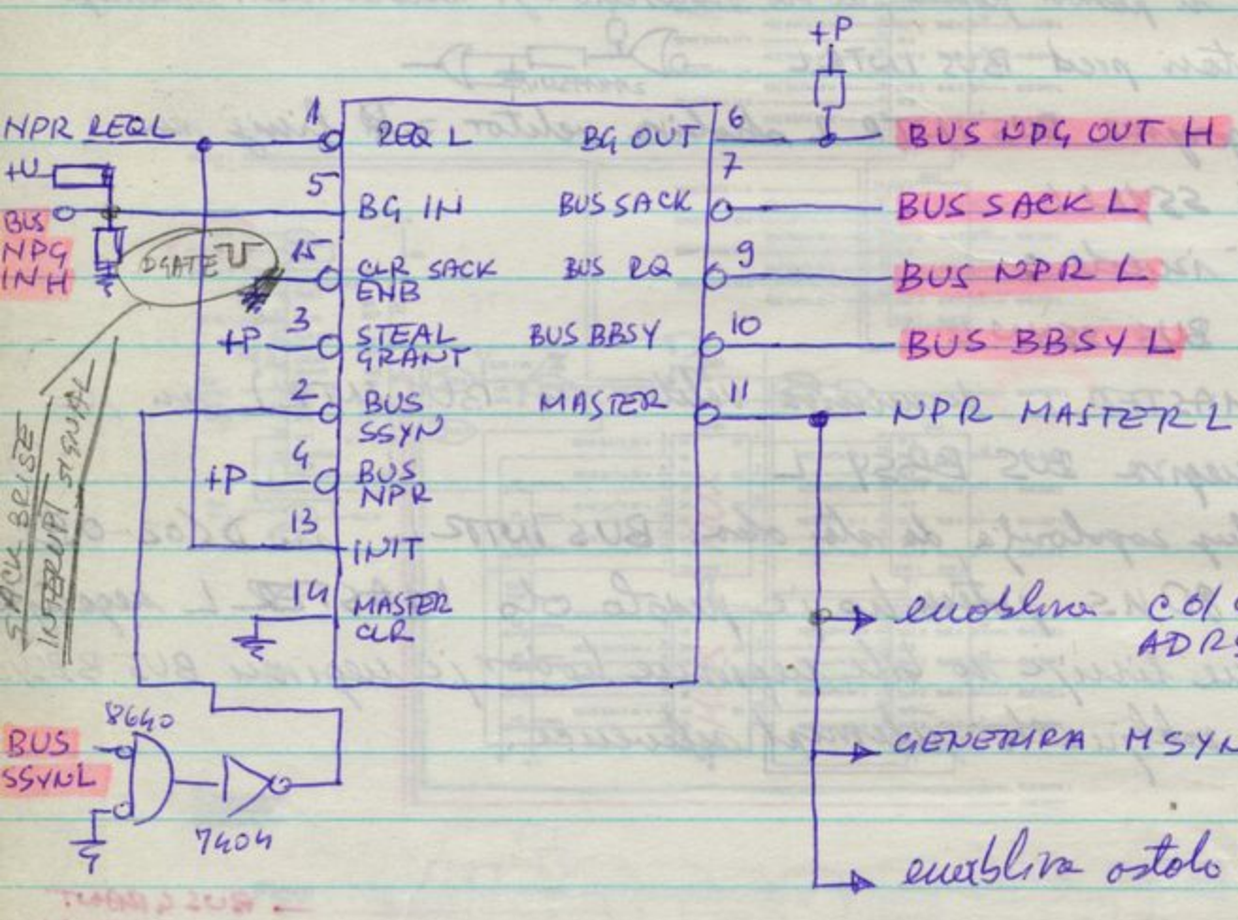
Uporaba Shottky zapolonja, da sta oba: BUS INTR L in D<02-09> negirana v 80 ns potem ko je prišlo do MASTER L negacije
Vse postavljene linije so bile negirane kadar je negiran BUS BBSYL
To omogoča zaključek interrupt sekvence.



NPR

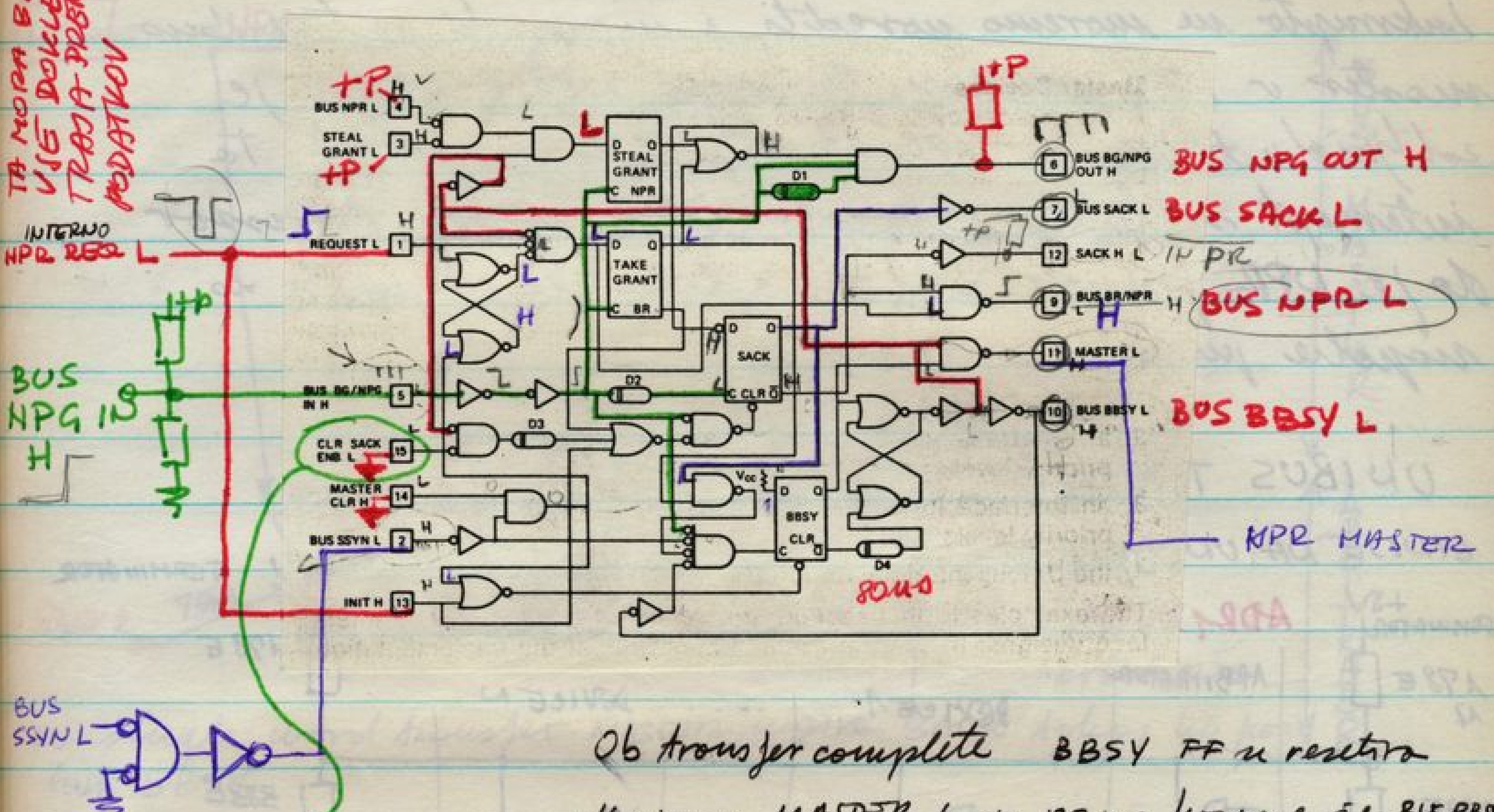


Drugo napravo na vmesnik



Kadar se zahteva NPR prenos, napravo, ki zahteva prenos izda NPR REQ L. Arbitrator prepozna ta request in izda NPG. Napravo ki zahteva request blokira grant, ki pozira skozi to enoto v naslednjo napravo in vsaka ne omogoči grant s signalom BUS SACK. Ko je bus SACK postavljen arbitrator neha grant in prenehe z arbitražo. Naprava ki je zahtevala prenos sprejme nepogoj BUS BBSY L, BUS SSYN L in BUS NPG H. Potem postavi BUS BBSY L

TA MORA BITI V L VSE DOKLER TRAJA PRENOS PODATKOV



Ob transfer complete BBSY FF u resetira
Negira u MASTER L in 80ns kasneje še BUS BBSY L

in s tem postane bus master. Kot bus master prične s prenosom podatkov.
NPR MASTER L se postavi istočasno kot BUS BBSY L. NPR MASTER L se uporablja za poganjanje zunanjega vezja. Generira (Enabla) lokalno podatke in odreže na data in adr linije, Generira lokalno MSYN L ..

Ko se prenos podatkov zaključi se kontrola nad busom zaključi z negiranjem INTERNI NPR REQ L signala. Ta signal moramo zopet postaviti če želimo naslednji prenos. NPR REQ L je direktno povezan na INIT H signal. Ko postavimo request mora ~~post~~ ostati postavljen dokler se prenos podatkov ne konča sicer se nadzor nad busom prezgodaj zaključi.

STEAL GRANT FF je disabliran
BUS NPR L = +3V
STEAL GRANT L = +3V

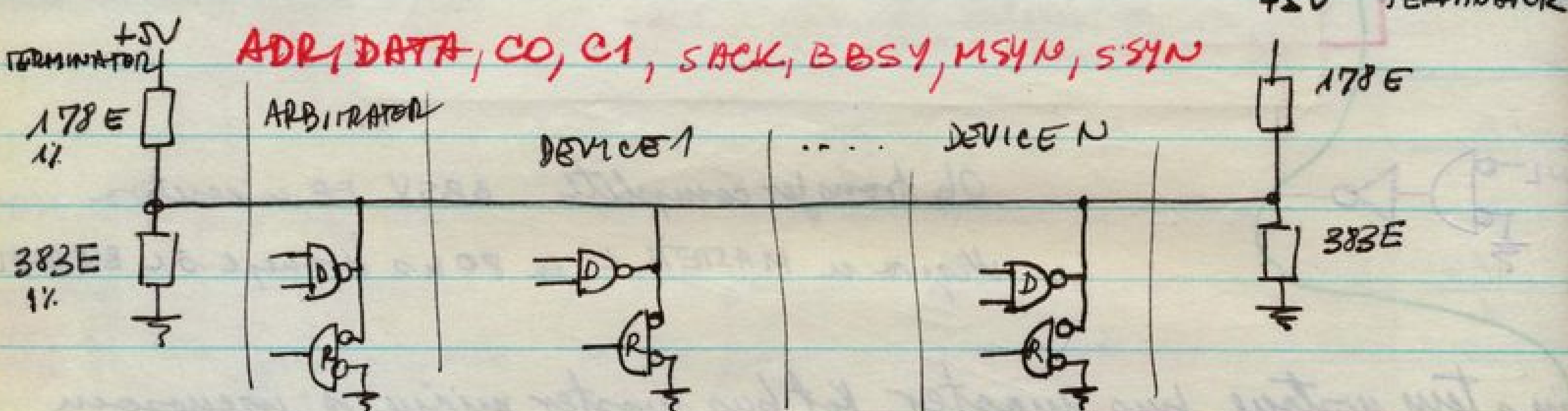
Negacija bus sack lahko preprečimo če potrebujemo več kot en podatkovni cikel. Clear SACK ENB je na High dokler ne začne zadnji bus cikel; Npr če prenašamo 100 NPR podatkovnih ciklov je ta vhod v high dokler se ne konča 99 podatkovnih ciklov. Ob koncu 100 prenosa pa dobi bus naprava z najvišjo prioriteto.

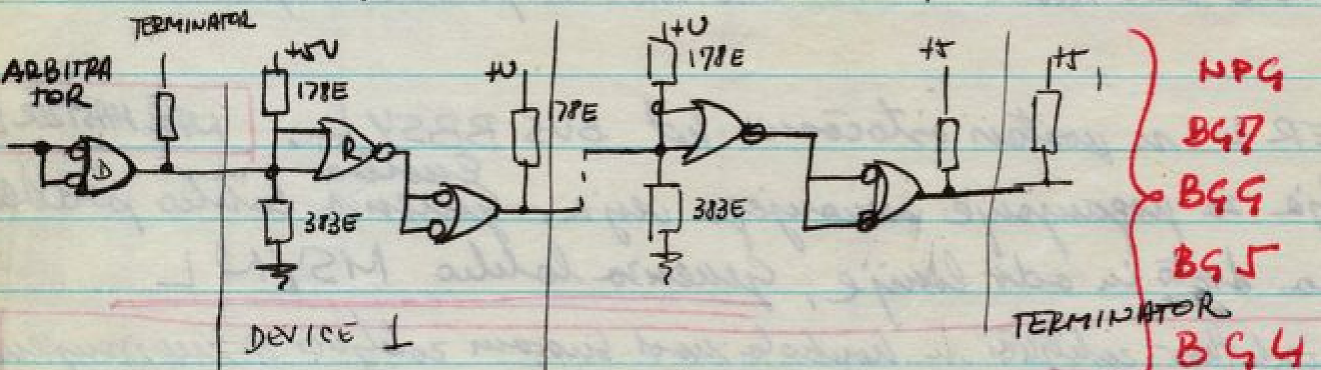
CLR SACK ENB je vezan na gnd tako da za vsak request naredimo en sam prenos.

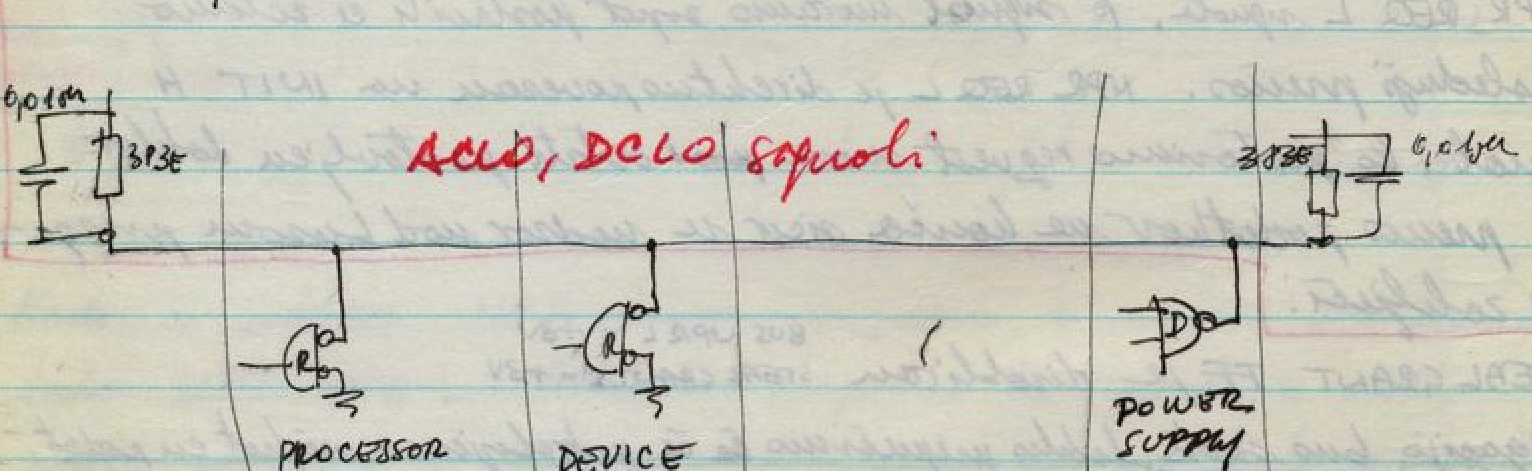
Interrupta ne moremo voroditi z napravo ki postane bus master v NPG sekvenci. Pri večini NPR aplikacij je zaključeku trenutnega NPR sekvence sledi interrupt. Ta interrupt se lahko uporabi da se obvesti processor da je NPR prenos zaključen ali pa da je prišlo do napake pri prenosu podatkov.

UNIBUS TIMING

LINIJE NA UNIBUSU:



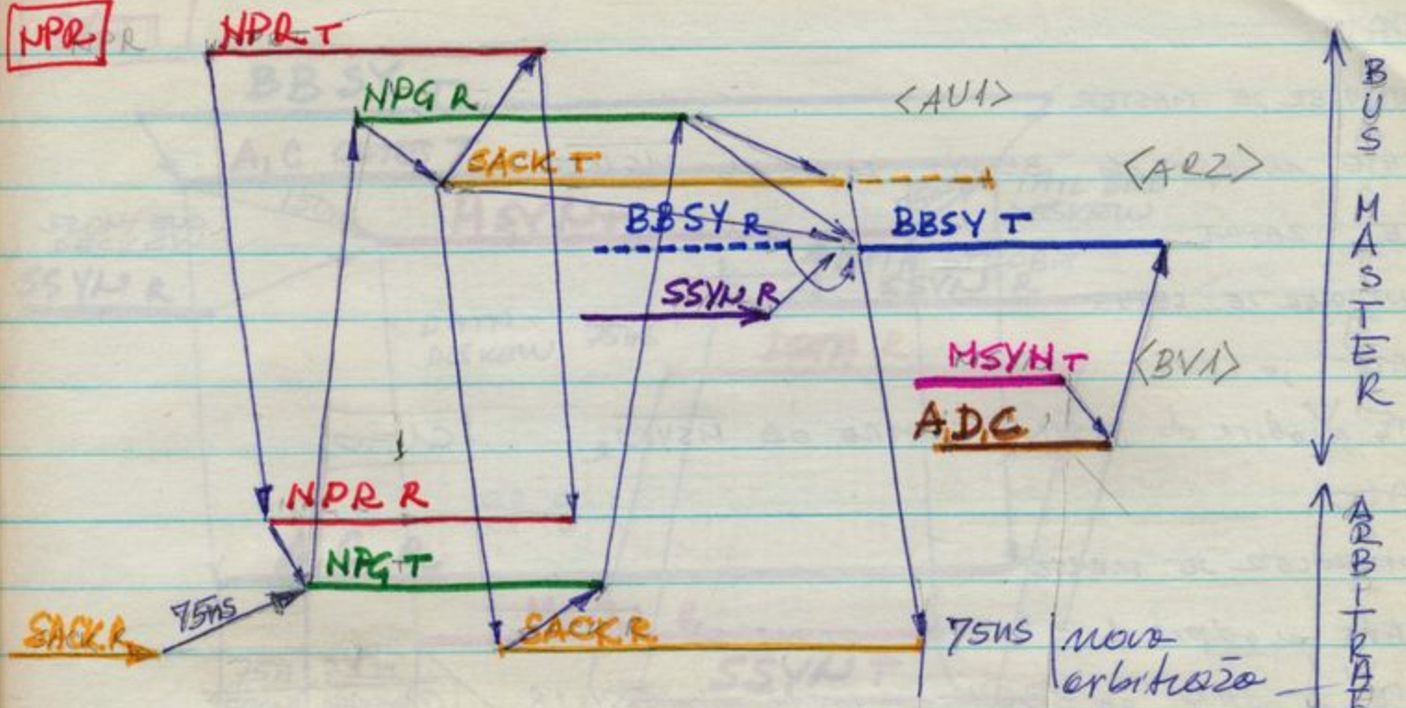




	C1	C0	PA	PB	
DATI	0	0	0	0	ni napake v glavi napora pri DATI in DATIP
DATIP	0	1	0	1	napaka v slave-u za DATI DATIP
DATO	1	0	1	X	neuporabljeno
DATOB	1	1			

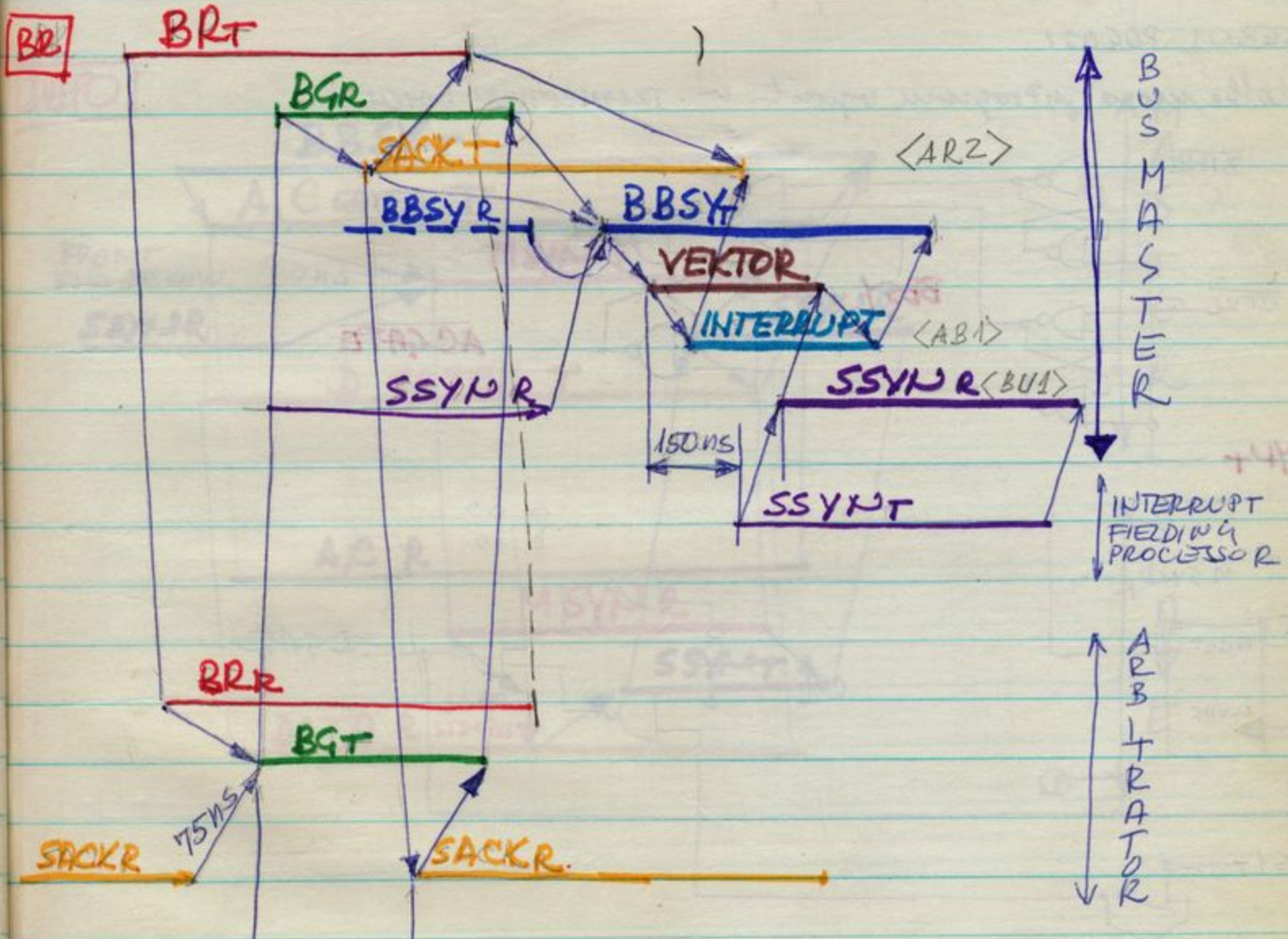
HERE IS | X

NPR



za single word transfer MASTER mgiva SACK takoj ko post tami BBSY

BR



Če je to čas 5-10 sek potem je BG mgiva avtomatsko

① DATI

a) CONTROLER JE MASTER

ACGATE se odpre ob BBSY↑, zapre ob $\overline{MSYN↑}$

DGATE ZAPRT

b) CONTROLER JE SLAVE

ACGATE JE ZAPRT

DGATE se odpre ob $MSYN_R$↓, zapre ob $\overline{MSYN_R}$ [C1=0]

② DATO

a) CONTROLER JE MASTER

ACGATE se odpre ob BBSY↑, zapre ob $\overline{MSYN↑}$

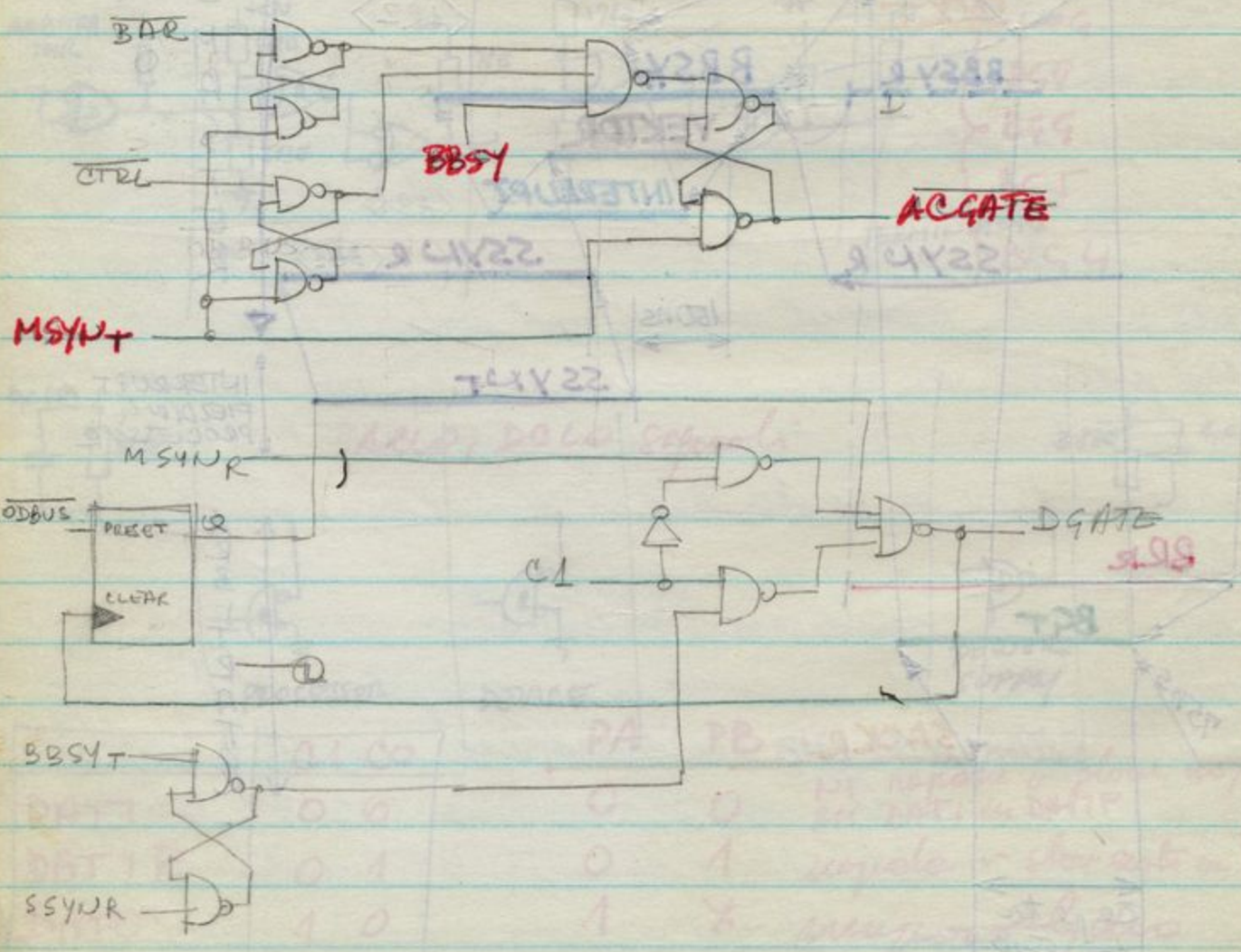
DGATE se odpre ob BBSY↑, zapre ob SSYNR [C1=1]

b) CONTROLER JE SLAVE

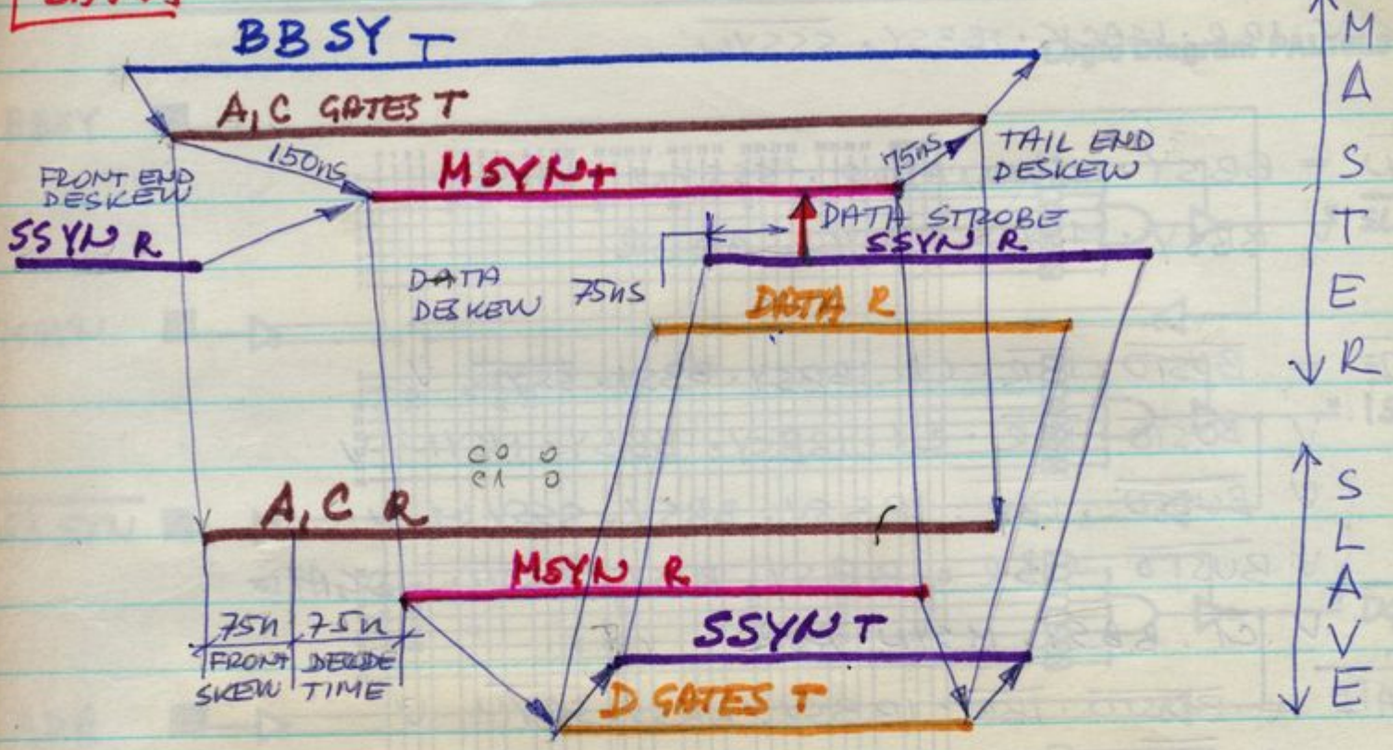
nimamo signalov za pošiljanje na vmesnik

POSEBNI POGOJI

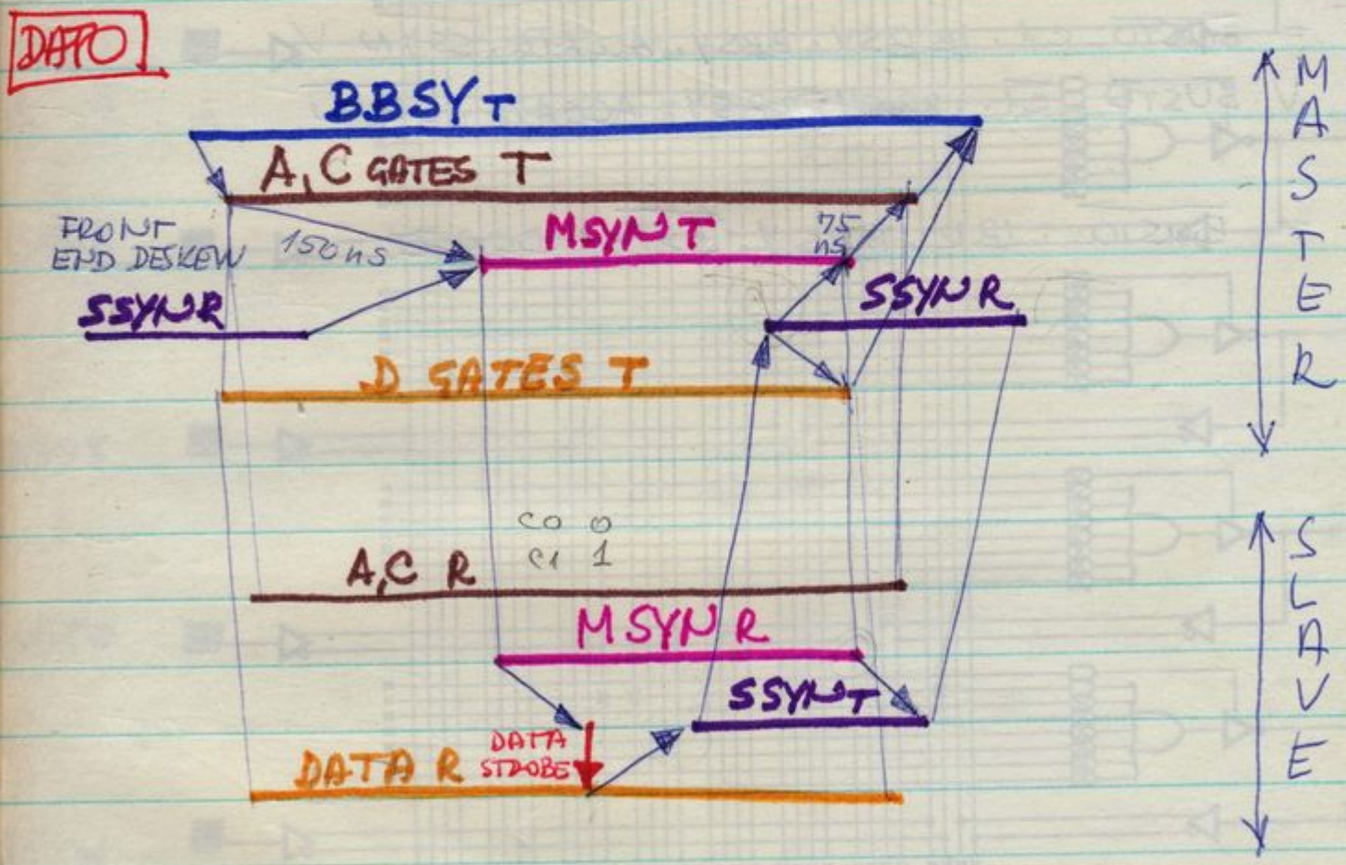
Podatke mora predpomniti v TRANSCEIVER LATCH



DATI


BBSY T
A,C GATES T
150ns
MSYN T
75ns
TAIL END DESKEW
FRONT END DESKEW
SSYN R
DATA STROBE
SSYN R
DATA DESKEW 75ns
DATA R
C0 0
C1 0
A,C R
MSYN R
75n
FRONT SKEW
75n
DECODE TIME
SSYN T
D GATES T
MASTER
SLAVE

DATO


BBSY T
A,C GATES T
FRONT END DESKEW
150ns
MSYN T
75 ns
SSYN R
SSYN R
D GATES T
C0 0
C1 1
A,C R
MSYN R
SSYN T
DATA R
DATA STROBE
MASTER
SLAVE

$\overline{SLAVE} = BBSY \cdot MSYN \cdot \overline{ISSYN} \cdot ADDR \cdot \overline{SSSYN} \vee$
$\vee IBR \cdot ISACK \cdot IBBSY \cdot \overline{SSSYN}$

$\overline{ISSYN} = BBSY \cdot MSYN \cdot ADDR \cdot SSSYN \vee$
$\vee BBSY \cdot MSYN \cdot ISSYN \cdot ADDR$

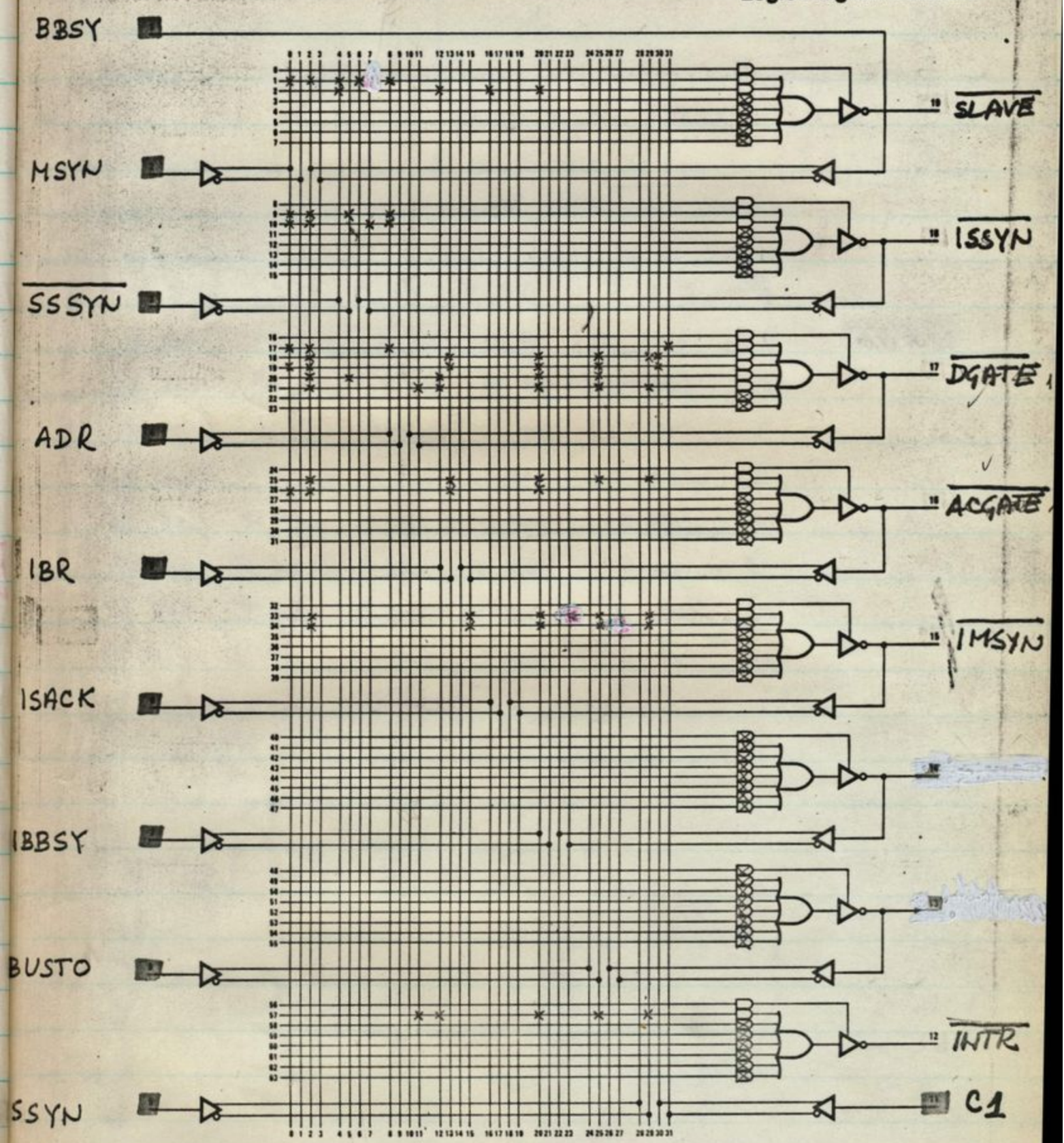
$\overline{DGATE} = \overline{BUSTO} \cdot \overline{IBR} \cdot C1 \cdot IBBSY \cdot BBSY \cdot \overline{SSYN} \vee$
$\vee \overline{BUSTO} \cdot \overline{IBR} \cdot C1 \cdot IBBSY \cdot BBSY \cdot MSYN \vee$
$\vee \overline{BUSTO} \cdot IBR \cdot IBBSY \cdot BBSY \cdot SSSYN \vee$
$\vee \overline{BUSTO} \cdot IBR \cdot IBBSY \cdot BBSY \cdot \overline{SSYN} \cdot DGATE$
$\vee \overline{C1} \cdot BBSY \cdot MSYN \cdot ADR$

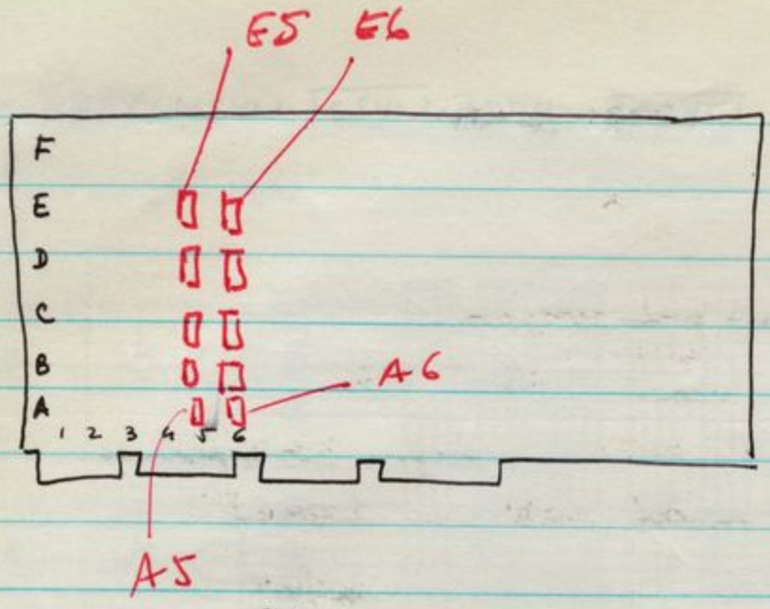
$\overline{ACGATE} = \overline{BUSTO} \cdot \overline{IBR} \cdot IBBSY \cdot BBSY \cdot \overline{SSYN} \vee$
$\vee \overline{IBR} \cdot IBBSY \cdot BBSY \cdot MSYN$

$\overline{IMSYN} = \overline{BUSTO} \cdot C1 \cdot IBBSY \cdot BBSY \cdot ACGATE \cdot \overline{SSYN} \vee$
$\vee \overline{BUSTO} \cdot \overline{C1} \cdot IBBSY \cdot BBSY \cdot ACGATE \cdot \overline{SSYN}$

$\overline{INTR} = \overline{BUSTO} \cdot IBR \cdot IBBSY \cdot \overline{SSYN} \cdot DGATE$

Logic Diagram PAL16L8





RAZMOSTITEV ROM-OV

PIN	E 6,5	D 6,5	C 6,5	B 6,5	A 6,5
~~E 6,6~~ 7	CWR 00	CWR 08	CWR 16	CWR 24	CWR 32
8	CWR 01	CWR 09	CWR 17	CWR 25	CWR 33
9	CWR 02	CWR 10	CWR 18	CWR 26	CWR 34
10	CWR 03	CWR 11	CWR 19	CWR 27	CWR 35
12	CWR 04	CWR 12	CWR 20	CWR 28	CWR 36
13	CWR 05	CWR 13	CWR 21	CWR 29	CWR 37
14	CWR 06	CWR 14	CWR 22	CWR 30	CWR 38
15	CWR 07	CWR 15	CWR 23	CWR 31	CWR 39

PODNOŽJE ZA 2716

27 S27 (22 pin)

1 + A3
2 + A4
3 + A5
4 + A6
5 + A7
6 + A8
7 + Q0
8 + Q1
9 + Q2
10 + Q3
11 + GND

Vcc − 22
A2 + 21
A1 + 20
A0 + 19
$\overline{E1}$ + 18
$\overline{E2}$ + 17
Cp + 16
Q7 − 15
Q6 − 14
Q5 − 13
Q4 − 12

PIN

9 + D0
10 + D1
11 + D2
13 + D3
14 — D4
15 — D5
16 — D6
17 — D7

PIN

8 + A00
7 + A01
6 + A02
5 + A03
4 + A04
3 + A05
2 + A06
1 + A07
23 + A08
22 — A09
18 — A10

24 + +5V
12 + GND
18 + CP

CIPHER streamers

GCR

emergency supply from London

900 X Vacuum tape drive

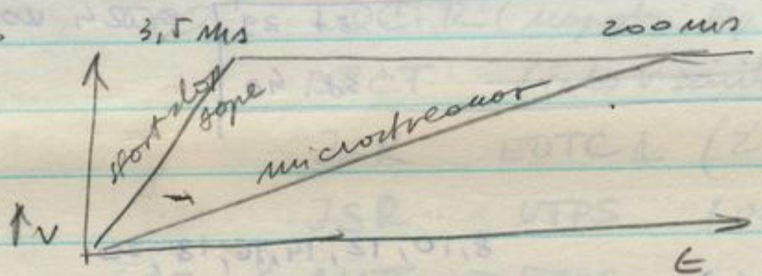
910 75 IPS 920 DIMASTRON 125 IPS

FORMATER je v DRIVE enoti toje 20% cene

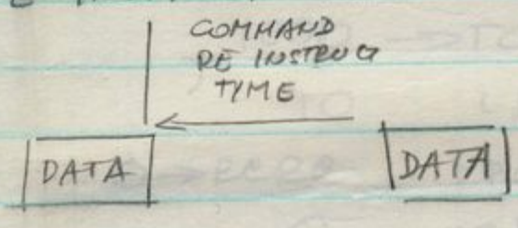
Potrebuje le interface med CPU in DRIVE

Microstreamer izgoje formater

ANSI IBM compatible



RE INSTRUCT TYPE



1,2 inch interrecord gap

Interface za računalnike:

PDP 11 / LSI 11

RS 232 C

VAX 750

IBM SERIES 1

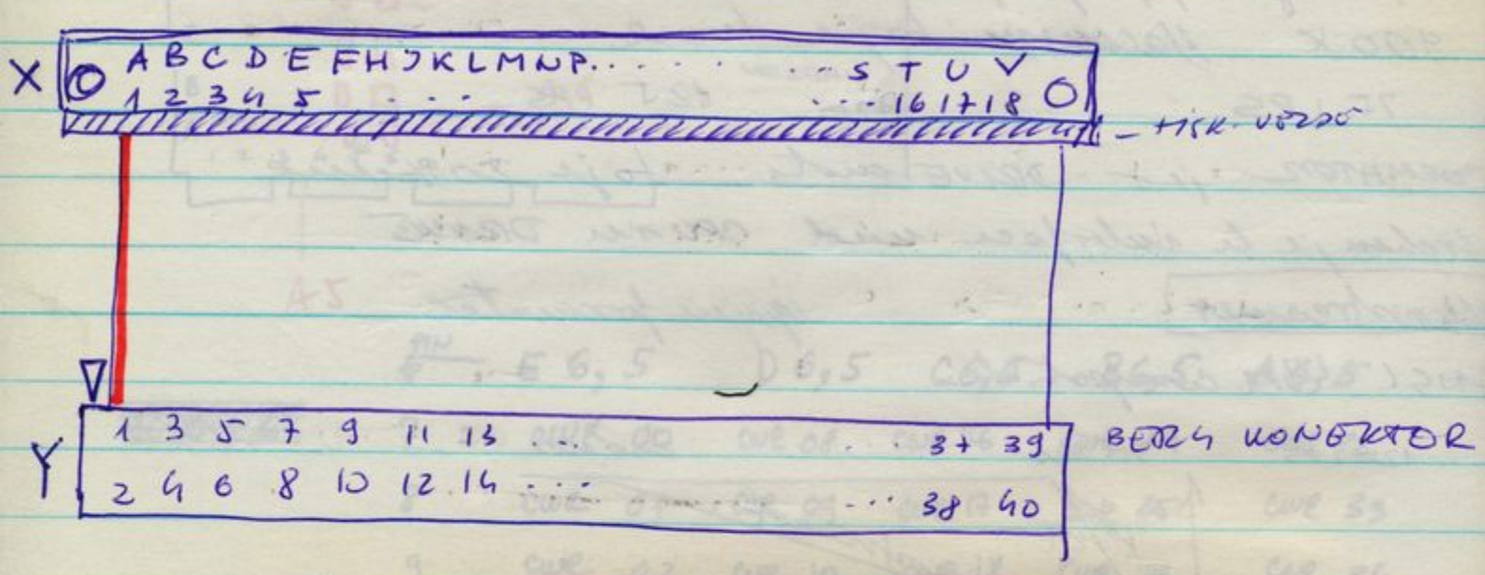
DIALOG coupler

DV 132 dela tudi na VAX ih

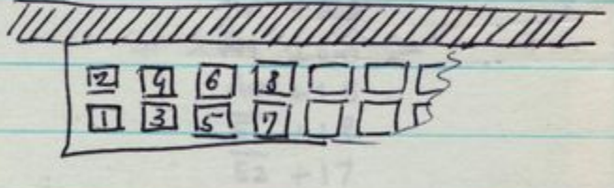
in za druge.

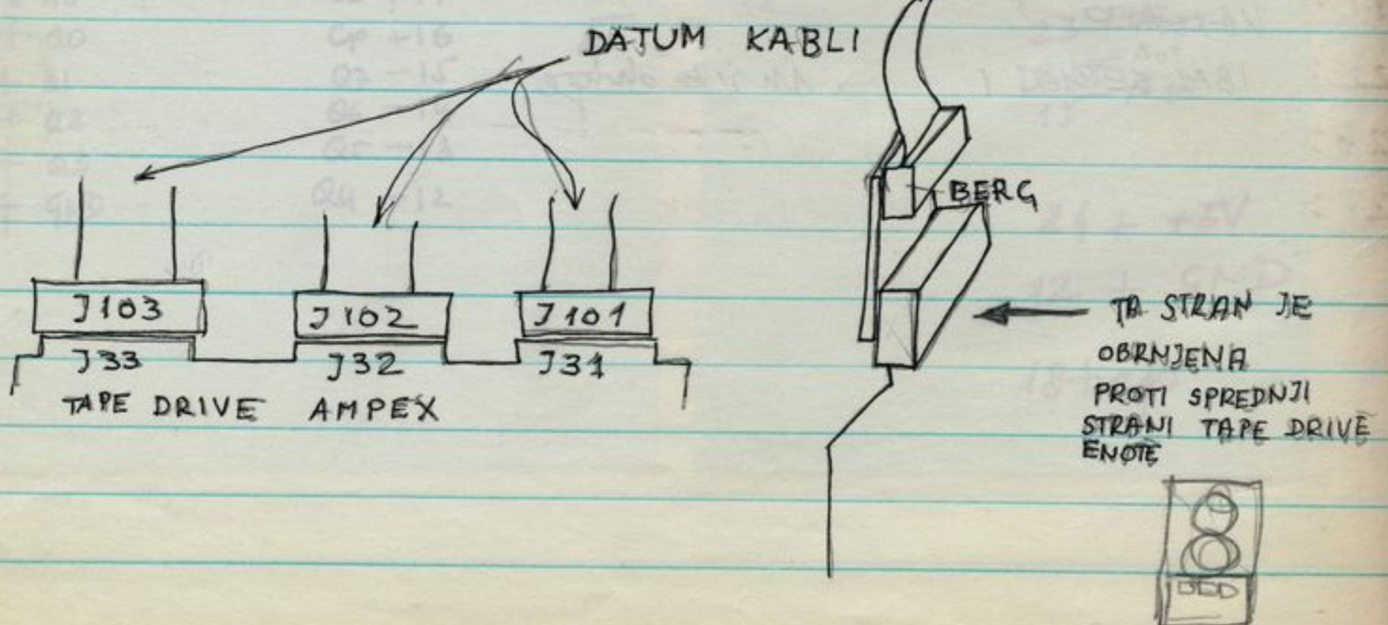
KONEKTOR NA TAPE UNIT

S STRANI KONTAKTOV



BERG Y	X	
1,	GND	8, 10, 12, 14, 16, 18, 20
2	OVERWRITE (–)	
3	FWD/STOP (–/+)	
5	DENSITY SELECT (HIGH)	
7	REV/STOP (–/+)	
9	REWIND (–)	
11	WRITE PERMIT (–)	
13	HIGH DENSITY STATUS (–)	
15	ON LINE & SELECTED (–)	
17	REWINDING STATUS (–)	
19	FILE PROTECT STATUS (–)	
21	BOT STATUS (–)	





REWIND & REWIND OFFLINE COMMAND EXECUTION ADR: 05E

```
                CUR
05E     10000000 V  AMTC  → AMTC                          D1
05F           0  +  TSDTA → Breg, FBUS, TEMP 2             D1
                INTERRUPT ENABLE
060       10 0000  &  ATEMP2 → YOUT                        D1
061   IF FBUS = Ø    JMP   REWOF  (67)                     D1
062                  JSR   EOTM   (2F)                     D1
063 1000 0000 0000 0000 V  AMR1   RCRCC                    D1
064          100 + 0   → FBUS, TCREG
065           10       LDCTR (nopolni Rep ctr z 10)
066                    RPCT  (čaka v zanki)
067                    JSR   EOTC1 (2B2)
068                    JSR   UTPS  (022)
                FBUS
069           10  &  AMTS → FBUS
06A            0  &  AMTS → FBUS  JMP          (2CA) END2
06B         1000  → TCREG
06C           10     LDCTR
06D    0 → RCRC, RPCT (čaka v zanki)
06E            0 → TCREG      JMP          (2CA) END2
```

BAD TAPE ERROR se postavlja v naslednjih primerih

ADR		ROUTINE	KONTROLIRA SE	BIT
178	MTS = AMTS OR 100	WRITE PROTECT CHECK	AMR2 & #2000	ID BURST
22E		CHARACTER TIMER SUB	—"— #8000	1 CHAR PRE
280		END OF FILE ROUTINE 2	—"— #4000	SPC REV (space cmd)
2FD		REW - READS PE ID BURST	—"— #2000	ID BURST

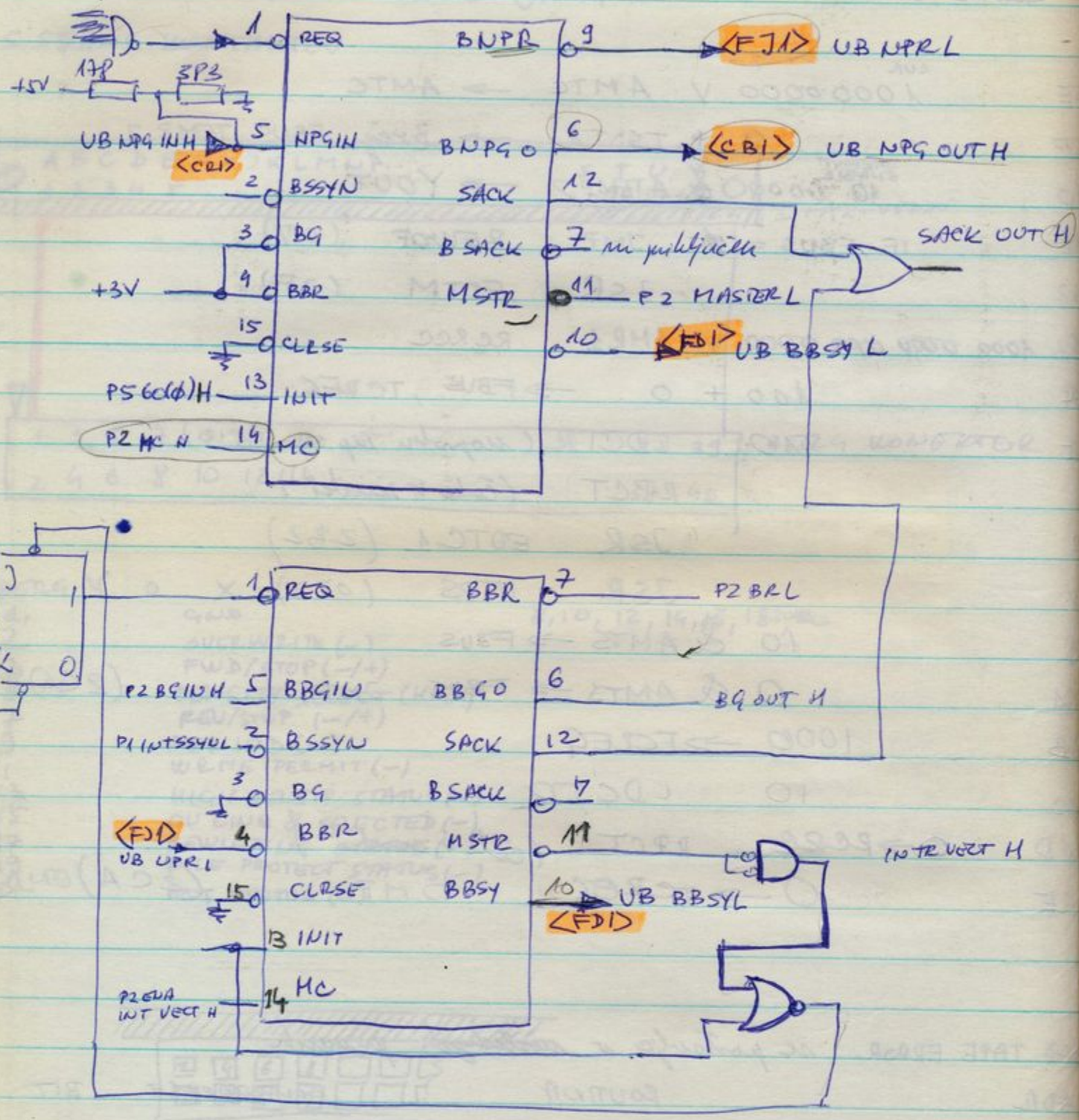
BOOT FUJITSU DISKA Z DIAGNOSTIKO:

NA TERMINALU SE POJAVI 177777707 165714

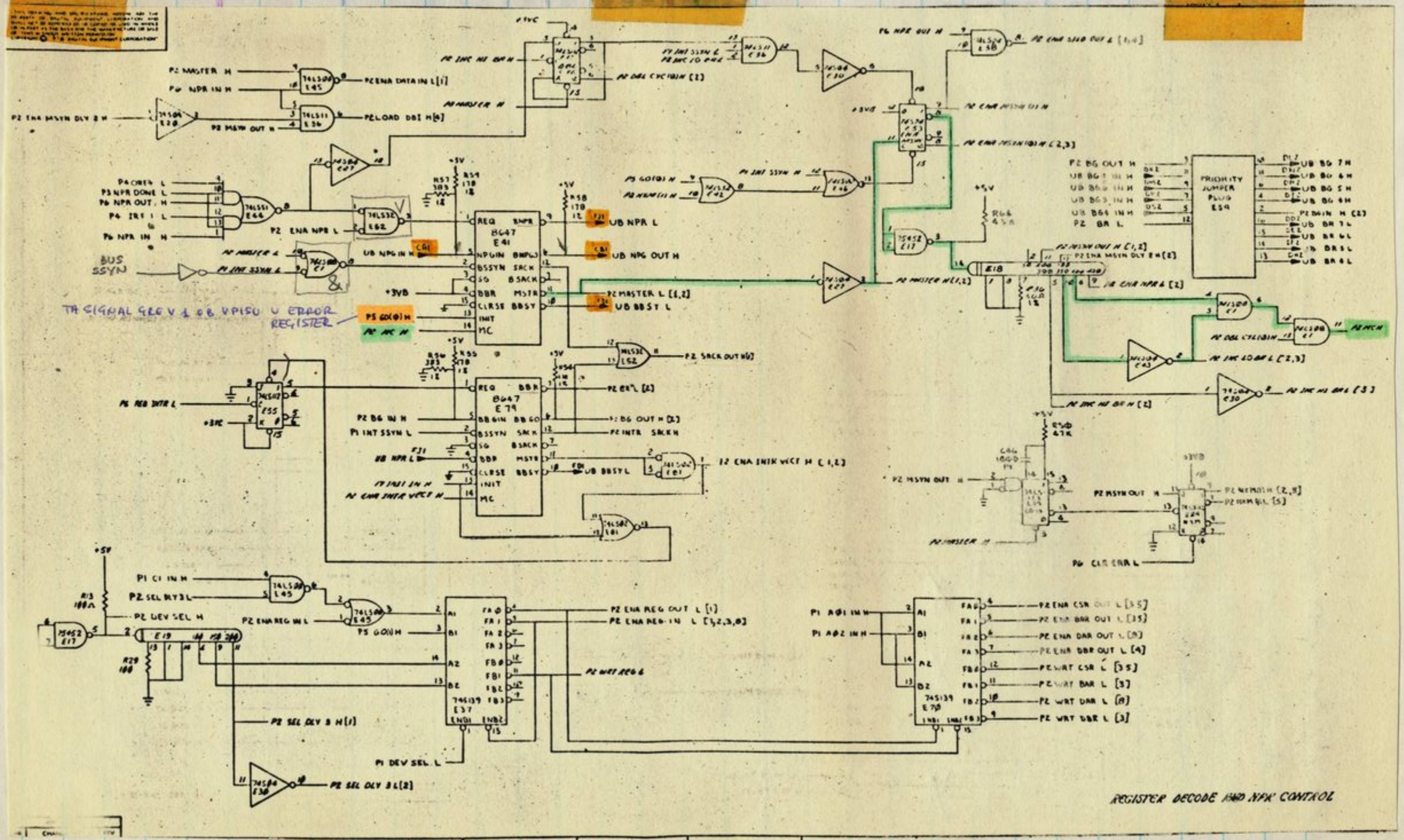
>>> B␣DB

odtipkamo ↑ in diagnostika teče

Verje na DL01



RL 11 KONTROLÉR



REGISTER DECODE AND NPR CONTROL

PDP 11/24 SPONKE NA MESTU PROCESORJA

FT2 BUS SACK L

CB1 NPG H

FJ1 BUS NPRL

DL2 BUS BG 7 H

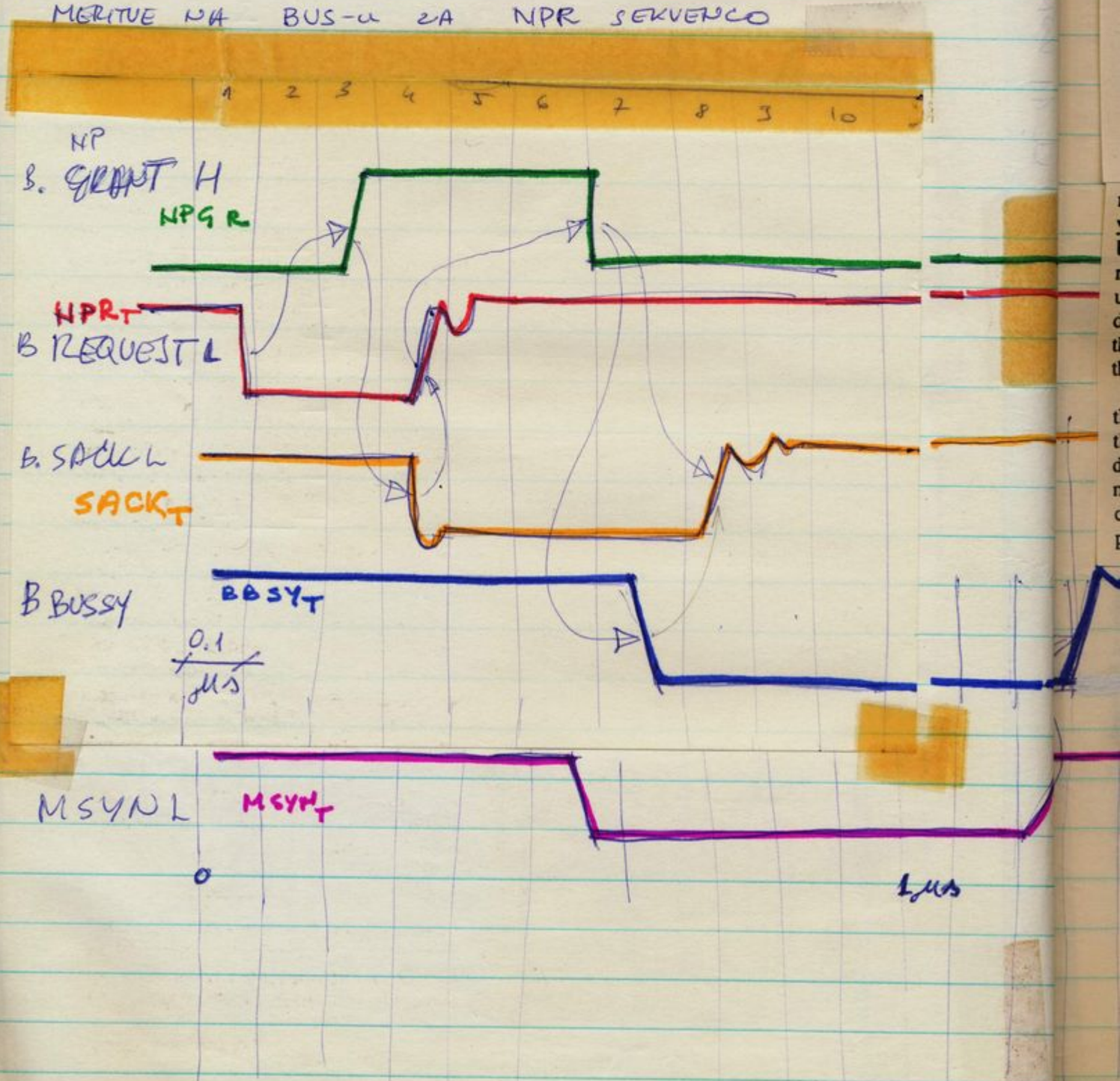
? DH2 BUS BG 6 H

? DR2 BUS BG 5H

DT2 BUS BG 4 H

~~FJ1~~ ~~BUS~~ ~~BG~~

MERITVE NA BUS-u ZA NPR SEKVENCO



V hardware-u se zove NPR sekvenca a SNPR signalom
ili je v µProg. ne lokacijah ODC in ODD to je
v subroutini NPR1

Q: I read your article about Unibus timing and found it very informative. I will be following your advice about using multiple cycles rather than trying to be bus master. There is one question, however: What signals do I hold on during these multiple cycles to hold the bus, and why?

A: Good question. I'm from the old school and would probably be more prone to follow the original methods, however, I've seen other methods used that don't appear to cause problems.

Sack Hold — I'll refer to this primary method as "Sack Hold," because it is the main signal that you are holding through your cycle. The Unibus, as I've explained before, has evolved over the years. I can't recall dates, but the early Peripherals Manual (a red book) covered the sack hold in diagrams. As the evolution progressed through purple and blue to the present day, this part of the diagram was left off.

The basis of "Sack Hold" was to hold the Sack Line down until the beginning of the last cycle. For example, if you were doing four consecutive cycles, you would maintain Sack until the end of the third cycle. The idea was to wait until the last possible minute before removing it so that arbitration would not begin until just before you ended a cycle. Probably this concept was attempting to give the highest priority device the best chance of getting the bus.

Some people, while holding Sack, would toggle BBSY (Bus Busy). What they would be doing was to attempt to reduce logic, by using the Bus Busy line to gate out Address, Control and Data. Whether or not this is advisable, will give you something to think about.

No Sack — We'll call this "No Sack" in order to distinguish it from releasing Sack at the beginning of the device's mastership of the bus. This typically is what you see in timing diagrams, since most timing diagrams only show a single cycle operation. Since more people have only seen the more recent peripheral manuals from DEC, this method is more prevalent in the industry. BBSY (Bus Busy) is the holding device (signal) in this method. Obviously, no other master can assume bus mastership until this signal is released. With more microprogrammed controllers coming to the bus, taking more time to go through the arbitration handshake process, this method would allow the controllers more time to get this accomplished.

In summary, both approaches have their advantages and should be considered on their own merits. Sack Hold allows arbitration to be withheld to a time just before the bus will be released (one memory cycle). For configurations that have many different DMA devices on it, this may be of importance — i.e., more possibility of arbitration conflicts. *No Sack* allows the entire time of the multiple cycle transaction to be used for arbitration handshake timing. This could give a better overlap of cycle readiness to slower microprogrammed devices, possibly giving a slightly better bus utilization in these configurations.

Which should you use? Well, I'll leave that to you. You'll probably find both approaches used.

(Questions or comments for the DEC Troubleshooter should be addressed to Ken O'Mohundro, Able Computer, 1732 Reynolds Ave., Irvine, Calif. 92714. Ed.)

XXDP

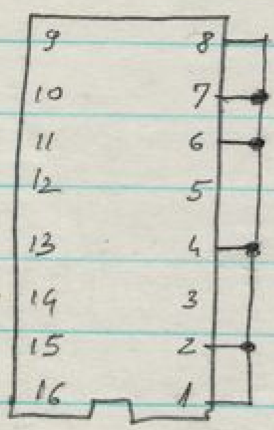
1) Adresa BOOT naprave je
173000 <CR>
{ XXDP
.L ZTMB ?? <CR>

.S 210 ... sk odpre storbono zato do testiranj samo eno napravo

SWR = 00 - - 0 NEW 401 ali 201 <CR>
SELECT UNITS 0, <CR>

TST	PAT	RLS	WMO	RMO
3	0	2	1	1

NASTAVITEV JUMPER-jev ZA HITROST NA DATUM TRAČNEM KONTROLERJU pri <u>45 Bpi</u>



Terminals are also generally designed to plug into modems and also use a connector as shown in Figure 26-1. In most computer installations, however, there are some terminals which are located close to the computer and need to be connected to the same multiplexer that has the modem equipped lines connected to it. The problem is shown schematically in Figure 26-2. Not only is there a problem with the sex of the connectors (both male), but there is also a problem with the pinning, as both connectors deliver data to be transmitted on pin 2.

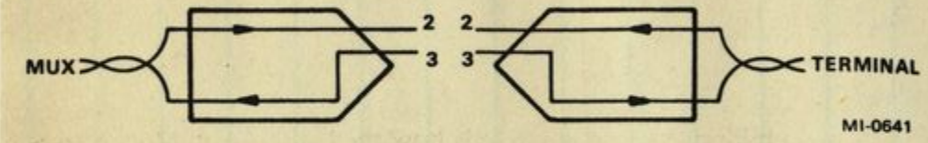


Figure 26-2. Attempted Connection of Multiplexer and Terminal Using DB25-P Connectors

Figure 26-3 shows one solution to this problem, the "null modem," and indicates how its use is similar to that of actual modems. In both Figure 26-3a and Figure 26-3b the "modem facility" provides a female connector for the mux to plug into and a female connector for the terminal to plug into. Furthermore, data applied to pin 2 of one of those connectors comes out pin 3 of the other.

The "null modem" may be implemented either as a length of cable with leads 2 and 3 (and some others) transposed, or as a cigarette box size container with a terminal board inside and female connectors at the ends. The cable implementation is much neater and provides some additional cable length. Many computer companies sell cables which accomplish this function without referring to them as "null modems" but rather as "cable for connection terminal to computer interface." As can be deduced from Figure 26-2, either the cable from the mux or the cable from the terminal can be altered to solve the connector problem.

The cigarette box type of null modem offers some advantages over the cable type, however. With a terminal strip inside the box a number of different wiring arrangements are possible. Figure 26-4 shows a typical terminal block.

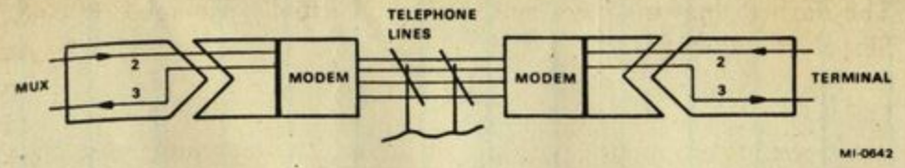


Figure 26-3a. Connection with Actual Modems

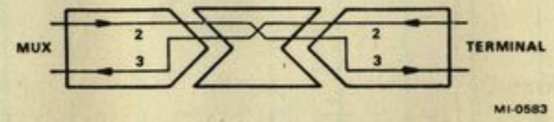


Figure 26-3b. Connection with Null Modem

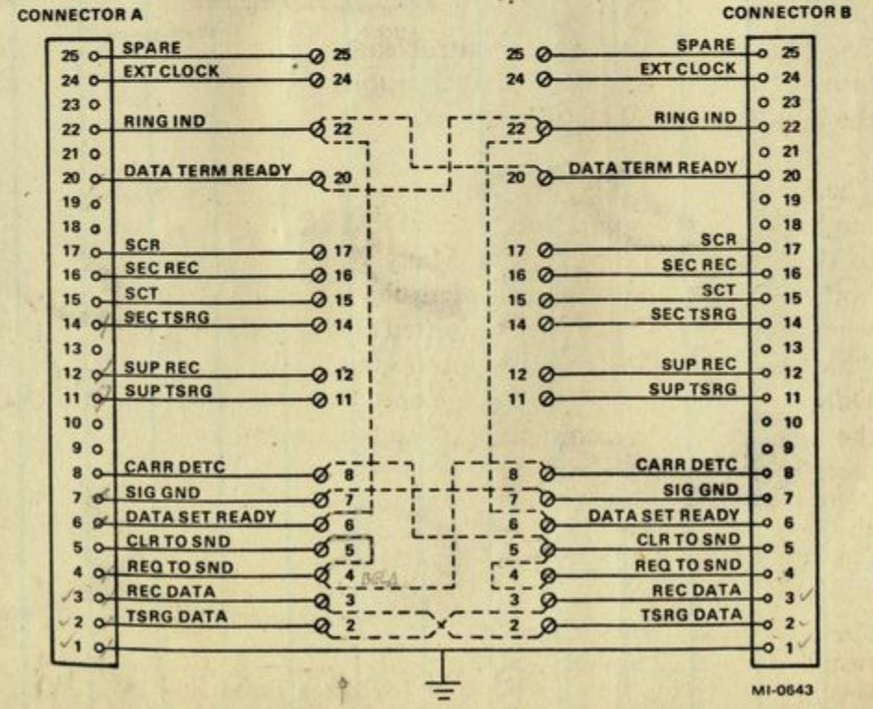


Figure 26-4. Null Modem Wiring Diagram

PROGRAM ZA SSYN signale

```
1000    12737
           1
        772524
          137
         1000
```

v reg vpisuje konstanto

PROGRAM ZA INTERRUPT SIGNALE

```
1000    12737
   2    100300
   4    772522
  16       1
  10     137
  12    1000
```

Popravi in prižge interrupte ERR=1 CUR=1 INT=1

Konstanta registra

WAIT

Skoči na 1000

```
2000    12737
   2        1
   4    772524
   6       2
```

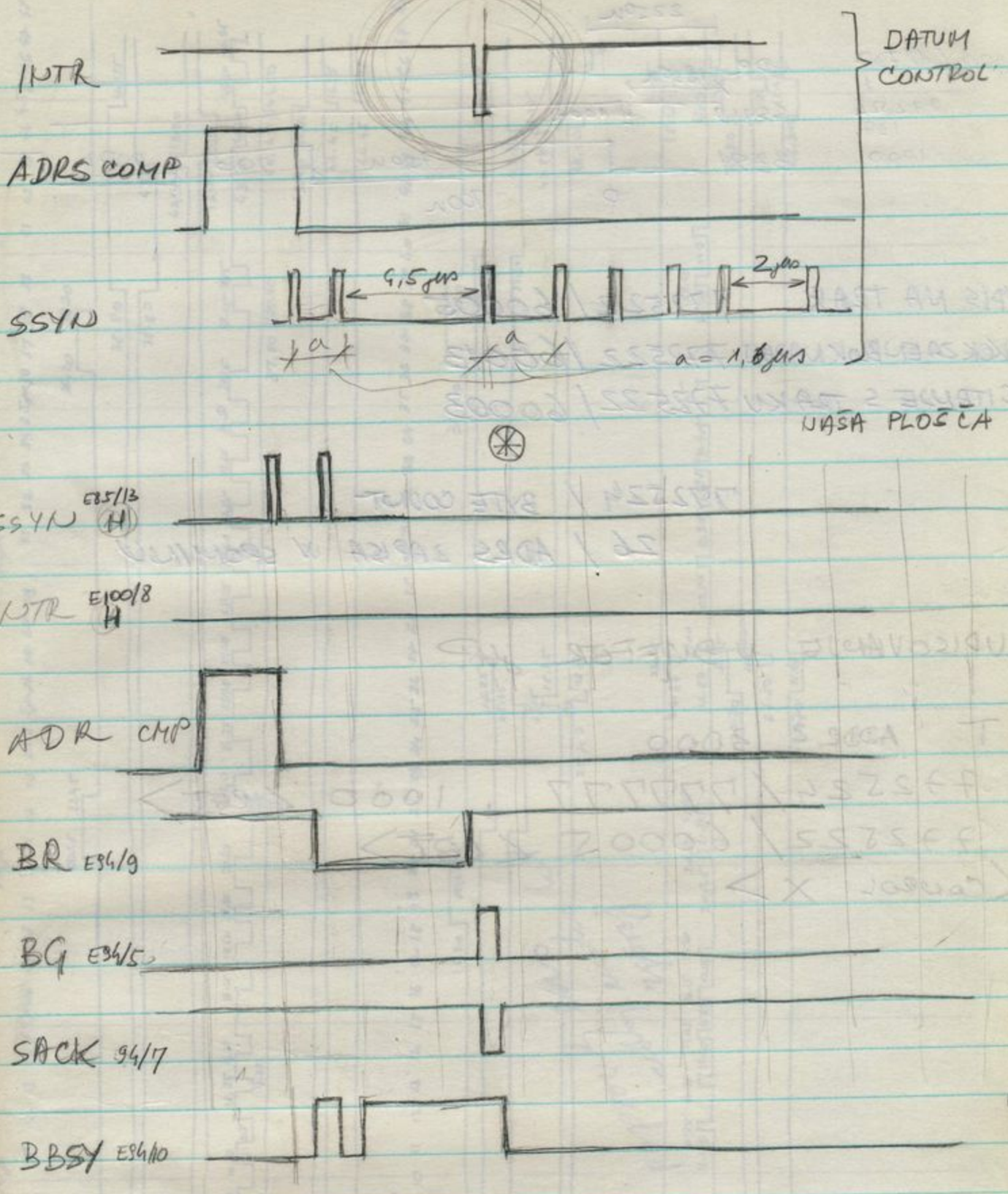
v byte count reg vpišemo 1

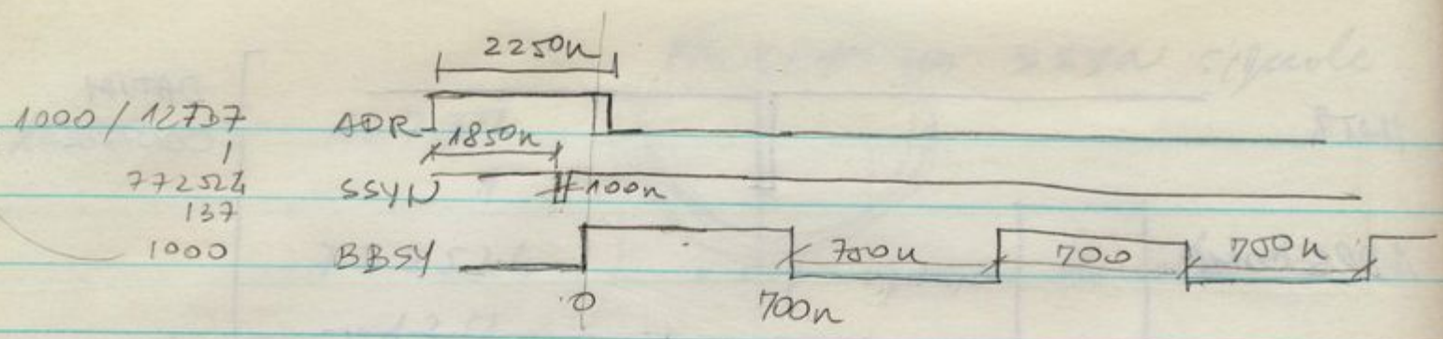
RTI

interrupt vektor

```
224     2000
226     340
```

PSW ima na prioriteti zapis 7 in 11


INTR
ADRS COMP
SSYN
6,5 μs
2 μs
a
a
a = 1,6 μs
DATUM CONTROL
NASA PLOČA
SSYN E85/13 H
INTR E100/8 H
ADR CMP
BR E84/9
BG E84/5
SACK 94/7
BBSY E84/10



VPIS NA TRAK 772522/60005

SKOK ZA EN BLOK NAZAJ 772522/60013

ČITANJE S TRAKU 772522/60003

)

772524 / BYTE COUNT

26 / ADRS ZAPISA V SPOMINU

UPISOVANJE V BUFFER MTP

T ADDR? 3000

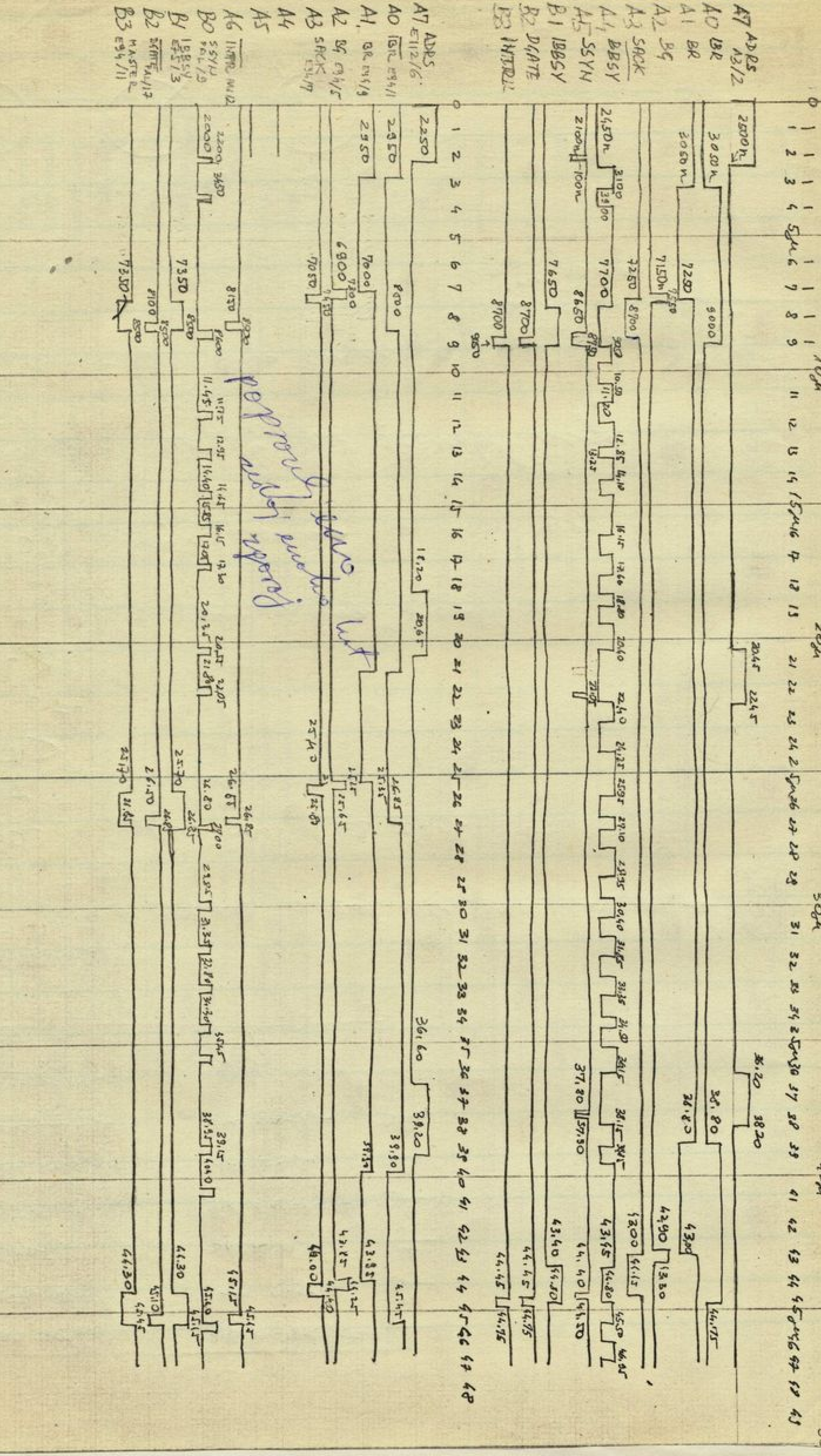
772524/777777 ↵ 1000 <RET>

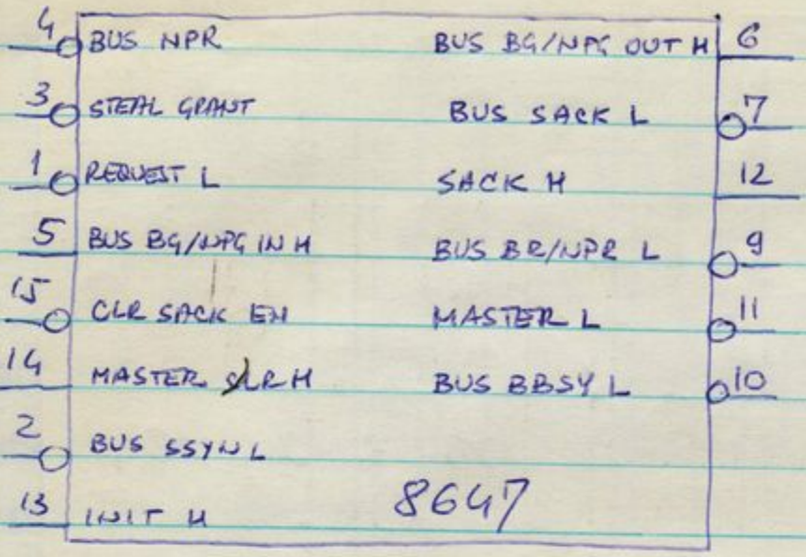
772522/60005 <RET>

<CONTROL X>

NAŠ KONTROLER

DATUM KONTROLER





ACOTON TERMINAL

2 R čitanje bralca, s tem locirano program na bralcu
(PAUZA)

B C 300 startno odreja EDITOR-ja
INIT } ustiplamo
READ }
(PAUZA OFF)
<CR>

ASM prevajan program asemblera

B 3800 zagon programa za analizer

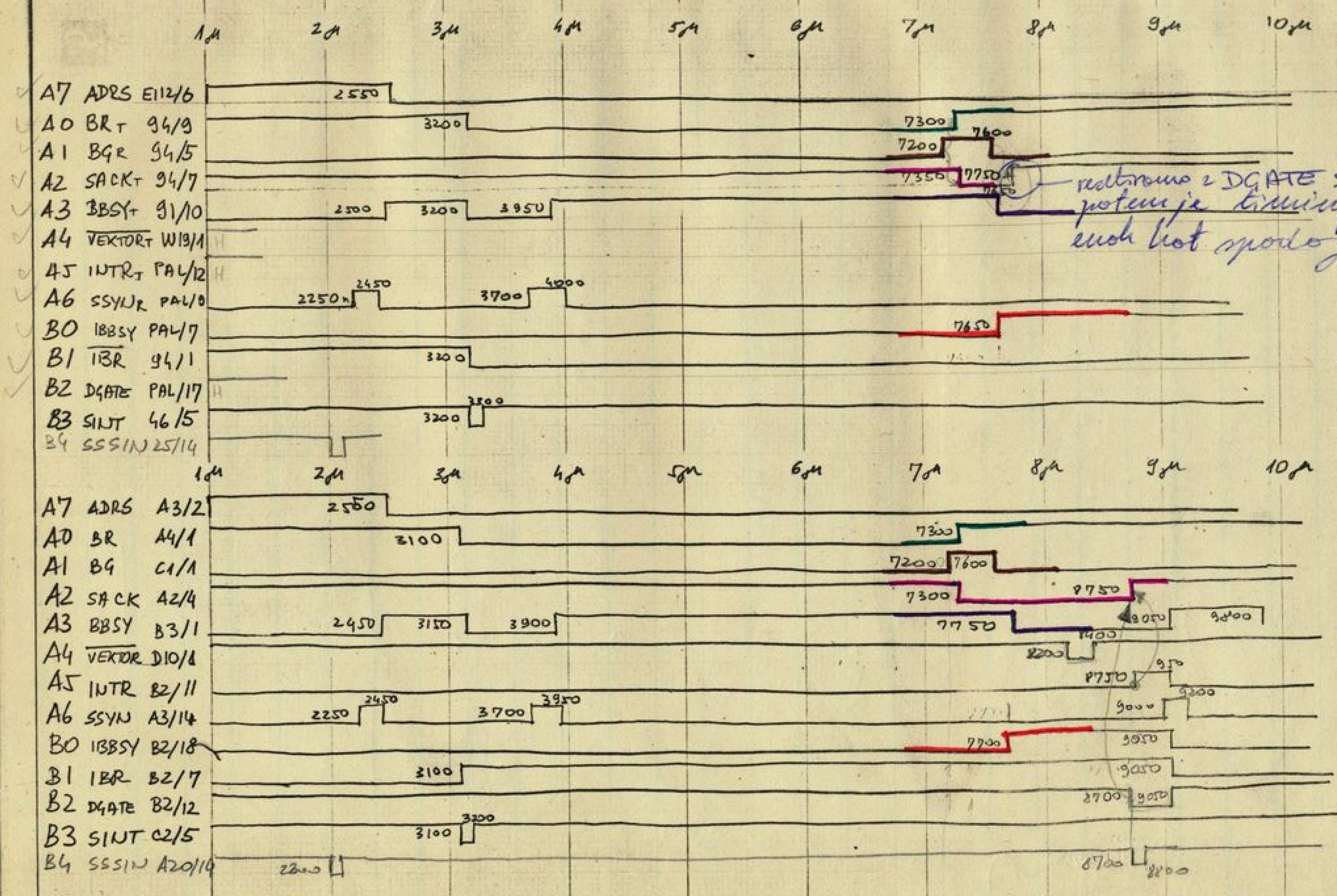
popravljanje napak pri asemblaciji

B (odreja hyperterm. naprave) ⊔ XX ⊔ YY
XX ... vrstica na tej odreji
YY ... nova vrstica

VOLUME BALANCE TONE

RIGHT LOW

22.06.82



razlikovno z DGATE SIGNALOM potem je timing enak kot spodaj.

DOBRO!
enak timing kot dotičnov kontroler

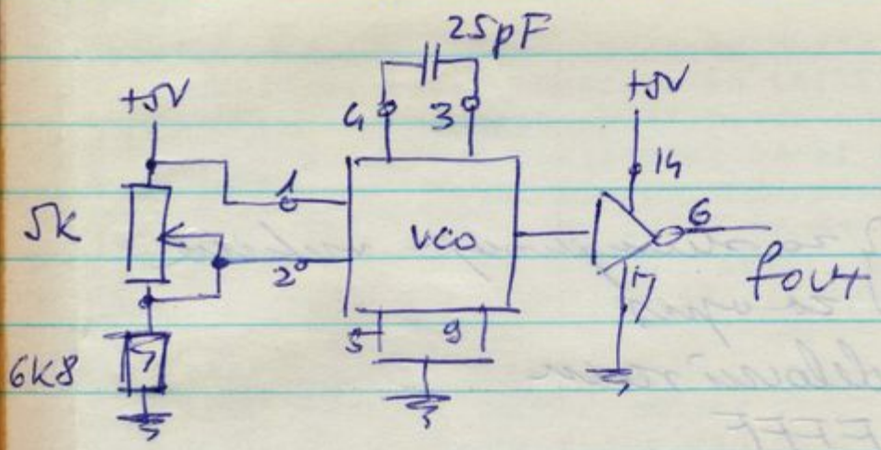
ZAPISIVANJE NA TRAK ILI OPAZOVANJE NA OSCILOSKOPU
(opcionalno lokalno signale RIR read input ready...)

1000	12737	
1002	777772	byte count number
1004	772524	BYTE COUNT REG = byte count number
1006	12737	
1010	4000	start odreza
1012	772526	ADRS REG = start odreza
1014	12737	
1016	60005	komanda za zapisivanje
1020	772522	komandni register
1022	13700	move status → R0
1024	772520	STATUS REG
1026	6000	ROR premaknemo ready bit v CARRY
1030	103 374	BRANCH IF CARRY CLEAR
1032	137	JMP
1034	1000	

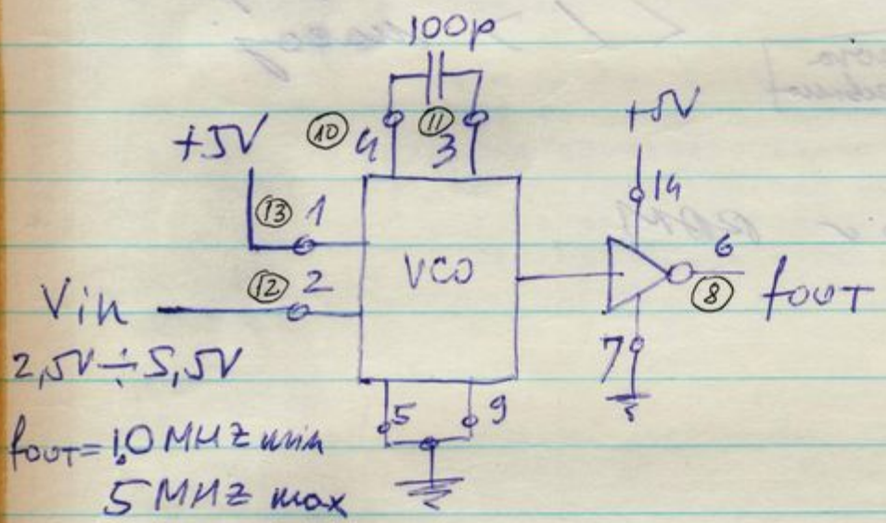
PAZI DA SE TRAK NE SNAME PREDEN PROGRAM NE TESTIRA EOT

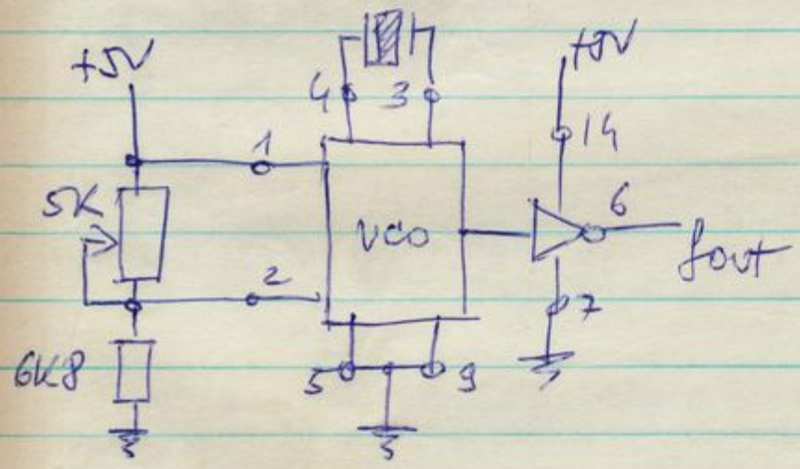
1032 / 12737	
1034 / 6017	skoknemo, pisanje na BOT
1036 / 772522	
1040 / 137	
1042 / 1000	

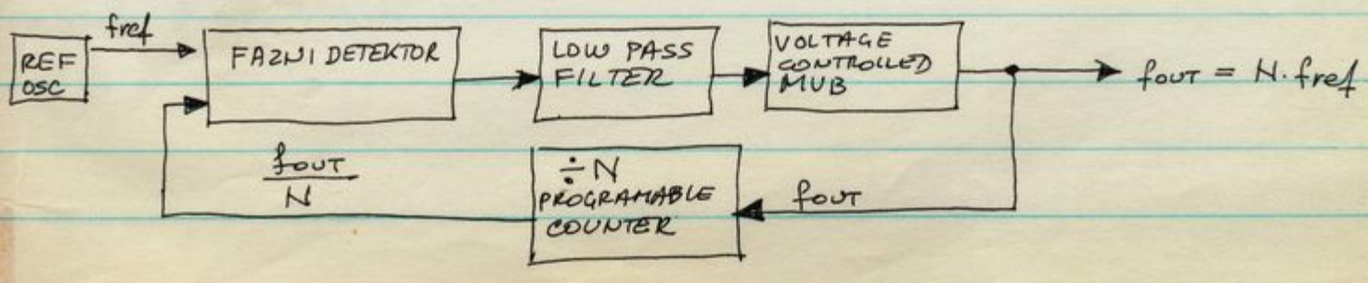
4024 DUAL VOLTAGE CONTROLLED OSC



Fout = 10 MHz







NAVODILO IN OPIS DELOVANJA EPROM EMULATORJA, VLAGALNIKA IN ZGALNIKA

Omogoca, da z RAM-om ali EPROM-om, ki sa upravljamo iz obstojecega sistema, emuliramo EPROM(2716) na sistemu, ki sa razvijamo. Vsebino pomnilnika je mozno shraniti na datoteki na racunalniku DELTA, kjer se lahko generirajo tudi nove datoteke, in jo zopet prepisati v pomnilnik. Dokoncna verzija programa pa se lahko vpise v stalni pomnilnik (EPROM 2716).

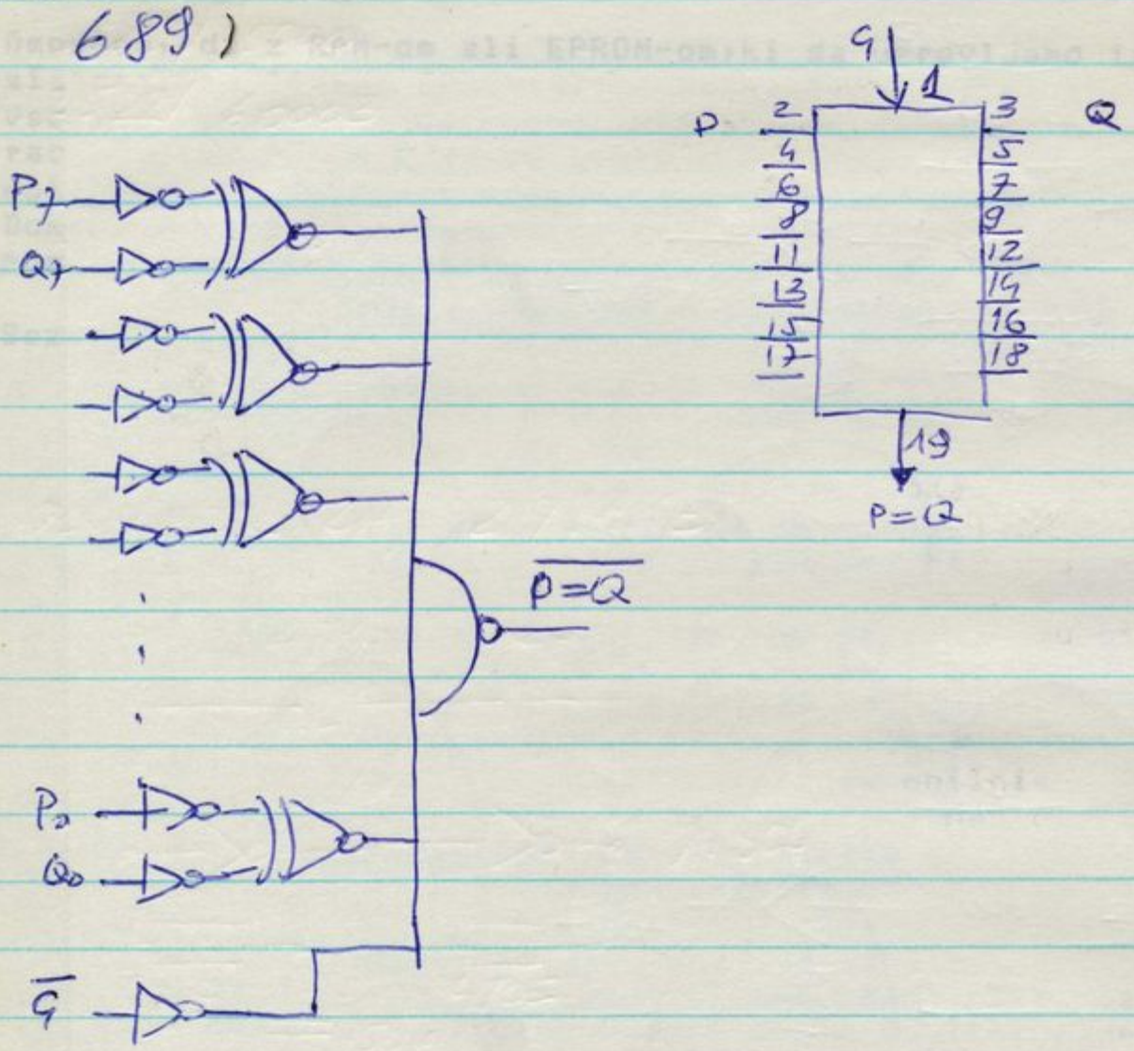
Seznam ukazov:

D aaaa dd		prikaz vsebine dd na lokaciji aaaa
	RET	prikaz naslednje lokacije
	/	prikaz prejsnje lokacije
	SP	vpis podatkov
	TAB	binaren prikaz vsabine dd
R ssss		start programa na naslovu ssss
^R		start programa na naslovu F000
M zzzz kkkk		prikaz vsebine pomnilniskih lokacij od naslova zzzz do naslova kkkk
	^S	zaustavitev izpisa
	^Q	nadaljevanje izpisa
	^C	prekinitev izpisa
U zzzz kkkk		prenos vsebine pomnilnika na racunalnik DELTA od lokacije zzzz do lokacije kkkk (oblika datoteke I)
P zzzz kkkk		prenos vsebine pomnilnika na racunalnik DELTA od lokacije zzzz do lokacije kkkk (oblika datoteke II)
V oooo aaaa		preverjanje enakosti bloka podatkov dolzine nnnn, ki se nahajata v pomnilniku ROM na naslovu oooo in v pomnilniku RAM na naslovu aaaa
	^C	prekinitev izpisa, ce so napake
^V		preverjanje enakosti bloka podatkov dolzine 07FF, ki se nahajata v pomnilniku ROM na naslovu 0000 in v pomnilniku RAM na naslovu A000
	^C	prekinitev izpisa, ce so napake
B oooo aaaa nnnn		vpis nnnn zlogov v pomnilnik EPROM, ki je naslovljen z oooo, iz pomnilnika RAM, ki je naslovljen z aaaa
^B		vpis vsebine celotnega RAM v EPROM
C oooo aaaa nnnn		vpis nnnn zlogov v pomnilnik RAM, ki je naslovljen z aaaa, iz pomnilnika EPROM, ki je naslovljen z oooo

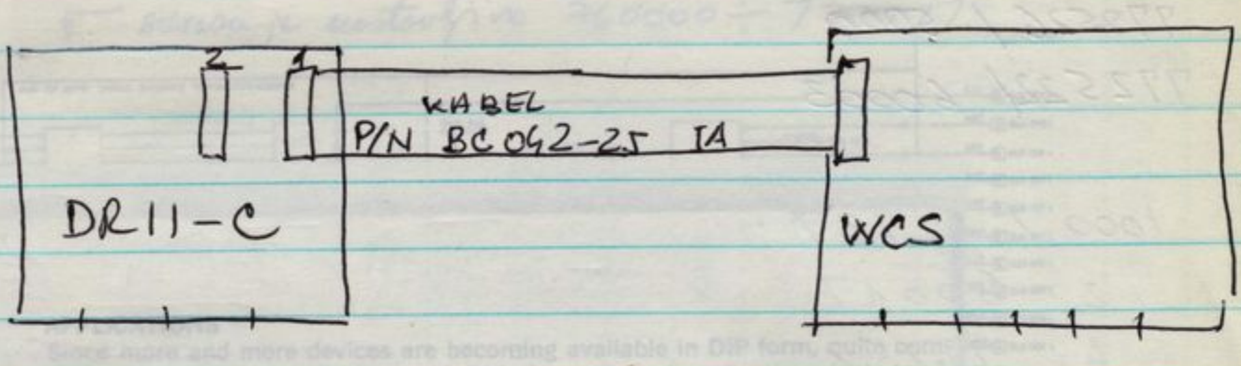
T	deluje kot terminal
E	deluje kot terminal in omogoca prenos podatkov iz datoteke na DELTI
L aaaa bbbb nnnn	prepis nnnn zlogov iz naslova bbbb na naslov aaaa
F	dostop do pomnilnika RAM/EPROM mozen iz emulatorja
N	dostop do pomnilnika RAM/EPROM mozen iz sistema,ki ga razvijamo
O	iz sistema, ki ga razvijamo mozen dostop do pomnilnika EPROM
A	iz sistema, ki ga razvijamo mozen dostop do pomnilnika RAM
^W	prepis celotne vsebine EPROM-a (podn.2) v RAM
^A	prepis celotne vsebine EPROM-a (podn.1) v RAM
Z	preverjanje vsebine EPROM PRAZEN je ce je vsebina vseh lokacij FF
J	izpis vrstice na tiskalnik ob vsakem vertikalnem poniku kazalca
K	prekinitev izpisa na tiskalnik

74LS688

6891



74LS 673 16 BIT serial in, serial out Shift reg with 16 parallel out outputs



ODT NA 11/44

>>> E␣ 17772520 [ret]
17772520 002101
>>> E/N:5␣ 17772520 [ret]
17772520 - - -
17772522 - - -
17772524 - - -
17772526 - - -
177725 30 - - -

>>> D␣ 17772524␣ 25..
>>> D␣+␣ 772524
vsebino je 16 bitna to je odvisno od BYTE ali WORD
>>> S␣ 1000 startanje programa

>>> D␣ SW␣ xxxxx loadanje SWITCH REGISTRA

772524 / 777700 TRACE IZVAJANJA INSTRUKCIJE

772526 / 5000

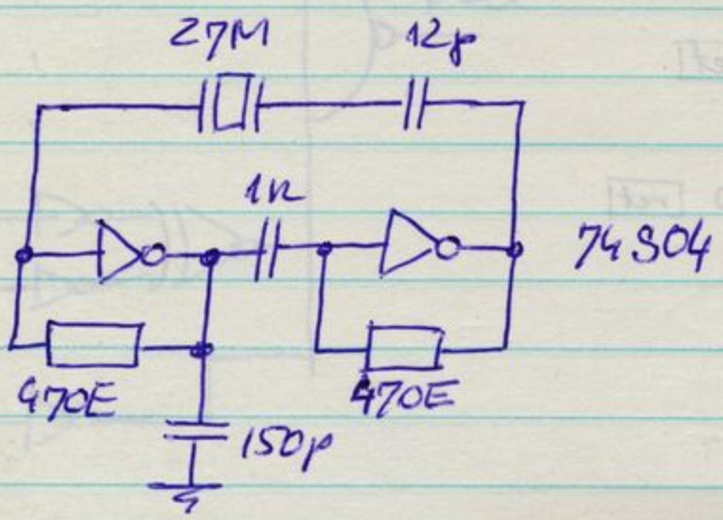
772522 / 60003

1000 / 12737

2 60017

4 77252

OSCILATOR ZA 27 MHz



APPLICATIONS

Since more and more devices are becoming available in DIP form, quite complex systems can be built on the M1710. Some typical applications include:

- Multiword input and/or output.
- Programmable instrument interfaces.
- Interprocessor buffers.
- Custom peripheral controllers.
- Interfacing of:
  - Microprocessors
  - A/D converters
  - Multiplexers
  - Counters
  - Shift registers
  - ROM and RAM memories
  - Arithmetic logic units
  - Programmable logic arrays (PLA)

FUNCTIONS

The M1710 can be divided into four functional sections: address selector logic, bus request logic, data bus interface, and miscellaneous logic.

Address Selector Logic

The address selector logic provides gating signals for up to 16 full 16-bit device registers. Addresses which can be chosen by the user range from 760000 to 777777. The basic M1710 address selection is similar in function to the M105 Address Selector Module. The input signals for the address selector logic consist of: 18 address lines BUS A<17:00>; two bus control lines, BUS C<1:0>; and a master synchronization lines, BUS MSYN. The address selector decodes the 18-bit address on lines BUS A<17:00>; receives MSYN and issues SSYN.

Bus Request Logic

The M1710 contains the circuitry required to make a bus request and gain control of the bus at either the NPR level or at one of the BR levels. The module also includes circuitry required for transferring a vector address during an interrupt operation.

Data Bus Interface

The M1710 contains standard UNIBUS receivers which provide a buffered bus signal output for each of the 16 data lines OUT 00 H through OUT 15 H. Output drive capability of these receivers is seven TTL unit loads.

The module also includes 16 bus drivers which drive data lines IN 00 H through IN 15 H. Input loading to each driver is one standard TTL load. All 16 drivers have two common gate line enables (DRIVER ENABLE 1 and DRIVER ENABLE 2) which require a logic Low for assertion. Each enable represents four TTL unit loads.

Miscellaneous Logic

The following additional circuitry is also provided on the M1710:

- Inverted and noninverted buffered initialize outputs (pins 58 and 59) capable of driving 28 and 30 TTL unit loads respectively.
- A general-purpose flip-flop with all input and output pins available for wire wrap (pins 51 through 57).
- A +3-volt source (in 50) capable of driving 30 TTL unit loads.

DESCRIPTION

The M1710 UNIBUS® Interface Foundation Module is a general-purpose board that provides for the construction of custom interface designs using integrated circuits (ICs). The M1710 lets users build their own interfaces between a wide variety of peripheral equipment and any PDP-11 processor. All essential UNIBUS logic, such as device address selection, interrupt circuitry, and bus receivers and drivers, is provided on the lower portion of the module. The remainder of the board contains IC mounting pads with wire-wrappable pins for custom logic designs. These pads accommodate combinations of all common type of DIP (dual-in-line-package) integrated circuits with up to 40 pins.

The M1710 is a versatile module, ideal for any type of application. The end user, such as a university laboratory familiar with ICs, will appreciate both the capabilities and cost-effectiveness of the module; no additional mounting panel or power supply is required. These features, coupled with the fact that the M1710 is capable of automatic wire wrapping, also should prove valuable to the Original Equipment Manufacturer (OEM) who requires many custom interfaces. And, in all cases, the module is easily adaptable to accommodate any changes in interface design.

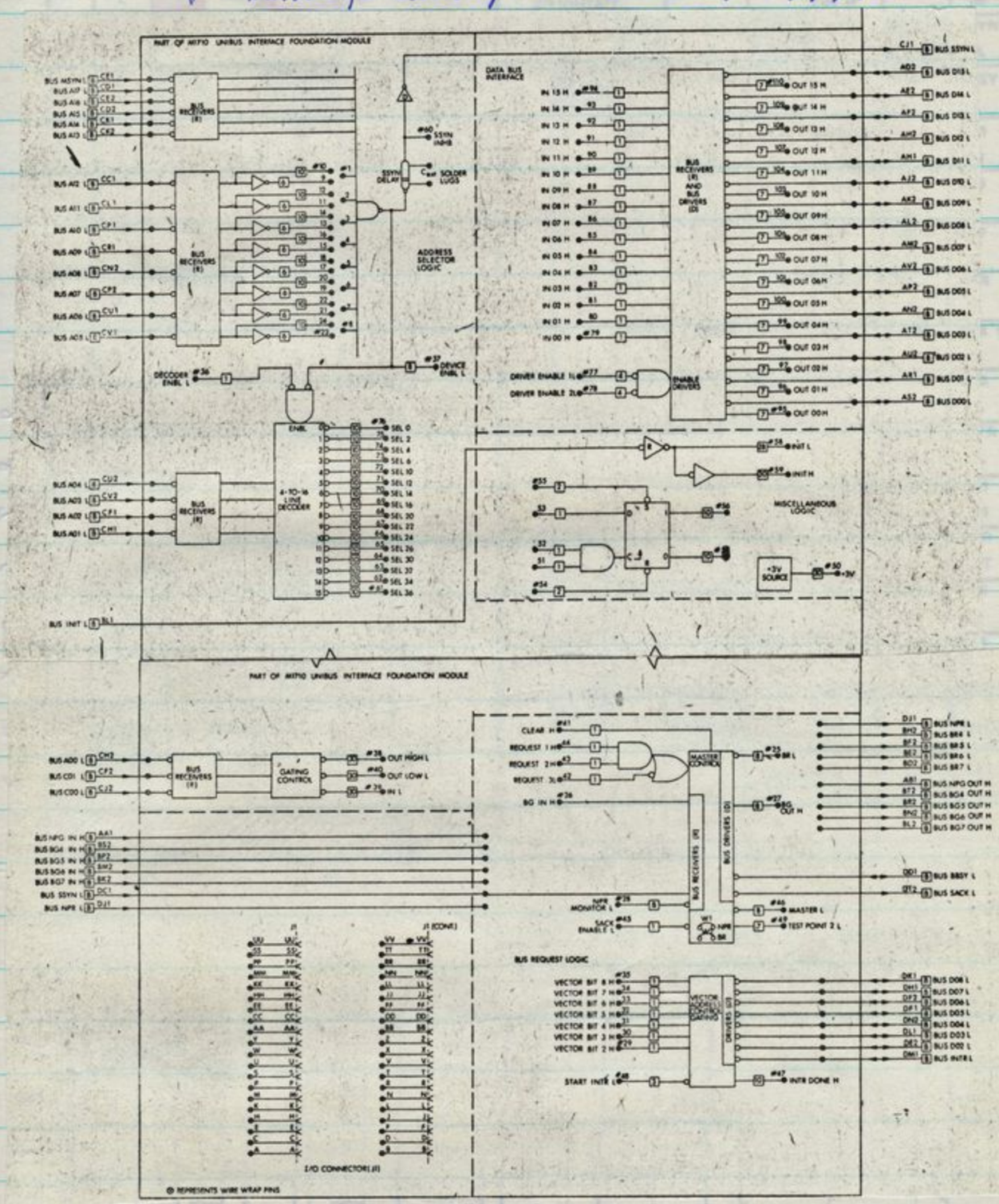
The M1710 plugs into any Small Peripheral Controller (SPC) slot of a DECkit11-M Instrument Interface or DD11 Peripheral Mounting Panel. Additionally, it may be used in a system unit such as the BB11-A. Connection to user equipment is made via a cable connector mounted on the M1710 module.

FEATURES

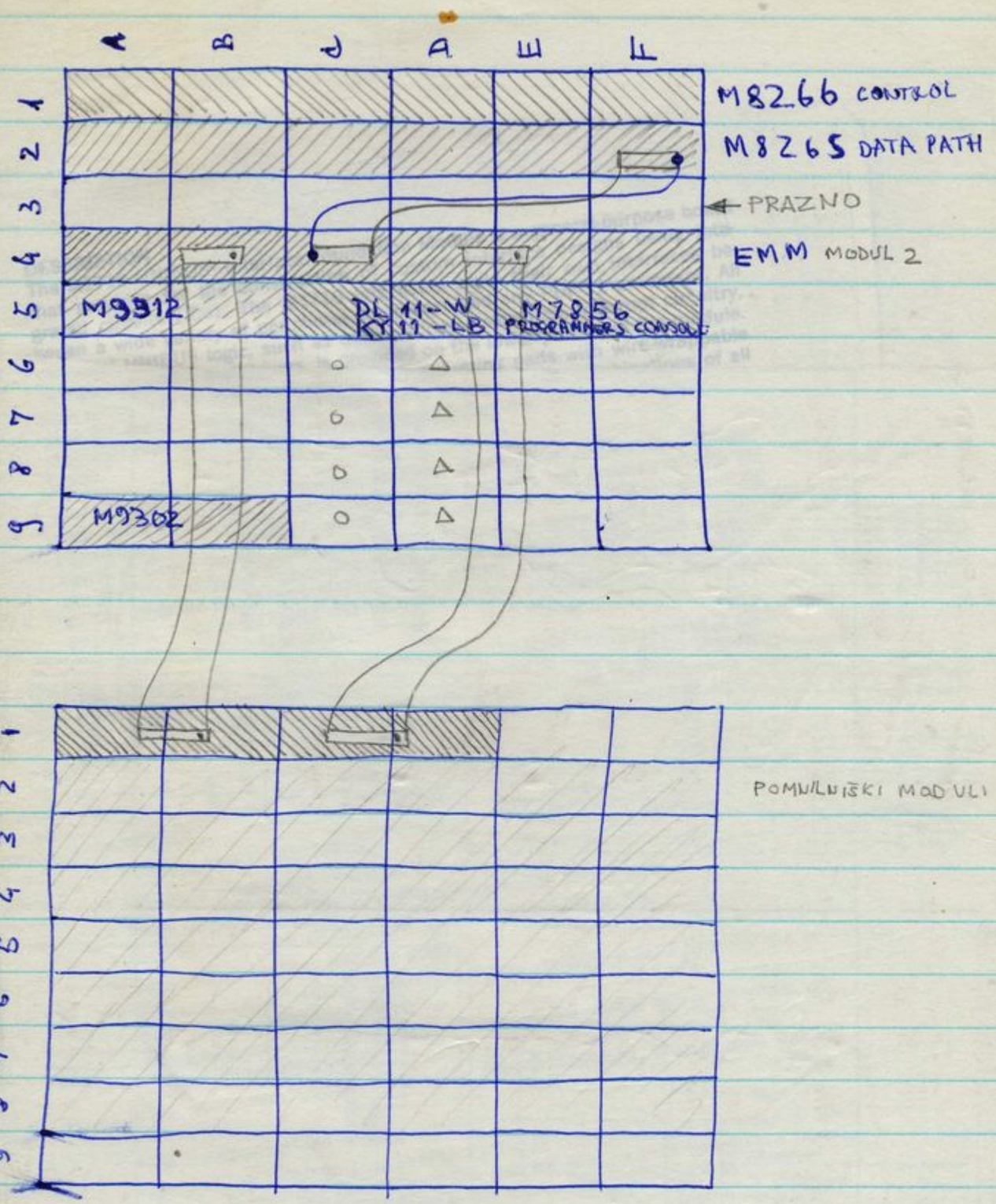
- "Do-it-yourself" interfacing.
- Complete single-card interface.
- Plugs directly into Small Peripheral Controller (SPC) slot.
- Can be used with DECkit11-M Instrument Interface Kit.
- Saves hardware and building costs.
- Preassembled/pretested UNIBUS circuitry eliminates need to build the required bus interfacing functions.
- Wire-wrappable interconnections—compact, 30-gauge wiring used for all IC lead interconnections.
- I/O connection directly to module board—standard 40-conductor cables available.
- All accessories and tools available.
- Accepts all common Dual-in-Line Packages (DIPs); mounts up to 16 of the 14- or 16-pin type plus a multi-use pad set that mounts two 40-pin types, three 24-pin types, four 14- or 16-pin types, or combinations of these.
- Additional bus driver and bus receiver ICs available—special high-impedance devices: DEC 8881, DEC 8640.
- Includes source of +3 V—convenient for tying unused TTL inputs high, etc.

M1710 UNIBUS INTERFACE FOUNDATION MODULE

↓ Adresa je postavljena 760000 ÷ 777777



DELTA 800 KONFIGURACIJA



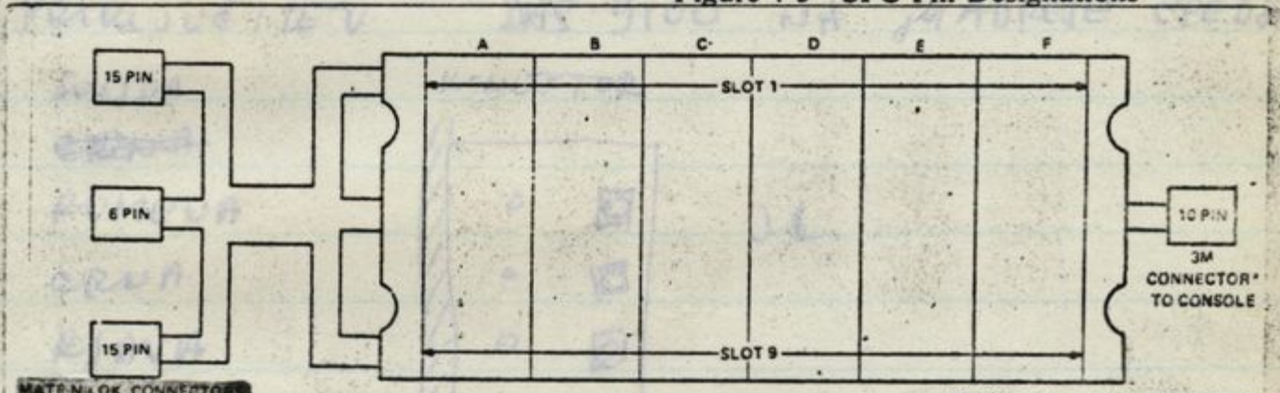
Modified Unibus Pin Designations

Side / Pin	Column A 1	Column A 2	Column B 1	Column B 2	
A	INIT L	+5V	RESV PIN	+5V	1
B	INTR L	TP	RESV PIN	TP	2
C	D00 L	GND	BR5 L	GND	3
D	D02 L	D01 L	+5 BAT	BR4 L	4
E	D04 L	D03 L	INT SSYN	PAR DET	5
F	D06 L	D05 L	AC LO L	DC LO L	6
H	D08 L	D07 L	A01 L	A00 L	7
J	D10 L	D09 L	A03 L	A02 L	8
K	D12 L	D11 L	A05 L	A04 L	9
L	D14 L	D13 L	A07 L	A06 L	10
M	PA L	D15 L	A09 L	A08 L	11
N	PAR P1	PB L	A11 L	A10 L	12
P	PAR P0	BBSY L	A13 L	A12 L	13
R	+15 BAT	SACK L	A15 L	A14 L	14
S	-15 BAT	NPR L	A17 L	A16 L	15
T	GND	BR7 L	GND	C1 L	16
U	+20 (CORE)	BR6 L	SSYN L	C0 L	17
V	+20 (CORE)	+20 (CORE)	MSYN L	-5 (CORE)	18

Side / Pin	Column C 1	Column C 2	Column D 1	Column D 2	Column E 1	Column E 2	Column F 1	Column F 2	
A	NPG (IN)	+5V	TP	+5V	GND A	+5V	ABG OUT	+5V	1
B	NPG (OUT)	-15V	TP	-15V	ASSYN IN H	-15V	ABG IN	-15V	2
C	PA L	GND	A SEL 6	GND	A12 L	GND	SSYN L	GND	3
D	LTC	D15 L	A OUT LOW	BR7 L	A17 L	A15 L	BBSY L	FO1 N1	4
E	TP	D14 L	A SEL 4	BR6 L	MSYN L	A16 L	FO1 V2	D02 L	5
F	TP	D13 L	A SEL 0	BR5 L	A02 L	C1 L	D05 L	D06 L	6
H	D11 L	D12 L	A IN	BR4 L	A01 L	A00 L	D07 L	A INT ENB B	7
J	A INT B	D10 L	A SEL 2	A BR OUT	SSYN L	C0 L	NPR L	GND A	8
K	TP	D09 L	A OUT	BG7 SO	A14 L	A13 L	D08 L	A INT B	9
L	A INT ENBB	D08 L	INIT L	BG7 OUT	A11 L	TP	D03 L	FO1 L2	10
M	TP	D07 L	AINT ENBA	BG6 SO	AIN	AOUT HIGH	INTR L	FO1 M2	11
N	DC LO	D04 L	A INT A	BG6 OUT	A OUT LOW	A08 L	FO1 N1	D04 L	12
P	HALT REQ	D05 L	TP	BG5 S0	A10 L	A07 L	ABR OUT	FO1 P2	13
R	HALT GRT	D01 L	TP	BG5 OUT	A09 L	ASEL 4	FO1 L2	FO1 N1	14
S	PB L	D00 L	TP	BG4 S0	ASEL 6	ASEL 0	FO1 M2	FO1 P2	15
T	GND	D03 L	GND	BG4 OUT	GND	ASEL 2	GND	SACK L	16
U	+15/+8 15V	D02 L	TP	ABG IN	A06 L	A04 L	A INT A	ABR OUT	17
V	AC LO	D06 L	ASSYN IN H	ABG OUT	A05 L	A03 L	A INT ENB A	FO1 FO1	18

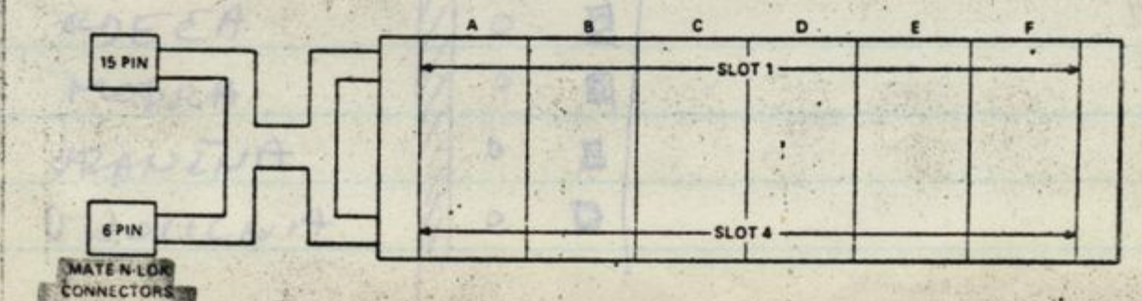
11-4630

Figure 4-5 SPC Pin Designations



*3M Connector installed on DD11-PK only

DD11-DK and DD11-PK NINE-SLOT BACKPLANES (module side)



DD11-CK FOUR SLOT BACKPLANE (module side)

NOTE
The 3M connector is installed only if the operator's console is present.

11-0456

oboje 9600 baud

DL 11 W NASTAVITEV ZA TU 58

	1	2	3	4	5	6	7	8	9	10
S5	F	N	F	N	F	N	N	N	N	F
S4	N	N	F	F	N	F	N	N	N	F
S3	N	N	F	F	N	N	F	N	F	N
S2	F	F	F	F	N	F	N	F		
S1	N	N	F	N	F	F	N	N	F	N

VEKTOR 300

ADR 176500

DL 11 W NASTAVITEV ZA KONZOLO

	1	2	3	4	5	6	7	8	9	10
S5	F	F	F	N	F	F	N	F	F	N
S4	N	F	F	F	N	F	F	F	F	F
S3	N	N	F	F	N	F	N	F	N	F
S2	F	F	N	F	F	N	F	F		
S1	F	F	N	N	F	N	F	N	F	N

PRIKLJUČITEV DAS 9100 NA MADREJE CPED800

SONDA	KONEKTOR	
~~CRNA~~		
RUMENA	o	1
ČRNA	o	2
RJAVA	o	3
ZELENA	o	4
RDEČA	o	5
MODRA	o	6
ORANŽNA	o	7
VIOLIČNA	o	8

)L

2.2 INTERFACE INFORMATION

The LA120 interfaces with EIA devices using an optional modem cable. The interface is compatible with Bell 103, 212A, and 202 modems and meets EIA specification RS232-C requirements. The interface conforms to CCITT recommendation V.2.4.

2.2.1 Interface Signals

Table 2-1 summarizes EIA interface signals. The following paragraphs describe interface signals.

2.2.1.1 Protective Ground – This conductor is connected to the LA120 chassis. It is further connected to external grounds through the third wire of the power cord.

2.2.1.2 Transmitted Data (TDX) – The direction of TDX is from the LA120. Signals on this circuit represent serially encoded characters generated by the LA120.

2.2.1.3 Received Data (RDX) – The direction of RDX is to the LA120. Signals on this circuit represent serially encoded characters generated by the user's equipment.

Table 2-1 Summary of LA120 EIA Interface Signals

Pin	Source	Name	Function	Circuit CCITT/EIA
1	–	–	Protective ground	101/AA
2	LA120	TXD	Transmitted data	103/BA
3	User	RXD	Received data	104/BB
4	LA120	RTS	Request to send	105/CA
5	User	CTS	Clear to send	106/CB
6	User	DSR	Data set ready	107/CC
7	–	–	Signal ground	102/AB
8	User	RLSD	Carrier detect	109/CF
9	–	–	–	–
10	–	–	–	–
11*	LA120	SRTS	Sec. req. to send	120/SCA
12†	User	SPDI	Speed indicator (FDX)	112/CI
	User	SRLSD	Sec. carrier det. (HDX)	122/SCF
13	–	–	–	–
14	–	–	–	–
15	–	–	–	–
16	–	–	–	–
17	–	–	–	–
18	–	–	–	–
19*	LA120	SRTS	Sec. req. to send	120/SCA
20	LA120	DTR	Data term. ready	108.2/CD
21	–	–	–	–
22	User	RI	Ring indicator	125/CE
23*	LA120	SPDS	Speed select (FDX)	111/CH
24	–	–	–	–
25	–	–	–	–

* Pins 11, 19, and 23 are driven by a common circuit whose function is determined by the modem and secondary channel SET-UP commands.

† Pin 12 is SPDI for full-duplex operation. For half-duplex operation, pin 12 is SRLSD.

2.2.1.4 Request To Send (RTS) – The direction of RTS is from the LA120. The on condition of RTS means that the LA120 intends to transmit data. After turning this circuit on, the LA120 waits for a clear to send (transmit enable) condition before starting transmission.

2.2.1.5 Clear To Send (CTS) – The direction of CTS is to the LA120. Although the LA120 physically receives this signal, it is not used for any purpose. Depending on the modem control protocol in use, either RLSD, SRLSD, or a timeout after asserting RTS is used to provide a clear to send (transmit enable) condition.

2.2.1.6 Data Set Ready (DSR) – The direction of DSR is to the LA120. The on condition of DSR indicates that the user's equipment is capable of transmitting and receiving data signals. The off condition of DSR causes the LA120 to ignore all other interface inputs except ring indicator (RI). In full duplex without EIA control, this circuit is assumed to always be in the on condition.

2.2.1.7 Signal Ground – This circuit establishes the common ground reference potential for all interface circuits except protective ground. The circuit is permanently connected to the protective ground circuit.

2.2.1.8 Carrier Detect (RLSD) – The direction of RLSD is to the LA120. The on condition of RLSD indicates that data transmission from the user's equipment to the LA120 is enabled. In full duplex without EIA control, this circuit is assumed to always be in the on condition.

2.2.1.9 Secondary Request To Send (SRTS) – The direction of SRTS is from the LA120. In certain half-duplex modes, the on condition of SRTS indicates that the LA120 is capable of successfully processing the received data from the user's equipment. In restraint mode, the off condition of SRTS indicates that the user's equipment should temporarily suspend the transmission of data. When SRTS goes on, transmission may be resumed.

2.2.1.10 Speed Indicator (SPDI) (Full Duplex Only) – The direction of SPDI is to the LA120. The on condition of SPDI indicates that the baud rate is 1200, regardless of the rate selected by the operator. The off condition indicates that the operator-selected baud rate is being used.

2.2.1.11 Secondary Carrier Detect (SRLSD) (Half Duplex Only) – The direction of SRLSD is to the LA120. The on condition of SRLSD indicates that the user's equipment is capable of successfully processing the transmitted data from the LA120.

2.2.1.12 Data Terminal Ready (DTR) – The direction of DTR is from the LA120. The on condition of DTR indicates that the LA120 is capable of transmitting and receiving data signals. The off condition of DTR may cause the user's equipment to set the DSR (data set ready) to the off condition. When DTR is off, the LA120 ignores all interface inputs except ring indicator (RI).

2.2.1.13 Ring Indicator (RI) – The direction of RI is to the LA120. If data terminal ready (DTR) is off, then the on condition of RI causes DTR to turn on. DTR remains on until data set ready (DSR) turns on or 30 seconds elapses, whichever occurs first. Then DTR turns off. If DTR is on, the on condition of RI causes a 30-second timeout. If no data is received in 30 seconds, DTR is pulsed low for 233 ms ± 10 percent.

2.2.1.14 Speed Select (SPDS) (Full Duplex Only) – The direction of SPDS is from the LA120. If the operator-selected baud rate is 600 or higher, the LA120 asserts an on condition of SPDS; otherwise, the LA120 holds this circuit in the off condition.

2.2.2 EIA Interface Cables

BC22A and BC22B interface cables are described in the following paragraphs.

2.2.2.1 BC22A-10, -25 – BC22A interface cables come in 10 and 25 foot lengths for hookup between LA120 and computer.* Each end is terminated with a female molded connector. The cable is shielded, contains six conductors, and is wired in a null modem configuration.

2.2.2.2 BC22B-10, -25 – BC22B interface cables come in 10 and 25 foot lengths for hookup between LA120 and modem.† They can also be used for cable extension. Connectors are molded with a male connector at one end and a female at the other end. Cable is shielded and has 14 conductors.

2.2.3 Impedance of Terminator

The terminating impedance of the receiving end of the interface circuits has a dc resistance of not less than 3000 ohms or more than 7000 ohms. When the interface plug is disconnected, interface voltage on the terminator circuits is –2 V to +2 V.

2.2.4 Rise and Fall Times

The circuitry that receives signals from an interface circuit depends only on signal voltage and conforms to RS232-C rise time and fall time. For control interface circuits, the time required for the signal to pass through the transition region (–3 V to +3 V) during a change in state does not exceed 1 μs. For the transmitted data circuit, the rise time and fall time do not exceed 16.7 μs through the 6 V range (–3 V to +3 V).

2.2.5 Open Circuit Voltages

The open circuit driver voltage for signal ground on any interface circuit does not exceed –12 V to +12 V. The terminator on an interface circuit is designed to withstand any input signal within a –25 V to +25 V range. When the terminating impedance is in the proper range (3000 to 7000 ohms) and the terminator open circuit voltage is zero, the potential at the point of interface is not less than –5 V to +5 V or more than –12 V to +12 V. An open circuit or applied voltage more negative than +0.6 V will be interpreted the same as a legitimate negative voltage (–3 V to –25 V).

2.3 LA120 SPECIFICATIONS

The LA120 specifications include data on the printer, keyboard, communications, physical aspects, and paper. The data in each of these categories is given in Tables 2-2 through 2-6.

TU 58 RADIAL SERIAL PROTOCOL (RSP)

Podatkovni paketi: ukazi — command
podatki — data
končna sporočila — end message { INIT, CONTIN, XOFF

FLAG BYTE prve vrednost paketa
- biti 7–5 so rezervirani

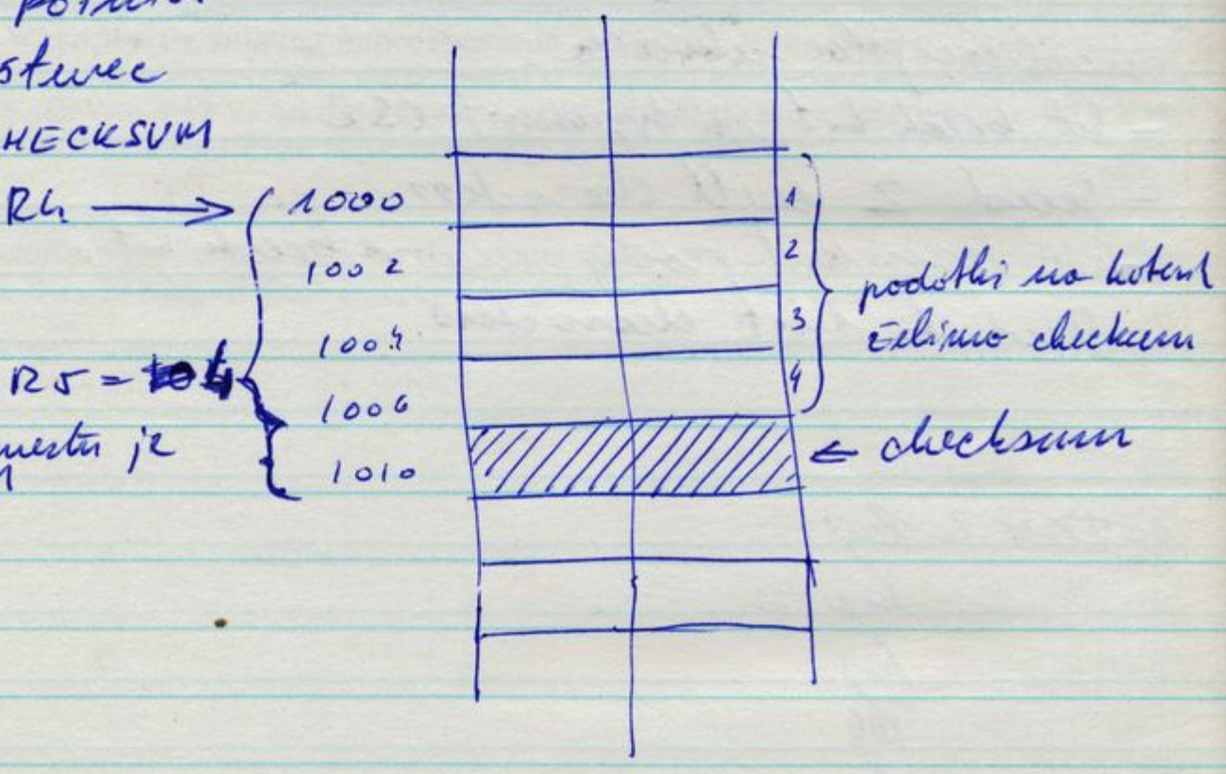
01_8	00001	DATA
02_8	00010	CONTROL (COMMAND)
04_8	00100	INIT.
10_8	01000	BOOTSTRAP
20_8	10000	CONTINUE
23_8	10011	XOFF

Inicializacijska sekvenca:
- Set break bit in transmit CSR
- send 2 null characters
- IF transmit ready, remove break bit
- Send 2 init characters

A27204 2 109
102
104
106
110

100	12704	MOV #1000, R4
102	1000	
104	12705	MOV #10, R5
106	10	
110	5000	CLR R0
112	62400	ADD (R4)+, R0
114	103405	BCS (126)
116	5305	DEC R5
120	100374	BPL (112)
122	10024	MOV R0, (R4)+
124	000	HALT
126	5200	INC R0
130	772	BR (116)

R4 = pointer
R5 = števec
R0 = CHECKSUM



LOAD TRAKA STREAMER

[1] [2] [3] [4] [5]

A 45335 PRIMIZ RONO,DA

This boot ...

```
777 177004 : 1000/012701        MOV #176500, R1
             1002/176500
             1004/012702        MOV #176504, R2
             1006/176504
             1010/010100        MOV R1, R0
             1012/005212        INC (R2)
             1014/105712    C:  TSTB (R2)
             1016/100376        BPL C
             1020/006300        ASL R0
             1022/001004        BNE B
             1024/005012        CLR (R2)
             1026/005200        INC R0
             1030/005761        TST 2(R1)
             1032/000002
             1034/042700    B:  BIC #20, R0
             1036/000020
             1040/010062        MOV R0, 2(R2)
             1042/000002
             1044/001363        BNE C
             -------------------------------
             1046/005003        CLR R3
             1050/105711    D:  TSTB (R1)
             1052/100376        BPL
             1054/116123        MOVB 2(R1), (R3)+
             1056/000002
             1060/022703        CMP #1000, R3
             1062/001000
             1064/101371        BHI D
             1066/005007        CLR PC     (4KT)
```

Delo ne /24 ! a) profilem ↑
b) progi + ...

... DL11-W + ...

LOAD TRACK1 STREAMER

1 2 3 4 5

A 45335 — PRIMER KOMANDNE VSTAVLJE VRTENJE

B LOAD

C ON LINE

MERJENJE DOLŽINE INSTRUKCIJ

```
START  MOV #40000, R2          100  12702
LOOP   MOV #2000 , R1          102  40000
ZANKA  SOB  R1, ZANKA          104  12701
       SOB  R2 , LOOP          106   2000
       MOV  BELL , @#RBUFF     110  77710 1
       JMP  START              112  77204
                               114  12737
                               116      7
                               120  777566
                               122    137
                               124    100
```

izvajanje tega programa je 56,6 s

to povečamo toliko da zanka skoči na instrukcijo, ki jo testiramo.

$40000_8 = 16384_{10}$

$2000_8 = 1024_{10}$

16384 × 1024

16.777.216

Merimo čas med pisanjem na konzolo.

Vrnimo ta gre ponovno v zanko konzola zapiska.

Primer merjenja instrukcije MOV #AAA, @#BBB

```
START   MOV #40000, R2
LOOP    MOV #2000, R1
ZANKA   instrukcija ki jo testiramo MOV #AAA, @#BBB
        SOB R1, ZANKA
        SOB R2, LOOP
        MOV BELL, @#RBUFF
        JMP START
```

```
100     12702
102     40000
104     12701
106     2000
110     12737
112     100
114     1000
116     77710
120     77207
122     12737
124     7
126     777566
130     137
132     100
```

to testiramo

to zanka traja 2 min in 40 sek = 160 sek

$$t_{(MOV+SOB)} \quad t_{(SOB)} = t_{(MOV)}$$

$$t_{mov} = \frac{160\,s - 56{,}64\,s}{16\,777\,216} \doteq 6{,}4\,\mu s$$

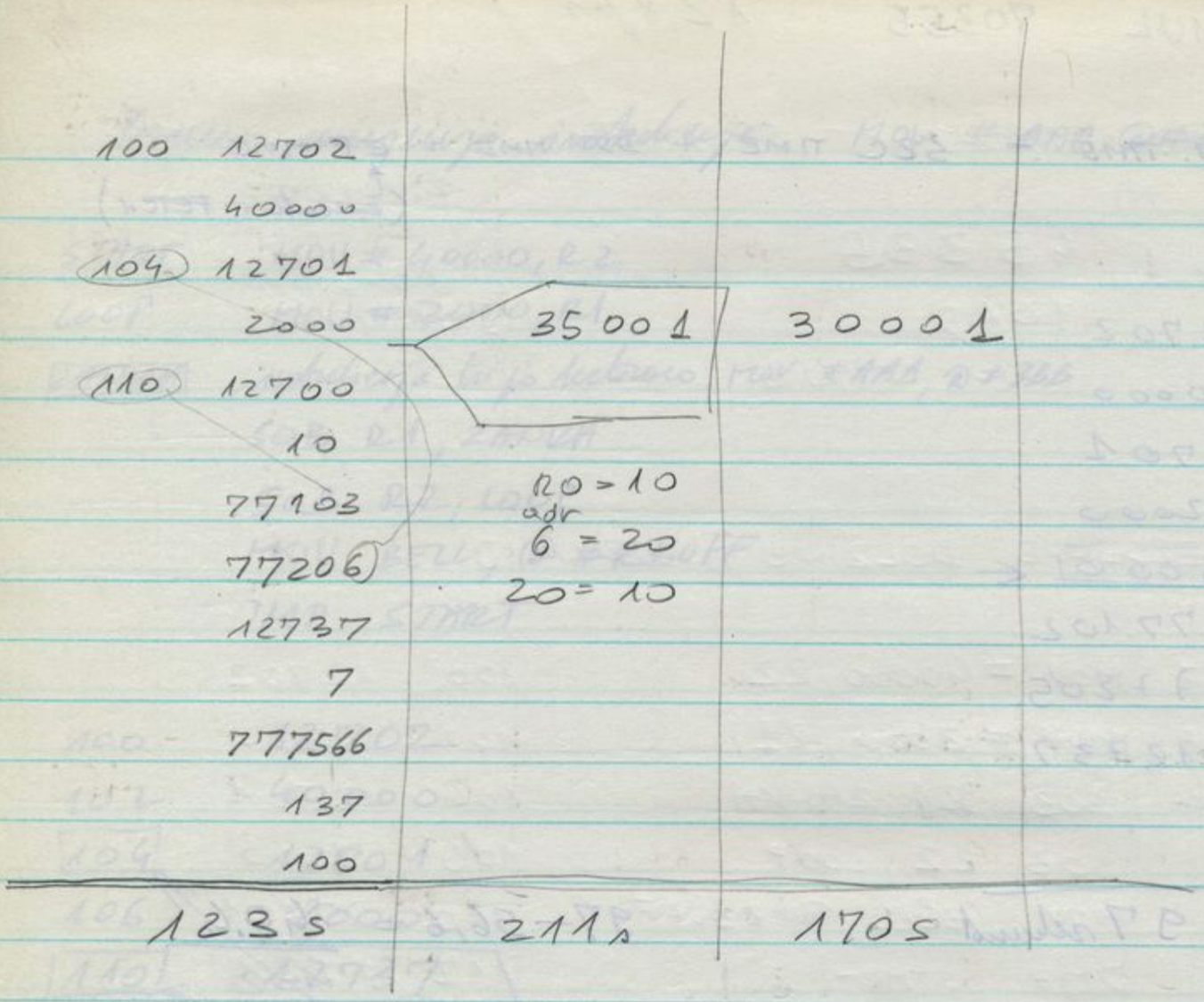
MUL 70355 12,8 µs

INST. TIME = SRC TIME + DST TIME + EF TIME
(execute + FETCH)

1000 12702
2 40000
4 12701
6 2000
10 [50001] ←
12 77102
14 77205
16 12737
.
.
.

97 zehntel 97 − 56,6 = 40,4 / 102

1000 12702			
2 40000			
4 12701			
6 2000			
1010 32020	← 32000	30020	33000
5000			
77103			
77206			
12737			
7			
77756			
137			
1000			
184 zehntel · 97/87	164 zehntel · 97/67	164 zehntel	185 zehntel
5,186 µs	3,99 µs	3,99 µs	5,2 µs



211
− 170
41 : 16 777 216 = 2.45
+0.33
2.78

BIT SSDD	INST TIME µs	SOURCE TIME	MODE
30001	2.8	0	0
31101	3.7	0.9µ+0.4	1.30	1
32701	4	1.2µ+0.4	1,60	2
33701	5.2	2,4 +0.4	2,80	3
34701	4	1.2 +0.4	1.60	4
35001	5.25	2.45 + 0.4	2.85	5
36701	5,6	2,8 + 0.4	3,20	6
37701	6.6	3.8 + 0,4	4.20	7

BIT SS DD	µs INST TIME	µs	DST TIM	MODE	ČAS PROGRAMU Δ ZAŠKRT BREZ INST 123
30001	2.80		0	0	170
30014	3.87	1.07		1	188
				2	
				3	
				4	
				5	
				6	
				7	

BIS SS DD	µs INST. TIME	µs	DST TIM	MODE	
50001	2.8µs		0	0	45.4
50011	4.4µs	1.60 +0.1	1.75	1	
50021	4.6µs	1.80	1.90	2	
50032	5.9µs	3.1	3.30	3	
50042	4.62µ	1.82	1.92	4	
50052	5.9µ	3.1	3.30	5	
50062	6.2µ	3.4	3.50	6	
50072	7.35µ	4.55	4.70	7	

MOV	INST TIME	INST TIME - 2.40 - 0.3	DST TIME	
010000	2.73	0		0
010011	3.80	1.1		1
010021	4.20	1.5		2
010032	5.00	2.3		3
010042	4.20	1.5		4
010052	5.00	2.3		5
010062	5.40	2.70		6
010072	6.55	3.85		7

50
420
-.80

EXEC + FETCH + SERV + MDO + MSO

	MODE	
ADD	0	2.4 µ
SUB	ostali	2.6 µ
CMP		2.4 µ
BIT		2.2 µ
BIC		2.1 µ
BIS	0	2.36
	ostali	2.56
XOR	0	2.36
	ostali	2.56
MOV	0	2.40
	ostali	2.60

SOURCE ADRS TIME = CELOTEN ČAS − (ČAS EXEK (0.3) + SERVICE (0.3) + FETCH (1.3) + MODE 0 MIKRONST (0.3) + DMO)

	SM	ST	
	0	0 CIKLOV	0.00 µs
DOUBLE OPERAND	1	1	1.3
	2	1	1.6
	3	2	2.8
	4	1	1.6
	5	2	2.85
	6	2	3.20
	7	3	4.20

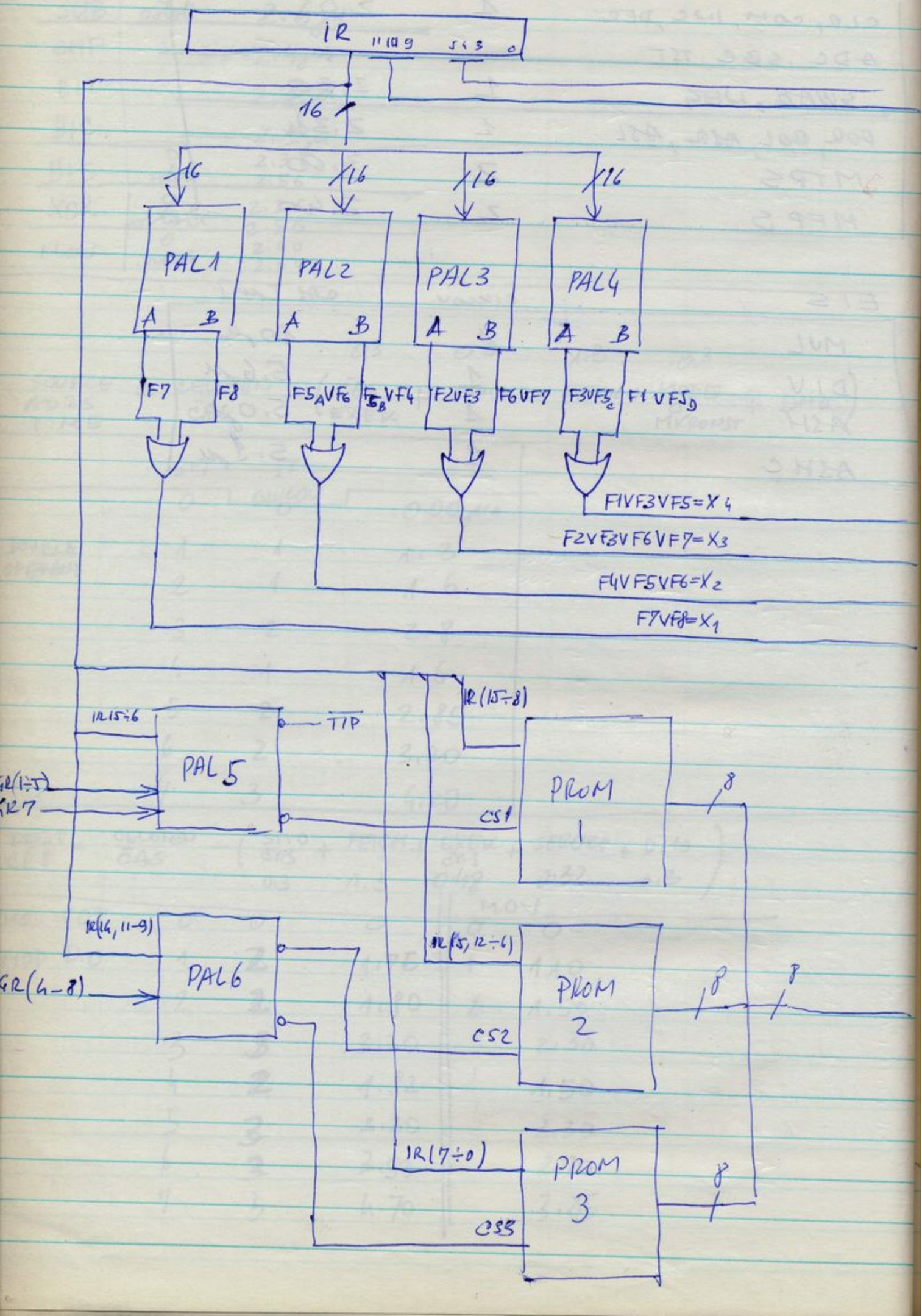
DEST ČAS = CELOTEN ČAS − (SM 0 ČAS (0.3) + FETCH (1.3) + EXEK ČAS (0.48) + SERVICE (0.32) + DMO (0.3))

				MOV	
MOD SOP	0	0	0	0	0
MOD D.O.	1	1	1.75	1	1.10
	2	1	1.80	2	1.50
	3	3	3.30		2.30
	4	1	1.82		1.50
	5	3	3.30		2.30
	6	3	3.50		2.70
	7	3	4.70		3.85

SOP	CIKLOV	CAS [μs]
CLR, COM, INC, DEC	1	2.0
ADC, SBC, TST		
SWAB, NEG	1	2,30
ROR, ROL, ASR, ASL	1	2.31
MTPS	2	3.60
MFPS	2	2.40

ETS	CIKLOV	CAS [μs]
MUL	1	10 μ
DIV	1	5.6 μ
ASH	1	5.0 μ
ASHC	1	5.9 μ

Dekodiranje instrukcij za DELTA 16/B-S



SOURCE ADRS TIME = CELOTEN ČAS − (ČAS EXEK. + SERVICE + FETCH + MIKRO IN MODE 0 SMO + DMO)

	ČAS EXEK.	SERVICE	FETCH	MIKRO IN MODE 0 SMO	DMO	
24 MHz	0.3	0.3	1,3	0.3	0.3	[µs]
27 MHz	0.27	0.27	1.27	027	0.27	[µs]

INSTRUKCIJA	SM	ST MEM CIKLOV	t/24 MHz CLOCK [µs]	t/27 MHz CLOCK [µs]
DOP	0	0	0.00	0.00
	1	1	1.30	1.13
	2	1	1.60	1.43
	3	2	2.80	2.63
	4	1	1.60	1.43
	5	2	2.85	2.68
	6	2	3.20	3.03
	7	3	4.20	4.03

DEST. TIME = CELOTEN ČAS − (ČAS EXEK. + SERVICE + FETCH + SMO + DMO)

	ČAS EXEK.	SERVICE	FETCH	SMO	DMO
24 MHz →	0.48	0.32	1.3	0.3	0.3

INSTRUKCIJA	DM	ST MEM CIKLOV	t/24 MHz CLOCK [µs]	t/27 MHz CLOCK [µs]
MODIFYING	0	0	0	0
SOP in DOP	1	2	1.75	1.58
	2	2	1.90	1.73
	3	3	3.30	3.13
	4	2	1.92	1.75
	5	3	3.30	3.13
	6	3	3.50	3.33
	7	4	4.70	4.53

MERITVE ČASOV

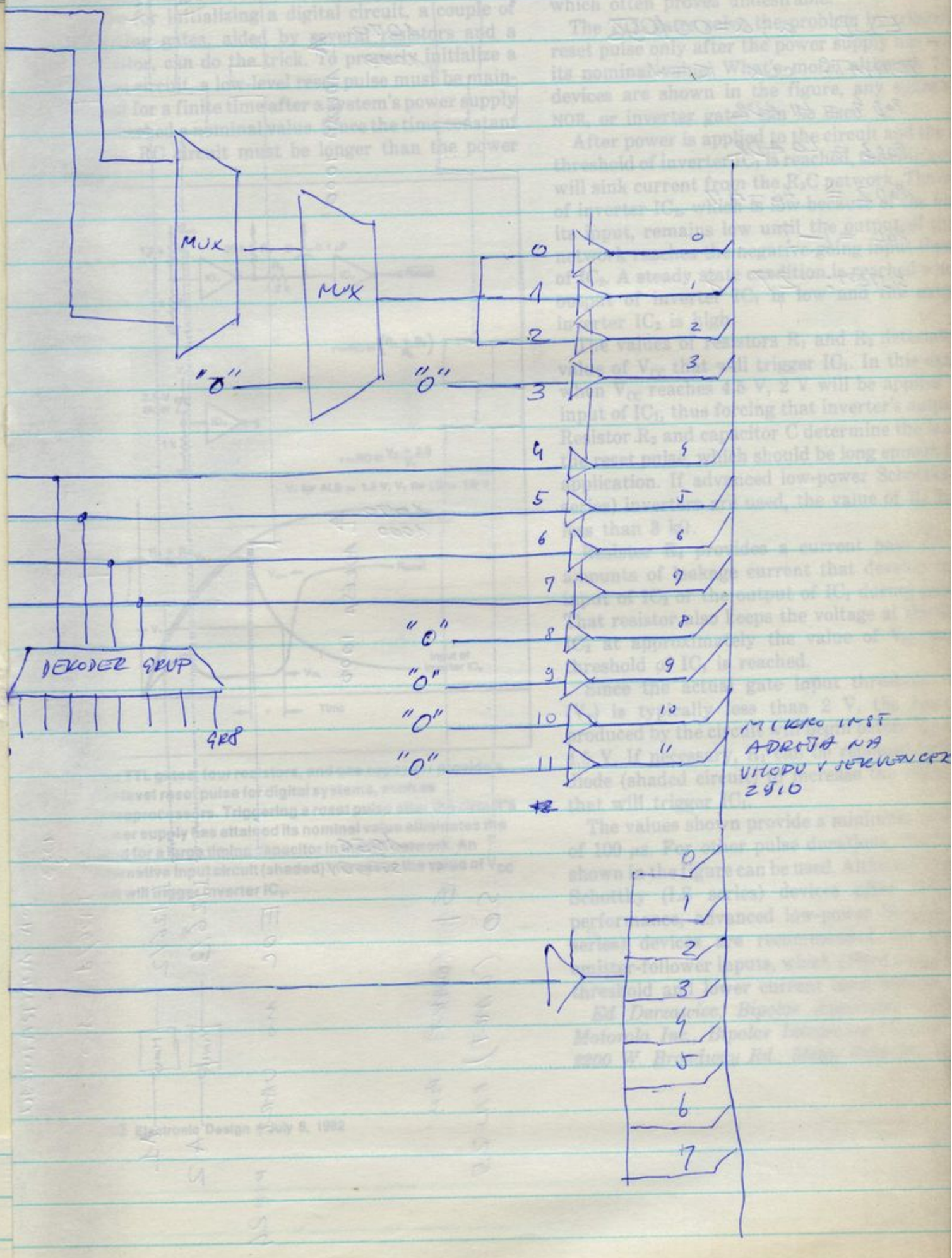
DEST. TIME

INSTR	MODE	št. pom. ciklov	t/24 MHz CLK [μs]	t/27 MHz CLK [μs]
	0	0	0	0
MOV	1	1	1.10	0.93
	2	1	1.50	1.33
	3	2	2.30	2.13
	4	1	1.50	1.33
	5	2	2.30	2.13
	6	2	2.70	2.53
	7	3	3.85	3.68

EXECUTE + FETCH + SERV + MDO + MSO

INSTRUKCIJA	MODE	t/24 MHz clk [μs]	t/27MHz clk [μs]
ADD	0	2.4	2.29
SUB	OSTALI	2.6	2.47
CMP		2.4	2.29
BIT		2.1	2.02
BIC		2.1	2.02
BIS	0	2.36	2.25
	OSTALI	2.56	2.45
XOR	0	2.36	2.25
	OSTALI	2.56	2.45
MOV	0	2.40	2.29
	OSTALI	2.60	2.47

SOP EXEC TIME	ŠT CIKLOV	24 MHz CLK [μs]	27 MHz CLK [μs]
CLR, COM, INC, DEC, SBC, ADC, TST	1	2.0	1.93
SWAB, NEG	1	2.30	2.20
ROR, ROL, ASR ASL	1	2.31	2.21
MTPS	2	3.60	3.37
MFPS	2	2.40	2.29
EIS			
MUL	1	10	9.13
DIV	1	5.6	5.17
ASM	1	5.0	4.63
ASHC	1	5.9	5.44


MUX
MUX
"0"
"0"
DEKODER GRUP
GRB
"0"
"0"
"0"
MIKRO INST ADRESA NA VIHODU V SEKVENCER 2910

AVD016

TRACE : SINGLE

END a;a=001056

FORMAT: CLK K=↓

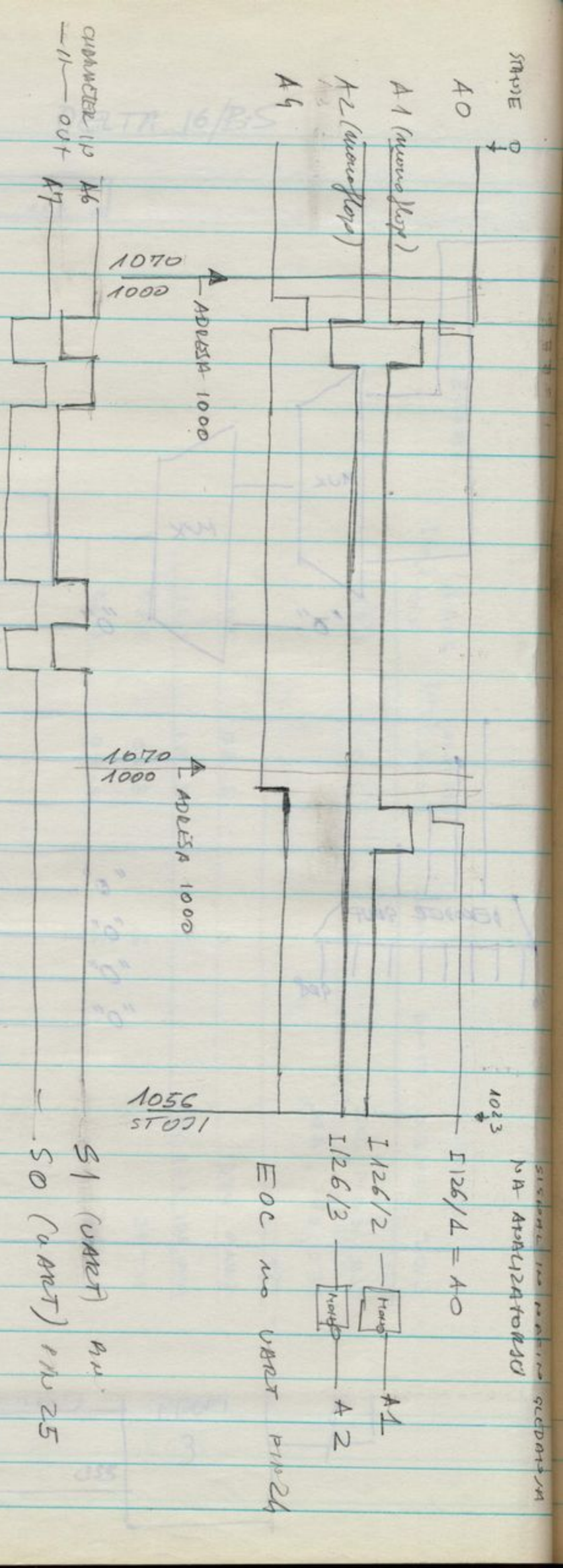
POD4 = HI ADR

POD3 = LO ADR

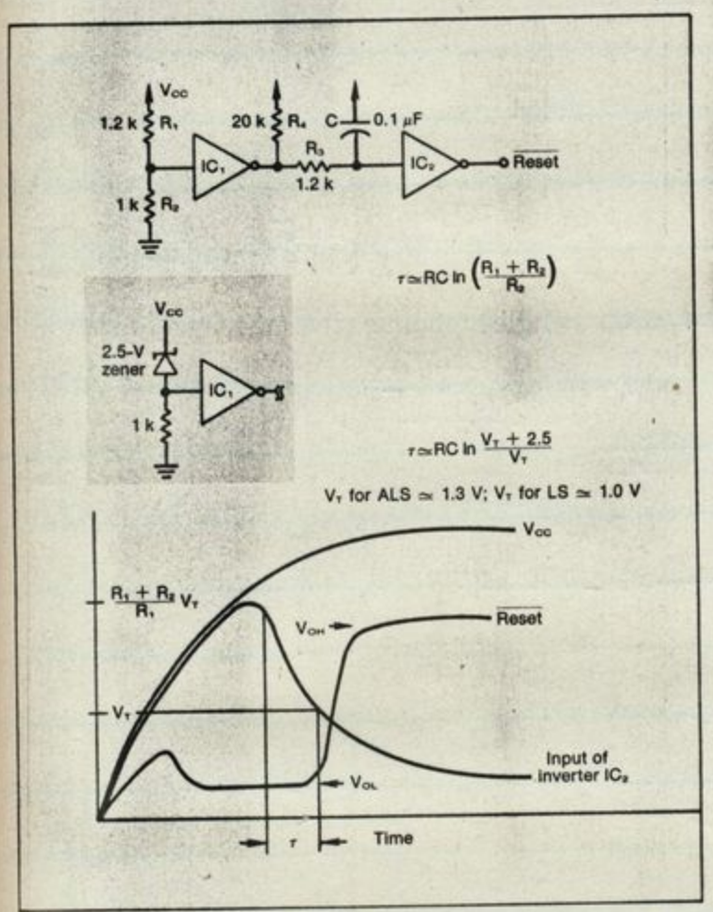
POD2 = $A_0 - A_7$

POL = [—]

SISTEM: 27 8



When simple RC reset circuits are not a practical choice for initializing a digital circuit, a couple of inverting gates, aided by several resistors and a capacitor, can do the trick. To properly initialize a digital circuit, a low-level reset pulse must be maintained for a finite time after a system's power supply has reached a nominal value. Since the time constant of the RC circuit must be longer than the power supply's, a large timing capacitor may be required, which often proves undesirable.



Two TTL gates, four resistors, and one capacitor provide a low-level reset pulse for digital systems, such as microprocessors. Triggering a reset pulse after the circuit's power supply has attained its nominal value eliminates the need for a large timing capacitor in the RC network. An alternative input circuit (shaded) increases the value of V_{CC} that will trigger inverter IC_1.

The TTL gates solve the problem by triggering a reset pulse only after the power supply has reached its nominal value. What's more, although 74LS04 devices are shown in the figure, any spare NAND, NOR, or inverter gates can be used.

After power is applied to the circuit and the input threshold of inverter IC_1 is reached, the chip's output will sink current from the R_3C network. The output of inverter IC_2, which is low because of the high on its input, remains low until the output of the R_3C network reaches the negative-going input threshold of IC_2. A steady state condition is reached when the output of inverter IC_1 is low and the output of inverter IC_2 is high.

The values of resistors R_1 and R_2 determine the value of V_{CC} that will trigger IC_1. In this example, when V_{CC} reaches 4.5 V, 2 V will be applied to the input of IC_1, thus forcing that inverter's output low. Resistor R_3 and capacitor C determine the length of the reset pulse, which should be long enough for the application. If advanced low-power Schottky (ALS series) inverters are used, the value of R_3 must be less than 3 kΩ.

Resistor R_4 provides a current path for small amounts of leakage current that develop into the input of IC_2 or the output of IC_1 during power-up. That resistor also keeps the voltage at the input of IC_2 at approximately the value of V_{CC} until the threshold of IC_1 is reached.

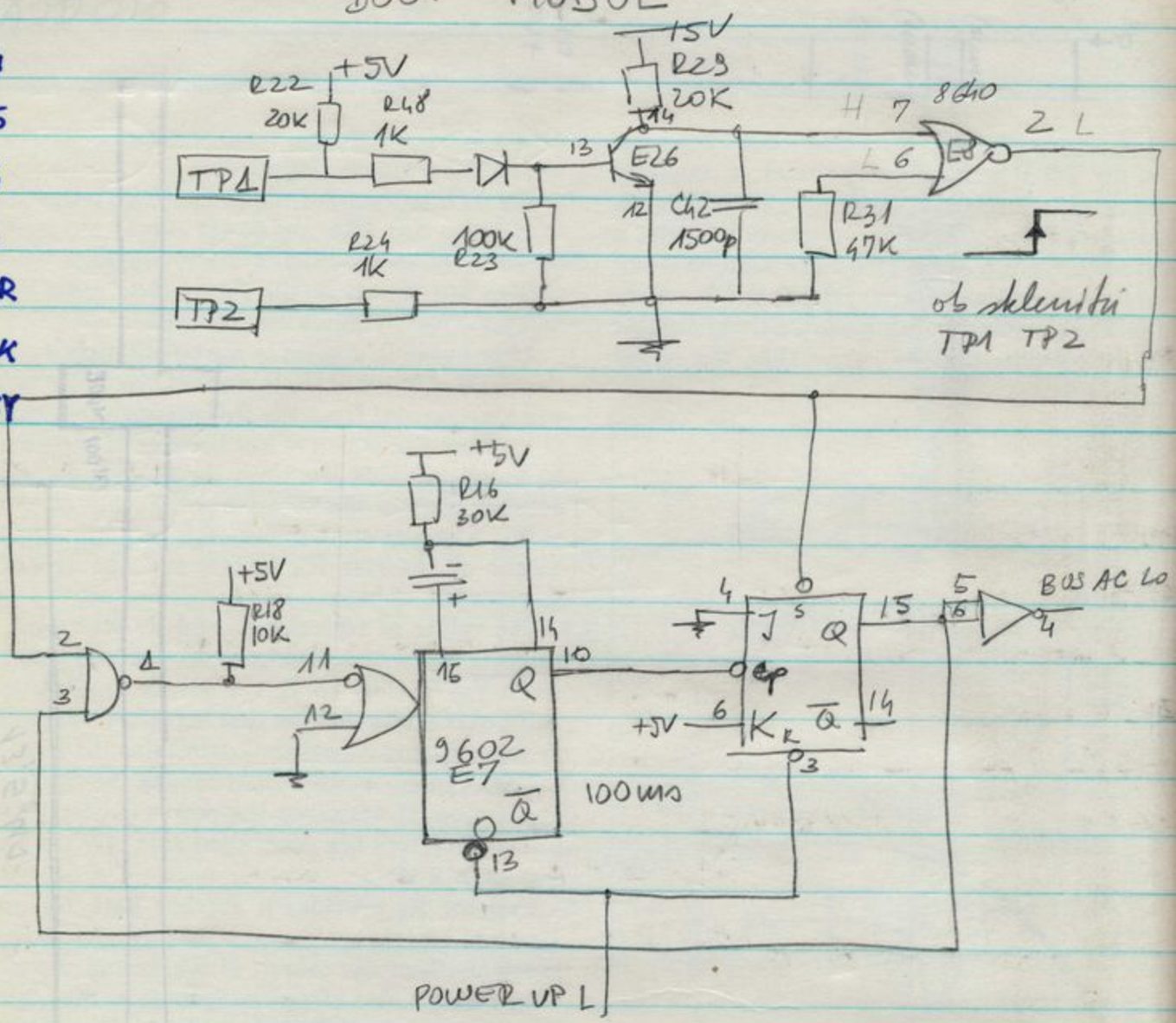
Since the actual gate input threshold voltage, (V_T) is typically less than 2 V, the reset pulse produced by the circuit will begin before V_{CC} reaches 4.5 V. If necessary, R_1 can be replaced by a zener diode (shaded circuit) to increase the value of V_{CC} that will trigger IC_1.

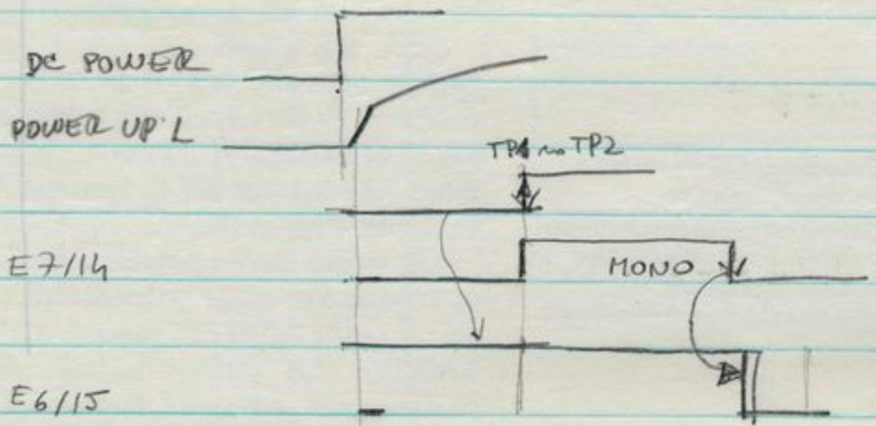
The values shown provide a minimum reset pulse of 100 μs. For other pulse durations, the formulas shown in the figure can be used. Although low-power Schottky (LS series) devices offer satisfactory performance, advanced low-power Schottky (ALS series) devices are recommended for their pnp emitter-follower inputs, which afford a higher input threshold and lower current consumption.

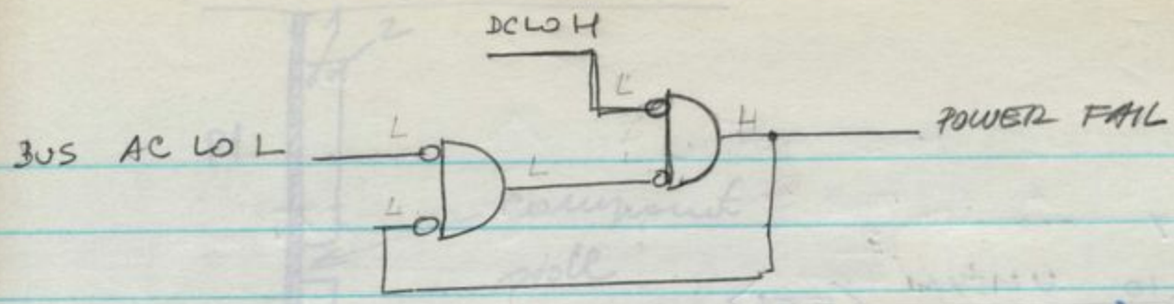
Ed Derzawiec, Bipolar Applications Engineer, Motorola Inc., Bipolar Integrated Circuits Group, 2200 W. Broadway Rd., Mesa, Ariz. 85202.

BOOT MODUL

- ⊗ RUN
- ○ BR4
- ○ BR5
- ○ BR6
- ○ NPR
- ○ INTR
- ○ SACK
- ⊗ BBSY

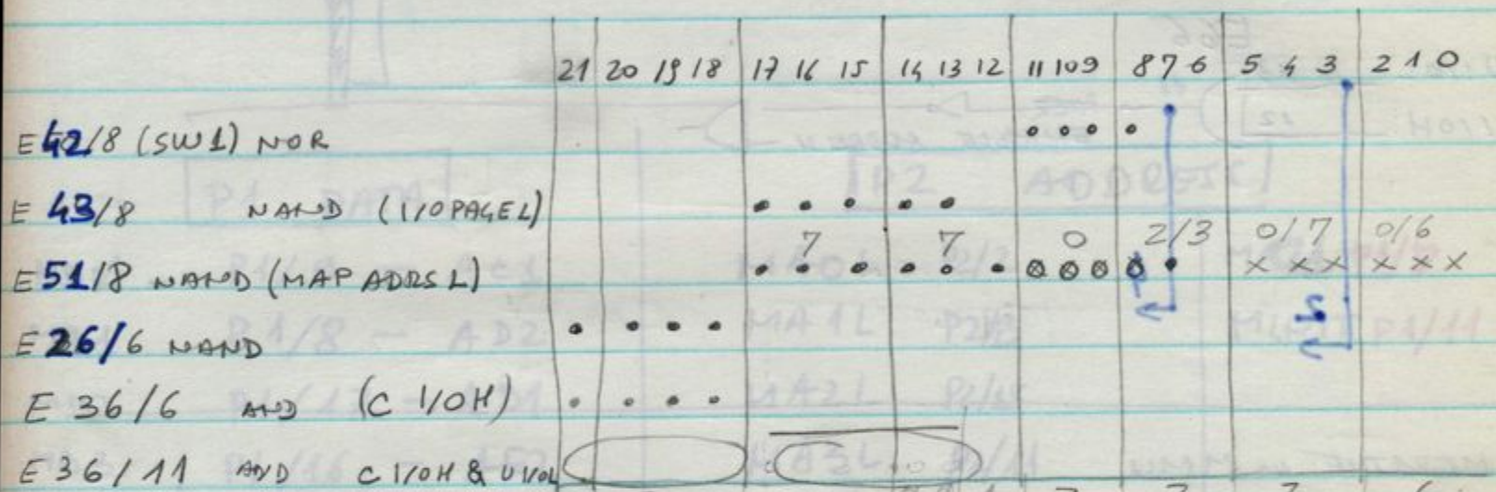


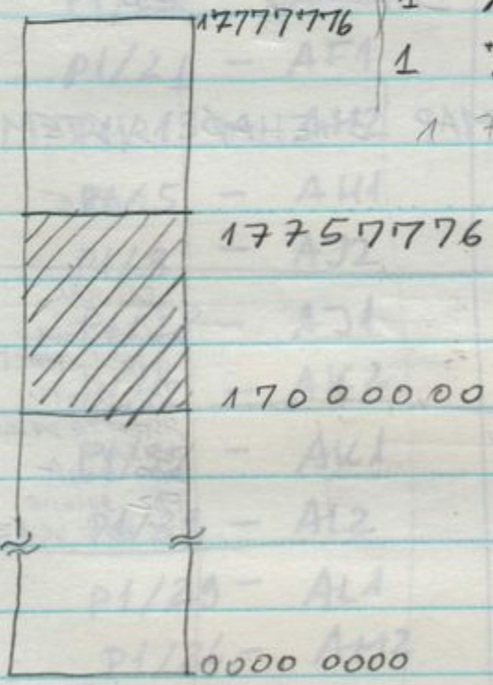




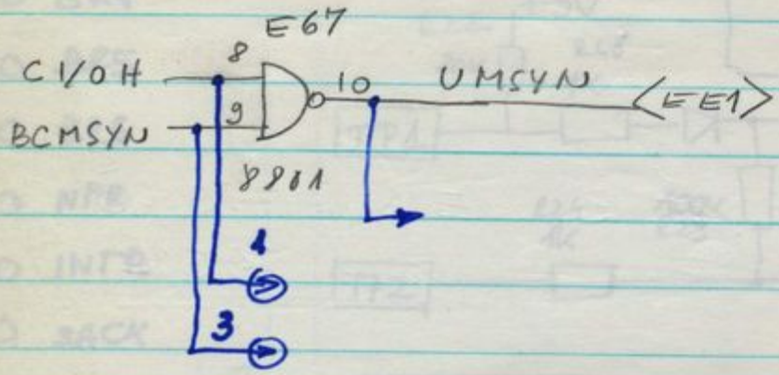
MMU

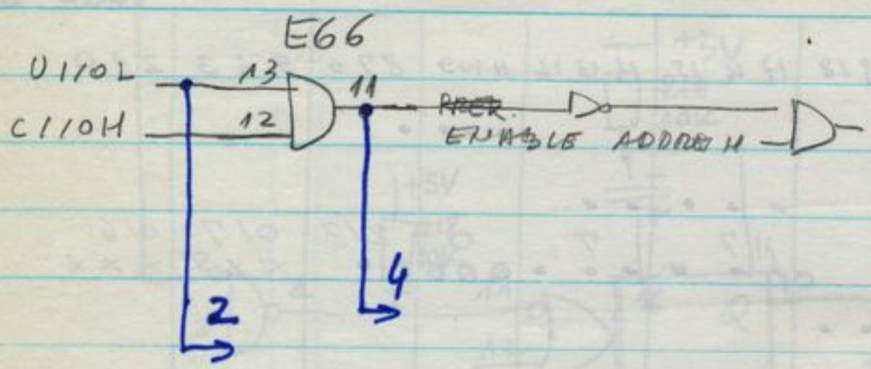
ADRESNI DEKODER





MMU

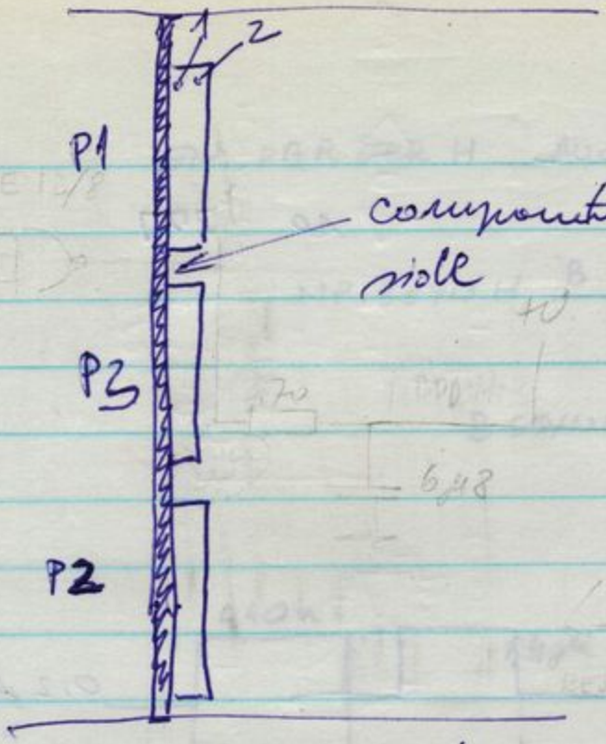




MERITVE na MMU

Moduli so označeni z: A in B. MPX in
A in B (predelan) RUP

1. meritev	34EPAR	34EMEM	34EMAP	34EHAP	SISTEM	GLED LED
MPX A RUP A	2X PASS	PC=2722 PARERR REG= 102741			PADE	1
MPX B RUP A	NE GRE	NE GRE			Se ne sveti gre v HALT PO BOOTU	(2)
SYSTIME RUP A	ne gre	ne gre	✓		SISTEM SE DVIGNE Z DISKA BOO DEB A FDZ DVIGNE SAMO BOOT DEB A Z FD	(3)
MPX A RUP B						(4)



P1 DATA

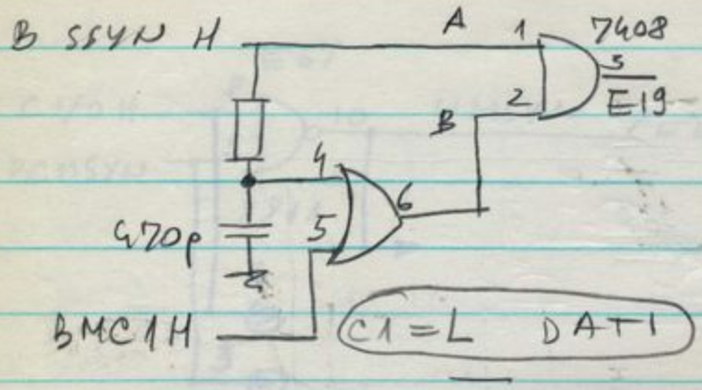
MDØ	P1/9	AC1
MD1	P1/8	AD2
MD2	P1/17	AD1
MD3	P1/16	AE2
MD4	P1/20	AE1
MD5	P1/25	AF2
MD6	P1/21	AF1
MD7	P1/13	AH2
MD8	P1/5	AH1
MD9	P1/3	AJ2
MD1Ø	P1/28	AJ1
MD11	P1/12	AK2
MD12	P1/37	AK1
MD13	P1/39	AL2
MD14	P1/29	AL1
MD15	P1/24	AM2
MCØL	P2/4	BU2
MBUS PBL	P1/36	MBUS PBL → AN2 connected for ECC

P1/2 = MSYN
P1/7 - UC1L
P1/32 = SSYN

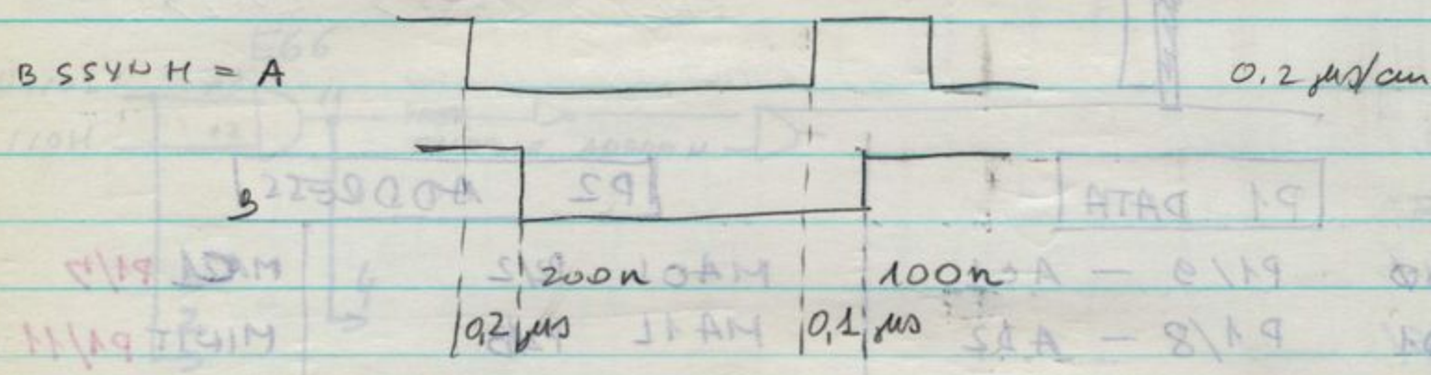
P2 ADDRESS

MA0L	P2/2	MA21 P1/7
MA1L	P2/3	MINIT P1/11
MA2L	P2/15	
MA3L	P2/11	
MA4L	P2/7	
MA5L	P2/3	
MA6L	P2/19	
MA7L	P2/~~21~~ 21	
MA8L	P2/~~29~~ 25	
MA9L	P2/~~31~~ 29	
MA10L	P2/~~27~~ 31	
MA11	P2/~~25~~ 27	
MA12	P2/~~17~~ 23	
MA13	P2/~~33~~ 17	
MA14	P2/~~35~~ 33	
MA15	P2/~~37~~ 35.	
MA16	P2/~~39~~ 37	
MA17	P2/~~40~~ 39	
MA18	P2/~~38~~ 40	
MA19	P2/~~36~~ 38	
MA20	P2/~~34~~ 36	
MA21	P2/34	

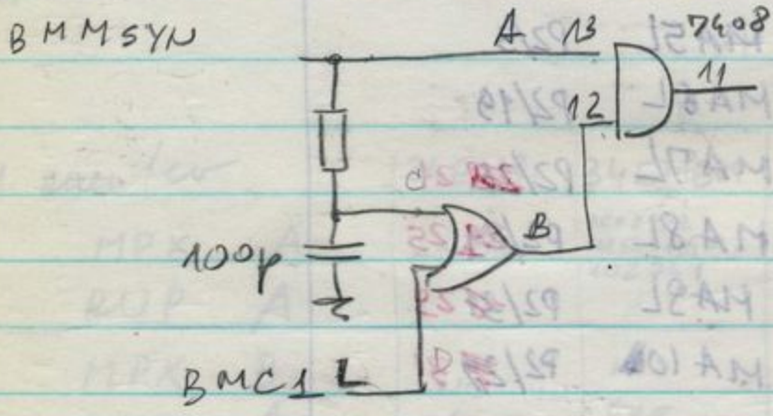
ORIGINAL SYSTEME MPC MODUL

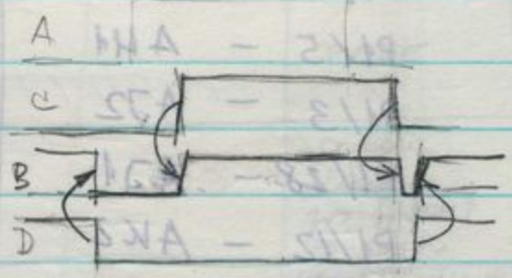


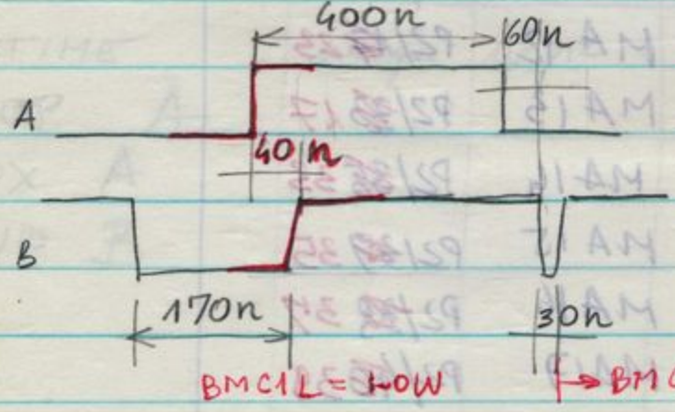
10 777

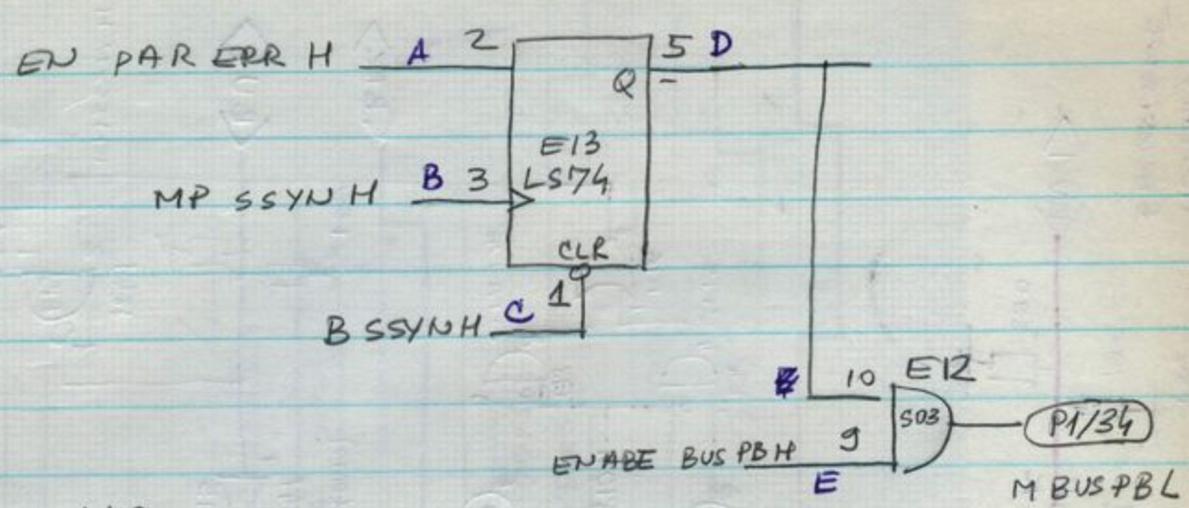


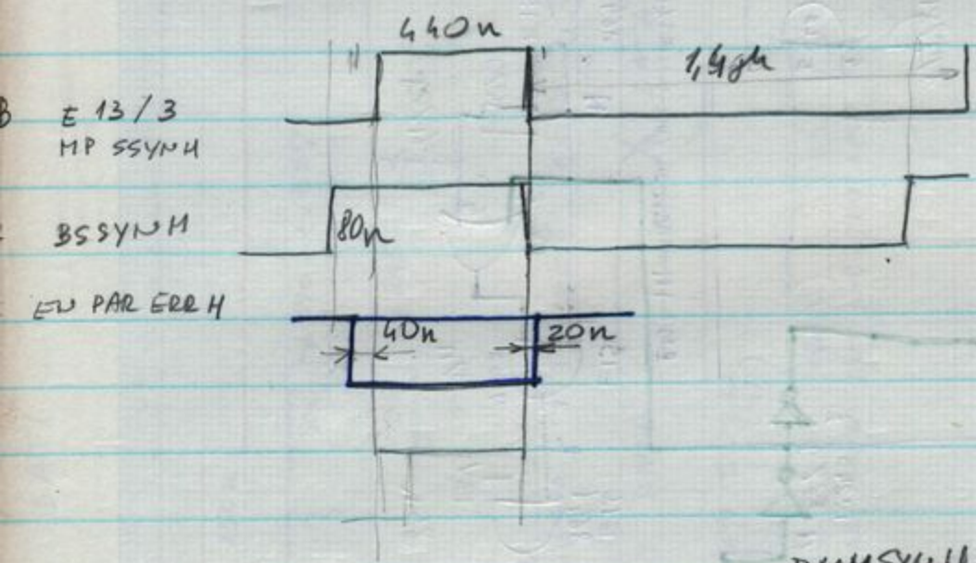
10 127700
12 100
14 10010 ← DATI
16 776 ← DATI

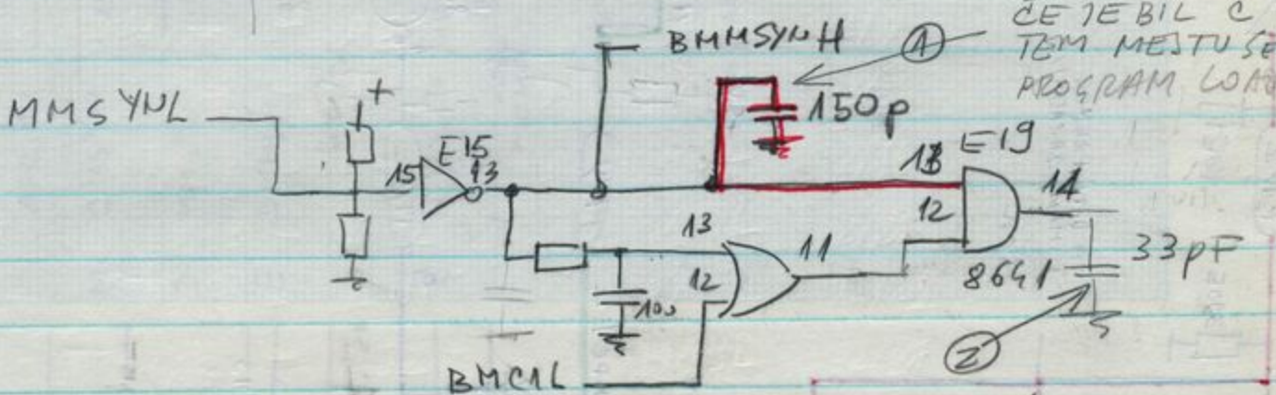




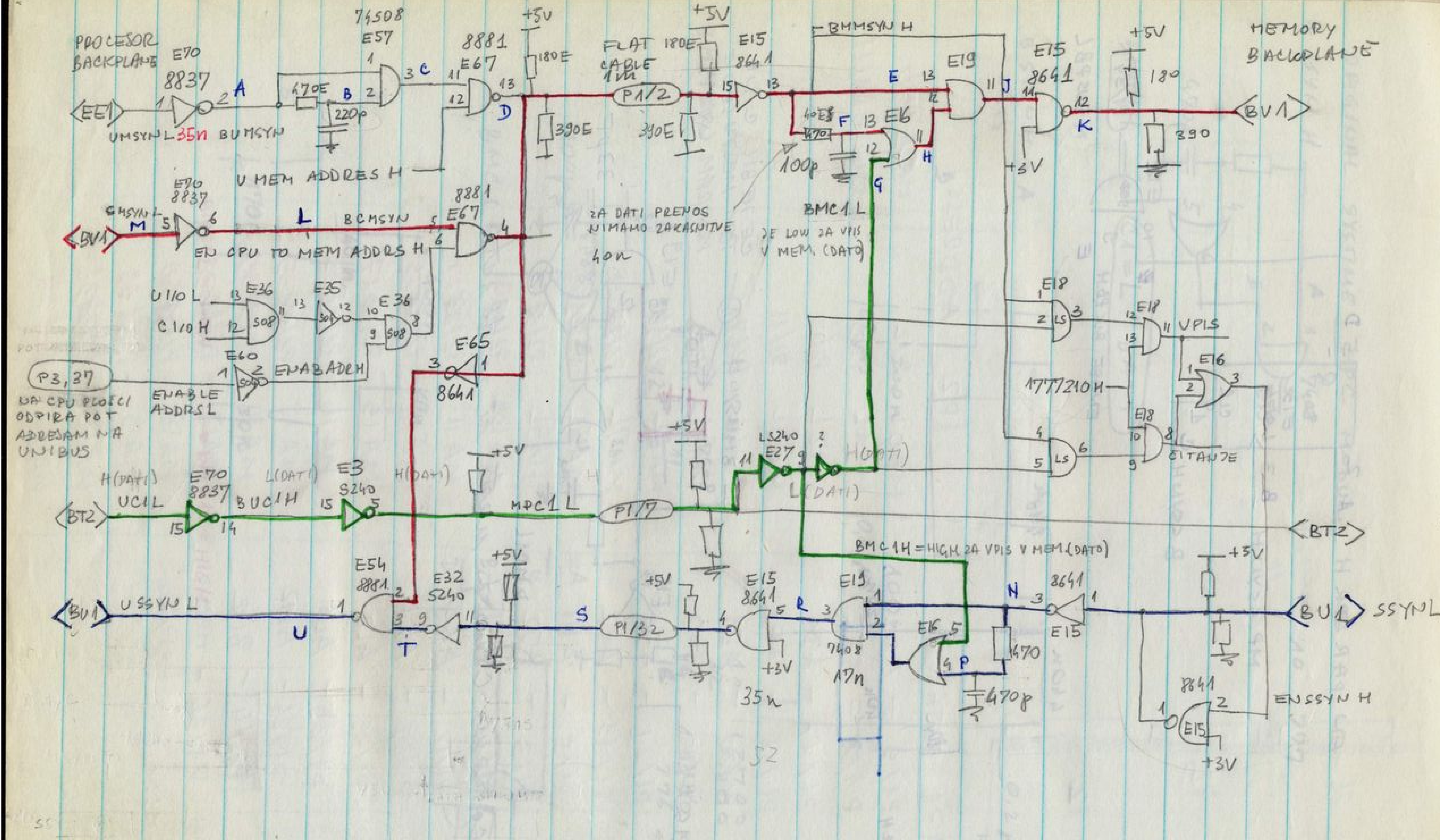




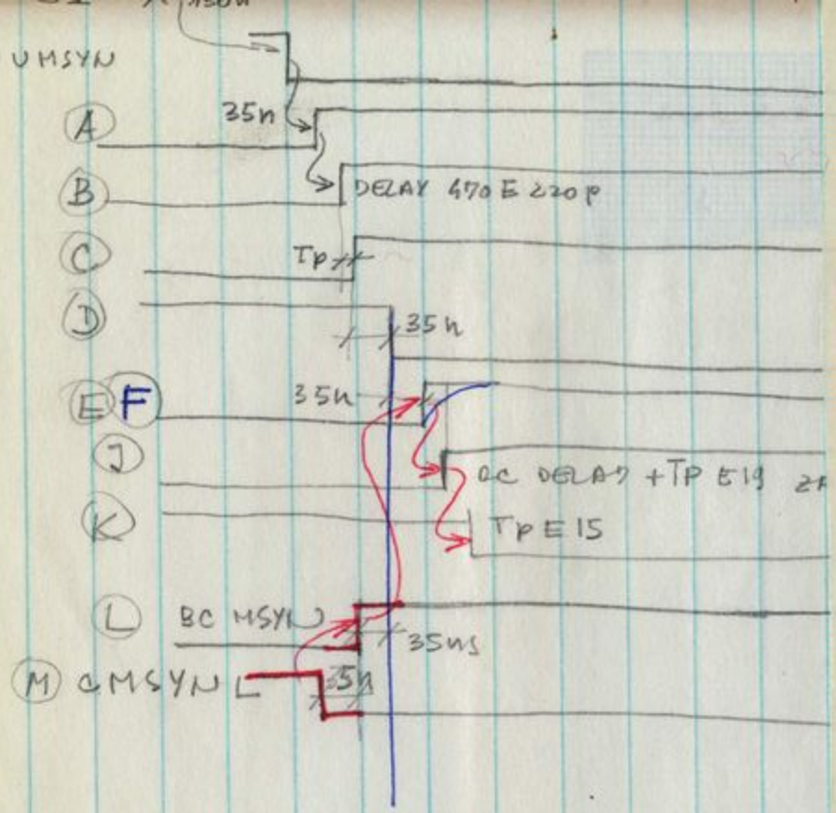


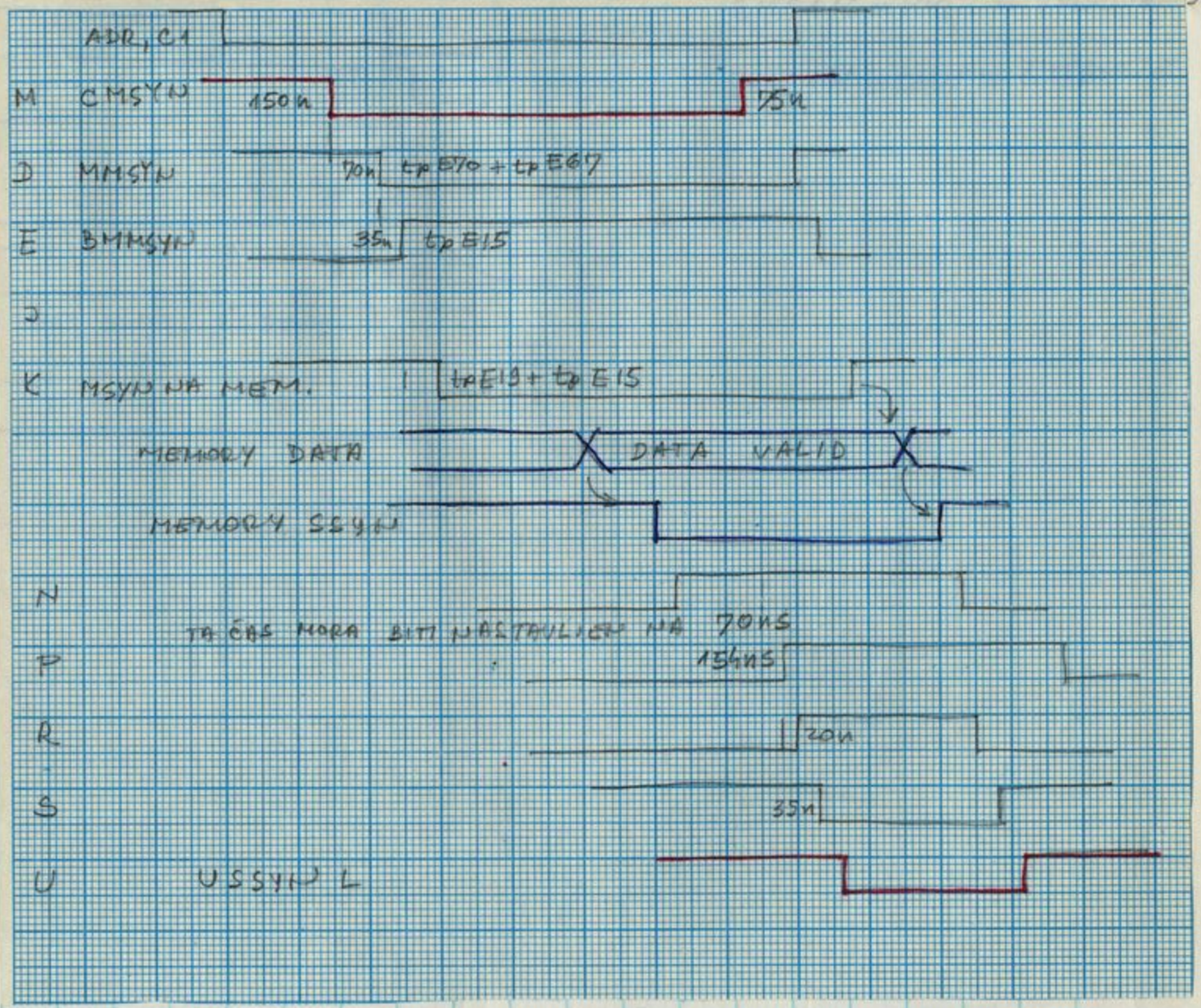


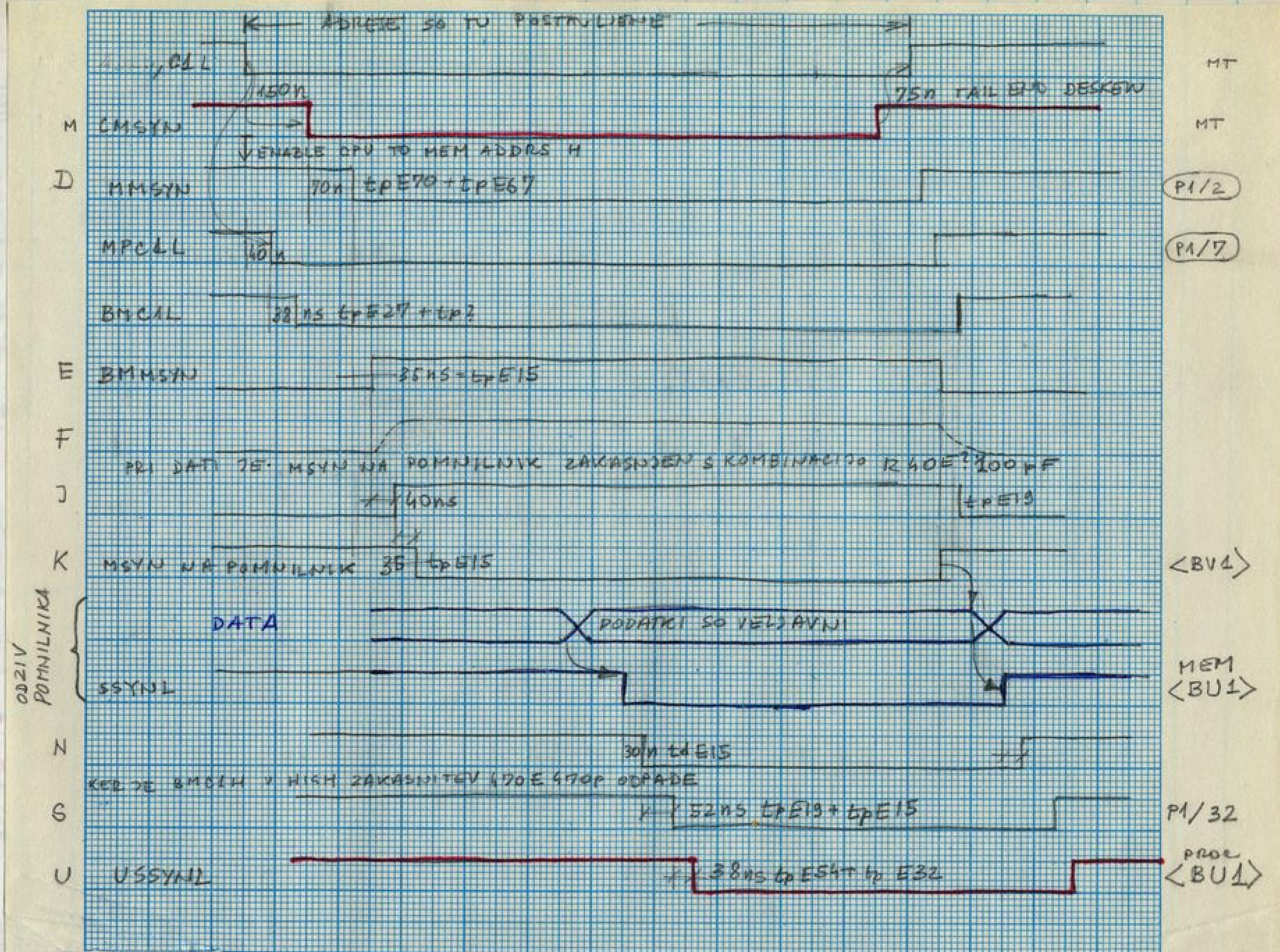
② če damo C na to mesto se zgodi isto -

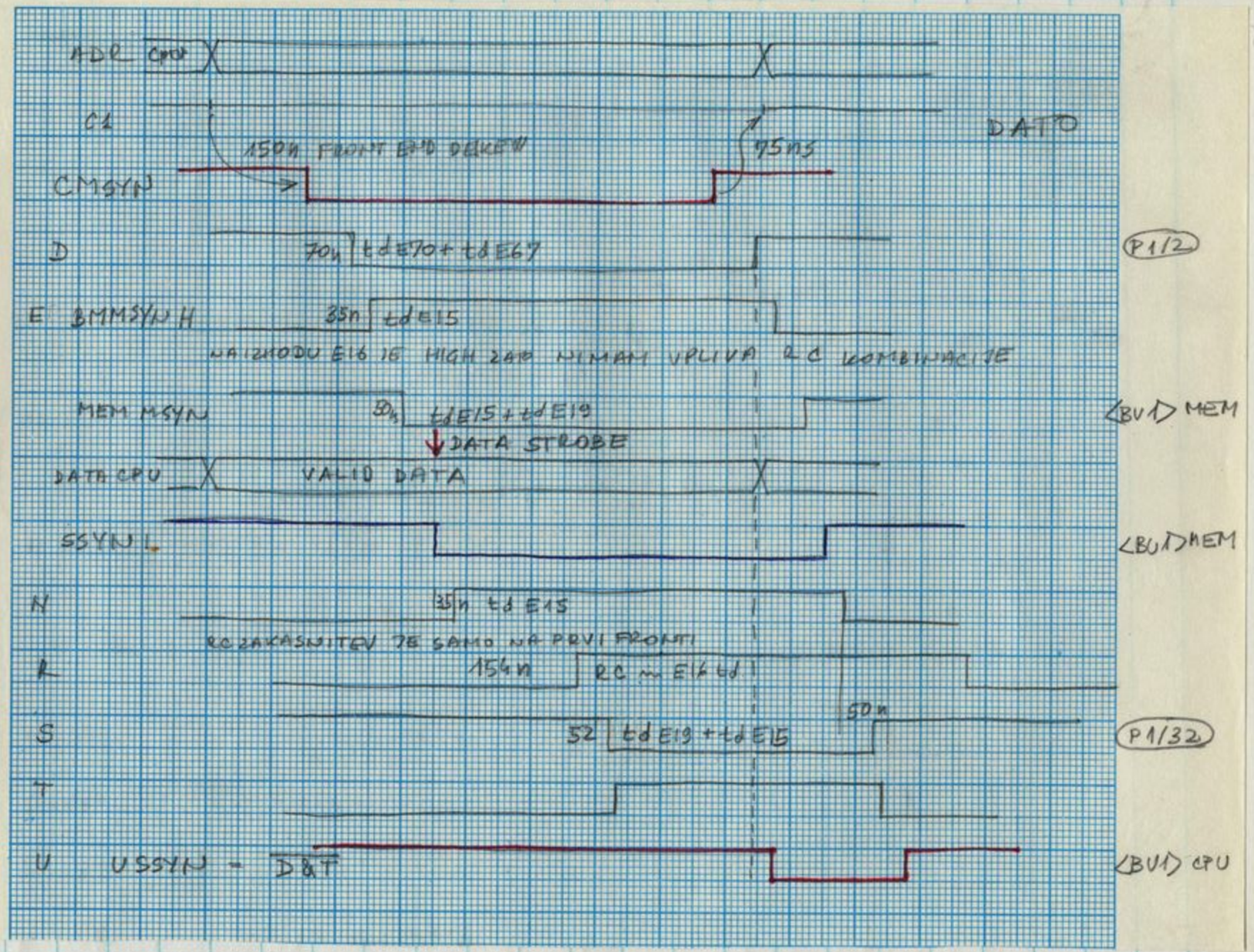

PROCESOR BACKPLANE
MEMORY BACKPLANE
E70 8837
UMSYN L 35n BUMSYN
74S08 E57
8881 E67
U MEM ADDRES H
FLAT CABLE 1m
P1/2
E15 8641
BMMSYN H
E19
E16
100p
BMC1 L
ZA DATI PRENOS NIMAMO ZAKASNITVE
40n
JE LOW ZA VPIS V MEM. (DATO)
EN CPU TO MEM ADDRS H
BCMSYN
U110 L
C110 H
E36
E35
E60
ENABADR H
ENABLE ADDRS L
P3,37
NA CPU PLOŠČI ODPIRA POT ADRESAM NA UNIBUS
E65 8641
E18
VPIS
177721OH
ČITANJE
UC1L
BUC1H
E3 S240
MPC1 L
P1/7
LS240 E27
H(DATI)
L(DATI)
BMC1H = HIGH ZA VPIS V MEM.(DATO)
E54 8881
E32 S240
USSYN L
P1/32
7408
17n
35n
470p
ENSSYN H
SSYN L
BU1
BT2
EE1

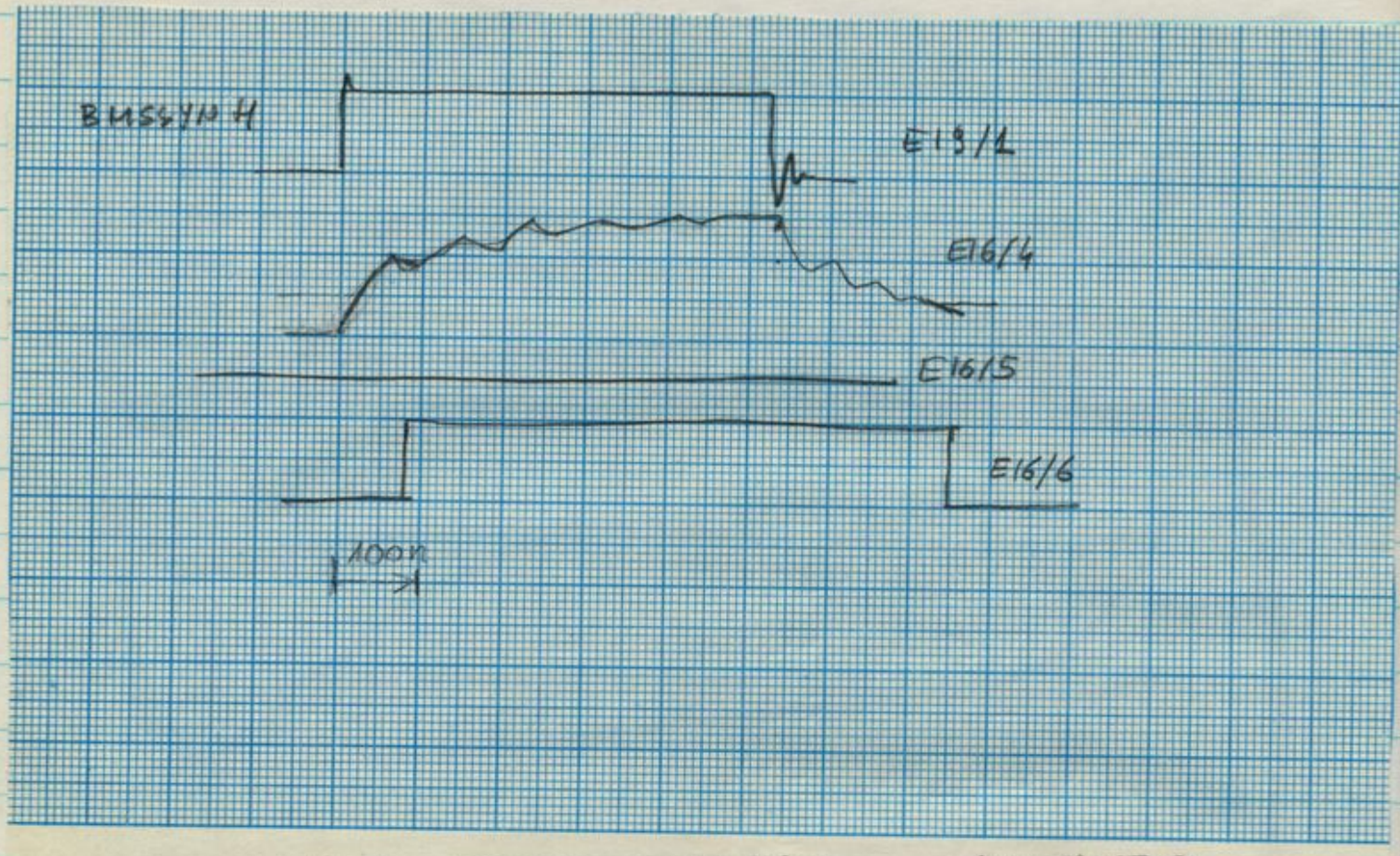
DATI


UMSYN
A
35n
B
DELAY 470 E 220 P
C
Tp
D
35n
E F
35n
J
QC DELAY + TP E19
K
TpE15
L
BC MSYN
35ns
M CMSYN L
35n

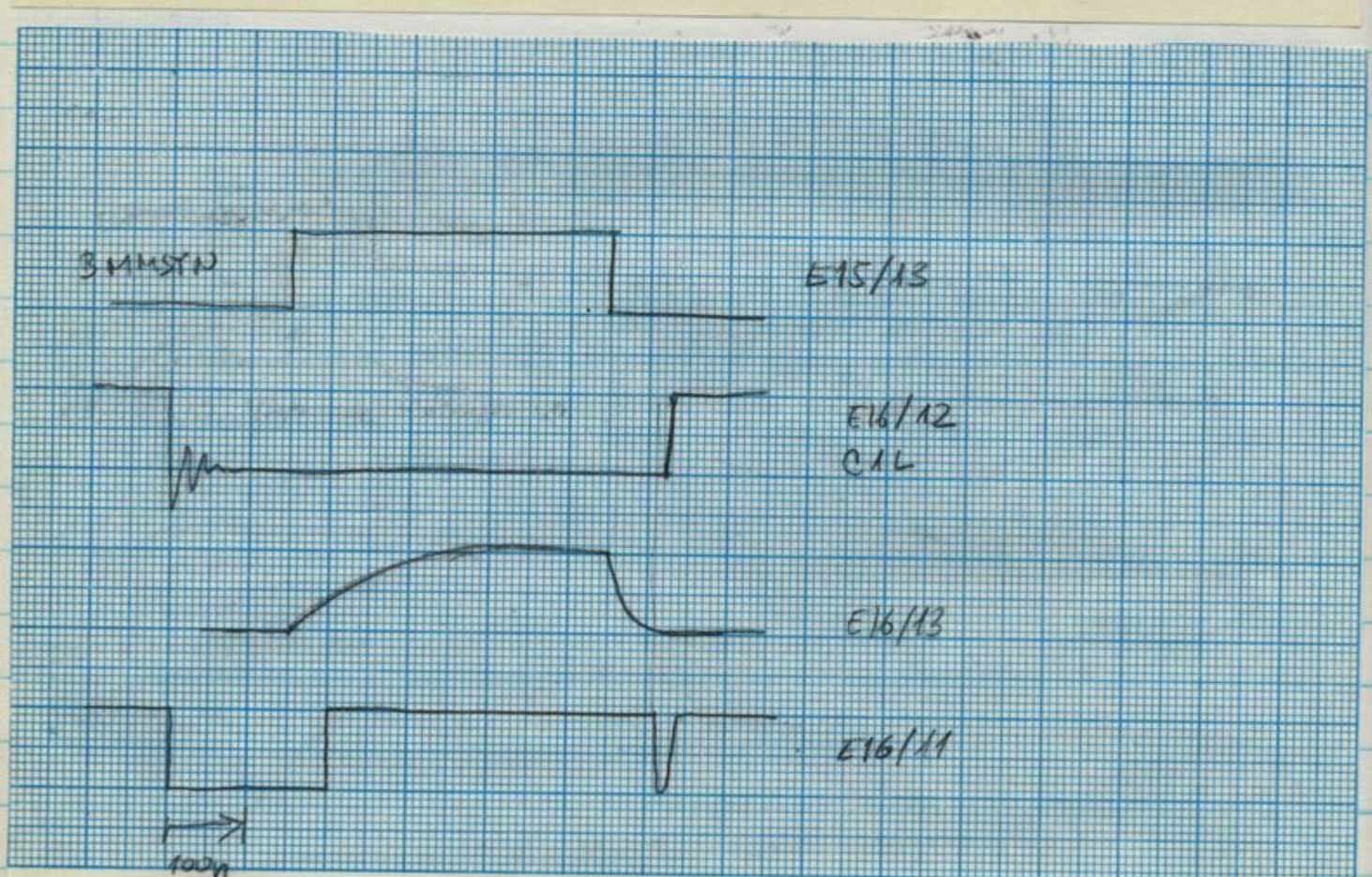

ADR, C1
M CMSYN
150n
75n
D MMSYN
70n
tpE10 + tpE67
E BMMSYN
35n
tpE15
J
K MSYN NA MEM.
tpE19 + tpE15
MEMORY DATA
DATA VALID
MEMORY SSYN
N
TA ČAS MORA BITI NASTAVLJEN NA 70NS
P
15ns
R
20n
S
35n
U
USSYN L


ADRESE SO TU POSTAVLJENE
CLL
150n
75n TAIL ... DESKEW
MT
M CMSYN
MT
ENABLE CPU TO MEM ADDRS H
D MMSYN
70n tpE70 + tpE67
P1/2
MPCLL
40n
P1/7
BMCLL
38 ns tpE27 + tp?
E BMMSYN
35ns + tpE15
F
PRI DATI JE MSYN NA POMNILNIK ZAKASNJEN S KOMBINACIJO R 40E? 100pF
J
40ns
tpE13
K MSYN NA POMNILNIK
35 tpE15
<BV1>
ODZIV POMNILNIKA
DATA
PODATKI SO VELJAVNI
SSYNL
MEM <BU1>
N
30n tdE15
KER JE BMCLH V HIGH ZAKASNITEV 470E 470p ODPADE
S
52ns tpE13 + tpE15
P1/32
U USSYNL
38ns tpE54 + tp E32
PROC <BU1>


ADR CPU
C1
DATO
150n FRONT ENO DELCEW
95n3
CMSYN
D
70n tdE70+tdE67
P1/2
E BMMSYN H
85n tdE15
NA IZHODU E16 JE HIGH ZATO NIMAMO VPLIVA R C KOMBINACIJE
MEM MSYN
80n tdE15+tdE19
<BU1> MEM
DATA STROBE
DATA CPU
VALID DATA
SSYN L
<BU1> MEM
N
35n tdE15
RC ZAKASNITEV JE SAMO NA PRVI FRONTI
R
154n
RC na E16 dd
50n
S
52 tdE19+tdE15
P1/32
T
U USSYN = DAT
<BU1> CPU



DATI ČASOVNI DIAGRAM IZMERJENO



DATO ČASOVNI DIAGRAM izmerjeno

MERITVE dne 24.4.85

Program 34EMEM.SAV se napolni v spomin ko ga startamo se sistem ustavi. **RUN LIGHT OFF**

Sistem se drugače normalno, SAM TECE NORMALNO

NASTAVITEV ANALIZATORJA FILENAME ~~34EMEM~~ ~~106~~

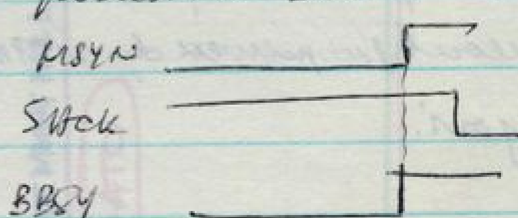
L 200 na S inside 200

202

204

17765210 } 24 μs timeout

↓ postavi se SACK



to je napačen start ker se pri drugem startu uspešno program pokvari.

Po LOADED L200 start

10pre reg 157776 051415 177774 005362

Program ~~ste~~ trip. 5362 timeout pri testiranju

napisano 17772100 trap to 4 62

4 2562 ←

6 go enable parity action

4

2562 4767

2564 2574

5362 32737 check if parity error detection is to be

5364 40 enabled, branch if not to be enabled

5366 176

176 ⇐ SW REGISTER

5370

5372

5374

5374

5376

5222 172100

17772100 } 24ms timeout

glej FILENAME 200

L 176 40000

L 200 S

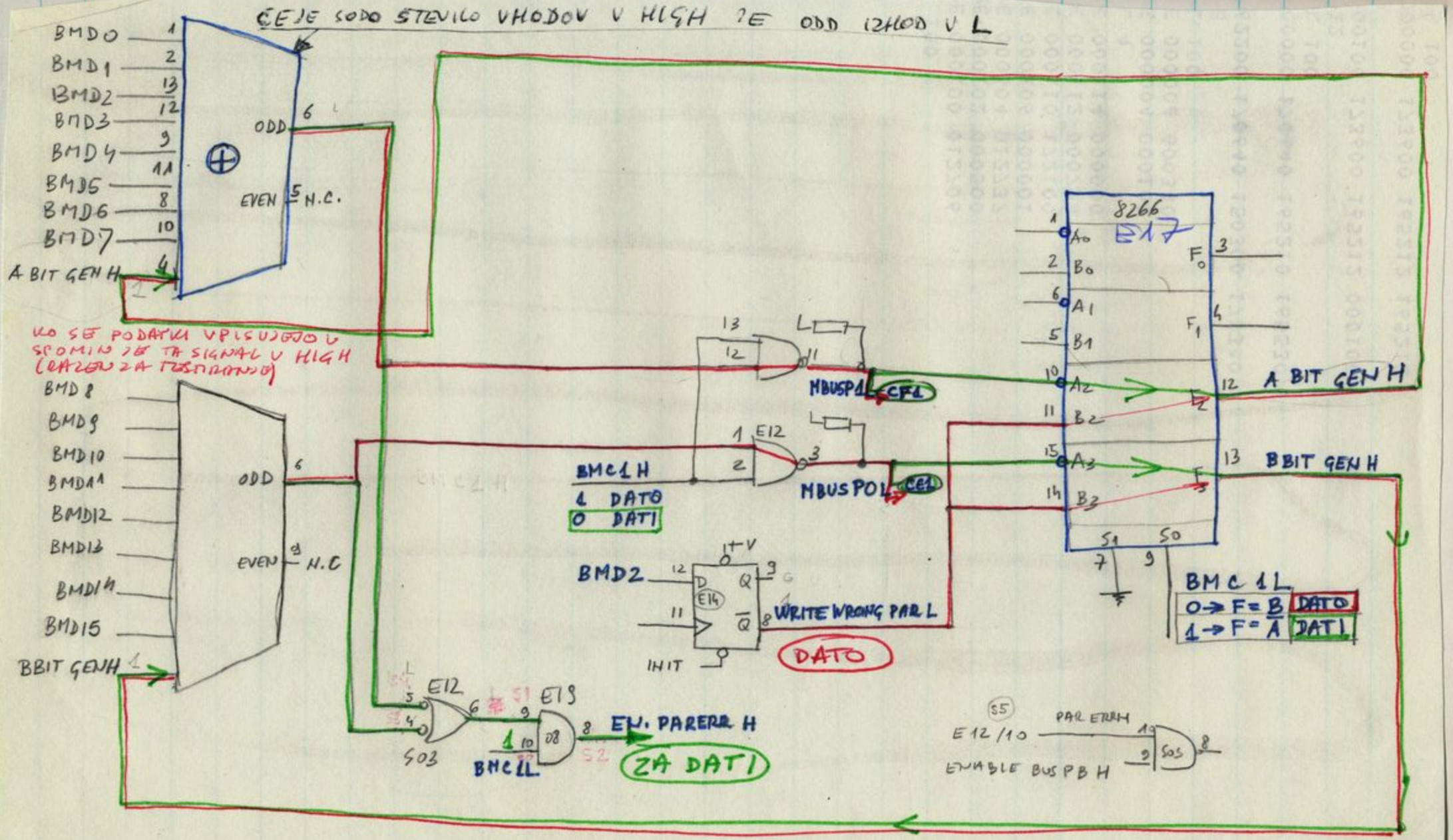
Program teče v zanki, kar se izvede timeout v primeru da se PARITY reg. ne javi.

LOOP ON SUBTEST

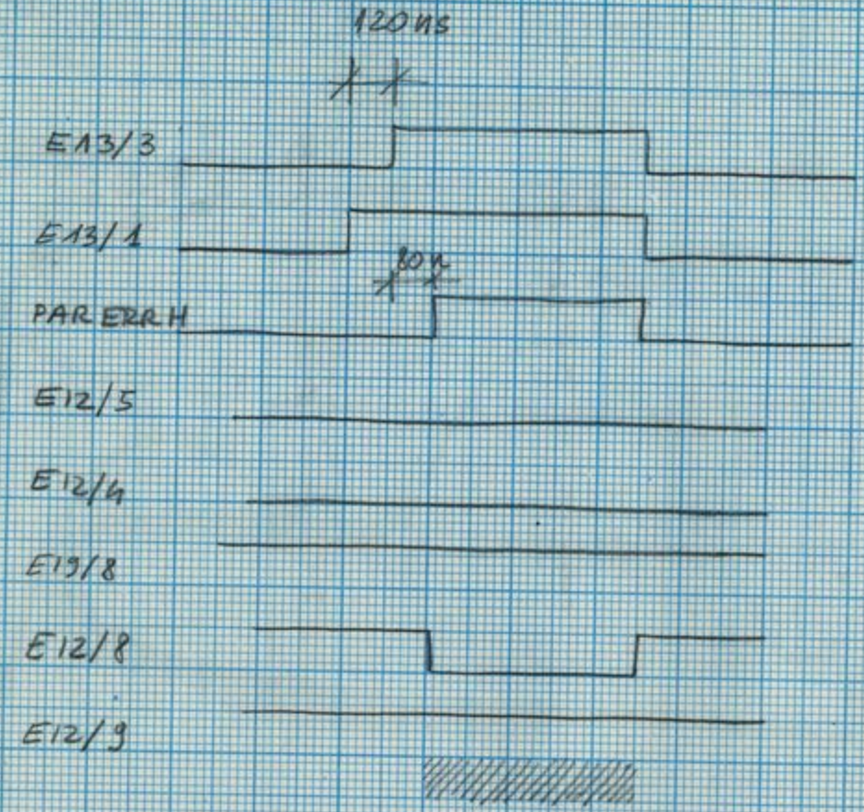
100	12706	LOAD SP
102	500	
104	12737	LOAD PARITY
106	1	REGISTER
110	172100	
112	774	
4	100	PC
6	340	PSW

TRAP 5000

če je trap v področje trap vektorjev potem R6 vsebuje zadnjo lokacijo predno je procesor padel v TRAP. NA TO lokacijo se nastavi log. analizator



Paritetna logika na MPC ploščí



SISTEM IZPIŠE
EXE: PARITY ERROR
KER JE NA DOLOČENIH
LOKACIJAH NAPAČNA
PARITETA (PODATKI SO
SICER PRAVILNI, ZATO SISTEM
DELUJE DE ODSTRANIMO
MBUS PB L SIGNAL.)

! POTREBNO JE PREVERITI
ČE MODUL V REDU
GENERIRA PARITETNE BITE!
OB VPISOVANJU V POMNILNIK
IN MERJENJU SE MORAJO IZRAČUNATI

PARITY ERR NA ADR. 7256 177777
PODATEK JE PRAVILEN, PARITETNI BITI SO NAPAČNI

MOV #570, R0 — 12700

570 ← odvrne na lokaciji kamor kaže R0

MOV #177777, R2 — 12702

177771 ← podatek, ki se zapiše na odr ↑

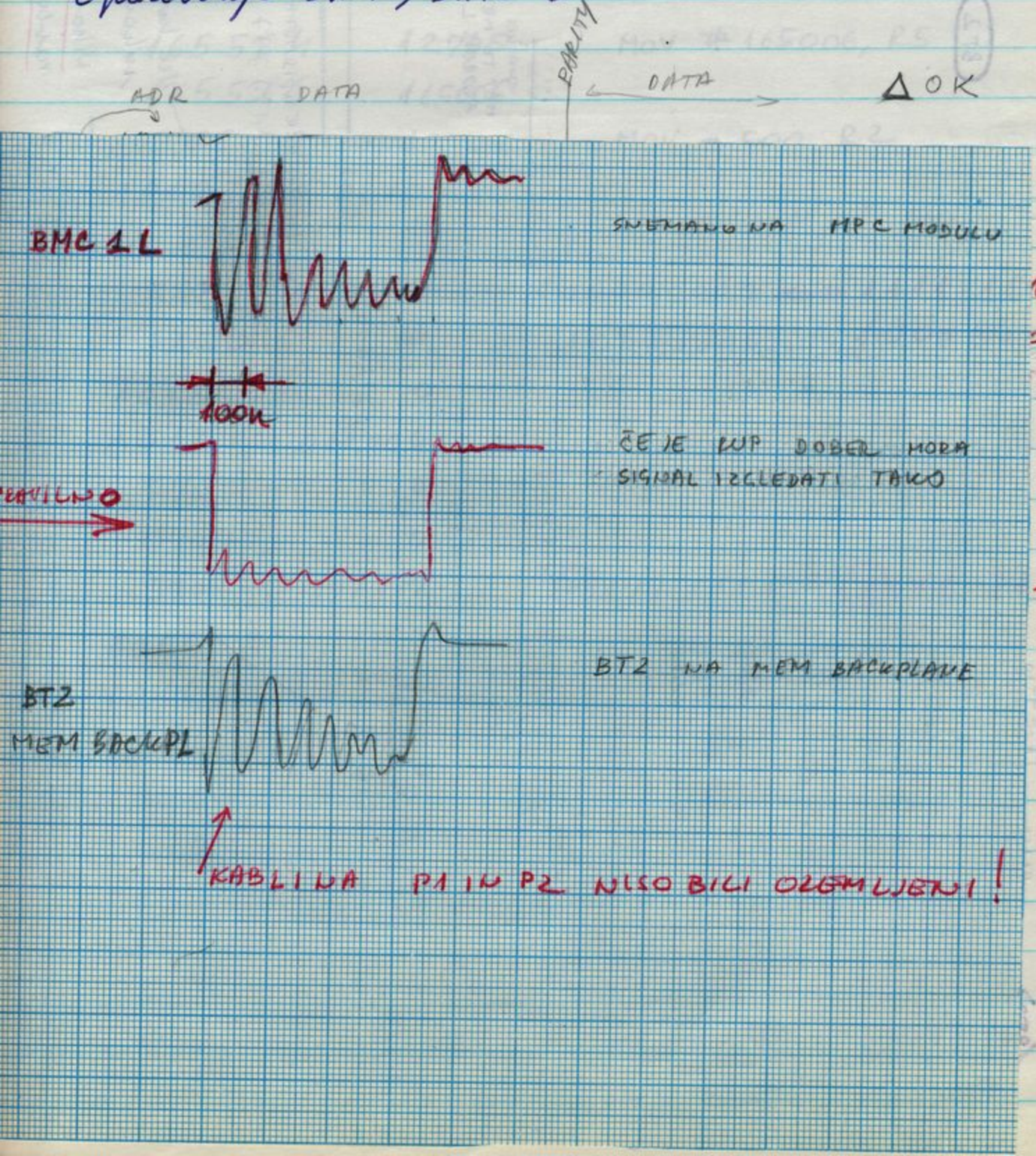
MOV R2, (R0) — 10210

MOV (R0), R4 — 11004

BR — 775

} vpiše na odrezek, in čita v register.

Opazovanje DATI, DATO cikla



MOV #570, R0 — 12700

570 ← odvero na kojeto se upisuje dato

MOV #177777, R2 — 12702

177777 ← dato, ki se upisuje na odr ↑

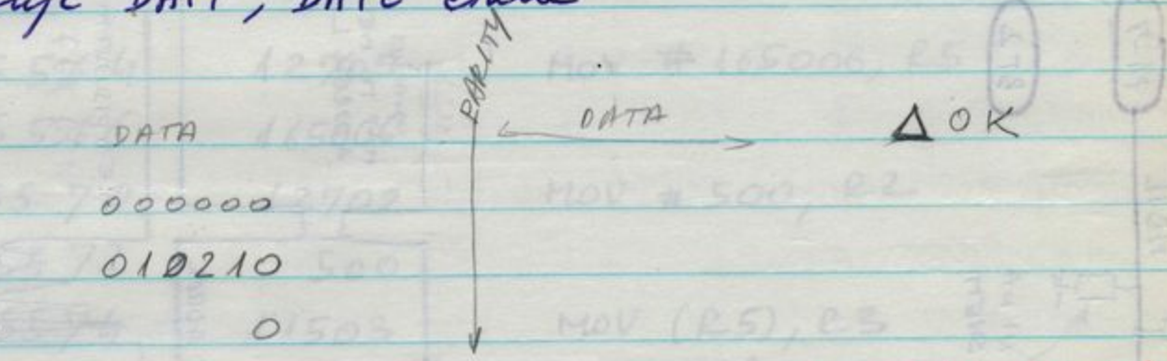
MOV R2, (R0) — 10210

MOV (R0), R4 — 11004

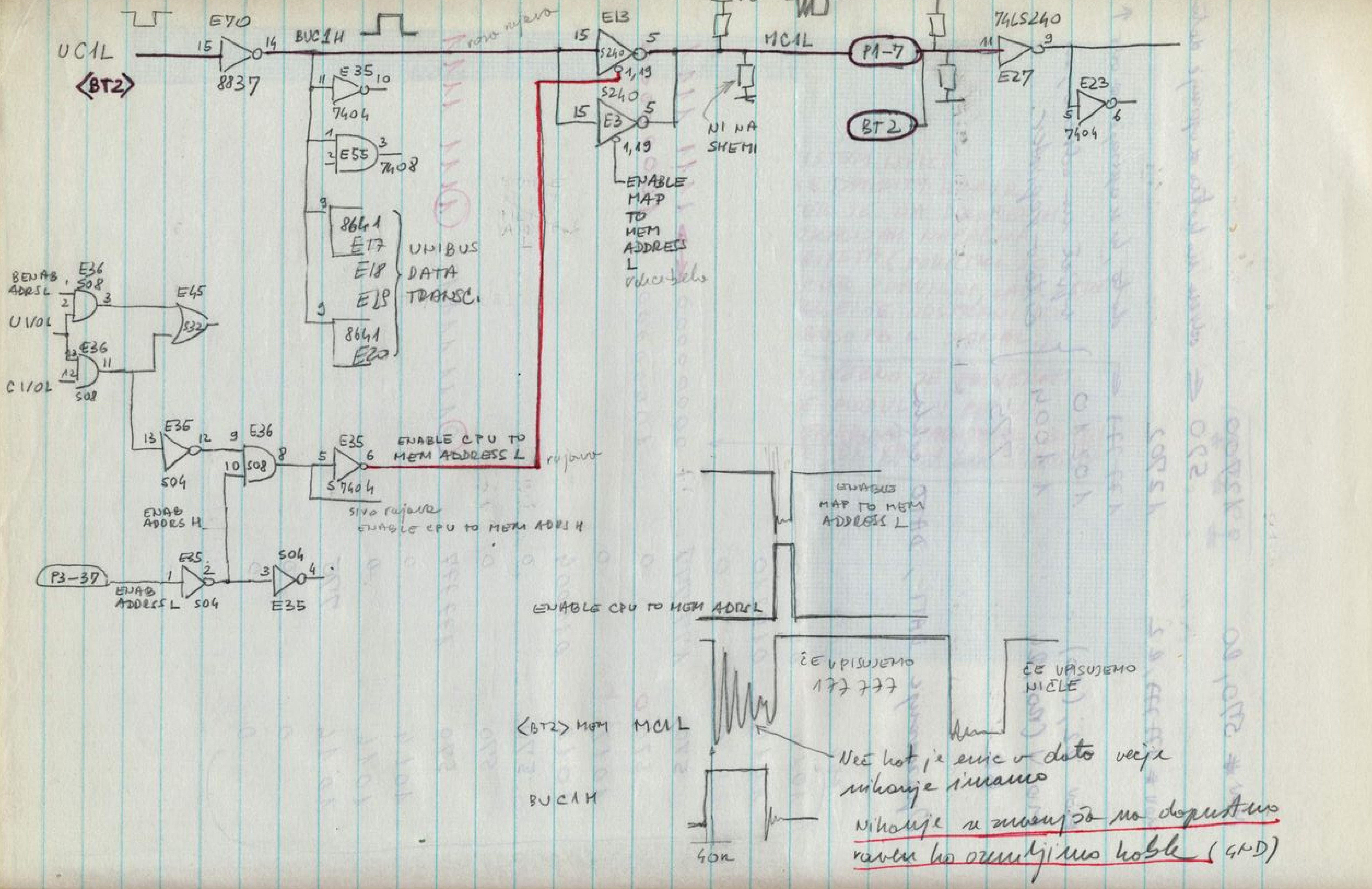
BR — 775

} upiše na odvero, in čita v register.

Opazovanje DATI, DATO cikla



ADR	DATA
1010	000000
1010	010210
0	0
570	177 777
570 0	0
1012	0
1012	011004
570	0
570	0
570	177777
1014	0
1014	0
1014	775
0	0
0	0


UC1L
<BT2>
E70
8837
BUC1H
E35 7404
E55 7408
8641 E17
E18
E19
8641 E20
UNIBUS DATA TRANSC.
E13 S240
E3 S240
ENABLE MAP TO MEM ADDRESS L
+U
NI NA SHEMI
MC1L
P1-7
BT2
74LS240
E27
E23 7404
BENAB ADRS L
U1/OL
C1/OL
E36 S08
E45 S32
E36 S04
E36 S08
E35 7404
ENABLE CPU TO MEM ADDRESS L
ENAB ADRS H
ENABLE CPU TO MEM ADRS H
P3-37
ENAB ADDRESS L
E35 S04
S04 E35
ENABLE MAP TO MEM ADDRESS L
ENABLE CPU TO MEM ADRS L
ČE VPISUJEMO 177777
ČE VPISUJEMO NIČLE
<BT2> MEM MC1L
BUC1H
40n
Več kot je enic v dato večje nihanje imamo
Nihanje v zmanjša na dopustno raven ko ozemljimo kable (GND)

BOOT TEST SE NE IZVEDE

no	R4	SP	PC
0	173420	165212	165550

← to lokacija se ne izvede več zadnja lokacija, ki se je izvedla je bila 165646, + lahko uporabimo za prožen je analizatorja

↑ NA TEJ LOKACIJI JE 544

Analizator prožimo z ... SACK ...

Procesor se ustavi ker je SACK stalno v LOW glej timing z analize trace-a vzgotovimo:

8837 E64 — E13 S240

Prekinjena je bila poverora BA0

TRACE

165564	12705	MOV #165006, R5
165566	165006	
165570	12702	MOV #500, R2
165572	500	
165574	11503	MOV (R5), R3
165006	177777	
165576	5012	CLR (R2)
500	0	
165600	112512	MOVB (R5)+, (R2)
165006	100377	
500	177777	
165602	5202	INC R2
165604	112512	MOVB (R5)+, (R2)
165007	177777	
501	177777	
165606	5302	DEC R2
165610	23512	COMP @(R5)+, (R2)
165006	177777	
500	177400	← tu bi moralo biti sama enica
165612	1015	BNE (Z=0)
165646	000000	HALT

Instaliranje SAM-a iz TP/AKO

```
>INS $BRU
>BRU /MOU/NOINI /VER
FROM:    MT:
TO:      DR:
BRU - STARTING TAPE 1 ON MT0:

BRU - END OF TAPE 1 ON MT0:

BRU - COMPLETED
```

> SET VIC /[2,300]
> @ SAM

Vzlomo pa s
CTRL C
MCR> ABO AT.

EKVIVALENTNA INTEGRIRANA VEZJA

DEC 8640	DS 8640	NATIONAL
DEC 8641	DS 8641	〃
DEC 8837	DS 8837	〃
DEC 8881	96101	FAIRCHILD

DELTA 800 CPU

Upisovanje konstante v SPM

1) HEX konzola priključena na J1 konektor
2) PAJ SER vezica

		IZVOR
K1-5 REG CLK H	E7/16	E112/11
K2-4 SP WRITE L	E7/15	E219/6
K1-1 OUTPUT STORE L	E7/13	E54/8
K1-10 ENAB GR L	E7/12	E58/8

BYTE SWAB SIVO-ROZA

1 0 0 0 1 1
1 1 0 0 0 1
1 1 0 0 0 0
1 1 0 0 0 0

F E 0 0 1 B (HALT)

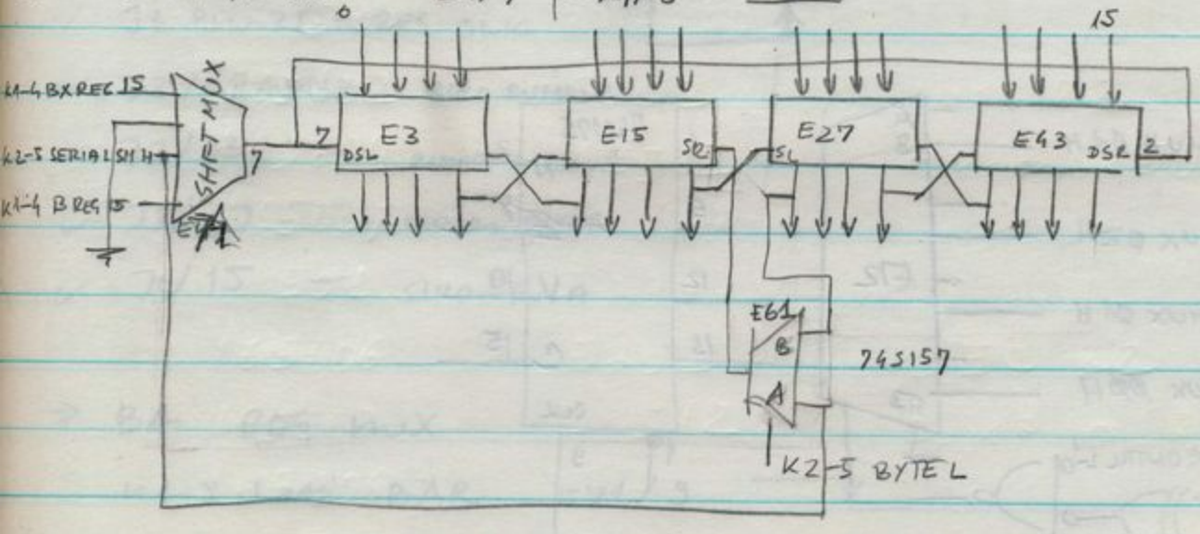
invertirano preko buff 0 1 F F E 0 (HALT)

* manjka upor R251, ki je na poziciji CX levo (označbo CX je odrezal) *

B REG, BX REG		IZVOR
K2-8 BMODE ØØL	E3/9	E219/8
K2-8 BMODE Ø1L	E3/10	E219/4
K1-5 PROC CLK L	E3/11	E112/6
K1-10 SHIFT IN BH	E43/2, E3/7	E74/7
K2-8 BX MODE ØØL	E4/9	E196/4
K2-8 BX MODE Ø1L	E4/10	E196/3
K1-10 SHIFT IN BX H	E4/7	E74/9

0 1 0 1 BELO RUMENA
0 0 1 1 RUMENO RJAVA
STRB
MODRO RDEČA

HOLD
POMIK DESNO
POMIK LEVO
PARALEL LOAD



* E71 pin 15 ni na gnd *

K1-5 REG CLK L ⊓⊔ IZVOR

K2-8 AUX CONTROL = H
L

SSMUX ØØ–Ø3 NA MZVC

CLK E11

K1-10 LOAD PSW LOW L ⊓⊔

SSMUX Ø5–Ø7 NA PSW Ø7–Ø5

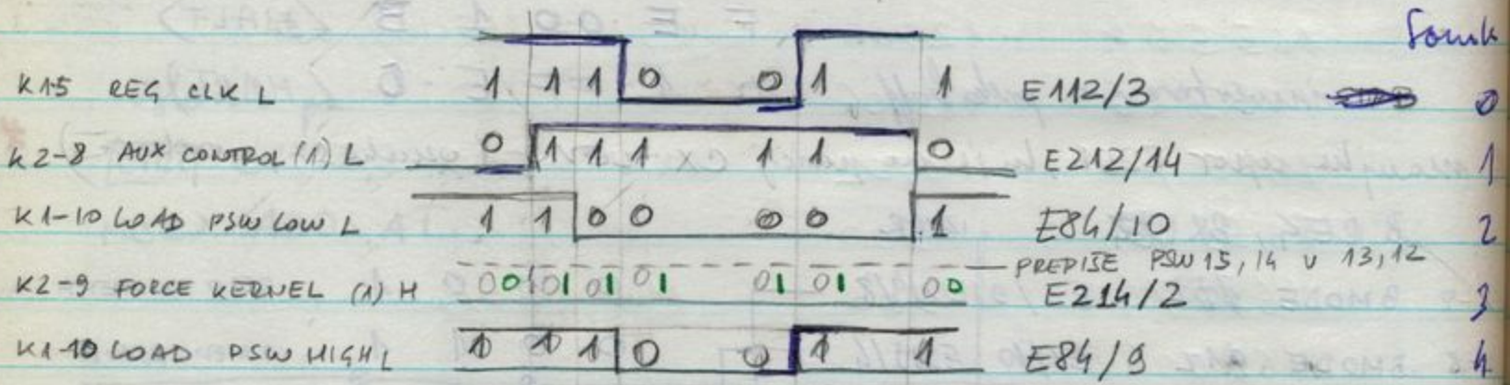
SSMUX Ø4 VPIS TBIT

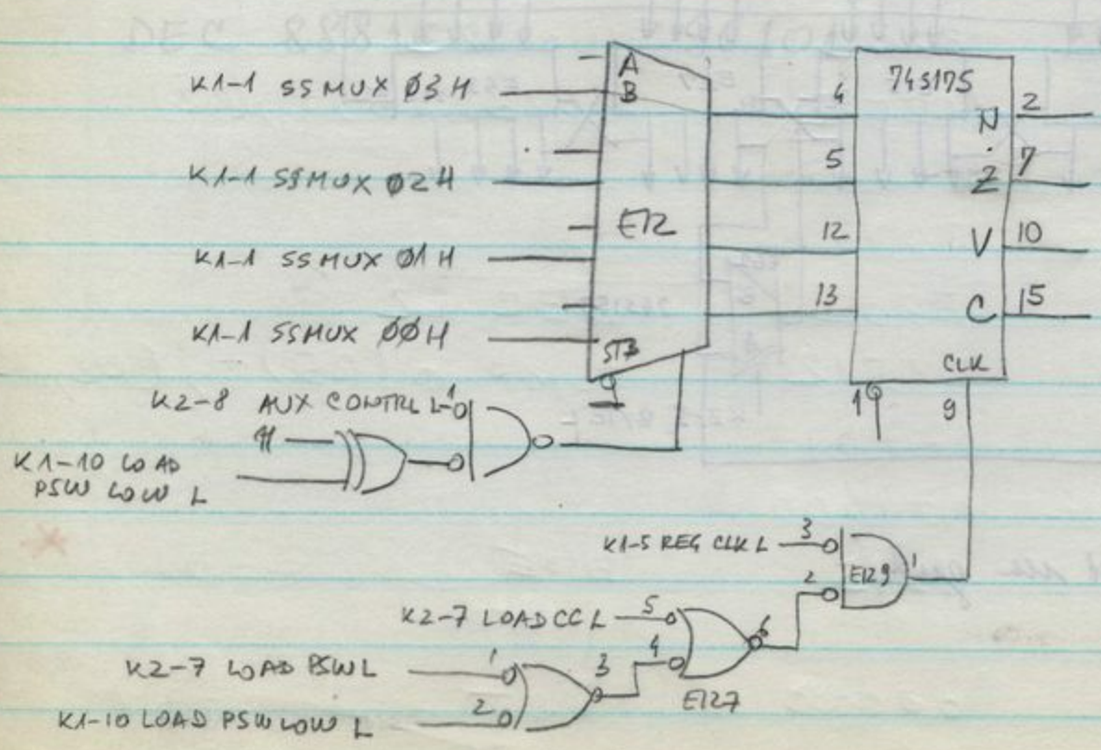
K2-9 FORCE KERNEL (1) H = L

K1-3 LOAD PSW H ⊓⊔

MUX SSMUX 13, 12

VPIS 13, 12 14, 15





15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

→ SPM ADRESE

✓ SPA0 E163/9 RUMENO-ZEL

✓ SPA1 E163/7 BELO-MODRA

✓ SPA2 E164/9 BELO-ZELENA

✓ SPA3 E164/7 RDEČ-ZEL

✓ K2-8 DISABLE UPPER BYTE H E195/12 = LOW

K1-5 REG CLK H E112/11

K2-4 SP WRITE L E213/6

K1-10 GEN REG L E86/9

Preveri TAP 30 (mora biti v HIGH)

Preveri K1-10 ENABLE GR L E58/8 (= LOW)

CLOCK IZ PLOŠČE VZAMEMO E71 (74S157) in E112 74S37

→ VPIS V VBA REG

LOAD VBA PIN E71/7

→ VPIS V PAR

J3 PIN 23 = $\overline{WE}$

J3 PIN 25 = REG CLK

✓ J3/19 = MSB — BELO RUMENA

✓ J3/13 — RUMENO MODRA

✓ J3/17 — MODRO RDEČA

✓ J3/15 — SIVO ROZA

→ BA REG MUX

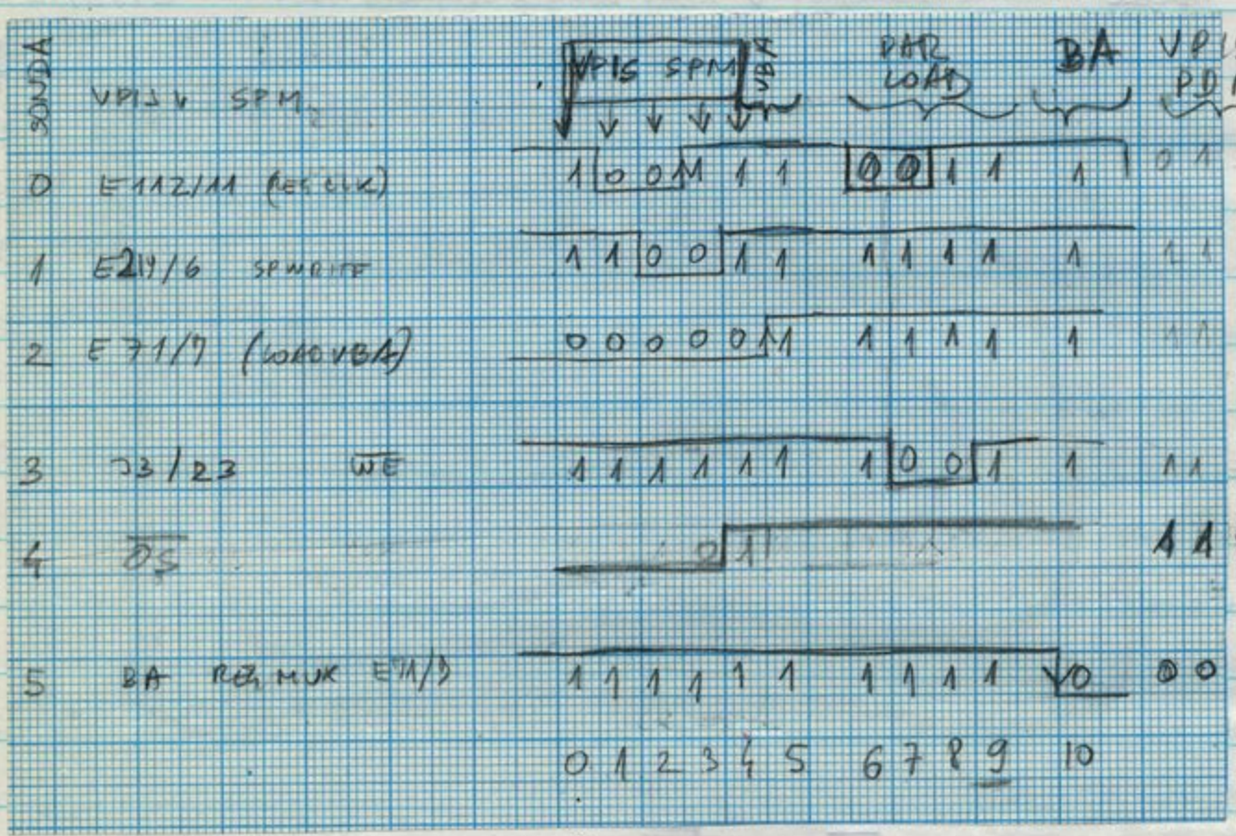
K1-5 LOAD BAR E71/9

E195/9 → LOW

E103/ → ODSTRANITI

85568

OD	$\overline{WE}$	CLK	$\overline{OS}$		IZHOD
0	X	X	0	OUTPUT STORE	data iz zadnje adresir. lokac.
X	0	⌐	X	WRITE	odvisno od OD in $\overline{OS}$
0	X	X	1	READ DATA	podatke shranjene na adresi
1	X	X	0	OUTPUT STORE	} high impedance
1	X	X	1	OUTPUT DISABLE	



6 LOAD PDR HIGH L

7 LOAD PDR LOW L

			TIP
✓ 23-110 A1	DOP EIS ROM	E206	32x8
✓ 23-349 A9	DOP	E205	512x4
✓ 23-172 A2	DOP	E204	256x4
✓ 23-A22 A2	BRANCH	E203	256x4
✓ 23-173 A2	SOP	E207	256x4
✓ 23-174 A2	SOP	E208	256x4

✓ 23-162 A2	E194	256x4
✓ 23-175 A2	E193	256x4
✓ 23-176 A2	E192	256x4

INSTRUKCIJSKI DEKODER

	ROM 349 A9 (512x4)	172 A2 (256x4)		PIN
A0	IR 12	IR 12	1	5
A1	DMØH	DMØH	0	6
A2	SMØH	SMØH	0	7
A3	IR DECODE 1H	IR DECODE (1) H	X	4
A4	BUT DEST L	BUT DEST L	0	3
A5	IR 14	IR 14	0	2
A6	IR 15	IR 15	0	1
A7	IR 13	IR 13	0	15
A8	FP ATTACHED		1	14

D1	IR CODE 00	X	X	K2-6 SEC H	12
D2	MPC ØSL	0	1	K2-6 MOV L	11
D3	MPC Ø6L	1	0	MPC Ø3L	10
D4	K2-7 MPC Ø7L	1	0	MPC Ø4L	9

C 2

D 3

TBC MODUL JE NAVIJEČ
V ROKE MIHOVCU (ali

TIP INSTR. PRIMER HEX	IR CODE OOL	MPC 09 MPC 08	MPC 07 MPC 06 MPC 05	MPC 04 MPC 03	MPC 02 MPC 01 MPC 00	
MOV (SMØ*DMØ)	1	1 1	0 1 1	1 Ø	1 1 0	← PROM BREZ OZNAKE V 1ST1 VREŠK
1 0 0 1 (MOV)	1	1 1	0 1 1	1 0	1 1 0	← 172A2 ORIGINAL
DOP ($\overline{\text{MOV+SUB}}$) MOD (SMØ*DMØ)	1	1 1	1 0 0	1 1	1 1 0	
4 0 0 1 (BIS)	1	1 1	1 0 0	1 1	1 1 0	← 172A2
SUB ($\overline{\text{SMØ*DMØ}}$)	1	1 1	1 0 1	1 0	1 1 0	
E 2 0 A (SUB)	1	1 1	1 0 1	1 0	1 1 0	← 172A2
OOP (SMØ*DMØ)	1	1 1	0 1 0	1 1	1 1 0	
E 0 0 1 (SUB)	1	1 1	0 1 0	1 1	1 1 0	← 172A2
LLEGAL	1	1 1	1 1 1	1 1	0 0 0	
F F F E	1	1 1	1 1 1	1 1	1 1 0	← 172A2
OOP NOV MOD (MØ*DMØ) (CMP, BIT)	1	1 1	0 1 0	Ø Ø	1 1 0	
3 0 0 1 (CMP)	1	1 1	0 1 0	0 0	1 1 0	← 172A2

v kujiji je tu Ø

SHEMANJE MIKROPROGRAMA POWER UP SEKVENCA

E220/1 = CLR Q = 1 ✓

9	8	7	6	5	4	3	2	1	0
H	H								
H	H								

RUP MODUL

TEST 1 GRE O.K.

TEST 2: 165054   12702

165056   165000

ne gre na naslednjo instr. → 165002   165000

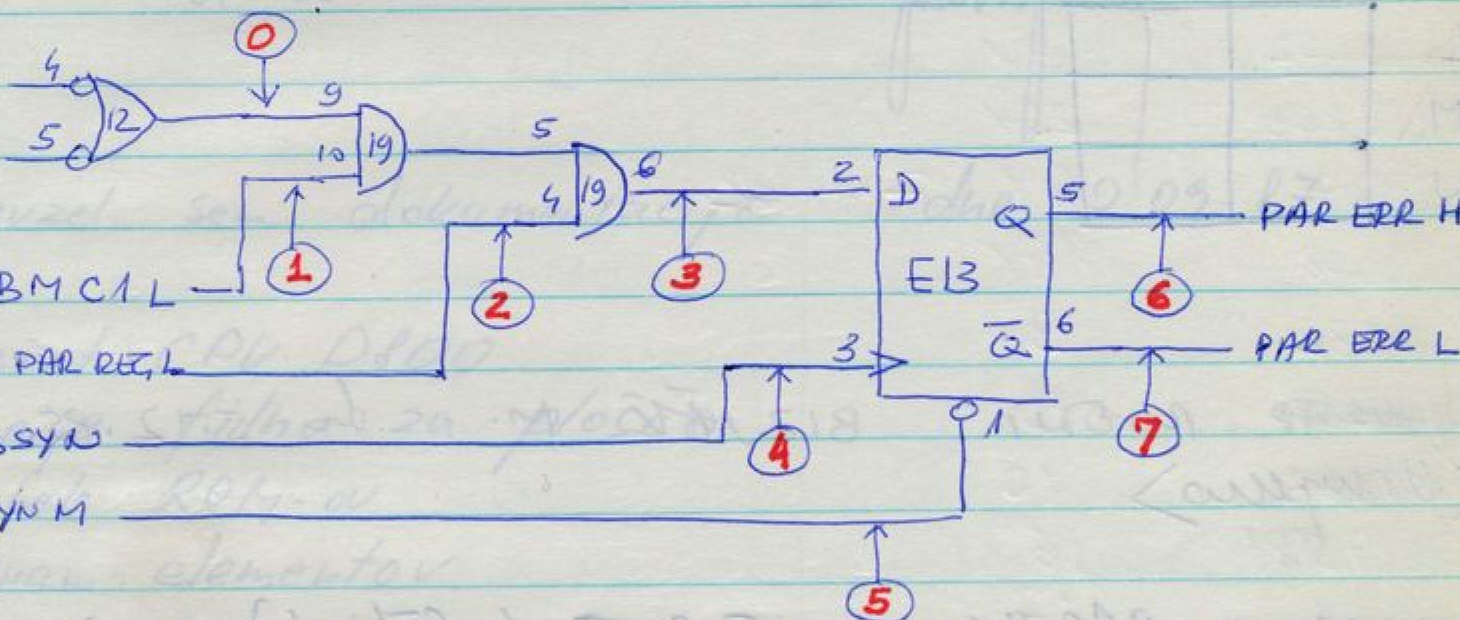
107776

56736

165004

PARITY ERROR LOGIKA

E9/6 ODD
E8/6 ODD
BMC1L
READ PAR REG L
M P SSYN
BBSYN M



PREJEL MPC MODUL 13.8.1985
GRABEC IGOR
I-D MB
(062) 25127

~~Podaljševalni kabel za unibus~~
~~IVANUŠ 25.9.85~~

TBC MODUL JE IVANUŠ DOSTAVIL V KRANJ V ROKE MIHOVCU (šef proizvodnje) ZAČETEK NOVEMBRA

DELTA 800 CPU KONEKTOR J2-1 J2-2

1	A	ČRNA 2	RUMENA 2
2	B		ČRNA 2
3	C		MODRA 1
4	D		ZELENA 1
5	E		RDEČA 1
6	F		MODRA 1
7	H		ORANŽ. 1
8	J		VIJOLIČ 1
9	K		
10	L		
11	M		
12	N		

POSODIL WRAP RODNI BIZJAK M. dne 5.12.85
<Urnijeno>

ANALIZATOR DAS 9100 + ~~5 SOND~~ 4 (štiri) SONDE
<Urnijeno>

PROM F3040 tip 256 X 4
8 vrzeljev po 25 elementov
Prevzel 6.12.1986 MENART JERNEJ

24. 12. 87

DOCUMENT DIRECTORY	1 kos
TEST LIBZARY PROGRAMMING	1 -"-
SYSTEM OPERATION	1 -"-
TEST SET PREPARATION	1 -"-
TEST FIXTURE KIT	1 -"-
TEST PROGRAMMING	1 -"-
SYSTEM MANAGER'S GUIDE	1 -"-
INSTRUCTION MANUAL	1 -"-

PREJEL : DEMŠAR JANEZ
OB. E. KRANJ

TANJAN J.
za

Prevzel sem dokumentacijo dne 30.09.87

- Načrt CPU D800
- kopije filma za ploščo
- Tabele ROM-ov
- seznam elementov.

Naučil sem se dekagirat modul z analizatorjem.
Tečaj sem končal 27.09.87

Učil»

BS 16 SPISEK MATERIALA

74S 260	20X
85S 68	3X
S 244	20X
LS 00	20X
Am 2904	3X
Am 2941	8X
MMC2147	40X + 20 + 20 + 20 + 20 + 20 + 20 + 20 + 22
S 472	10X
S 253	6X
AM2918	20X
25LS252	8X
DM 74S 253	25X
2907	4X + 7X + 42
74LS08	14X
74S32	9
LS 298	10
LS 243	10
DM 87S181	16
6381	17
FC 93419PC	9
S74	8
LS175	10
S 157	7
6305	3
29811	10
S 257	10
S 240	7
LS 02	10
LS 87	10
LS 132	10

S138	10x
LS11	2x
S240	3x
LS10	10x
S251	10x
S04	15x
Am 29803	10x
LS73	12x
H01	1x
S174	1x
LS86	2x
LS257	10x
2901	8x + 1 + 18
2909	10x
2716	4x
2904	~~9x~~ 17x
2901	~~14~~
MC6808	9x
2532	2x
Am 2841	10x + 2
SGS M5517	~~30~~ 15x + 15
6850	12x.
MC 145107	4x
745200	20x
MMI 6349	3x
2923	10x
~~2804~~ 2516	1x + 5x
74LS02	3x
29116	1x
74s04	
LS266	17x
Am 2911	20x

2914	8x + 1
2902	7x
74~~S~~374	10x
29~~H~~03	4x + 2
2910	2x
2904	1x
74LS374	6x
LS253	15x
LS273	8x
PAL 16L2	3x
16H2	1x
WW AUGAT QUAD	1x
AM 2920	1x

Pravd

Predal:

A6D2

28B7
28B9
28BD
28BD
28B1
2
3
4
032A
B
D get drive type code
G1C1 ← RAM ?? G1D1
032F
0332
0333
4
5
8
B
B
0033
36
1B
1C
1E
46

28B1 return to rejoin main flow
2
3
4
32A