a) Potrebno zaporedje je naslednje:

- odpiranje vrstice (izbira vrstice in banke)
- zakasnitev t_{RCD} zaokrožena na najmanjše celo število urinih period, ki je enako ali večje t_{RCD} čipa (v našem primeru 2 u.p. ali 20ns >= 18ns)
- branje vrstice (izbira stolpca)
- zakasnitev t_{CL} (v našem primeru 3 u.p. ali 30ns)
- prenos 4 besed po vodilu (4 u.p. ali 40ns)
- zapiranje vrstice
- pred ponovnim odpiranjem iste vrstice mora preteči še zakasnitev t_{RP} (v našem primeru 2 u.p. ali 20ns >= 18ns)

b) Čas prenosa (v u.p.) = t_{RCD} (v u.p.) + t_{CL} (v u.p.) + N (u.p.) = 2u.p.+3u.p.+256u.p. = 261 u.p. = 2,61μs kjer je N velikost vrstice v besedah

Imamo 8 neodvisnih modulov z velikostjo vrstice 1024x8 bitov. Ker novih vrstic ne odpiramo in lahko znotraj ene vrstice beremo (ali pišemo) podatke z maksimalno hitrostjo, ki jo dovoljuje DDR2 čip, lahko z maksimalno hitrostjo preberemo največ 8x1024x8 bitov ali 65536 bitov velik blok.

Da dosežemo maksimalno hitrost moramo na čip pošiljati ukaze za branje v takih časovnih presledkih, da se bo naslednje branje pričelo izvajati ravno takrat, ko se je prejšnje končalo. Vrsti red dostopa do stolpcev znotraj odprte vrstice je lahko poljuben, ravno tako lahko pri vsakem branju poljubno izbiramo modul.

N_{sec} - velikost sektorja v bajtih

RPM - hitrost vrtenja diska (število obratov na minuto)

B_{sri} - zunanja hitrost prenosa

B_{min} - najmanjša notranja hitrost prenosa

$t_{seek_{max}}$ - maksimalni iskalni čas

Minimalen čas dostopa je takrat, ko se želeni sektorji nahajajo v predpomnilniku. Takrat je čas dostopa:

$$t_{d_{min}} = \frac{2*N_{sec}}{B_{sri}} = \frac{2*512B}{200*10^6\ B/s} = \frac{2}{390625}s = 5.12\mu s$$

Maksimalni dostop časa pa imamo takrat, ko so hkrati izpolnjeni naslednji pogoji:

- sektorjev ni v predpomnilniku in ju moramo prebrati z diska;
- sektorjem v takem zaporedju, da traja pomik glave na ustrezno sled najdalj časa, se pravi iskalni čas maksimalni (za navedeni disk je enak 10ms);
- ko je glava na pravi sledi, je potrebno počakati, da se disk zavrti do pravega sektorja, kar v najslabšem primeru traja en obrat diska;
- vsaj en sektor se najprej prebere v predpomnilnik z najmanjšo notranjo prenosno hitrostjo.

$$t_{d_{max}} = 2*\left[t_{seek_{max}} + \frac{60s}{RPM} + \frac{N_{sec}}{B_{sri_{max}}} + \frac{N_{sec}}{B_{sri}}\right] = 2*\left[10ms + \frac{60s}{7200} + \frac{512B*8}{270*10^6\ bit/s} + \frac{512B}{200*10^6\ B/s}\right] =$$

$$= 2*\left[10ms + 8,\overline{3}ms + 15.17\mu s + 2.56\mu s\right] \approx 2*18,\overline{3}ms = 36,6ms$$

Pri enačbi nismo upoštevali, da se vsaj en sektor prebere hitreje kot z najmanjšo notranjo prenosno hitrostjo, saj sicer ne bi imeli dveh maksimalnih iskalnih časov. Ker pa vidimo, da so časi, ki so pogojeni s premikom glave in ploščo diska, tisočkrat večji od časov prenosa, jih lahko v tem primeru zanemarimo.

Teoretična pasovna širina vodila je: BW = 4B * 66M/s = 264 MB/s, vendar se podatki ne prenašajo s to hitrostjo, saj vsakem eksplozijskem prenosu porabimo en cikel za prenos naslova in kontrolnih signalov ter 1 cikel za zaključek prenosa. Za prenos 8 besed porabimo 10 urinih period vodila.

Efektivna pasovna širina je tako : BWef = 8/10 * BW = 211,2 MB/s

Podatki po vodilu se bodo prenašali z efektivno pasovno širino. Disk je veliko počasnejši od vodila, zato traja celoten prenos:

4,194304 MB / 70MB/s = 59,92 ms

Podatki se prenašajo v blokih. Disk pripravlja bloke podatkov (s svojo prenosno hitrostjo), posamezni bloki pa se nato iz diskovnega predpomnilnika prenesejo z maksimalno možno hitrostjo po vodilu. Tako traja prenos enega bloka:

16,384 kB / 211,2 MB/s = 77,58 us

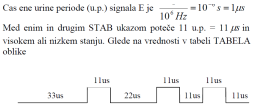
Vsega skupaj prenesemo 4,194304 MB / 16,384 kB = 256 blokov.

Prenos blokov po vodilu (to opravlja procesor) tako traja: 256 * 77,58 us = 19,86 ms.

Procesor je s prenašanjem blokov zaseden: (19,86 ms / 59,92 ms) * 100% = 33,14% časa prenosa.

Čas ene urine periode (u.p.) signala E je $\frac{1}{10^8\ Hz} = 10^{-8}s = 1\mu s$

Med enim in drugim STAB ukazom poteče 11 u.p. = 11 μs in to je tudi najkrajši možen čas, ko je lahko signal visokem ali nizkem stanju. Glede na vrednosti v tabeli TABELA program potem na PB0 izhodu vezja PIA tvori sign oblike



Signal je periodičen s periodo 110 μs .

Pravilni odgovor je : B.

Razlaga: Najprej na naslovno vodilo postavimo naslov vrstice ter aktiviramo signal RAS (ukaz ACTIVATE). S tem naslovimo vrstico v izbrani banki ter podatke prenesemo v register vrstice. Nato na naslovno vodilo postavimo naslov stolpca ter aktiviramo signal CAS (ukaz READ ali WRITE). Podatki bodo na podatkovnem vodilu veljavni po poteku CAS latence, zato jih smemo brati šele po poteku CL urinih period.

Pri branju mora signal potovati v obe smeri. Prenosni cikel torej znaša vsaj 2*tau+vse zakasnitve

tau = 0.25m * 6ns/m = 1.5ns

T >= 2*1.5ns + 2ns + 12 ns + 3ns = 20 ns

prenosna hitrost = 32 B * 1/20ns = 4B * 50 * 10^6 Hz = 200 MB/s

Pravilen odgovor je pod izbiro a. Pri branju iz SDRAM se mora na signalih A10-A0 najprej pojaviti naslov vrstice, aktivira se signal RAS# (aktiven v nizkem stanju), signala CAS# in WE# pa sta v neaktivnem stanju (visoko). Nato se mora na naslovnih signalih pojaviti naslov stolpca ter aktivirati signal CAS#. Signala RAS# in WE# sta neaktivna. Zatem je potrebno počakati CL urinih period (CAS latenca) in šele potem lahko beremo podatek.

Pri prenosu brez čakalnih stanj mora biti dostopni čas EPROMa manjši od:

dve periodi: 2/25E6 – 22ns – 5ns =53ns

vsaka čakalna perioda traja 40ns, torej je potrebno vstaviti 3:

53ns + 3*40ns=173ns, kar pomeni, da mora biti dostopni čas EPROMa manjši od 173 ns

a) Pri programskem vhodu-izhodu z uporabo prekinitev procesor pri čakanju na pripravljenost naprave ni zaseden (0%), brez uporabe prekinitev pa mora stalno preverjati pripravljenost naprave (zaseden 100%).

b) Pri obeh načinih je procesor pri prenosu polno zaseden (100%).

c) Hitrejši odzivni čas pri večini procesorjev dosežemo brez uporabe prekinitev. Večina procesorjev potrebuje več časa za opravila pri vstopu v prekinitveno servisni program (shranitev PC in drugih registrov, skok v PSP, ...) kot za izvršitev ukazov, ki so potrebni za preverjanje ali je naprava pripravljena na prenos.

d) Pri obeh načinih je sam prenos enako hiter. V obeh primerih mora procesor prebrati podatek iz V/I naprave ali pomnilnika ter ga nato zapisati v V/I napravo ali pomnilnik – odvisno od smeri prenosa.

Najmanjši čas: blok je že v diskovem predpomnilniku:

t = 512.66E6 = 7,75 us.

Povprečni čas:

t = (8 ms + 0,5*60/(7200) + 512*8/200E6)*0,95 + 0,05*7,75 us = 11578,18 us

Za izračun dostopnega časa moramo od časa ene periode urinega signala odšteti:

- maksimalni čas, ki je potreben, da se naslovni in kontrolni signali stabilizirajo in je v našem primeru 15ns;
- minimalen čas, ki je potreben, da so podatkovni signali stabilni pred koncem urine periode (v našem primeru 7ns);
- zakasnitev dekodirnih vezij.

t_0 = 1/20MHz = 50ns

t_{min} = 50ns – (15 ns + 7 ns + 13 ns) = 15ns

Pri počasnejši pomnilniških vezjih moramo zmanjšati frekvenco ure.

7200 obratov na minuto / 60 = 120 obratov na sekundo.

Na notranji sledi je hitrost večja ali enaka 17 MB na sekundo, torej se v enem obratu prenese 17MB/120 bajtov oziroma vsaj 17MB/120/512B = **277 sektorjev** (zaokrožitev navzgor, sicer je notranja hitrost prenosa manjša od 17MB/sekundo).

Na zunanji sledi je hitrost manjša ali enaka 28 MB na sekundo, torej se v enem obratu prenese 28MB/120 bajtov oziroma vsaj 28MB/120/512B = **455 sektorjev** (zaokrožitev navzdol, sicer je notranja hitrost prenosa višja od 28 MB/sekundo).

3a)

blok je potrebno iz predpomnilnika diska prenesti vsakih

2048 B/63E6 B/s = 0,032 ms.

Za to je potrebnih 2048/4 = 512 obhodov zanke, vsak obhod v povprečju traja 10*1,2=12 urinih period torej se za prenos bloka s pomočjo zanke porabi 512*12=6144 period.

Pri frekvenci ure f_{CPU}=3 GHz, je za to potrebnih 6144/3E9 = 0,002048 ms. Torej je zasedenost procesorja zaseden 0,002048 ms/0,032 ms = 6,4% časa.

3b)

Za vsak blok se porabi 18 urinih period, torej 18/3E9 = 6 ns. Zasedenost je torej 6E-9 s/ 32E-6 s = 0,018755%.

Zaporedje servisiranj naprav podano v č.e. (od - do vključno)

Naprava 2:	20 č.e. – 24 č.e.	
Naprava 1:	25 č.e. – 39 č.e.	
Naprava 2:	40 č.e. – 49 č.e.	; do konca servisiramo prekinitev na nivoju 2
Naprava 3:	50 č.e. – 59 č.e.	
Naprava 2:	60 č.e. – 74 č.e.	
Naprava 3:	75 č.e. – 79 č.e.	; do konca servisiramo prekinitev na nivoju 3
Naprava 4:	80 č.e. – 94 č.e.	

Povprečni odzivni čas = ((20 – 20) + (25 – 25) + (50 – 25) + (80 – 40) + (60 – 60)) / 5 =
= (0 + 0 + 25 + 40 + 0) / 5 = 13 č.e.

1. Maksimalno hitrost prenosa podatkov (lahko ji rečemo tudi maksimalna efektivna hitrost prenosa) označimo z $BW_{ef\ max}$. Dosežemo jo pri prenosu z največjo dolžino eksplozijskega prenosa:

$$BW_{ef\ max} = \frac{ds}{t_{prenosa}} = \frac{255*4B}{257\ u.p.} = \frac{255*4B}{257*1/33MHz}$$

$$BW_{ef\ max} = \frac{255*4B*33*10^6}{257s} = \frac{255}{257}*132MB/s = 130,9727MB/s$$

2. Tukaj moramo upoštevati dve dolžini eksplozijskega prenosa:
   a) Prenos pri največji dolžini eksplozijskega prenosa:
      Pri največji dolžini eksplozijskega prenosa se podatki prenašajo z maksimalno efektivno hitrostjo prenosa $BW_{ef\ max}$. Ker je zunanja hitrost za trdi disk manjša od pasovne širine vodila, določa trdi disk maksimalno hitrost prenosa podatkov. Čas prenosa 4MB v/iz diska je tako:

$$t_{prenosa} = \frac{4MB}{50MB/s} = 80ms$$

Čas prenosa enega bloka (le-ta se prenese s hitrostjo $BW_{ef\ max}$) je:

$$t_{blok} = \frac{8kB}{130,9727MB/s} = \frac{8kB}{130,9727MB/s} = 61,1\mu s$$

Med prenosom bloka je procesor zaseden le prvih 256 ciklov cikla, v zadnjem ciklu lahko izvaja druge ukaze. Zato je procesor za prenos bloka zaseden le 256/257 časa t_{blok}! Vseh blokov, ki jih mora procesor prenesti je 4MB/8kB = 500. Zato je procesor zaseden:

$$zasedenost_procesorja = \frac{256}{257}*\frac{1}{t_{blok}}*500 = 38,027\%$$

   b) Prenos pri najmanjši dolžini eksplozijskega prenosa:
      Pri najmanjši dolžini eksplozijskega prenosa je efektivna hitrost prenosa podatkov:

$$BW_{ef\ min} = \frac{1*4B}{3\ u.p.} = \frac{1}{3}*132MB/s = 44MB/s$$

Zakaj je zunanja hitrost za trdi disk večja od pasovne širine vodila, določa vodilo maksimalno hitrost prenosa podatkov? Čas prenosa 4MB v/iz diska je tako:

$$t_{prenosa} = \frac{4MB}{44MB/s} = 90ms$$

Zakaj je v tem primeru ozko grlo vodilo, procesor ves čas prenaša bloke. Prost je le v zadnjem ciklu eksplozijskega prenosa. Vsak prenos se izvede v 3 ciklih vodila pri čemer je procesor prost v 3. ciklu. Zato je procesor zaseden le 2/3 časa celotnega prenosa oz. 66, 66 %.

S	– velikost podatkov za prenos	B_d	– prenosna hitrost diska
S_b	– velikost bloka podatkov	n_i	– čas CPE v u.p. za inicializacijo DMA
N_b	– število blokov za prenos	n_p	– čas CPE v u.p. za obdelavo prekinitve
B_v	– prenosna hitrost vodila	t_{CPE}	– procesorski čas
f	– frekvenca vodila	f_{CPE}	– frekvenca urinega signala procesorja
w	– širina vodila	t_{pr}	– čas prenosa podatkov
n	– ur. period na prenos	t_{int}	– čas med prekinitvami

$$N_b = \frac{S}{S_b} = \frac{1,073741824GB}{8192KB} = 131072$$

$$B_v = \frac{f*w}{n} = \frac{66MHz*8B}{2} = 264MB/s$$

$B_d = 200MB/s < B_v$; Prenosna hitrost vodila je večja od prenosne hitrosti diska, se pravi, da je ; vodilo dovolj hitro, da se podatki z diska prenašajo z maksimalno hitrostjo.

$$t_{CPE} = (n_i + N_b*n_p)*t_0 = \frac{n_i + N_b*n_p}{f_{CPE}} = \frac{50000+10000*131072}{1,5GHz} = 0,87s$$

$$t_{pr} = \frac{S}{B_d} = \frac{1,073741824GB}{200MB/s} = 5,37s$$

$$p_{CPE} = \frac{t_{CPE}}{t_{pr}} = \frac{0,87s}{5,37s} = 0,16 = 16\%$$

$$t_{int} = \frac{S_b}{B_d} = \frac{8,192KB}{200MB/s} = 40,96*10^{-6}s = 40,96\mu s$$

Pri rešitvi smo v času prenosa zanemarili CPE čas (t_{CPE}). Ker je ta čas le približno sedem krat manjši, je bolj natančna rešitev, ki upošteva tudi ta čas naslednja:

$$t_{pr} = \frac{S}{B_d} + t_{CPE} = \frac{1,073741824GB}{200MB/s} + 0,87s = 6,24s$$

$$p_{CPE} = \frac{t_{CPE}}{t_{pr}} = \frac{0,87s}{6,24s} = 0,14 = 14\%$$

$$t_{int} = \frac{S_b}{B_d} + \frac{n_p}{f_{CPE}} = \frac{8,192KB}{200MB/s} + \frac{10000}{1,5GHz} = 47,63*10^{-6}s = 47,63\mu s$$

Za programski prenos enega bajta podatkov potrebujemo torej 5 ukazov (ukaze pred zanko zanemarimo). Vsak ukaz se izvede v dveh urinih (E) periodah, pri čemer je dolžina ene E periode enaka

$$\frac{1}{4MHz} = 250ns$$

Vsak ukaz se torej izvede v 0,5 us, prenos enega bajta pa traja 5 * 0,5 us = 2,5 us.

S programskim prenosom lahko torej dosežemo

$$\frac{1B}{2.5\mu s} = 400kB/s$$

b)

Pri DMA prenosu smo omejeni s hitrostjo pomnilnika in če DMA krmilnik podpira leteči (fly-by) način prenosa, lahko en bajt podatkov prenesemo v 100ns in dosežemo prenosno hitrost 10MB/s.

Pri prenosu podatkov iz pomnilnika v pomnilnik mora DMA krmilnik delovati v pretočnem načinu (flow-through) in za vsak prenos enega bajta podatkov potrebujemo eno branje in eno pisanje. Tako za en bajt podatkov potrebujemo 200ns. Prenosna hitrost je tako dvakrat manjša kot v prejšnjem primeru in je 5MB/s.

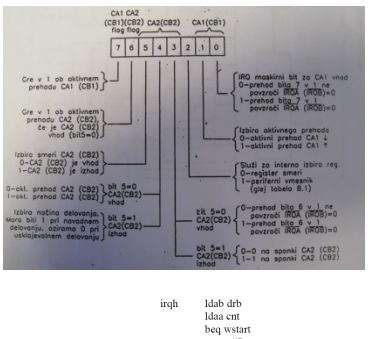
Da krmilnik podpira pretočni način prenosa, je tudi potrebni pogoj, da lahko prenašamo podatke iz pomnilnika v pomnilnik.

```
7   ACCUMULATOR A      0  7   ACCUMULATOR B      0
15          DOUBLE ACCUMULATOR D                 0
15          INDEX REGISTER X                     0
15          INDEX REGISTER Y                     0
15          STACK POINTER                        0
15          PROGRAM COUNTER                      0
                                                 0
            7 |  |  |  |  |  |  |  |  0
            S | X | H | I | N | Z | V | C
            |   |   |   |   |   |   |   |
            |   |   |   |   |   |   |   CARRY
            |   |   |   |   |   |   OVERFLOW
            |   |   |   |   |   ZERO
            |   |   |   |   NEGATIVE
            |   |   |   INTERRUPT MASK
            |   |   HALF-CARRY
            |   X INTERRUPT MASK
            STOP DISABLE
```

```
        org $c000
dtmf    fcb %1010111,%1101110,%1011110,%0111110,%1101101
        fcb %1011101,%0111101,%1101011,%1011011,%0111011

dial    ldaa #9
        ldy #stev
di_p    ldab 0,y        ;števki n v ak. b
        ldx #dtmf       ;ustreza kombinaciji n v tabeli dtmf
        abx             ;naslovu tabele prištejemo ak. b
        ldab 0,x        ;na dobljenem naslovu je ustrezna kompibacija
        stab dra        ;zapišemo v podatkovni reg. a
        bsr delay       ;zakasnitev
        iny
        deca
        bne di_lp
        ldaa #$ff       ;izklopimo vse oscilatorje
        staa dra
        rts

delay   ldd tcnt
        std tocl
        ldx #2                  ;2x 50 ms zakasnitev
de_lp   ldd tocl                ;k tocl prištejemo 50 ms
        addd #61440             ;50 ms
        std tocl
        ldaa #%10000000         ;brisanje zastavice
        staa tflg1
de_lp1  ldaa tflg1              ;po 50 ms se zastavica postavi
        bpl de_lp1
        dex
        bne de_lp
        rts

POLJE   RMB 4
STEVEC  RMB 1
..
PSP:    ldx   #$1800
        ldy   #POLJE
        ldab  #4
        stab  STEVEC

ZANKA:  tst   1,x #%11000000  ; preverimo prisotnost prekinitev
        beq   NI_PREK

        ldaa  0,x             ; slepo branje
        ldaa  0,y
        staa  0,x

NI_PREK:  abx
          iny
          dec   STEVEC
          bne   ZANKA
          rti

PSP     LDX     #PIA
        BCLR    1,X #%00001000      ; aktiviramo RD signal
        LDAB    0,X                 ; preberemo LSB
        LDAA    2,X                 ; preberemo MSB
        BSET    1,X #%00001000      ; deaktiviramo RD signal
        LDX     NASLOV
        STD     0,X                 ; shranimo podatek
        INX                         ; ustrezno povečamo naslov
        INX
        STX     NASLOV              ; in ga shranimo v spr.
        RTI                         ; povratek iz prekinitve

PIA     EQU     $1800
DRA     EQU     0                   ; PIA – periferni register vrata A
DRB     EQU     2                   ; PIA – periferni register vrata B

PODPROG PSHX
        PSHB
        LDX     #PIA
        LDAB    #8
LOOP    BSET    DRA,X #%00000001    ; izhod PA0 v visoko stanje
        STAA    DRB,X
        LSRA
        BCLR    DRA,X #%00000001    ; izhod PA0 v nizko stanje
        DECB
        BNE     LOOP
        PULB
        PULX
        RTS
```

```
STEVEC  RMB  1
...
PSP:    TSX                 ; vrh sklada v IX
        LDY  #PROC          ; naložimo naslov pod. str. PROC v IY
        LDAB  #9
        STAB  STEVEC        ; zamenjamo 9 8-bitnih vrednosti
LOOP:   LDAB  0,Y
        LDAA  0,X           ; zamenjujemo vrednosti med strukturo
        STAA  0,Y           ; in skladom
        STAB  0,X
        INX                 ; povečamo oba kazalca
        INY
        DEC   STEVEC
        BNE   LOOP          ; če ni konec, ponovimo
        RTI

SPREJEM LDX      #REG_BASE
        LDAA     #%00000001
        STAA     TFLG1,X            ; pobrišemo zastavico za f. IC3
WAIT1   BRCLR    TFLG1,X #%00000001 WAIT1 ; čakamo neg. fronto IC3
        LDD      TIC3,X             ; nastavimo alarm na sredino prvega
        ADDD     #90                ; bita podatkov, od 96 u.p. odštejemo
        STD      TOC1,X             ; 6 u.p. za LDAB in del BRCLR ukaza
        LDAA     #%10000000
        STAA     TFLG1,X            ; pobrišemo zastavico za alarm TOC1
        LDAA     #8                 ; 8 ponovitev
        PSHA
LOOP    BRCLR    TFLG1,X #%10000000 WAIT2 ; čakamo alarm TOC1
WAIT2   LDAB     PORTA,X            ; preberemo vrata A
        LSRB                        ; podatkovni bit v zastavico C
        ROR      PODATEK            ; nato v spremenljivko podatek
        LDD      TOC1,X             ; nastavimo alarm za sredino
        ADDD     #64                ; naslednjega bita
        STD      TOC1,X
        LDAA     #%10000000         ; pobrišemo zastavico za alarm TOC1
        STAA     TFLG1,X
        PULA
        DECA                        ; če nismo še obdelali 8 podatkovnih
        BNE      LOOP               ; bitov, ponovimo
WAIT3   BRCLR    TFLG1,X #%10000000 WAIT3 ; čakamo alarm TOC1
        LDAB     PORTA,X            ; preberemo stop bit
        ANDB     #%00000001
        EORB     #%00000001         ; negiramo vrednost in
        STAB     PE                 ; jo zapišemo v spr. PE
        LDAB     #1                 ; za konec sprejema postavimo spr.
        STAB     SPREJEM            ; SPREJEM na 1
        RTS

PIAIRQ  INC COUNTER
        LDAA COUNTER
        CMPA #7
        BEQ SWITCH
        CMPA #10
        BNE ENDIRQ
        CLR COUNTER
SWITCH  LDAA DRA                    ;DRA brati tudi zaradi potrjevanja
        EORA #%00000010
        STAA DRA
ENDIRQ  RTI
PONOVI: TSX
        LDAB  2,X           ; preberemo podatek o dolžini s sklada
        LDX   3,X           ; in naslov polja podatkov s sklada
        LDY   #STATUS       ; v IY damo naslov spr. STATUS
        BRCLR 0,Y #%00000001 CAKAJ  ; čakamo da v STATUS bit0 = 1
        LDAA  0,X           ; preberemo
        STAA  VMESNIK       ; in zapišemo 8-bitni podatek
        INX                 ; povečamo naslov v polju podatkov
        DECB                ; zmanjšamo števec
        BNE   CAKAJ         ; če nismo končali, ponovimo

        BRCLR 0,Y #%10000000 PONOVI  ; preverimo če v STATUS bit7 = 1

        RTS

SPREJEM LDX  #PIA_A                 ; naslov vrat A vezja PIA v IX
        CLRB                        ; brišemo reg. B
        LDAA #8                     ; števec ponovitev = 8
W_SS    BRSET  0,X #%00000100 W_SS  ; počakamo, da je signal SS=0
W_SCK_0 BRSET  0,X #%00000001 W_SCK_0  ; počakamo, da je SCK=0
W_SCK_1 BRCLR  0,X #%00000001 W_SCK_1  ; počakamo, da je SCK=1
        LDAA  0,X           ; zdaj imamo pozitivno fronto
        LSLB                ; rezultat za bit v levo
NADALJUJ BRCLR 0,X #%00000010 NADALJUJ  ; sprejeti bit 0 ali 1?
        INCB                ; če 1, upoštevamo v rezultatu
        DECA                ; zmanjšamo števec zanke
        BNE   W_SCK_0       ; če ni konec, ponovimo zanko
        RTS

PSP     LDX     #TABELA     ; v IX naslov spremenljivke TABELA
        TSY                 ; IY = SP+1
        XGDY
        ADDD
        STD     #8          ; prištejemo 8, da dobimo vrednost SP pred
        TSY                 ; prekinitvijo
        LDAA    0,X         ; shranimo s spr. TABELA
        LDAB    0,Y
        STAB    #9          ; nato v zanki prepišemo v spremenljivko
        INX     0,Y         ; TABELA še vrednosti ostalih registrov,
        INY     2,X         ; ki so na skladu in tudi povratni naslov
        DECA
        BNE     LOOP
        RTI
```

```
IRQPSP  LDAA    PIA_DRB     ; beremo vrata B (hkrati IRQ potrditev)
        LDAB    NELEM
        CMPB    #8
        BHI     ENDIRQ      ; če že 8 elementov, zaključimo
        LDX     #MEDPOM
        LDAB    GLAVA
        ABX                 ; izračunamo naslov v MEDPOM
        STAA    0,X         ; shranimo podatek
        INCB
        ANDB    #%000000111 ; povečamo odmik (GLAVA) po modulu 8
        STAB    GLAVA
        INC     NELEM       ; povečamo število elementov
ENDIRQ  RTI

PROC    LDAA    NELEM
        CMPA    #5
        BLO     ENDPROC     ; je manj kot 5 elementov?
        LDY     #MEDPOM
        LDAB    REP
        ABY                 ; izračunamo naslov podatka v MEDPOM
        LDAA    0,Y
        STAA    0,X         ; prepišemo na ciljni naslov
        INCB
        ANDB    #%000000111 ; povečamo odmik (REP) po modulu 8
        STAB    REP
        INX                 ; povečamo ciljni naslov
        DEC     NELEM       ; zmanjšamo število elementov
        BNE     LOOP        ; prepisali vse?
ENDPROC RTS
```




```
BLOK    LDAB CRB
        BPL FRONTA
        LDAB DRB
        LDAB 0,X
        PSHB
        ORAB #%00010000
        STAB DRB
        ANDB #%00001111
        STAB DRB
        PULB
        LSRB
        LSRB
        LSRB
        ORAB #%00010000
        STAB DRB
        ANDB #%00001111
        STAB DRB
        INX
        DECA
        BNE BLOK
        RTS
```

Pri frekvenci ure 1 MHz je čas cikla 1 μs.

```
POSLJI  LDX  #DRA
        LDAB #8
        SEI
ZANKA   BCLR 0,X,%00000001
        STAA 0,X
        LSRA
        PSHB
        LDAB #12
ZAKAS   DECB
        BNE  ZAKAS
        PULB
        BSET 0,X,%00000001
        DECB
        BNE  ZANKA
        RTS
```

Za navedeno rešitev velja: t_{pos}=4 μs (STAA 0,X), t_{ZAK}=82 μs (LSRA, PSHB, LDAB #12, ZANKA, PULB, BSET), t_{rec}=12 μs (DECB, BNE, BCLR).

```
irqh    ldab drb
        ldaa cnt
        beq wstart
        cmpa #8
        ble dbit

        clr cnt
        clr error
        lslb
        bcs eirqh
        inc error
        bra eirqh

dbit    lslb
        ror buffer
        inc cnt
        bra eirqh

wstart  andb #%10000000
        bne eirqh
        inc cnt

eirqh   rti
```

```
TABELA  FCC "0123456789ABCDEF"

IZPIS   LDAA #8
IZANKA  LDAB 0,X
        PSHX
        PSHB
        LSRB
        LSRB
        LSRB
        LSRB
        LDX #TABELA
        ABX
        LDAB 0,X
        STAB 0,Y
        PULB
        ANDB #$0F
        LDX #TABELA
        ABX
        LDAB 0,X
        STAB 1,Y
        LDAB #' '
        STAB 2,Y
        LDAB #3
        ABY
        PULX
        INX
        DECA
        BNE IZANKA
        RTS
```

```
IRQPSP  LDAA    DRA         ;brisanje zastavice
        LDX     #$4000      ;naslov A/D pretvornika
        LDAB    CH1
        ABX                 ;prištejemo št. kanala
        LDAA    0,X         ;kanal, katerega št. je v CH1
        LDX     #$4000
        LDAB    CH2
        ABX
        ADDA    0,X         ;rezultat je v splošnem 9-biten
        RORA                ;uporabimo ROR, da upoštevamo še prenos pri seštevanju
        STAA    $4010       ;D/A pretvornik
        RTI

TMP_Y   RMB     2           ; pomožna spremenljivka
...
PALINDROM PSHX
          PULY              ; prepišemo IX v IY
SRCH    TST     0,Y         ; in pošiljemo konec niza
        BEQ     FOUND
        INY                 ; ce še ni konec povecamo IY
        BRA     SRCH        ; in nadaljujemo z iskanjem
FOUND   DEY                 ; IY kaže na 0, zato ga zmanjšamo za 1
TEST_PAL STY    TMP_Y
         CPX    TMP_Y       ; preverimo ali smo že preverili cel palindrom
         BHS    OK          ; ce smo, gremo na konec
         LDAA   0,X
         CMPA   0,Y         ; sicer preverimo naslednja znaka
         BNE    NOT_OK      ; ce nista enaka, ni palindrom – vrnemo
                            ; nenicelno vrednost v A
         INX
         DEY                ; ustrezno povecamo,zmanjšamo IX,IY
         BRA    TEST_PAL    ; in nadaljujemo s preverjanjem
OK       CLRA               ; našli smo palindrom – v A vrnemo 0
NOT_OK   RTS                ; konec podprograma
```

b7	b6	b5	b4	b3	b2	b1	b0
OC1F	OC2F	OC3F	OC4F	I4/O5F	IC1F	IC2F	IC3F

Timer Interrupt Flag 1 Register (TFLG1)