LUXOR

Testlåda 1600

Bruksanvisning

Programbeskrivning

flödesschema

Programlistning

Innehåll

LUXOR DATORER
Håkan Heldmann
850509

1 Inledning

Denna dokumentation skall vara en liten beskrivning om vad man kan använda testutrustningen till och hur man skall tolka dom olika felutskrifter,felmeddelanden som ges genom LYSDIODER ,SJUSEGMENT och SKÄRM, samt en liten vägledande beskrivning på hur 1600 fungerar för varje enskild test. Det finns två valmöjligheter vid testningen 1: test prom
2: boot prommet

Med hjälp av en omkopplare kan man välja mellan dessa två prom.
Var i testprogrammet man är indikeras på SJUSEGMENTEN, FEL-ledtrådar ges med hjälp av LYSDIODERNA .Skärmen ger även vissa ledtrådar efter att DARTEN har blivit testad. Testprogrammet skall ge en ledning var på kortet felet kan ligga, samt ge en möjlighet att mäta på vissa viktiga signaler som ENABLE, svarsignaler,bussledningar osv . Kortet indelas i olika block : segmentRAM, pageRAM, primärminne, 0-1M, DART, CIO+ klocka, floppy, nvram, test av paritetskontrollkretsarna, dma och test av busskort. Testen skall ge en ledning om vad som är fel inom ett block .
Det är inte helt säkert att FELET ligger i komponenten som blir utpekad,det kan tex vara styrsignaler från andra komponenter som gör att utpekad komponent inte fungerar riktigt. För att kunna mäta sig fram till vilka signaler som är felaktiga står programmet och loopar på den plats där felet har uppkommit.När felet är åtgärdat går programmet vidare i testen.
Genom att använda olika mätinstrument, titta på ritningar, flödesschema samt programlistning plus erfarenhet hur olika fel kombinationer ser ut skall dom uppkomna felen gå att hitta. För att kunna felsöka effektivt och kunna förstå hur signalerna ser ut i olika fall behövs det datablad för dom olika kretsarna som finns på CPU-kortet samt en viss förståelse hur det kortet arbetar.
ABC-burken skall anslutas till VIDEO-kontakterna på cpu-kortet.
Boot-prommet på CPU-kortet skall bytas ut mot testkabeln från ABC-burken (eller EPROM direkt på kortet) .
Någon typ av TERMINAL (tex ABC+802 +tangentbord) som kopplas till DISPLAY/CONSOLE kontakten på cpu kortet behövs. Byglingen på ABC802 dip switch skall vara 1=off,2=off,3=on,4=on,5=off,6=off,7=on,8=off (byglingen innebär att 802:s klocka läggs ut på pinne 6 i channel B).

1.1 Övrig kringutrustning

För att kunna testa floppy-controllen skall en floppydriver (tex typen Teac 55f) anslutas till cpu-kortet. Testskivan som används i testen skall vara formaterad och inte skrivskyddad.

Vid test av busskortet skall det sitta ett SAZI-interface (databoard 4105-10) i någon av busskontakterna.
Till SAZI interfacet skall ett kontrollerkort TYP XEBEC 1410A CONTROLLER anslutas (med tillhörande driver).

2 Segmentram

Mac står för memory-access-controller. Den flyttar CPU:s logiska adresser till fysiska (relativa) adresser i primärminnet. Utan fungerande MAC går det inte att ladda primärminnet med program.

Mac-testen är uppdelad i två BLOCK.

Block 1: SEGMENT-rammet
2: PAGE-rammet

2.1 Teknisk beskrivning av taskregistret

TASK-registret används för att peka ut vilken process som körs . Vi kan ha 0-15 processer igång samtidigt i datorn. Innan maccen kan testas och sättas upp måste ett process nummer laddas in i taskregistret. Det görs genom att lägga ut ett dataord som klockas in i taskregistret av signalen W(c) (adresss 80005) som kommer från avkodningspalen 16L8 (17E). Taskregistret innehåller förutom processnumret även BOOT-prom enable på låga adresser (BOOTE*) samt MACIC bit (CT*).

2.2 Teknisk beskrivning av SEGMENT-RAM

Segmentramet delar upp cpu:s primärminne i olika segment. Vi kan max ha 32st segment (per process).
Varje segment kan sedan ha 16 sidor "page" i pageramet. Segment-rammet består av två ram-kretsar (2149) som har beteckningen 17G,15G. Två st 74245 (transiver) sköter inskrivningen av data i SEGMENT-ramet. Adresseringen av segmentramet sker genom TASK-registret + A15-A19.
Vid uppsättning av maccen används A8 istället för A19, A19 används för att indikera att vi skriver i maccen och inte i BOOT-prommet vid LÅGA adresser. Den avkodningen görs av OR grindarna (20F) samt pal 16L8 (5D). Den finessen används inte sedan vid körningen, det görs NOENABLE BOOT-prom via TASK-registrets bit 7 (läggs "0"). Avkodning av chip select till SEGMENT-ramet görs via PAL16L8 (17E). Avkodningen sker via adressledningarna A0-A3,A19 ,chip selecten ligger på adress 80003 för SEGMENT-ram och 80005 för taskregistret.

Programbeskrivning segmentram

Testprogrammet för "MACCEN" går ej att välja från meny. Maccen testas direkt efter RESET av CPU-kortet med adressering och bittest. Med hjälp av ABC-burken ser man hur långt testningen har kommit innan något fel har uppkommit. Var i testen man är indikeras av två SJUSEGMENT (0-5). Om något fel uppstår ges vissa "ledtrådar" om var felet kan ligga av ÅTTA st LYSDIODER, någon utskrift på skärmen kan ej ske innan maccen är uppsatt.

Innan det kan laddas in data i segmentrammet skrivs det in vilken process som gäller i TASKREGISTRET 19E. I testen skrivs det in process "0" (£/40), det genereras en puls på ben 9 (W(c)) som klockar in det i taskregistret. Vid reset skall det även komma en puls på ben 1 (HRST).

3.1 segram "bittest"

Första delen av testen testar skrivbarheten (bittest) i kretsarna.En vandrande etta får gå igenom alla minnesceller ,som sedan kontrolläses. Testen testar båda segmentram kretsarna innan någon felutskrift skrivs ut,med ledning av detta kan man dra olika slutsatser om felet. Om något fel uppstår indikeras det på lysdioderna och programmet gör omtest på nytt. Under omtestningarna kan styrsignalerna iakttas.
Man kan kontrollmäta på CS o WE signalerna från pal 16L8 (17E). Segmentramets adress avkodas i palen genom adressledningarna A0-A3,A19 (adress 80003).

3.2 segram "adresstest"

I andra delen av testen testas det om adressbussarna är felaktiga.Det skrivs in ett mönster i segmentramen som sedan kontrolläses. Om det har blivit någon kopiering av det inskrivna mönstret är det troligtvis något fel på adressbussen eller adressavkodningen i minneskapseln (denna test förutsätter att bit-testen av segmentramet har gått bra. Om det uppkommer något fel vid adresstesten görs det en omtest av denna. OBS om man vill ta testen från början måste man göra en ny RESET.

3.3 segram "transparent

I tredje delen av testen görs segmentrammet transparent. Med det menas att en adress som kommer ut från segmentramet är den samma som adresserar segmentramet

3.1.1 Sjuseg 1

Här testas alla minnesbitar i segram

LYSDIOD:	Trolig FELORSAK:
00000001	bitfel i segram 15G fel på pal 17E fel på transiever 16G korslutning databuss kolla även R/W signalen från pal 17E
00000010	bitfel i segram 17G föv samma som ovan
00000011	troligen är det fel på styrningen av segmentramen. Kolla styrsignaler (enable, R/W). Att taskregistret är okey (19E).

Ovanstående ledtrådar är bara en hjälp att söka efter fel på rätt ställe.

3.2.1 Sjuseg 2

Här testas segmentramets adressbuss

LYSDIOD:	Trolig FELORSAK:
11111111	Det har skett ett adress- fel.Kan vara adressbuss A15-A18,A8 eller fel på logiken 20F. Även felaktig adressavkodning i minneskapseln är möjlig.

Ovanstående ledtrådar är bara en hjälp att söka efter felet på rätt ställe.

3.3.1 Sjuseg 3

Segmentramet görs transparent

5 Pageram

5.1 Teknisk beskrivning av pageram

Page-ramet består av tre ram-kretsar (2149) som har beteckningen 18G,22G,20G. Page-ramet får sin adress från segmentramet(5st) samt A11-A14 (4st). Det ger 16 sidor (page) per segment. Ett page ord består av 12 bitar,dom 10 första bitarna blir X-ADRESS (X11-X20) och övriga bitar 2 bitarna (högsta) använd som

1: WRITE PROTECT (ger busserror vid skrivförsök)
2: NON EXISTING (anger om sidan existerar eller ej)
(buss error on try)

Write protect (WPV) går via grind 7402 (28F) och 7464 (2E) som genererar buss error.
Non exist (NONX) gå ovanstående väg.

För att kunna mata in data i pageramet används 2 st transivrar (19G och 21G). Dom och pageramen får sina styrsignaler från PAL16L8 (17E) via multiplexen 74158 (21F). Adressavkodning sker via adresserna A0-A2,A19 (80000)

5.2 Programbeskrivning pageram

5.2.1 Bittest pageram

Det första som görs är reset av hela pageramet med efterföljande bit-test. Där testas skrivbarheten (bit-test) i kretsarna.En vandrande etta får gå igenom alla minnesceller ,som sedan kontroll läses. Testen går igenom alla TRE kretsar innan någon felutskrift ges. Om något fel uppstår indikeras det på lysdioderna och programmet testar på nytt.Under omtestningarna kan styrsignalerna iakttas (viktiga signaler är chip select av page-ram,W/R till pageram samt enable av transivrarna). OBS vill mån göra testen från början behövs en RESET. Med hjälp av felutskriften på lysdioderna kan man dra den slutsatsen om det gäller pageramen eller kontroll kretsarna runt omkring, om det är mer än ett fel kan man tex dra slutsatsen att det är kontrollkretsarna till pageramen som är felaktiga.

5.2.2 Adresstest pageram

Andra delen testar att det inte blir något adress fel (kopiering över till andra minnesceller).
Det skrivs ett ord på en adress ,varefter det kontroll läses på övriga adresser att det inte har skett någon kopiering över till dessa. Om fel hittas görs det omtest av adresstesten . OBS vill man att hela testen skall göras om får man göra RESET.

5.2.3 Sjuseg 4

Bit-test av pageramet

LYSDIOD:	Trolig FELORSAK:
00000001	fel i pageram 20G krets 20G transivern 21G 2/1 multiplexern 21F
00000010	fel i pageram 22G samt som ovan
00000100	fel på pageram 18G kolla transiever 19G samt som ovan

Kombinationer av ovan kan ske .-
Det kan vara till hjälp vid felsökningen.Om man får fel på alla pageramen så kan det antas att det är fel på kringkretsarna till pageramen ,tex chip select till transivrarna 19G,20G .På så sätt kan man följa upp felet bakvägen till sin felkälla.

5.2.4 Sjuseg 5

Här testas adresseringen till pageramen

LYSDIOD:	Trolig FELORSAK:
1111111	Det har blivit ett adressfel i pageramen. Kolla adressbussarna A11-A14. Det kan även vara internt avkod- ningsfel i minneskapslarna.

6 Sjuseg 6

Primärminnet testas för att kunna sätta upp en stack (för subrutinanrop). För att det inte skall bli något interupt sätts paritetsbiten "1" genom att skriva i krets 13E .Det är signalen PARTST som styr paritets-kretsarna 19D. Det gör att signalen PTY inte genereras till paritetskretsarna vid paritets fel.
Om testprogrammet stannar här kan det vara fel på IOACK* (i/o ack) signalen. Den signalen genereras när en I/O krets har blivit adresserad. Om kontrollkretsarna för denna signal inte fungerar stannar testen och reset görs. Vid fel kan tex klocksignalen till vipporna 7474 :9C kollas (8MHZ),om det kommer någon IOACK* signal från 7474 :9C, signalen från pal 16R6 11E (ben13) samt GDS* signalen mellan palarna 16R4 :12E och 16R6:11E (ben 12 och 9).

Minnestest för stack

Dom första 2K av primärminnet BIT-testas för stacken. Om bittesten går bra sätts stackpekarn upp. Om det blir bit-fel görs det en omtest på det skrivna ordet tills det inte visar bit-fel, dvs om en minneskapsel är trasig står testen och lopar tills den kretsen har kollats (bytts).
Om det blir många skrivfel visar det tex att det är dåliga ramar eller att styrsignalerna till primärminnet inte fungerar riktigt.

7.1 Programbeskrivning

Först sätts maccen upp för I/O-PAGE (ligger högst upp i adressrymden). Det måste göras för att vi skall kunna adressera SPECIALKONTROLLREGISTRET 74259 (13E). Nästa steg är att sätta den PARTST biten ,det görs genom att skriva på adress 7FE00 som genererar en strobe från krets 74138 (14G) varvid den signalen ETTSTÄLLS (utgång ben 4 (13E). Nästa steg är att sätta upp maccen för 0K-2K (det området stacken ligger på). På dessa 2K:na gör sedan BIT-test.(OBS inom dessa två kilobyte kan det ligga fel i paritetskretsen testas ej).
När bit-testen är klar initieras stackpekaren på adress 0800 i primärminnet (logisk adress 60800 ,har med att vi inte skall skriva BOOT prommet)

7.2 Sjuseg 7

Om det finns något bitfel i dom första 2K får man vägledning om i vilken krets fel ligger via lysdioderna. Dom visar vilken krets som bitfelet har uppkommit i (1-8). Vid fel görs det omtest ,DVS programmet står och LOOPAR tills felaktig krets är åtgärdad (bytt).

Om paritetslampan tänds under testen kolla då att biten PARTST verkligen sätts.

8 Dart

8.1 Teknisk beskrivning dart

Darten används för monitor ,printerkanal och keybord. Viktiga signaler som styr darten är DRT*. Denna signal kommer från kretsen som styr ENABLARNA för I/O-kretsar 74138: 4E (pinne 13). Utgången skall ligga lågt när cpukortet körs mot darten.
Resetsignalen RST* kommer från resetkretsarna via inverteraren 7406 :8D. Denna signal är "låg" vid reset. Signalen INT5* går till pal16L8 :5D som genereras när darten vill göra "interupt" på cpu:n.
Signalen RD* (read cycle status) anger om någon memory eller I/O process är igång. Denna signal kommer ifrån pal16R6 :11E, signalen skall ligga "låg" vid READ (WRITE "hög"). Signalen MINT5* används vid interuphantering. IORQ* används i kombination med dom övriga signalerna vid överföring av data och commando,den skall ligga låg när WRITE- eller READ-cycle görs. Signalen går till PAL16R4 (6E).

Vad darten skall göra styrs av adressledning A1,A2 via inverteran 7419 (15D) (kontroll port A "/4",kontroll-port B "/0",port A "/6£,port B "/2").

8.2 Programbeskrivning

För att kunna kommunicera med omgivningen måste DARTEN testas och initieras för yttre terminal. Genom att låta darten skriva ut text på skärmen kollas att SÄNDNINGEN fungerar (jämn paritet, 7 tecken per ord). För att kolla att mottagningen fungerar knappar man in ett antal tecken på tangentbordet (ekas ut på skärmen). Om darten har gått igenom testen skrivs det ut en MENY på bildskärmen där man kan välja nästa moment i testningen.

3Sjuseg 8

Om den utskrivna texten är oläslig kan man testa att byta ut DARTEN. Kolla dess ENABLE signaler (ritning 3 kretsar 15D(inverterare)).
Det kan även vara fel på transiever- och reciever buffrarna (15A,14A) på dartens utgångar.
Printerkanalerna testas inte i denna test.

9 PRIMÄRMINNE

9.1 Teknisk beskrivning primärminne

Primärminnet består av dynamiska ramkapslar, dessa dynamiska ramkapslar behöver "refrechas" då och då. Refrech görs under vissa tidsintervall som styrs av räknarna 7474 (24G) som delar E clockan från CPU med 2 Denna 400KHZ signal går till räknaren 74163 (23G) som delar ner frekvensen med 8. Denna utfrekvens triggar flip-flop 7474 (24G) som genererar utsignalen RFRQ*. Denna utsignal går till krets 74175 (24F) som i sin tur styr multiplexer 74352 (26F) som hindrar att vi får WE* och CAS strobe. Dessa åtgärder vidtas bara när vi har signalen MEMW från 74260 (26G) . För att ange datariktningen in till primärminnet använd signalerna RACK*,DS* och R/W* som NORas ihop i grind 7427 (8c) som styr riktningen. När en refrech görs ligger RACK* signalen hög och då går det inte att skriva i primärminnet . Den signalen kommer ifrån 7474 (25F). CAS och RAS signalerna fås genom att multiplexen 74253 (26F) genererar en RAS*-strobe som sedan fördröjs i en digital delay line som sedan tas ut som CAS signal. Denna signal går via en demultiplexer som är byglad för hur stor kapacitet varje minneskapsel har. Storleken på primärminnet anges genom att bygla ingången till grind 74260 (26G).

9.2 Programbeskrivning primärminne

Primärminnet testas på samma sätt som Macen. Testen är uppdelad i block om 256K minne. I varje block görs först en bit-test, typ vandrande ETTA. Adresstesten utföres med ett vandrande mönster som skrivs in i blocket. Detta mönster kontrollläses sedan så att inget adresseringsfel har inträffat.Om det blir fel skrivs adressen ut på bildskärmen samt texten "skriv eller bitfel" ,programmet står sen och snurrar vid felaktig adress tills det är åtgärdat.

Man kan välja vad man vill testa i dom olika blocken. (bit,adress,paritetstest).

Programmet testar själv om det finns 1M minne inmonterat i apparaten.

(anm: Adresser som slutar på 280 eller 380 görs inte test på, denna adress används för att avkoda testutrustningen.

9.2.1 Sjuseg 9,12,15,18

Här testas varje enskild bit i PRIMÄR- MINNET.
Genom att svara "j" på frågan om minnesarean skall bit testas görs en BIT-test av vald minnesarea (annars hoppas till adresstest).
När testningen börjar skrivs det TESTNING..... på bildskärmen. När minnes arean är testad får man meddelandet MINNES-AREAN ÄR BITTESTAD på skärmen.
Vid fel skrivs feladressen ut på skärmen samt felutskriften "skriv eller bitfel".programmet står och snurrar vid felaktig bit tills felet är åtgärdat.

LYSDIOD:	Trolig FELORSAK:
xxxxxxxx	ljusdioder pekar ut felaktig krets. Programmet står och snurrar på felaktig bit tills motsvarande krets byts ut. Om skrivfel har skett testar programmet att skriva om i biten. Om detta går forsätter programmet vidare i sin test (ingen åtgärd behövs göras). Om många skrivfel uppkommer tyder det på att något fel finns i styrningen av PRIMÄRMINNET (kan säkert orsaka´dykningar´hos maskinen). Det även vara fel på databussen tex kortslutning mellan två ledningar.

9.2.2 Sjuseg 10,13,16,19

Här testas ADRESSERINGEN till PRIMÄRMINNET.
Genom att svara "j" på frågan om minnesarean skall adress testas görs en ADRESS-test av vald minnesarea (annars hoppas till paritets test).
När testningen börjar skrivs det TESTNING..... på bildskärmen. När minnesarean är adresstestad skrivs MINNESAREAN ÄR ADRESSTESTAD ut på skärmen.
Feladressen skrivs ut ,med hjälp av denna kan man få fram vilken adressledning som är felaktig.

LJUSDIODER:	Trolig FELORSAK:
11111111	Adressfel har uppkommit. Kolla adressledningarna så att inga kortslutningar finns, att multiplexrarna inte är trasiga samt att rätt styrsignaler styr dom. Det kan även vara fel på internavkodningen hos minneskapslarna.

10 Paritetstest

10.1 Teknisk beskrivning

Om det finns felaktiga bitar i primärminnet kan CPU:s körning äventyras ,även viss data kan bli förstörd. För att förhindra att detta skall uppkomma finns det en paritetstest medtagen i konstruktionen . Den fungerar på det sättet att vid JÄMN inläsning i primärminnet genereras det en "0" i paritetskretsen (9 ramkretsen),vice versa vid udda antal ettor. Denna bit används sedan vid läsning av primärminnet. Om denna bit inte stämmer med utlästa dataordet genereras det ett PARITETSFEL. Inskrivning av paritetsbiten sker med kretsen 74280 (19D) som styr av signalen PARTST (från specialkontrollregistret).

10.2 Programbeskrivning

10.2.1 Sjuseg 11,14,17,20

I föregående tester har paritets kretsarna varit inaktiva för att inte störa minnestesten. Om man önskar göra paritetstest på NIONDE minneskapseln svarar man "j" på frågan "om minnesarean skall paritetstestas". Testprogrammet skriver först "0" på PARTST biten i specialkontrollregistret ,sedan skrivs det /ff i primärminnet som genererar en NOLLA i paritetskretsen. Sedan läses ordet direkt av.Om det uppstår ett paritetsfel TÄNDS lysdioden på CPU-kortet samt WATCHDOG dioden på ABC-burken. I detta skede gör processorn reset på sig själv (200 första korten, nyare kort hoppar till en avbrottsrutin där felutskrift på skärmen visas). Under testens gång skrivs det ut på skärmen "JÄMN PARITETS TESTNING..... .

Om ovan går igenom skrivs det hex 07 som genererar en ETTA i paritetskretsen. Testen görs sedan som ovan. Under testens gång skrivs det ut på skärmen "UDDA PARITETS TESTNING..... .

Vid fel måste det göras RESET för att kunna testa vidare (dvs testen startas på nytt,gäller dom första 200 korten). Dom senare 200 korten gör ett NMI avbrott vid paritets fel .Det skrivs ut på skärmen att paritetskretsarna fungerar samt att det uppstått paritetsfel i minnet.CAUSE-registret skrivs ut på lysdioderna som indikering på vad som skett (D1 och D2 är "1" om OKEY) .Återhopp till meny sker efter paritetsfel.

Om paritetsfel uppstår kolla paritetskretsen för motsvarande minnesarea (9 ramet).Om det även då blir paritetsfel kan man kolla att paritetsgenererande kretsar är OKEY samt att PARTST-signalen fungerar som den ska. Ta även en titt på JK- vippan 18B.Om det forfarande blir paritetsfel kan det bero på att vi har råkat på en AVKODNINGSADRESS till testlådan (väldigt liten sannolikhet " men den finns").

11 Kontroll av paritetskretsarn

11.2 Programbeskrivning

11.2.1 Sjuseg 21

Test av kontrollkretsarna för paritetsfel (paritetsbiten).
Genom att skriva data på stacken med felparitet (paritetsbiten satt),och sedan nollställa motsvarande bit vid läsning genererar vi ett medvetet paritetsfel. Om kretsarna fungerar riktigt görs ett avbrott (reset och paritetsdioden tänds (200 första korten) ,för övriga kort görs hopp till en avbrottsrutinen NMI. NMI-rutinen skriver att "paritetskretsarna fungerar(causeregistret skrivs ut på dioderna)" . Sedan testas det om paritetsfelet har uppkommit under paritetsfel eller under minnestest, om så är fallet skrivs det på skärmen " det har blivit fel i paritets test".
Om det inte är paritetsfel vid inhopp i NMI-rutinen skrivs det på skärmen "kontroll kretsarna felaktiga för paritetstest (eller något annat som ger NMI interupt)" samt läggs CAUSE registret ut på lysdioderna. Med hjälp av den informationen kan man dra slutsatser om vad som gått fel.

12 Cio

12.1 Teknisk beskrivning

CIO står för counter I/O .Den sköter avbrottshantering från busskortet (I0-I7) på PORT A. PORT B har med externa avbrott från bussar ,spänningsavbrott osv att göra . Klockan +NVRAM är kopplad till PORT C. Den kan även programeras till att lämna interuptsignal INT2* .
Denna signal plus INT,INT5* prioriteras i pal16L8 (5D) som sedan blir interuptsignal I till cpu:n.
Chip select (CIO*) kommer från krets 74138 (4E), den skall ligga låg om CIO är enablad.
Signalerna RD (read), WR (write) kommer från PAL16R4 Dom indikerar att CPU läser eller skriver till CIO.
Vid reset ligger båda dessa låga och det tolkar cpu som en intern reset. A1 och A2

12.2 Programbeskrivning

12.2.1 Sjuseg 22

CIO testen testar inte alla möjliga timer eller alla interupter. Testens preliminära uppgift är att se om enable signaler,adressavkodningen fungerar samt att det inte är kortslutningar på databussen.
Vid test av CIO:n används klockan som interface. För att kunna tala med klockan måste CIO fungera. Cio:s port c sätts upp så att kommunikation med klockan är möjlig. Om detta inte fungerar blir ingen tid utskriven på bildskärmen. Först görs enable CIO genom att skriva till "enablekretsen" 74138 (4E) och "0" bit CIO*. . Sedan följer en TIDSVISNING under 8 sekunder .

Om fel uppkommer kan man tex testa följande saker :

1: CIO får inga ENABLEsignaler

Åtgärder: mät med instrument på enable CE (ben 36),det skall gå lågt om avkodningen är okey,om inte får man nysta sig bakåt därifrån till en felaktig krets.
Kolla även RW,RD (ben 5,6) samt adressledningarna A1,A2.

2: CIO felaktig

Åtgärder: mät på port c .På ben 19 skall det ligga en klocksignal ut till cmosklockan. Om inte är det troligen fel på CIO

3:Klockan trasig

Åtgärd: Kolla om klockan får någon extern klockfrekvens från kristallen med ett mätinstrument,samt att batteri-backupen fungerar (den kan vara urladdad).Om ovan OKEY byt klockkrets.

13 KLOCKTEST

Om man vill testa klockan kan det göras här. Om svaret är "j" på frågan görs en klocktest. Under klocktesten sparas klockan innehåll undan i en spararea för att man senare skall kunna återställa klockan. För att kunna testa klockans alla register initierar man klockans register med År 99, dag i veckan 07, månad 12, dag 31, timmar 23 minuter 59, sekunder 56 som sedan räknas upp var sekund. Om allt står rätt till med klockan kommer den att slå om till 000101000000 (det räknas upp 8sekunder från start).
Om man vill ställa om tiden gör man det efter klocktest varvid man skriver in data i följande ordning YYDDMMDDHHMMSS (den gamla tiden läggs inte tillbaka).
OBS : För att kunna ställa om klockan måste man göra en klocktest först.

14 Floppy

14.1 Teknisk beskrivning

För att kunna köra med floppydriver finns det en floppy-disk-controller FD 1797 + floppy-disk-interface circuit på cpukortet. Dessa kontroller styrs via data bussen som anger vad för kommando som skall göras. Dom kontrollsignalerna som behövs är:FLP* :anger att floppyn skall användas,signalen kommer från krets 74138 (4E) (det är kretsen som gör enable på alla I/O prylar) .För att styra dom olika signalerna till floppy-controllen används två kretsar 74273 ,74174 (7B,8B) . Vilken som skall kopplas in av dessa väljs av signalerna FW1*,FW0* som genereras av adress A0,A7 via demultiplexern 74139 (10F) + (A8-A10 via 14G)(adressen är XXb00), Övre halvan av ett skrivet WORD hamnar till DISK INTERFACE CIRIUIT via 74273 (7B) medans den låga delen hamnar direkt till floppy drivens kabelanslutning via 74179 (8B) och bufferen (9B) . Signalen DRQ indikerar att dataregistret är tomt vid WRITE-operation samt att det ligger data i registret vid READ-operation.

14.2 Programbeskrivning

14.2.1 Sjuseg 23

För att kunna testa floppy-controllen behövs det en formaterad floppyskiva som det går att skriva och läsa ifrån. Testen börjar med att göra RESET av floppykontrollen ,det läggs ut "0000" på data bussen samt signalerna FW0*, FW1* :74139 (10F) visar i vilket av registrena bytena skall hamna i. Nästa word initierar floppyn samt startar floppy motorn. För att kolla att floppy-kontrollen har mottagit initieringen läses statusen . Om kontrollen inte blir READY skrivs det ut på skärmen "floppyn ej ansluten eller fel på kontrollen" ,sedan görs det en ny TEST från början. Om drivern är ready stegar "huvudet" fram till spår 45 ,varefter det görs en kontrolläsning av TRACKregistret (adress xx002). Om trackregistret inte är rätt skrivs det ut på skärm "stegning till fel spår/kontrollen felaktig/floppyn ej formaterad" , sedan görs OMTEST från början. Felaktigt spår kan bero på icke formaterad skiva,felaktig kontroll eller något fel på data bussen. Därefter skrivs sektor 1 in i sektorregistret (adress xxx004).

När ovan är gjort är floppy-kontrollen klar för att ta imot indata som skall skrivas på floppydisken, som sedan skall kontrolläsas. Indata skrives in från tangentbordet . Floppy kontrollen hämtar sedan data från lagrad sträng. Om kontrollen inte gör DATA REQUEST vid WRITE-mod stannar programmet och WATCH-DOGENS lysdiod tänds. Om vi får felkod vid skrivningen skrivs DET ut på skärmen (FELAKTIG SKRIVNING (SE DATA BLAD) samt statusregistret läggs ut på lysdioderna. Om det blir fel vid skrivningen görs det ett nytt försök att skriva (obs om skrivningen inte går och man vill testa från början måste RESET göras på cpu:n).

Vid läsningen skrivs det in först att vi skall läsa sektor 1 som sedan görs. Om floppy kontrollen inte gör datarequest vid READ träder WATCHDOG kontrollen in (lysdioder tänds). Efter READ läses statusen av ,om den är felaktig skrivs detta ut på skärmen "Läsningen fungerar ej riktigt,status på lysdioderna (gör omtest)" , statusregistret läggs ut på lysdioderna . Vid läsfel görs nytt försök att läsa, om vi vill ta testen från början måste det göras RESET på cpu:n.

Den lästa texten från floppydrivern skrivs sedan ut på skärmen ,om denna är samma som inskriven fungerar kontrollen bra.

Testen avslutas med att floppymotorn stannar.

14.3 Testordning vid floppykontroll

1: Inskrivning av text på FLOPPY-disk
2: Läsning på FLOPPY-disk

A: Reset av floppy-kontroll (min 50u)

B: initiering av floppy (motor startar)
klockpuls till krets 8B, d3 går hög på ben 3.

C: Det testas om floppy-drivern är ansluten samt att skiva är isatt.
Om ej skrivs texten "Floppyn ej ansluten eller fel på kontrollen" ut på skärmen.

D: Skrivhuvudet stegar fram till spår 45
Om stegningen inte går rätt skrives det ut på skärmen att "kontrollen felaktig eller floppyn ej formaterad. Programmet stoppar där och reset måste göras för vidare testning.

E: Programmet frågar efter en testtext som skall knappas in. Denna text skrivs ut på floppy-drivern.
Statusregistret läggs ut på LYSDIODERNA om det har blivit fel vid skrivning samt texten "felaktig skrivning (status på lysdioderna" . Sedan görs det omtest.
(se data-blad FD179X-02 "status för skrivning")

Lathund

BIT:
0 kontrollen upptagen
1 inget i skrivregistret
2 tecknet ej översänt till kontrollen
3 CRC ERROR
4 sektorspår ej funnet (skivan ej formaterad)
5 WRITE FAULT "skrivfel"
6 SKRIVSKYDDAT
7 DRIVEN inte klar "READY"

F: Om det har blivit något fel vid läsning skrivs STATUS ut på lysdioderna samt texten "LÄSNING FUNGERAR EJ RIKTIGT,STATUS PÅ LYSDIODERNA" annars texten "om skrivning/läsning fungerar skrivs texten ut på skärmen".
Om det är samma text som skrivs ut fungerar kontrollen "troligen" bra.
Med hjälp av STATUS bitarna kan man dra slutsatser om vad som är fel vid läsning. (se datablad ,status för läsning)

15 DMA

15.1 Teknisk beskrivning

DMA används för att överföra data snabbt mellan två st enheter. Om det gäller intern överföring av data inom cpu skall SYSFS bit8 i SPECIAL CONTROLL REGISTRET (4E) sättas till "1". För att ange vilka block överföringen skall gå mellan måste det skrivas i DMAMAPEN innan någon överföring kan ske. Dmamappen består av 4st kretsar (1G,2G,10G,1E). För att kunna skriva in block adresserna i dmamappen genereras signalen DMP* (indikerar att vi har gjort chip select på (1G och 2G)), denna signal kommer ifrån dekodern 74138 (14G) ,adresserna ledningarna A8-A10 och X11-X12 genererar stroben. När data skrivs in i DMAMAPPEN genereras stroben GDS* :pal16R4 (12E)
När FLOPPY vill sända över data via DMA skickas signalen DRQ* ut från floppy-kontrollen som tas emot i krets 74158 (6D) som sedan genererar signalen RDY* (dma ready) till DMA0. Dma:n lägger då ut signalen BRQ* (buss request).Om signalen DMADIS (specialkontrollregistret (4E)) är "1" genererar BR* (buss request) som indikerar att DMA vill ha bussarna. När bussarna är lediga sänder cpu tillbaka BG* (buss grant ack). Signalen BGACK* (buss grant ack) genereras när alla bussar är släppta och DMA kan starta sin överföring. Signalen genereras av 7410 (18C) som grindar ihop signalerna BG*, DMADIS och IOC* (visar att bussarna är helt släppta av CPU) . Denna signal går till BAI på DMA som startar överföringen (om den är initierad för dataöverföring).
Signalen DMAOK* kommer från 7410 (18C).Generering av DMAOK* ,signalen indikerar att det bara är dmamappen som adresserar på adressbussen (maccen har släpp bussen helt och hållet) fås från maccen och vippan (10D),innan det genereras någon BGACK* måste maccen vara frånkopplad från bussen.

15.2 Programbeskrivning

15.2.1 Sjuseg 24

DMA0 testas genom att man läser från floppy till ett raderat minnesområde. Detta område kontrolläses efter överförandet för att kolla att DMA har sänt över rätt information på rätt adress. Sedan kontrolläses övriga minnesområden så att ingen feladressering har skett av DMA0. Vid överföringen skrivs floppykontrollen och DMA status register ut på ljusdioderna om det blir något fel. Ordningen vid testandet:

A: Resetar skrivarean för DMA överföringen
B: Skriver in data till FLOPPY (spår NOLL).Detta förutsätter att floppytesten fungerar.

15.3 Testordning vid DMA

A: Resetar skrivarean för DMA-överföringen
B: Skriver in data till FLOPPY (spår NOLL).Detta förutsätter att floppytesten fungerar.
C: BIT "sysfs" i kontrollregistret sätts till "1" (krets 13E,pinne 12)
D: Det skrivs i DMA-mappen (kretsarna 1G,2G,10G,1E) Koll tex WE-signalen ben 3 om okey
E: DMA resetas 5gånger .Koll tex CE/W signalen på DAM0 5G pinne 16.
F: Signalen "DMADIS" "1" ställes i controllregistret. koll tex krets 13E pinne 10 "1"
G: DMA enablas
H: Floppy drivern startas .Genererar RDY SIGNAL till DMA .
koll tex VIKTIG pinne 25 DMA0 skall gå låg.
I: När RDY signalen går låg skall BRQ signalen på DMA0 gå låg. Och generera signalen BR* (buss reqest) om DMADIS* är "1". Om allt är okey skall CPU:n låta BG* (buss grant) gå låg. Om nu alla bussar är släppta (indikeras av IOC*) skall signalen BGACK* genereras.
Koll tex VIKTIGT pinne 15 DMA0 skall gå låg. OM inte, fungerar ej DMA0 "busrequest" .Någon överföring kan ej göras då.
J: Status från floppy och DMA skrivs ut om det har blivit fel vid överföringen. Med hjälp av dessa ledtrådar kan man lista ut vissa fel. Status för floppy skall vara helt släckt. Status för DMA 11011000 om det hela är okey.
Vid andra felsekvenser får man titta i datablad floppy:sid104. Dma sid33 i ZILOG

LATHUND:

FLOPPY:		DMA:	
	0:BUSY		0:DMA TRANSFER
	1:DATA REQEST		1:READY BONE
	2:LOST DATA		2:X
	3:CRC ERROR		3:INTERRUPT
	4:RECORD NOT FOUND		4:MATCH FOUND
	5:WRITE FAULT		5:END OF BLOCK
	6:WRITE PROTECT		6:X
	7:NOT READY		7:X

K: Om DMA inte har gjort någon överföring skrivs ett felmeddelande ut på skärmen. Sedan testas om end of blockflaggan är satt ;om inte skrivs felmeddelande ut .DET GÖRS SEDAN OMTEST
L: Överfört mönster kollas. Om felaktigt skrivs detta ut på skärmen :DET GÖRS SEDAN OMTEST M: Sedan kollas om det har skett någon kopiering till angrensande minnesområden. Vid felaktig överföring skrivs detta ut på skärmen. Kan vara något fel på adressbussen. DET GÖRS SEDAN OMTEST

16 Busskorts test

16.1 Teknisk beskrivning

TEST av BUSS-kortet görs för att kunna kommunisera med Winchester .Den kanalen är viktig för att kunna ladda in andra testprogram och köra igång operativ systemet. För att kunna testa lite av busskorten måste man bestycka det med ett SASI -interface. För att CPU skall veta var interfacet är inkopplat genereras signalerna CSA* och CSP* från pal16R4 (11E) .Dessa signaler klockar ut signalen Cs från buss kontakterna. För att kunna läsa av var interface kortet sitter genereras stroben RCSB* :74139 (10F). Denna signal klockar in CSB* signalerna från dom olika busskontakterna, det görs via kretsarna 74244 (4A) (styrs av signaln RCSB*) och kretsen 74373 (5A) som får sin enable från signalen IORQ* : pal 16R6 (11E)). Signalen BUS0 visar om det är buss 0 eller buss 1,2 som används. När vi skall skriva till buss korten görs det via trancivrarna 74245 (1B och 6B). Dessa styrs av signalerna DIR (riknning på data) och E12* ,E0* (enable buss 1,2 eller 0) :pal16R4 (12E)

16.2 Programbeskrivning

16.2.1 Sjuseg 25

Testen börjar med att generera signalerna CSA* och CSP* :pal16R4 (12E) som klockar ut stroben CS på busskontakterna.
CSB* signalerna "latchas till 74373 (5A) av signalen IORQ* :pal16R6 (11E).För att läsa av CSB* signalerna genereras signalen RCSB* :74139 (10F) som klockar in status på data bussen. På lysdioderna kan man se avläst status från CSB* signalerna samt läsa var SASI-interfacet sitter på skärmen . Om CPU inte hittar rätt kort skriv detta ut på skärm. Om det inte avkodas rätt CSB* signal skrivs "fel på kortval" på skärmen, sedan görs det omtest från början.

```
BUSS0:          10000000
BUSS 0 EXT:     01000000
BUSS1:          00000010
BUSS2:          00000001

BUSS EXTERN: 00xxxx00     Det skall bara tändas en dom

Ovan skall gälla om kortval är rätt.
```

17 Strobe decoder

17.1 Teknisk beskrivning

VIKTIGA kretsar för att få kontakt med yttre enheter via busskortet är 7A och 6A (BUSS 0) ,2A och 1A (BUSS1 och BUSS2).Dessa kretsar genererar strobepulser via SASI-interfacet till XEBEC-kontrollen som sedan svarar på vissa kommandon via busskortet.
Vid reset av cpu-kortet görs det automatiskt reset av kontrollen (RST*), vid reset mjukvarumässigt genereras det en strobe C3* (74138 (2A)) under minst 100ns. Val av en viss kontroll görs genom att "latcha" in ett databyte som väljer vilken kontroll som är kopplad till bussen (kan vara mer en i samma busskontakt), detta byte latchas in med hjälp av stroben OUT* (74138 (2A).
Det valda kortet selectas sedan med strobe C1*. Vilken del av dekoderarna (1A,2A,7A,6A) som genererar stroben styrs av signalen DIR W/R*, DIR. När statusen skall läsas från kontrollen genereras stroben STAT*, mostvarande signal INP* när vi skall läsa DATA. Vid överföringen av DATA till kontrollen måste den först latchas in med hjälp av stroben OUT*

17.2 Programbeskrivning

17.2.1 Sjuseg 25

För att kunna kolla en del av STROBE-DECODERNS signaler samt BUSSKORTETS adress- och data-bussar görs det en test via ett SASI interface + XEBEC disk controller (XEBEC S1410) . Den test som görs är DRIVE TEST READY, om testen ger ERROR tillbaka skrivs error-bytet ut (innehåller error-code och error-type) på skärmen och lysdioderna. Under testens gång genereras det olika strobe-pulser från strobedetektor (busskortet) som går att mäta på. Testprogrammet börjar med att göra ett mjukvarureset genom att generera en strobe på C3* :74138 (2A) under en viss tid (3ms). I nästa steg kollas det om kontrollen är ledig, detta görs genom att lägga ut en strobe STAT* från strobe dekodern. Om det avlästa bytets bit d2 är "0" är kontrollen ledig.
Om kontrollen är upptagen genereras det en ny strobe.
Om kontrollen inte blir ledig efter en viss tid skrivs det på skärmen att "kontrollen vill inte bli ledig ,status på lysdioderna" . Sedan görs det omtest från början. Genom STATUSEN på lysdioderna ges det en ledning om det finns fel på DATA-bussen (förutsätter att kontrollen fungerar samt att stroben STAT* genereras).

Om kontrollen blir ledig "latchas" /01 (via data bussen) in till kontrollen med stroben OUT* :74138 (2A) (detta databyte anger vilken kontroll vi vill arbeta imot). Nästa steg blir att göra "select" av utvald controller,det görs genom att stroben C1* genereras (går att titta på). Denna strobe skall göra att controllen blir aktiv ,detta kollas genom att stroben STAT* genereras och data bussen läses av. När kontrollen har blivit "aktiv" skickas kommandot "test drive ready". Kommandot sänds över när kontrollen vill ha "commando". Det kollas med en STAT* strobe ,data bytet vi får tillbaka skall ha bitarna d0:"1", d2:"1", d3:"1" satta (se data blad).
Om kontrollen vill ha "commando" sänds det över sex st COMMANDO-BYTE med ovanstående test mellan varje byte.
Om kontrollen inte vill ha kommando efter en viss tid skrivs det ut på skärmen "controllen tar inte kommando,status på lysdioderna". Med hjälp av denna information kan man avgöra fel ,tex på databussen.
Efter översänt "commando" skickar kontrollen tillbaka ett statusbyte som indikerar om "driven är ready".
Om inte går testprogrammet ut och hämtar 4st fel-BYTE. Hämtningen går till på samma sätt som när kommandot "test drive ready" gjordes (pulserna följer samma mönster). Det första felbytet som anger ERROR-CODE och ERROR-TYPE skrivs ut på lysdioderna och skärm. Sedan görs en omtest från början. Om kontrollen inte vill sända över ERRORBYTET efter en viss tid skrivs det ut på skärmen "kontrollen vill inte sända status,status på lysdioderna" varefter det görs en omtest från början.

18 NVRAM

Teknisk beskrivning:

NVRAMET används till att lagra vissa viktiga parametrar som används till att starta programmen med. Dess storlek är 16x16 bitar. Nvramet är kopplat till port c på CIO:n. Det får batteribackup från batteriet som laddas automatisk när cpu-kortet har spänning.

18.2 Programbeskrivning

Sjuseg 26

Det som står i NVRAMMET förstörs när det testas.
När testen börjar skrivs det ut TESTNING.....
på skärmen. Om NVRAMMET fungerar skrivs NVRAM OKEY på
bildskärmen.
Om ej så kan man kolla att NV-RAMet får CHIP-SELECT
("1") samt att det klockas.Det kan även vara fel på
CIO i detta fall.

20 Sjusegtabell

1	Bittest segram	
2	Adresstest segram	
3	Segmentram transparent	
4	Bittest pageram	
5	Adresstest pageram	
6	Paritetsflaggan sätts	
7	Primärminnet testas för stack (32K)	
8	Darten testas och meny skrivs ut	
9	Primärminne 32K-256K	bit
10		adress
11		paritets
12	Primärminne 256K-512K	bit
13		adress
14		paritets
15	Primärminne 512K-768K	bit
16		adress
17		paritets
18	Primärminne 768-1M	bit
19		adress
20		parites
21	Paritetskontrollkretsarna	
22	CIO+KLOCKA	
23	FLOPPY	
24	DMA	
25	test av BUSSKORT (val av busskontakt)	
26	test av NVRAM	

21 Teknisk beskrivning av testlådan

Testlådan skall kopplas till VIDEO-kontakterna på cpu-kortet.Spänningsmatningen till testlådan sker genom bankabeln från videokontakterna (så någon extern spänningsmatning behövs ej). Lådans funktion bygger på att man avkodar dom TIO lägsta adresser på adressbussen från cpu-kortet. Att avkodningen görs på dom 10 lägsta adresserna och inte på dom högsta beror på att X11-X20 inte är tillgängliga innan MACCEN har blivit testad och uppsatts. Avkodningen bygger på att testlådan skall ha FYRA SKRIVNINGAR AV SAMMA ADRESS EFTER VARANDRA för att enablas. Styrningen av SJUSEGMENTEN sker på adress 380 och utskriften på LYSDIODERNA på adress 300. Klockfrekvensen till CPU-kortet genereras från en kristall som sitter i testlådan ,så någon extern kristall behövs ej anslutas.
Man kan välja mellan att köra TESTPROGRAM eller BOOT-PROMMET med en omkopplare. Om man kör bootprommet bryts PACK* signalen från testlådan. Vid start av testprogrammet eller boot-prommet görs det RESET från testlådan. Om CPU INTE kommer igång inom 2 sekunder eller testen hakar upp sig någon stanns i programmet träder WATCHDOGEN in och gör än ny reset av cpu:n. För att indikera detta finns det en lysdiod (märkt watchdog) på testlådan som tänds vid fel (den går att släcka genom att göra reset igen). Var programmet har hakat upp sig kan utläsas på SJUSEGMENTEN (anger var i testen man är)

```
        .text


DIODLATCH       =       /0280
SEGCLOCK        =       /0380
SYSSP           =       /60800
WDOG            =       /80007
BURK            =       /20000          öreset av burk

SEGRAM          =       /80003
PROCSTART       =       /40
TASKREG         =       /80005
SEGSTART        =       /80003
STARTBIT        =       /01

PAGESTART       =       /80000
SPCTROL         =       /7fe00
PARIT           =       /7fe00
IOSTART         =       /fe000
BURK            =       /7ff00

BUS             =       /40008          ö resetvektor 512K
ö-----------------------------------------------------------------
WR0A            =       /7f204          ö kontrollport A
WR0B            =       /7f200          ö kontrollport B
DARTA           =       /7f206          ö ut/in port A
DARTB           =       /7f202          ö ut/in port B
ö-----------------------------------------------------------------
CIC             =       /7f700          öcio port c
CIB             =       /7f702          öcio port b
CIA             =       /7f704          öcio port a
CIK             =       /7f706          öcio kontrollport
ö-----------------------------------------------------------------
FLOPCMD         =       /7f000          öcommandoreg
FLOPSTA         =       /7f000          östatusreg
FLOPTRACK       =       /7f002          öspårregistret
FLOPSEK         =       /7f004          ösektorregister
FLOPDATA        =       /7f006          öDATA register

FLOPCTR         =       /7fb00          öfloppy TTl control register
ö-----------------------------------------------------------------
DMAMAP0         =       /7fd06
DMAMAP1         =       /7fd04
DMAMAP2         =       /7fd00

DMA0            =       /7f300
DMA1            =       /7f400
DMA2            =       /7f500

ö-----------------------------------------------------------------
BUSSSTA         =       /7eca0          övar korten sitter

ERRDAT          =       /7e4a0          ögenererar INP          (W)
COMDAT          =       /7e4a0          ögenererar INP          (W)
SELCTRL         =       /7e4a4          öSELECT PULS C1*        (W)
SELSTROB        =       /7e4a4          ögenererar selectstrobe(W)

DATASTROBE      =       /7e4a0          ö LATCH till controllen

CARDSTA         =       /7e4a2          östatus på controllen   (R)

CONRESET        =       /7e4a8          ögenererar C3* puls för reset
```

```
ö-------------------------------------------------------------------
EPROM1          =          /180000        öEXTRA EPROM (add efter mac)
EPROM2          =          /1a0000
EPROM3          =          /1c0000
EPROM4          =          /1E0000
ö-------------------------------------------------------------------


IN              =          /60000         ötangenbords text (lagring)
REF             =          /60080         öreferens till klocka
MODE            =          /60081         övad som skall testas
LOOP            =          /60082         öloopräknare
REL             =          /60084

OPCODE          =          /60094         öopcode till NVRAM
DATA            =          /60090    ödata  skrivas in i NVRAM
REGI            =          /60088         övilket register NVRAM

TRAC            =          /60088         öSPÅR FLOPPY
SEK             =          /60090         öSEKTOR FLOPPY
BYTE            =          /60094
SAVE            =          /60100         öspar AREA

MEMTEST         =          /60090         övisar att det är minnes test


org:
.long RESET,SYSSP,HIGH,QW,QW1,QW2,QW3,QW4,QW5,QW6,QW7
.long QW8,QW9,QW10,QW11,QW12,QW13,QW14,QW15,QW16,QW17,QW18,QW19,QW20
.long QW21,QW22,QW23,QW24,QW25,QW26,QW27,QW28,QW29


        ö*******************************************************************
        ö*******************************************************************


QW: movb #3,d3
 jsr DIOD1
 movl ADRINT,a1
 jsr OUTPUT                          öadressinterupt
 jmp TRAP
QW1: movb #4,d3
 jsr DIOD1
 jmp TRAP
QW2: movb #5,d3
 jsr DIOD1
 jmp TRAP
QW3: movb #6,d3
 jsr DIOD1
 jmp TRAP
QW4: movb #7,d3
 jsr DIOD1
 jmp TRAP
QW5: movb #8,d3
 jsr DIOD1
 jmp TRAP
QW6: movb #9,d3
 jsr DIOD1
 jmp TRAP
QW7: movb #10,d3
 jsr DIOD1
 jmp TRAP
```

```
QW8: movb #11,d3
 jsr DIOD1
 jmp TRAP
QW9: movb #12,d3
 jsr DIOD1
 jmp TRAP
QW10: movb #13,d3
 jsr DIOD1
 jmp TRAP
QW11: movb #14,d3
 jsr DIOD1
 jmp TRAP
QW12: movb #15,d3
 jsr DIOD1
 jmp TRAP
QW13: movb #16,d3
 jsr DIOD1
 jmp TRAP
QW14: movb #17,d3
 jsr DIOD1
 jmp TRAP
QW15: movb #18,d3
 jsr DIOD1
 jmp TRAP
QW16: movb #19,d3
 jsr DIOD1
 jmp TRAP
QW17: movb #20,d3
 jsr DIOD1
 jmp TRAP
QW18: movb #21,d3
 jsr DIOD1
 jmp TRAP
QW19: movb #22,d3
 jsr DIOD1
 jmp TRAP
QW20: movb #23,d3
 jsr DIOD1
 jmp TRAP
QW21: movb #24,d3
 jsr DIOD1
 jmp TRAP
QW22: movb #25,d3
 jsr DIOD1
 jmp TRAP
QW23: movb #26,d3
 jsr DIOD1
 movl #CIOINT,a1
 jsr OUTPUT                    ö CIOINTERRUPT
 jmp TRAP
QW24: movb #27,d3
 jsr DIOD1
 jmp TRAP
QW25: movb #28,d3
 jsr DIOD1
 jmp TRAP
QW26: movb #29,d3
 jsr DIOD1
 movl #DARTINT,a1
 jsr OUTPUT                    öDART ELLER SCC INT
 jmp TRAP
QW27: movb #30,d3
 jsr DIOD1
 jmp TRAP
```

```
QW28: movb #31,d3
 jsr DIOD1
 jsr NMI                öparitetsfel
 jmp TRAP
QW29: movb #32,d3
 jsr DIOD1
 jmp TRAP


TRAP:   movl #VEKTOR,a1
 jsr OUTPUT
 jmp stopp



      ö**************************************************************
      ö**************************************************************

         ö++++++++++++++++++++++++++++++++++++++++++++++
         ö+                  SUBRUTIN                  +.
         ö+  lysdioder. indata i d3.                   +.
         ö++++++++++++++++++++++++++++++++++++++++++++++.

DIOD:    clrl   d0
DIO:     addql  #1,d0
         movb   d3,DIODLATCH            ö lysdioderna .
         cmpl   #4,d0
         bne    DIO
         jmp    a6@


DIOD1:   clrl   d0
DIO1:    addql  #1,d0
         movb   d3,DIODLATCH            ö lysdioderna .
         cmpl   #4,d0
         bne    DIO1
         rts
      ö**************************************************************
           ö++++++++++++++++++++++++++++++++++++++++.
           ö+              SUBRUTIN                +.
           ö+  sjusegment räknas upp med ett.      +.
           ö++++++++++++++++++++++++++++++++++++++++.

SJUSEG:  clrl   d0
SJU22:   addql   #1,d0
         movb   #/00,SEGCLOCK        öräknar upp sjusegment.
         cmpl   #4,d0
         bne    SJU22
         jmp    a6@

      ö**************************************************************
           ö++++++++++++++++++++++++++++++++++++++++.
           ö+              SUBRUTIN                +.
           ö+  sjusegment räknas upp med d1 gånger +.
           ö++++++++++++++++++++++++++++++++++++++++.

SJUSEG1: clrl   d2
SJU2:    clrl   d0
SJU1:    addql   #1,d0
         movb   #/00,SEGCLOCK        öräknar upp sjusegment.
         cmpl   #4,d0
         bne    SJU1
         addl #1,d2
```

```
            cmpl d1,d2
            bne SJU2
            rts

        ö*************************************************************
        ö*************************************************************
              ö++++++++++++++++++++++++++++++++++++++++.
              ö+                SUBRUTIN                +.
              ö+     delay rutin  .indata d1            +.
              ö+  ungefär 15mikrosek per enhet          +.
              ö++++++++++++++++++++++++++++++++++++++++.

DELAY:      clrl    d2
DEL:        addql   #1,d2
            movb    WDOG,d7          ökick DOG
            cmpl    d1 ,d2
            bne     DEL
            jmp a6@

DELAY1:     clrl    d2
DEL1:       addql   #1,d2
            movb    WDOG,d7          ökick DOG
            cmpl    d1 ,d2
            bne     DEL1
            rts


        ö*************************************************************
        ö*************************************************************


SEGDAFEL:       movl #H3,a6
                movb d3,d5
                movb d3,d4
                clrl d3
                andb #/0f,d4
                cmpb #0,d4
                beq H4
                orb #/01,d3
H4:             andb #/f0,d5
                cmpb #0,d5
                beq H5
                orb #/02,d3
H5:             jmp DIOD
H3:             jmp AC                   ö testar på nytt (stopp)


        ö*************************************************************
        ö*************************************************************


SEGADFEL:   movb #/ff,d3             öåtta lysdioder som indekerar
            movl #FELADD2,a6         öfel på adressbussen
            jmp DIOD
FELADD2:    jmp AB                   ötestar på nytt  (stopp)

        ö*************************************************************
        ö*************************************************************


SEGRES:     movb #/40,TASKREG
            movl #/40,d2                        öprocess ETT
            movl #SEGSTART,a0                   östart adress segram
A11:        movb #/00,a0@
```

```
                addl #/100,a0
                movb #/00,a0@
                addl #/7f00,a0
                cmpl #/100003,a0                ösista adressen
                bne  A11                        önästa process
                cmpb #/4f,d2                    öom process 15
                beq  A2
                addb #1,d2                      önästa process
                movb d2,TASKREG
                movl #SEGSTART,a0
                movb WDOG,d7                    ökick watchdog
                jmp A11
A2:             jmp a6@

        ö***************************************************************
        ö***************************************************************


SEGDATA:        movb   WDOG,d7                  ökick DOG
                clrl d3                         öfel register
                movb #/01,d4                    östart mönstret
B1:             movb #PROCSTART,d0              öprocess NOLL
B3:             movl #SEGSTART,a0
                movb d0,TASKREG                 öprocess x
                movb d4,d2                      ötestmönster
B0:             movb d2,a0@
                addl #/100,a0
                movb d2,a0@
                addl #/7f00,a0
                cmpl #/100003,a0                öom sista adressen
                bne  B0
                addb #1,d0
                movb   WDOG,d7                  ökick DOG
                cmpb #/50,d0                    öom sista processen
                bne B3                          öMAGICbiten ej satt

        ö---------------------------------------------------------------ö
        ö testar att det är inskrivet rätt på alla bitar ,i alla processer.  ö
        ö---------------------------------------------------------------ö

test1:          movb   WDOG,d7                  ökick DOG
                movb #/40,d0                    öprocess noll
                movl #SEGSTART,a0               östart adress
B2:             movb a0@,d2                     öhämtar
                eorb d4,d2                      öom någon felaktig bit
                orb d2,d3                       öläggs den i d3
                addl #/100,a0
                movb a0@,d2
                eorb d4,d2                      öom rätt bit
                orb d2,d3
                addl#/7f00,a0
                cmpl #/100003,a0                öom sista adressen
                bne B2
                movb   WDOG,d7                  ökick DOG
                lslb #1,d4                      öskiftar testmönstret ett HACK
                cmpb #/80,d4                    öom alla bitar är testade
                bne B1                          öMAGICbiten testas ej här
                cmpb #0,d3
                bne SEGDAFEL                    öom bitfel

                jmp a5@

        ö***************************************************************
        ö***************************************************************
```

```
SEGADD:
            movb WDOG,d7              ökick watchdog
            movb #/40,TASKREG         öprocess NOLL
            movb #/7f,d1
            movb d1,a0@               ölägger ut testmönster
            movl #SEGSTART,a1         ötestar alla andra adresser
A6:         cmpl a1,a0                öså ingen kopiering har skett
            beq A4                    öi process NOLL
A7:         movb a1@,d2
            andb #/7f,d2              ömaskar av magic biten
            cmpb #0,d2
            bne SEGADFEL              öadressbussen troligen felaktig
A4:         addl #/100,a1
            cmpl a1,a0
            beq A5
            movb a1@,d2
            andb #/7f,d2              ömaskar av magic biten
            cmpb #0,d2
            bne SEGADFEL              öadressbussen troligen felaktig
A5:         addl #/7f00,a1
            cmpl #/100003,a1          ösista adressen i processen NOLL
            beq  A50
            cmpl a1,a0
            beq A4
            jmp A7
ö-------------------------------------------------------------------ö
ö testar i övriga processer på adress a0 att inte /7f är skrivet   ö
ö processnummret pekas ut av d0                                    ö
ö adressen pekas ut av a0                                          ö
ö-------------------------------------------------------------------ö

A50:        movb WDOG,d7              ökick watchdog
            movb #/41,d4              öprocess ETT
A9:         movb d4,TASKREG
            movb a0@,d5
            cmpb #0,d5
            bne SEGADFEL              öinskriven data i fel process
            addb #1,d4
            cmpb #/50,d4              öom process 15 är testad
            bne A9
            movb #/40,TASKREG
            movb #/00 ,a0@
A91:        jmp a6@

      ö*******************************************************************
      ö*******************************************************************

ö+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SEGRAMM:    movl #AC,a6               ösjusegment ETT
            jmp SJUSEG                ösegmentramets bittestas
ö+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ö-----------------------------------------------------ö
ö                  bittestar segram                   ö
ö-----------------------------------------------------ö

AC:         movl #G8,a5
            jmp SEGDATA             ötestar bitmässigt i segram

G8:         movl #B11,a6
            jmp SEGRES              öresetar segram för adresstest


ö+++++++++++++++++++++++++++++++++++++++++++
```

```
B11:        movl #AB,a6             ösjusegment TVÅ
            jmp SJUSEG              ösegramets adressbuss skall testas
ö++++++++++++++++++++++++++++++++++++++++++++
ö--------------------------------------------ö
ö              testar adressbuss  segram     ö
ö--------------------------------------------ö
AB:         movl #SEGSTART,a0
A12:        movl #A23,a6
            jmp SEGADD              ötestar adressbuss ,segram
A23:        addl #/100,a0           öA8
            movl #A10,a6
            jmp SEGADD
A10:        addl #/7f00,a0          öA15-A18
            cmpl #/100003,a0         ösista adressen
            bne A12

ö++++++++++++++++++++++++++++++++++++++++++++
            movl #B14,a6            ösjusegment TRE
            jmp SJUSEG              ösegramets görs TRANSPARENT
ö++++++++++++++++++++++++++++++++++++++++++++

B14:        movl #B12,a6
            jmp SEGTRAN             ögör segmentrammet transparent

B12:        jmp a7@

      ö*****************************************************************
      ö*****************************************************************


SEGTRAN:    movb #/40,TASKREG       öprocess NOLL
            movl #/80003,a0
            clrl d2                 östart adress
C2:         movb d2,a0@             öskriver in det i segram
            addb #1,d2
            movb WDOG,d7            ökick watchdog
            addl #/8000,a0          önästa adress i SEGRAM
            cmpl #/100003,a0        öom sista adress A19=0 (A8=0)
            bne  C2
            jmp a6@

      ö*****************************************************************
      ö*****************************************************************

PAGERES:    movb #/00,d2                 östart segram adress 80003
E2:         movb d2,SEGSTART             öläggs på första adressen
            movl #PAGESTART,a0           östart pageram
E1:         movw #/0000,a0@              öskriver NOLLOR
            addl #/800,a0
            cmpl #/88000,a0              ösista adressen skriven i pageram
          ö  15sidor
            bne E1
            addb #1,d2                   önästa adress från segram
            cmpb #/40,d2                 öom sista adressen i segram
                 ö64st block
            bne E2
            movb WDOG,d7                 ökick watchdog
            jmp a5@

      ö*****************************************************************
      ö*****************************************************************

PAGEADFEL:   movl #H1,a6
             movb #/ff,d3
```

```
            jmp DIOD
H1:         jmp C8                  ögör ett nytt försök (stopp)

      ö*****************************************************************
      ö*****************************************************************

PAGEDAFEL:  clrl d6
            movw d3,d5
            andw #/000f,d5
            cmpw #0,d5
            beq H6
            orb #/01,d6             öfel i pageram 18G
H6:         movw d3,d5
            andw #/00f0,d5
            cmpw #0,d5
            beq H7
            orb #/02,d6             öfel i pageram 22G
H7:         movw d3,d5
            andw #/0f00,d5
            cmpw #0,d5
            beq H9
            orb #/04,d6             öfel i pageram 20G
H9:         movw d6,d3
            movl #H2,a6
            jmp DIOD
H2:         jmp C10                 ögör ny test (stopp )

      ö*****************************************************************
      ö*****************************************************************

PAGEDATA:   movb WDOG,d7            ökick watchdog
            clrl d3
            movw #/0001,d5          ö/0200???  starttestord
G4:         movb #/00,d4
G2:         movb d4,SEGSTART        östartadress till segram
            movl #PAGESTART,a0      östartadress PAGERAM
G1:         movw d5,a0@
            addl #/800,a0
            cmpl #/88000,a0         ösista adressen
            bne G1
            movb WDOG,d7            ökick watchdog
            addb #1,d4
            cmpb #/40,d4            öom sista segadress
            bne G2
            movb WDOG,d7            ökick watchdog

            movb #/00,d4            ötestar att rätt bitfel
G10:        movb d4,SEGSTART
            movl #PAGESTART,a0
G3:         movw a0@,d6
            eorw d5,d6              öger felaktiga bitar
            orw d6,d3               öläggs i d3
            addl #/800,a0
            cmpl #/88000,a0         ösista adressen
            bne G3
            movb WDOG,d7            ökick watchdog
            addb #1,d4
            cmpb #/40,d4
            bne G10
            rorw #1,d5              öroterar vänster av testmönstret
            cmpw #/4000,d5
            bne G4                  önästa testmönster
            andw #/c3ff,d3
            cmpw #0,d3
```

```
            bne PAGEDAFEL              öom bitfel
            jmp a6@
        ö*************************************************************
        ö*************************************************************

PAGEADD:    movb WDOG,d7               ökick watchdog
            movb #/00,d4
F5:         movb d4,SEGSTART           östartadress till segram
            movw #/c3ff,d5             ötestord
F3:         movl #PAGESTART,a0         östartadress PAGERAM
F6:         movw d5,a0@
            movl #PAGESTART,a1         ötest i övriga adresser
F2:         cmpl a1,a0
            beq F1
            movw a1@,d6                öläser minnesinnehållet
            andw #/c3ff,d6
            movb WDOG,d7               ökick watchdog
            cmpw #/0000,d6             öskall inte stå nått
            bne  PAGEADFEL
F1:         addl #/800,a1
            cmpl #/88000,a1            ösista adressen
            bne F2
            movb WDOG,d7               ökick watchdog
            movw #/0000,a0@
            addl #/800,a0              önästa adress i pageramet
            cmpl #/88000,a0
            bne F6
            movb #/00,d6               ötest på a0 i alla segadresser
F8:         movb d6,SEGSTART           öförsta segadress
            cmpb d4,d6
            beq F7
            movw a0@,d2
            andw #/c3ff,d2
            cmpw #/0000,d2
            bne PAGEADFEL
            movb WDOG,d7               ökick watchdog
F7:         addb #1,d6
            cmpb #/40,d6
            bne F8

F9:         movb WDOG,d7               ökick watchdog
            addb #1,d4
            cmpb #/40,d4               öom sista adress i segram
                    ö64st block
            bne F5                     öny testadress


            jmp a5@
        ö*************************************************************
          ö+++++++++++++++++++++++++++++++++++++++.
          ö+          SUBRUTIN PAGERAM           +.
          ö+ testar pagerammet                   +.
          ö+++++++++++++++++++++++++++++++++++++++.

PAGERAM:

ö+++++++++++++++++++++++++++++++++++++++++++
            movl #C16,a6           ösjusegment FYRA
            jmp SJUSEG             öpagerammet bittestas
ö+++++++++++++++++++++++++++++++++++++++++++

C16:        movl #C10,a5
            jmp PAGERES
```

```
ö-------------------------------------------------ö
ö         testar skrivbarheten i pagerammet       ö
ö-------------------------------------------------ö

C10:        movl #C11,a6
            jmp PAGEDATA

C11:       movl #C5,a5
           jmp PAGERES

ö++++++++++++++++++++++++++++++++++++++++++++++++
C5:         movl #C8,a6            ösjusegment FEM
            jmp SJUSEG             öpagerammet adressbuss test
ö++++++++++++++++++++++++++++++++++++++++++++++++
ö-------------------------------------------------ö
ö  testar att adresseringen fungerar samt att ingen    ö
ö feladressering sker                                  ö
ö-------------------------------------------------ö

C8:         movl #C12,a5
            jmp PAGEADD

C12:        jmp a7@


        ö*****************************************************************
        ö*****************************************************************

PAGTRAN:        movb #/40,TASKREG           öprocess NOLL
                movl #/a0000,a0             östart adress pageram
                orl  #/4000,d3              ögör all minnesceller skriv
H11:            movw d3,a0@                 ö och läsbara
                addl #1,d3                  önästa inadress
                addl #/800,a0
                cmpl #/e0000,a0             öom 256K,man räknar upp 128
                bne H11
                jmp a7@
ö-----------------------------------------------------------------------
PAGTRAN1:       movb #/40,TASKREG           öprocess NOLL
                movl #/a0000,a0             östart adress pageram
                orl  #/4000,d3              ögör all minnesceller skriv
H111:           movw d3,a0@                 ö och läsbara
                addl #1,d3                  önästa inadress
                addl #/800,a0
                cmpl #/e0000,a0             öom 256K , räknar upp 128steg
                bne H111
                rts
ö-----------------------------------------------------------------------

PAGTRAN2:       movb #/40,TASKREG           öprocess NOLL
                movl #/a0000,a0             östart adress pageram
                orl  #/4000,d3              öall minnesceller skriv och
H112:           movw d3,a0@                 öläsbara
                movb WDOG,d7                ökick watchdog
                addl #1,d3                  önästa inadress
                addl #/800,a0
                cmpl #/c8000,a0             öom 32K , räknar upp 128steg
                bne H112
                rts

        ö*****************************************************************
        ö*****************************************************************
              ö++++++++++++++++++++++++++++++++++++++++++.
              ö+              PRIMADD                    +.
```

```
                ö+     primärminnet adress testas        +.
                ö++++++++++++++++++++++++++++++++++++++++.

PRIMADD1:       movl #RI3,a1
                jsr OUTPUT                  öom ADRESSTEST SKALL GÖRAS
                jsr INPUT                   ökollar svaret
                movb IN,d1
                cmpb #106,d1                öom j-tangenten
                bne TI79

                movl #TEST,a1
                jsr OUTPUT                     ötestning utföres

                movl #/20000,a0                östart adress primärminne
                clrl d6
PRI111:         clrl d3
T21:            cmpb #9,d3                  öom NIO skrivningar =FEL
                beq PRI911
                addb #1,d3                  öANTAL SKRIVNINGAR
                movl d6,a0@                 ötestmönstret två gånger
                movl d6,a0@
                movl a0@,d5
                movb d1,BURK
                movb WDOG,d7                ökick watchdog

                eorl d6,d5                  ötar fram felaktigt ord
                cmpl #0,d5                 öom inskrivet ord felaktigt
                bne T21                     ökorrekt läsning

T23:            addl #4,a0
                addl #1,d6

                movb WDOG,d7                   ökick watchdog
                cmpl #/60000,a0                öom sista adressen 256K
                bne PRI111                     ölägger ut ett långt word

ö---------------------------------------------------------------
        öläsning av utskrivet mönster

PRI31:          movl #/20000,a0                östart adress primärminne
                clrl d6
PRI53:          clrl d7
PRI51:          movl a0,d5
                cmpb #/280,d5
                beq RR1
                cmpb #/380,d5
                beq RR1
                movl a0@,d3
                addl #1,d7
                cmpl #9,d7
                beq PRI91                      öOM NIO FELLÄSNINGAR HOPP
                eorl d6,d3                     ötar fram felbit
                cmpl #0,d3
                bne PRI51                      öadressfel
RRR:            movb WDOG,d7                   ökick watchdog
RR1:            addl #4,a0
                addl #1,d6
                cmpl #/60000,a0                öom sista adressen 256K
                bne PRI53
                movl #RTT2,a1             öadresstesten klar
                jsr OUTPUT
TI79:           rts
```

```
PRI91:      movl #FEL2,a1              öskriver ut adressfel
            jsr OUTPUT
            jsr ADD                    öskriver  feladress på skärm
            movl #/fff,d1
            jsr DELAY1
            jmp RRR


PRI911:     movl #ADDFEL,a1   ösvårt att skriva
            jsr OUTPUT
            jmp stopp

ö-------------------------------------------------------------------
ö-------------------------------------------------------------------

PRIMADD2:   movl #RI3,a1
            jsr OUTPUT                 öADRESSTEST SKALL GÖRAS
            jsr INPUT                  ökollar svaret
            movb IN,d1
            cmpb #106,d1               öom j-tangenten
            bne TI35

            movl #TEST,a1              öatt vi testar
            jsr OUTPUT

            movl #/20000,a0            öTESTAR DOM 2K-256K ADRESS
            clrl d6                    öförsta k:na
RI111:      clrl d3
TI21:       cmpb #9,d3                 öom 9 skrivningar =FEL
            beq RI911
            addb #1,d3                 öANTAL SKRIVNINGAR
            movl d6,a0@                ötestmönstret två gånger
            movl d6,a0@
            movl a0@,d5
            movb d1,BURK
            movb WDOG,d7               ökick watchdog

            eorl d6,d5                 öfelaktigt ord (ej inskrivet)
            cmpl #0,d5                 öom inskrivet ord felaktigt
            bne TI21                   ökorrekt läsning

            addl #1,d6
            addl #4,a0
            movb WDOG,d7               ökick watchdog
            cmpl #/5f800,a0            öom sista adressen 256K
            bne RI111                  ölägger ut ett långt word


ö-----------------------------------------------------
          öutläsning av inskrivet mönster


RI31:       movl #/20000,a0          östart adress primärminne
            clrl d6
RI53:       clrl d7
RI51:       movl a0,d5
            cmpb #/280,d5
            beq RI96
            cmpb #/380,d5
            beq RI96
            movl a0@,d3
            addl #1,d7
            cmpl #9,d7
            beq RI91                       öOM NIO FELLÄSNINGAR HOPP
```

```
                eorl d6,d3                  ötar fram felbit
                cmpl #0,d3
                bne RI51                    öadressfel
RI94:           movb WDOG,d7                ökick watchdog
RI96:           addl #4,a0
                addl #1,d6
                cmpl #/5f800,a0             öom sista adressen 256K
                bne RI53
                movl #RTT2,a1  öadresstesten klar
                jsr OUTPUT

TI35:           rts

RI91:           movl #FEL2,a1               öskriver ut adressfel
                jsr OUTPUT
                jsr ADD                     öskriver ut feladress på skärm
                movl #/7ffff,d1
                jsr DELAY1
                jmp RI94



RI911:          movl #ADDFEL,a1             ösvårt att skriva
                jsr OUTPUT
                jmp stopp

        ö*************************************************************
        ö*************************************************************
                ö++++++++++++++++++++++++++++++++++++++++
                ö+              PRIMDATA                +
                ö+       bitfel i primärminnet          +
                ö+ vissar felet och står kvar och       +
                ö+ snurrar på felaktig adress.          +
                ö++++++++++++++++++++++++++++++++++++++++

PRIMDATA1:          movl #RI2,a1
                    jsr OUTPUT              öfrågar om BITtest skall göras
                    jsr INPUT               ökollar svaret
                    movb IN,d1
                    cmpb #106,d1            öom j-tangenten
                    bne DA79

                    movl #TEST,a1
                    jsr OUTPUT

                    movl #/01,d4            ötestmönter
                    clrl d3
DA81:               movl #/20000,a0         östart adress 256 första k:na


DA31:               jsr BIT                 ölägger ut testmönster
                    cmpl #/60000,a0
                    bne DA31

DA71:               lslb #1,d4
                    cmpb #/00,d4
                    bne DA81
                    movl #RTT1,a1
                    jsr OUTPUT

DA79:               rts

        ö*************************************************************
        ö*************************************************************
```

```
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+              PRIMRAM                 +.
            ö+ här testar vi primärminnet           +.
            ö+++++++++++++++++++++++++++++++++++++++.


ö+++++++++++++++++++++++++++++++++++++++
PRIMRAM:  movl #PR01,a6       ösjusegment SEX
          jmp SJUSEG          öPARITETSBITEN sätts
ö+++++++++++++++++++++++++++++++++++++++


PR01:     movl #IOSTART,a0    öadress 504K-512K
          movl #/43fc,d3      östartadress FYSISK
P0:       movw d3,a0@         öpageram upsatt för I/O
          addl #1,d3
          addl #/800,a0
          cmpl #/100000,a0
          bne P0

ö+++++++++++++++++++++++++++++++++++++++++++++
ö vi sätter specialkontroll registret
öADRESS 7fe00 med 0c ,det gör att paritetsfel
öej gör  RESET på cpu:n  samt att paritetsbiten
öskrivs med fel paritet (den kan kollas då)
ö+++++++++++++++++++++++++++++++++++++++++++++

          movb #/0c,PARIT     öskriver i paritetsbiten

ö+++++++++++++++++++++++++++++++++++++++
          movl #PR0,a6      ösjusegment SJU
          jmp SJUSEG        öprimärminnet testas för
                            ö STACKEN 2K
ö+++++++++++++++++++++++++++++++++++++++



ö---------------------------------------------------
ötestar 2K av primärminnet och sätter upp stacken
ö logisk adress stack 60800-60000
ö---------------------------------------------------

PR0:      movl #PPR02,a7
          movl #/0000,d3      östart noll i primärminnet
          jmp  PAGTRAN        ögör pageram transparent  första 256K

PPR02:    movl #/01,d4        ötestmönter
          clrl d3
PDA8:     movl #/20000,a0     östart adress 2 första k:na

PDA3:     movw a0,d7          öKOLLAR SÅ ATT VI INTE
    öFÅR SVAR FRÅN BURKEN
          andw #/2ff,d7
          cmpw #/280,d7
          beq PTE60
          cmpw #/380,d7
          beq PTE60

PTE7:     movb d4,a0@         ölägger ut testmönstret
PTE3:     movb a0@,d5
          movb d3,BURK        öFÖR ATT RESETA BURKEN
          movb WDOG,d7        ökick watchdog
```

```
            eorb d4,d5          ötar fram felaktigt ord (ej inskrivet)
            cmpb #/00,d5        öom inskrivet ord felaktigt
            beq PTE4
            movl #PTE,a6

            movb d3,BURK        öFÖR ATT RESETA BURKEN

            movb d5,d3
            jmp DIOD
PTE:        movl #PTE2,a6
            movl #/ffff,d1
            jmp DELAY
PTE2:       movl #PDA3,a6       öom skrivning på ord
            clrl d3
            jmp DIOD
PTE4:       addl #1,a0
            movb WDOG,d7        ökick watchdog
            cmpl #/20801,a0     ötest av 2K primärminne
            blt PDA3

PDA7:       lslb #1,d4
            cmpb #/00,d4
            bne PDA8

            movl #/e0000,a0     ö sätter upp MACCEN
            movl #/4000,d3      öför stacken 2K (60000-60800)
PPR35:      movw d3,a0@
            addl #1,d3
            orl #/4000,d3       öminnet skriv och läsbart
            movb WDOG,d7        ökick watchdog
            addl #/800,a0
            cmpl #/e0800,a0
            bne PPR35

            movl #SYSSP,a7      östart adress för stacken
            jmp a5@

ö++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
öOm farlig adress 280,300,380
ö++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


PTE60:      clrl d3  ötest åtta gånger innan utskrift
PTE70:      addb #1,d3
            cmpb #9,d3
            beq PTE80
            movb d4,a0@
            movb d4,a0@
            movb a0@,d5
            movb d3,BURK        öresetar burk
            eorb d4,d5
            cmpb #0,d5
            bne PTE70
            jmp PTE4            öom det hela är rätt

PTE80:      movb d5,d3
            movl #PTE85,a6
            jmp DIOD
PTE85:      movl #PTE86,a6
            movl #/ffff,d1
            jmp DELAY
PTE86:      movl #PTE90,a6      öom skrivning på ord
            clrl d3
```

```
          jmp DIOD

PTE90:    jmp PTE60              öomförsök


ö-------------------------------------------------------------

 ö+++++++++++++++++++++++++++++++++++++++++++
   öprimärminnet bittestas 2K-256K (1)
 ö+++++++++++++++++++++++++++++++++++++++++++

SJU:      movl #1,d1
          jsr SJUSEG1            ösjuseg 9
          movl #/01,d3
          jsr PAGTRAN1           ösätter upp MACEN för 2K-256K (1)
          movl #/01,d4           ötestmönter
          clrl d3
          movl #RI2,a1
          jsr OUTPUT             öfrågar om BITtest skall göras
          jsr INPUT              ökollar svaret
          movb IN,d1
          cmpb #106,d1           öom j-tangenten
          bne PD23

          movl #TEST,a1
          jsr OUTPUT

PPDA8:    movl #/20000,a0        östart adress 32K och upp till 256K
          movl #/fffe0800,REL    öanger att adressen börjar från 2K
     öffffe8000
PPDA3:    jsr BIT
          cmpl #/5f800,a0        öom sista adressen 254K
          bne PPDA3

          lslb #1,d4
          cmpb #/00,d4
          bne PPDA8
          movb WDOG,d7           ökick watchdog
          movl #RTT1,a1          öminnesarean är bittestad
          jsr OUTPUT


 ö+++++++++++++++++++++++++++++++++++++++++++
          öprimärminnet adresstestas
 ö+++++++++++++++++++++++++++++++++++++++++++
PD23:     movl #1,d1
          jsr SJUSEG1          ösjuseg 10
          jsr PRIMADD2         ötestar adresseringen i primärminnet
          movb WDOG,d7         ökick watchdog

 ö+++++++++++++++++++++++++++++++++++++++++++
          öparitetstest
 ö+++++++++++++++++++++++++++++++++++++++++++

          movl #1,d1
          jsr SJUSEG1          ösjuseg 11
          jsr PARITS           öparitetsbit test

          movl #97,d1
          jsr SJUSEG1          ösjuseg 8 MENY
   bra WE1                 öhoppar till meny ????
          rts
```

```
ö-------------------------------------------------------------------
 ö+++++++++++++++++++++++++++++++++++++++++
NIO:     öprimärminnet  256K ANDRA
 ö+++++++++++++++++++++++++++++++++++++++++

          movl #4,d1
          jsr SJUSEG1          ösjuseg 12
          movl #/80,d3
          jsr PAGTRAN1         ö256K (2)
 ö+++++++++++++++++++++++++++++++++++++++++
          ö 256K (2)
 ö+++++++++++++++++++++++++++++++++++++++++
          movl #/20000,REL     öadressen börjar på 256K
          jsr PRI              öBIT ,ADRESS ,PARITETSTEST
          movl #94,d1
          jsr SJUSEG1          ösjuseg 8 MENY
          rts


ö-------------------------------------------------------------------


ö-----------------------------------------------
ötest om primärminnet är större än 512K
ö-----------------------------------------------
ELVA:     movl #/100,d3             ö sätter upp MACCEN för 256K (3)
          movb WDOG,d7              ökick watchdog
          jsr PAGTRAN1
          movl #/20000,a0
          movb #/ff,a0@             ötestar  att skriva i övre 512K

ö-----------------------------------------------------
öom ej 1M ges buss error här
öprogrammet hoppar till rutinen HIGH
ödär räknas sjusegment upp till 14
öDART testas  sedan
ö-----------------------------------------------------

 ö+++++++++++++++++++++++++++++++++++++++++
          ö 256K (3)
 ö+++++++++++++++++++++++++++++++++++++++++

          movl #7,d1
          jsr SJUSEG1          ösjuseg 14
          movb WDOG,d7         ökick watchdog
          movl #/60000,REL     öanger adress 512K
          jsr PRI              öBIT ,ADRESS ,PARITETSTEST
          movl #91,d1          ösjuseg 8 MENY
          jsr SJUSEG1
          rts


ö-------------------------------------------------------------------

ö-----------------------------------------------
ötest om primärminnet är större än 512K
ö-----------------------------------------------
TRETTON:  movl #/180,d3             ö sätter upp MACCEN för 256K (3)
          movb WDOG,d7              ökick watchdog
          jsr PAGTRAN1
          movl #/20000,a0
          movb #/ff,a0@             ötestar att skriva i övre 512K
```

```
ö+++++++++++++++++++++++++++++++++++++++++++
          ö 256K (4)
ö+++++++++++++++++++++++++++++++++++++++++++
          movl #10,d1
          jsr SJUSEG1          ösjuseg 17
          movl #/180,d3        ö sätter upp MACCEN för 768K (4)
          movb WDOG,d7         ökick watchdog
          jsr PAGTRAN1
          movl #/a0000,REL     öadressen från 768K
          jsr PRI              öBIT ,ADRESS ,PARITETSTEST
          movl #88,d1
          jsr SJUSEG1          ösjuseg 7 MENY
          rts

        ö*************************************************************
        ö*************************************************************
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN PRI              +.
              ö+ testar bit ,adress och paritetsbit  +.
              ö+ i primärminnet för 256K             +.
              ö+++++++++++++++++++++++++++++++++++++++.

ö+++++++++++++++++++++++++++++++++++++++++++
           öbittest 256K
ö+++++++++++++++++++++++++++++++++++++++++++

PRI:      movb WDOG,d7         ökick watchdog
          jsr PRIMDATA1


ö+++++++++++++++++++++++++++++++++++++++++++
            öadresstest 256K
ö+++++++++++++++++++++++++++++++++++++++++++

          movl #1,d1
          jsr SJUSEG1          ösjuseg
          movb WDOG,d7         ökick watchdog
          jsr PRIMADD1         öadresstestar 256K


ö+++++++++++++++++++++++++++++++++++++++++++
          öparitetstest
ö+++++++++++++++++++++++++++++++++++++++++++

          movl #1,d1
          jsr SJUSEG1          ösjuseg
          jsr PARITS           öparitetsbit test
          movb WDOG,d7         ökick watchdog
          rts

        ö*************************************************************
        ö*************************************************************
              ö+++++++++++++++++++++++++++++++++++++++++
              ö+           SUBRUTIN BIT                +
              ö+ Lägger ut bit mönster och testar      +
              ö+ så att det är rätt                    +
       ö+ INDATA d4,a0             +
              ö+++++++++++++++++++++++++++++++++++++++++


BIT:      movl a0,d7             öKOLLAR SÅ ATT VI INTE FÅR
    ö SVAR FRÅN BURKEN
          andw #/03ff,d7         öom farlig adress
          cmpw #/280,d7
```

```
            beq BIT20
            cmpw #/380,d7
            beq BIT20

            movb d4,a0@             ölägger ut testmönstret
            movb a0@,d5
            movb WDOG,d7            ökick watchdog


            eorb d4,d5              ötar fram felaktigt ord (ej inskrivet)
            cmpb #/00,d5            öom inskrivet ord felaktigt
            beq BIT4

BIT22:      jsr ADD                 öskriver ut adress om BITfel
   movb d3,BURK  öresetar burk
            movl #FEL1,a1
            jsr OUTPUT              öskriver ut skriv eller bitfel

            movb d5,d3
            jsr DIOD1               öskriver ut på dioder

            movl #/ffff,d1
            jsr DELAY1
            clrl d3
            jsr DIOD1
            jmp BIT                 ögör om-försök med LÄSNING STÅR


BIT4:       addl #1,a0
            movb WDOG,d7            ökick watchdog
            rts


öOm vi adresserar på "burkens " adresser 280,380


BIT20:      clrl d3
BIT21:      cmpb #9,d3               öom NIO skrivningar =FEL
            beq BIT22
            addb #1,d3               öANTAL SKRIVNINGAR
            movb d4,a0@              ölägger ut testmönstret två gånger
            movb d4,a0@
            movb a0@,d5
            movb d1,BURK
            movb WDOG,d7             ökick watchdog

            eorb d4,d5               öfelaktigt ord (ej inskrivet)
            cmpb #/00,d5             öom inskrivet ord felaktigt
            bne BIT21                ökorrekt läsning

   jmp BIT4


        ö**************************************************************
        ö**************************************************************
              ö++++++++++++++++++++++++++++++++++++++++++++
              ö+          SUBRUTIN  ADD                   +
              ö+ skriver ut adress på skärm om bitfel+
              ö+ utdata HEX på skärm.indata a0        +
              ö++++++++++++++++++++++++++++++++++++++++++++

ADD:        movl #IN,a1              övar feldata skall ligga
            movl a0,a5               öRELATIVT A4
            addl REL,a5             öabsolut adress i primärminnet vid fel
```

```
            movl a5,d7              öfeladress
            swap d7
            rorw #8,d7
            rorb #4,d7
            andb #/0f,d7
            cmpb #9,d7
            bgt HA
            jsr DEC                 ö 1:a
            jmp HA2
HA:         jsr HEX

HA2:        movl a5,d7
            swap d7
            rorw #8,d7
            andb #/0f,d7
            cmpb #9,d7
            bgt HA3
            jsr DEC                 ö 2:a
            jmp HA4
HA3:        jsr HEX

HA4:        movl a5,d7
            swap d7
            rorb #4,d7
            andb #/0f,d7
            cmpb #9,d7
            bgt HA5
            jsr DEC                 ö 3:a
            jmp HA6
HA5:        jsr HEX

HA6:        movl a5,d7
            swap d7
            andb #/0f,d7
            cmpb #9,d7
            bgt HA7
            jsr DEC                 ö 4:a
            jmp HA8
HA7:        jsr HEX

HA8:        movl a5,d7
            rorw #8,d7
            rorb #4,d7
            andb #/0f,d7
            cmpb #9,d7
            bgt HA9
            jsr DEC                 ö 5:a
            jmp HA10
HA9:         jsr HEX

HA10:       movl a5,d7
            rorw #8,d7
            andb #/0f,d7
            cmpb #9,d7
            bgt HA11
            jsr DEC                 ö 6:a
            jmp HA12
HA11:       jsr HEX

HA12:       movl a5,d7
            rorb #4,d7
```

```
          andb #/0f,d7
          cmpb #9,d7
          bgt HA13
          jsr DEC                    ö 7:a
          jmp HA14
HA13:     jsr HEX

HA14:     movl a5,d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA15
          jsr DEC                    ö 8:a
          jmp HA16
HA15:     jsr HEX


HA16:     movb #94,a1@               ćavslutar med ^
          movl #IN,a1
          jsr OUTPUT                 öskriver ut feladress på skärm

          rts


        ö*************************************************************
DEC:      addb #48,d7                öger ascii värde
          movb d7,a1@                öskriver in ascii värde 0-9
          addl #1,a1
          rts
        ö*************************************************************
HEX:      addb #55,d7                öger hex A-F
          movb d7,a1@                ölägger i minne
          addl #1,a1
           rts
        ö*************************************************************


        ö*************************************************************
        ö*************************************************************
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN PARITS           +.
              ö+ testar paritetsbiten för minnesarea  +.
              ö+ om fel hoppas det till RESETADRESS   +.
              ö+ där cause registret läses av         +.
              ö+ om paritesfel visas det på skärm och +.
              ö+ lysdioder (FF) (PROGRAMMET STANNAR   +.
              ö+ FÖR DOM 200 FÖRSTA APPARATERNA       +.
              ö+ för efterföljande apparater hoppas   +.
              ö+ det till subrutinen NMI              +.
              ö+++++++++++++++++++++++++++++++++++++++.

PARITS:          movl #RI1,a1
                 jsr OUTPUT              öom paritetstest skall göras
                 jsr INPUT               ökollar svaret
                 movb IN,d1
                 cmpb #106,d1            öom j-tangenten
                 bne PAR5

    movl #TET1,a1
  jsr OUTPUT
```

```
  movb #/ff,d6          öatt vi gör paritetstest (NMI)
                movb #/04,PARIT       ötar bort paritets flaggan

                movl #/20000,a0       östart adress 256 första k:na
                movl PAR,a6           öVI HAR KRETS MED PARITETSFEL
PAR1:           movb #/ff,d4          ötestmönter
                movb #/ff,d3
                movb d4,a0@
                movl a0,d5
          cmpb #/280,d5               öom farlig adress
                beq PAR14
  cmpl #/380,d5
  beq PAR14
                movb a0@,d4           öger paritetsfel om paritetskrets FEL
PAR14:          movb #0,d3
                movb WDOG,d7          ökick watchdog
          addl #1,a0
                cmpl #/60000,a0
                bne PAR1
                jmp PAR2
PAR:            movl FEL4,a1          öskriver  att paritetskrets felaktig
                jsr OUTPUT
                jmp PARITS            östår och snurrar vid fel



PAR2:           movl #TET2,a1
  jsr OUTPUT

                movl #/20000,a0       östart adress 256 första k:na
PAR10:          movb #/07,d4          ötestmönter
                movb #/07,d3
                movb d4,a0@
          movl a0,d5
          cmpb #/280,d5
                beq PAR15             öom farlig adress
  cmpb #/380,d5
  beq PAR15
                movb a0@,d4           öger paritetsfel om paritetskrets FEL
PAR15:          movb #0,d3
                movb WDOG,d7          ökick watchdog
          addl #1,a0
                cmpl #/60000,a0
                bne PAR10
                movl #RTT3,a1         öparitetstesten klar
                jsr OUTPUT

PAR5:           movb #/0c,PARIT       ösätter paritetsflaggan igen
  movb #/00,d6          öatt paritetstest är klar (NMI)
                rts

       ö*************************************************************
       ö*************************************************************
 ö+++++++++++++++++++++++++++++++++++++++++
 öparitets kretsarna testas
 öskrivs först med fel paritet sedan ändras
 öparitetsbiten varvid fel skall uppkomma
 ötesten görs på stacktoppen
 ö+++++++++++++++++++++++++++++++++++++++++

KOLL:     movl #13,d1
```

```
            jsr SJUSEG1
            movb #/0c,PARIT         öändrar paritetsflaggan
            movb #/ff,/60000        öpå stacktoppen ger etta i paritet
            movb WDOG,d7            ökick watchdog
            movb #/04,PARIT         öändrar paritetsflaggan
            movl PPR36,a6           öåterhoppsadress från reset
            movb /60000,d1          öGER PARITETS FEL SKRIVS UT PÅ SKÄRM
            movl FEL3,a1
            jsr OUTPUT              öSKRIVER UT PÅ SKÄRM ATT FEL I
      öKONTROLLKRETS
            movl #/0f,d3            ölysdioder vid fel i paritetskrets
            jsr DIOD1
            jmp KOLL                öom test ger fel

PPR36:      movb #/0c,PARIT         öändrar paritetsflaggan
            movb #/ef,/60000        öger nolla paritetbit
            movb WDOG,d7            ökick watchdog
            movb #/04,PARIT         öändrar paritetsflagga
            movl PPR37,a6           öåterhoppsadress från reset
            movb /60000,d1          öGER PARITETS FEL SKRIVS UT PÅ SKÄRM
            movl #/f0,d3            ötänder dioder vid fel i paritetskrets
            jsr DIOD1
            jmp PPR36               öom test ger fel

PPR37:      movl #87,d1
            jsr SJUSEG1             öom återhopp FEL
            rts

         ö**************************************************************
         ö**************************************************************
               ö++++++++++++++++++++++++++++++++++++++++++.
               ö+           SUBRUTIN OUTPUT              +.
               ö+ skriver ut textsträng.INDATA a1        +.
               ö+avsluta texten med ^ (ascii dec 94)     +.
               ö++++++++++++++++++++++++++++++++++++++++++.

OUTPUT:     movb a1@,d0             ötecken från sträng
            cmpb #94,d0
            beq LI9

            jsr TX                  ökollar om transmit BUFFERTEN är tom
            movb d0,DARTB           öskriver ut tecken på skärm
            addl #1,a1
            jmp OUTPUT              ögår att skriva en bit till

LI9:        jsr TX                  ökollar om transmit BUFFERTEN är tom
            movb #10,DARTB          öline feed
            jsr TX                  ökollar om transmit BUFFERTEN är tom
            movb #13,DARTB          öreturn
            rts
         ö**************************************************************
         ö**************************************************************

TX:         movb WR0B,d1            ökollar  sändnings BUFFERTEN är tom
            movb WDOG,d7            ökick watchdog
            andb #/04,d1            ömaskar av Tx BUFFERT EMTY
            cmpb #/04,d1
            bne TX
            rts
         ö**************************************************************
RX:         movb #/00,WR0B
            movb WR0B,d0
            movb WDOG,d7            ökick watchdog
```

```
            andb #/01,d0
            cmpb #/01,d0            ökollar om något i inbufferten
            bne RX
            rts
          ö**************************************************************
          ö**************************************************************
               ö+++++++++++++++++++++++++++++++++++++++.
               ö+          SUBRUTIN INPUT             +.
               ö+ utdata i sträng IN.Hoppar ut vid    +.
               ö+ return. tecknet ekas på skärm.      +.
               ö+ första tecknet läggs i d1           +.
               ö+++++++++++++++++++++++++++++++++++++++.

INPUT:      movl #IN,a1
IN3:        jsr RX                  öom inbuffert tom
            movb DARTB,d2           ölägger ascii värdet i register d2
            jsr TX                  ökollar om transmitBUFFERT tom
            movb d2,DARTB           öekar tecknet
            cmpb #13,d2             öom return hoppa ut
            beq IN1

            movb d2,a1@             öanvänder toppen (botten) på stacken
            addl #1,a1
            jmp IN3                 önästa tecken

IN1:        jsr TX                  ökollar om transmitBUFFERT tom
            movb #10,DARTB          öläger ut linefeed
            movb #94,a1@            ösluttecken
            rts

          ö**************************************************************
          ö**************************************************************

 ö++++++++++++++++++++++++++++++++++++++++++++
               ösjusegment 8
DART:        movl #1,d1
            jsr SJUSEG1                 öDMA testas
 ö++++++++++++++++++++++++++++++++++++++++++++
            movb WDOG,d7                ökick watchdog
            movb #/18,WR0B              ö kanalreset B
            movb #/01,WR0B              ö  REGISTER 1
            movb #/00,WR0B              ö reset
            movb #/03,WR0B              ö  REGISTER 3
            movb #/41,WR0B              ö 7bits/character
            movb #/04,WR0B              ö  REGISTER 4
            movb #/47,WR0B              ö paritet osv
            movb #/05,WR0B              ö  REGISTER 5
            movb #/28,WR0B              ö bitar vid sändning

            jsr TX
            movb #12,DARTB              öraderar skärm
            movl #TEXT,a1
            jsr OUTPUT
            movl #TEXT1,a1
            jsr OUTPUT
            jsr INPUT
            movl #TEXT2,a1
            jsr OUTPUT

            rts

          ö**************************************************************
          ö**************************************************************
               ö++++++++++++++++++++++++++++++++++++++++.
```

```
            ö+          SUBRUTIN CIO                +.
            ö+        testar cio:n                  +.
            ö++++++++++++++++++++++++++++++++++++++++.

CIO:        movl #14,d1
            jsr SJUSEG1         ösjuseg 22

   jsr CIOENABLE      ögör enable på cio

CIO3:       jsr CLOCKVISI
            movl #TEXT4,a1
            jsr OUTPUT          öfrågar om klockan skall testas
            jsr INPUT           öhämtar svaret
            movl #IN,a1
            movb a1@,d1
            cmpb #106,d1
            bne CIO4
            jsr CLOCKTEST       ötestar clockan
            movl #TEXT6,a1
            jsr OUTPUT          öfrågar om klockan skall andras
            jsr INPUT           öhämtar svaret
            movl #IN,a1
            movb a1@,d1
            cmpb #106,d1
            bne CIO4
            movl #TEXT7,a1
            jsr OUTPUT
            jsr INPUT
            movl #IN,a1         öhämtar data
            jsr CLOCK           öändrar tiden på klockan


CIO4:       movl #86,d1
            jsr SJUSEG1         ösjuseg 8 MENY
            rts
        ö**************************************************************************
        ö**************************************************************************
            ö++++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN CIOENABLE          +.
            ö+ gör enable på CIO                    +.
            ö++++++++++++++++++++++++++++++++++++++++.

CIOENABLE: movb #/00,CIK       ökontrollreg
           jsr DELA
           movb #/01,CIK       öreset port c
           jsr DELA
           movb #/00,CIK       ökontrollreg
           jsr DELA
           movb #/02,CIK       öåterställer reset port c
           jsr DELA
           movb #/07,CIKöspecialcontrol register normal input or output
           jsr DELA
           movb #/00,CIK
           jsr DELA

           movb #/05,CIK
           jsr DELA
           movb #/00,CIK       öpolarity register
           jsr DELA

           movb #/1e,CIK       ömode specifikation reg
           jsr DELA
           movb #/00,CIK
           jsr DELA
```

```
            movb #/04,CIC       önot enable clock+nvram
                  jsr DELA

            movb #/01,CIK       öport c SELECT
            jsr DELA
            movb #/10,CIK
            jsr DELA

            movb #/04,CIC       önot enable clock+nvram
            jsr DELA

    rts

        ö**************************************************************
        ö**************************************************************
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN CLOCKVISI        +.
              ö+ skriver ut (YYDDMMDDHHMMSS) på skärm +.
              ö+ räknar åtta sekunder                +.
              ö+++++++++++++++++++++++++++++++++++++++.

CLOCKVISI:movb #0,LOOP
VIS1:       jsr SECPUT          öhämtar data från klockan
            movl #/60000,a1      övar talen skall ligga
            jsr OUTPUT

            movl #120000,d1
            jsr DELAY1          öväntar en sek

            addb #1,LOOP
            cmpb #8,LOOP
            bne VIS1
            rts

        ö**************************************************************
        ö**************************************************************
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN SEC              +.
              ö+ tar fram data ur  klockan           +.
              ö+++++++++++++++++++++++++++++++++++++++.


SECPUT:     movl #6,d6          ötitta på ÅR först
            movl #IN,a1
TTT:        jsr CIOREADC
            jsr ASCIIOUT
            addl #1,a1
            subl #1,d6
            cmpb #/ff,d6
            bne TTT
ö:          addl #1,a1
            movb #94,a1@        öavslutar med ^
            rts

        ö**************************************************************
        ö**************************************************************
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN CLOCK            +.
              ö+ ändrar tiden i klockan              +.
              ö+ indata a1 :var data ligger          +.
              ö+ avsluta med ^                       +.
              ö+++++++++++++++++++++++++++++++++++++++.

CLOCK:      movb #6,MODE
```

```
CLOCK1:  movb a1@,d3
         cmpb #94,d3
         beq CLOCK2
         rolb #4,d3
         andb #/f0,d3
         addl #1,a1
         movb a1@,d4
         cmpb #94,d4
         beq CLOCK2
         andb #/0f,d4
         orb d4,d3
         movb MODE,d6
         jsr CIOWRITE      öskriver ut data i klockan
         addl #1,a1
         subb #1,MODE      önästa mode
         jmp CLOCK1
CLOCK2:  rts

      ö*****************************************************************
      ö*****************************************************************
           ö+++++++++++++++++++++++++++++++++++++++++.
           ö+           SUBRUTIN CLOCKTEST          +.
           ö+ skriver ut (YYDDMMDDHHMMSS)           +.
           ö+ 99071231235956 som sedan räknas upp   +.
           ö+ till 00010101001004 om klockan fung   +.
           ö+++++++++++++++++++++++++++++++++++++++++.


CLOCKTEST:jsr SECPUT
          movl #IN,a1        ösparar undan nuvarande tid
          movl #SAVE,a2      övar det skall lagras
          movb #0,LOOP
TEST1:    movb a1@,a2@
          addl #1,a1
          addl #1,a2
          addb #1,LOOP
          cmpb #16,LOOP      öallt inläst
          bne TEST1


          movl #6,d6         öår 99
          movl #/99,d3
          jsr CIOWRITE
          movl #5,d6         öday of week
          movl #/07,d3
          jsr CIOWRITE
          movl #4,d6         ömonth
          movl #/12,d3       ödec
          jsr CIOWRITE
          movl #3,d6         ödate
          movl #/31,d3       ö31:a
          jsr CIOWRITE
          movl #2,d6         öhour
          movl #/23,d3       ö23 timman
          jsr CIOWRITE
          movl #1,d6
          movl #/59,d3       ö59 minuter
          jsr CIOWRITE
          movl #0,d6         ösek
          movl #/56,d3       ö56 sekunder
          jsr CIOWRITE


          jsr CLOCKVISI      öräknar upp den nya tiden 8SEK
```

```
            movl #SAVE,a1
            jsr CLOCK           öskriver tillbaka den gamla tiden
            rts
          ö*************************************************************
          ö*************************************************************

DELA:       movl #/01,d1        ö20mikrosek fördröjning
            jsr DELAY1
            rts

          ö*************************************************************
          ö*************************************************************
               ö+++++++++++++++++++++++++++++++++++++++.
               ö+          SUBRUTIN CIOWRITE          +.
               ö+ skriver ut på port c.indata d6:vad  +.
               ö+ d3:värde till vad                   +.
               ö+++++++++++++++++++++++++++++++++++++++.


CIOWRITE: movb #/06,CIK
          jsr DELA
          movb #/00,CIK       ödatarikningen alla UTGÅNGAR

          movb #/05,CIC       önoselect + clock HÖG
          jsr DELA
          movb #/00,CIC       öselect +clock låg
          jsr DELA

          jsr CIOADD          öadress vad som skall skrivas in sec-year

          movb #/00,CIC       öwrite mod läggs ut på port
          jsr DELA
          movb #/01,CIC       öklockas ut
          jsr DELA

          rorb #6,d3          öindata till adress (läggs på bit två)
          movb d3,d4
          jsr WRITE           ödataord 1

          movb #0,d5          öloop räknare för 7dataord
CIO1:     rolb #1,d3
          movb d3,d4
          jsr WRITE

          addb #1,d5
          cmpb #7,d5
          bne CIO1            öom inte sista dataordet
          movb #/01,CIC       önollpuls på slutet
          movb #/04,CIC       öno enable chip SET DATA

          rts

          ö*************************************************************
          ö*************************************************************
               ö+++++++++++++++++++++++++++++++++++++++.
               ö+          SUBRUTIN CIOREADC          +.
               ö+ läser från port c.indata d6:vad     +.
               ö+ utdata d3  BCD-format:värde på vad  +.
               ö+++++++++++++++++++++++++++++++++++++++.


CIOREADC:movb #/06,CIK
          jsr DELA
```

```
          movb #0,CIK       ödatarikningen ALLA ut
          jsr DELA
          movb #/05,CIC     öclockpuls + no chip enable
          jsr DELA
         movb #/00,CIC      öclock+chip select
          jsr DELA
         clrl d3
         jsr CIOADD         övad som skall tittas på
         movb #/02,CIC      ölägger ut READ mod
          jsr DELA
         movb #/03,CIC      öklockar ut
          jsr DELA
         movb #/00,CIC      öclocka låg
          jsr DELA
         movb #/01,CIC      öklocka hög för HIHG IMPEDANSBIT
          jsr DELA
         movb #/00,CIC      öclocka låg
          jsr DELA
         movb #/06,CIK
          jsr DELA
          movb #/02,CIK     ödatarikningen en INGÅNG ut
          jsr DELA
          clrl d3
          jsr READ
         rorl #1,d4
         orb d4,d3
         jsr READ
         orb d4,d3
         movl #1,d5         öloop räknare för bit 3-7
CIO2:    jsr READ
         rolb d5,d4
         orb d4,d3
         addb #1,d5
         cmpb #7,d5
         bne CIO2
         movb #/04,CIC      öno enable chip
 rts
        ö*************************************************************
        ö*************************************************************
              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+          SUBRUTIN ASCII               +.
              ö+ tar fram ascii-värdet                 +.
              ö+ indata d3 BCD .som läggs i adress a1  +.
              ö+++++++++++++++++++++++++++++++++++++++++.

ASCIIOUT:movb d3,d4
         rorb #4,d4
         andb #/0f,d4
         addb #48,d4
         movb d4,a1@
          movb d3,d4
         andb #/0f,d4
         addb #48,d4
         addl #1,a1
         movb d4,a1@
         rts



        ö*************************************************************
        ö*************************************************************
              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+          SUBRUTIN CIOADD              +.
              ö+ skriver till klockan vad som skall    +.
```

```
            ö+ skrivas eller tittas på                +.
            ö+++++++++++++++++++++++++++++++++++++++++.

CIOADD:  movb d6,d4
         rorb #1,d4
         jsr WRITE
         movb d6,d4
         jsr WRITE
         movb d6,d4
         rolb #1,d4
         jsr WRITE            öadress för vad som skal göras (sec-year)
         andb #/02,d4
         movb d4,CIC         öklocka låg,gör klart för read/write biten
         rts

        ö*************************************************************
        ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+          SUBRUTIN WRITE             +.
             ö+ skriver ut data till klocka         +.
             ö+ indata d4.  BIT TVÅ                 +.
             ö+ gör klocka låg först                +.
             ö+++++++++++++++++++++++++++++++++++++++.

WRITE:   andb #/02,d4
         orb #/00,d4           öno select nvram
         movb d4,CIC           ölägger ut data +no enable nvram
          jsr DELA
         orb #/01,d4
         movb d4,CIC           öklockar ut data
          jsr DELA
         rts
        ö*************************************************************
        ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+          SUBRUTIN READ              +.
             ö+ läser ut data från klocka           +.
             ö+ utdata d4.  BIT TVÅ                 +.
             ö+++++++++++++++++++++++++++++++++++++++.

READ:    clrl d4
         jsr DELA
         movb #/01,CIC         öklocka på utgång
         movb CIC,d4           öhämtar data
         jsr DELA
         movb #/00,CIC         öklocka låg
         andb #/02,d4
         rts

        ö*************************************************************
        ö*************************************************************

             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN FLOPPY           +.
             ö+         testar floppykontroller     +.
             ö+++++++++++++++++++++++++++++++++++++++.

FLOPPY:      movl #15,d1
             jsr SJUSEG1                ösjuseg 23

FLOP2:       jsr FLOPRES                öresetar floppyn ,startar motorn

             movb #45,TRAC              öinitiera 45 TRACK (sista spåret)
FLO:         jsr TRACK                  öSÖKER EFTER SPÅR
```

```
            movb   WDOG,d7              ökick DOG
            jsr TRACKTEST               ötestar att det är rätt spår

            movb #1,SEK
            jsr FLOPPYSEK               övilken sektor

            movl #FLOTEXT1,a1           ötest text
            jsr OUTPUT
            jsr INPUT                   övad som skall skrivas på floppy

             movl #IN,a2                öskriver in text på floppyn
            jsr WRITEF                  öskriver in data


            movb #1,SEK                 ösektor ett
            jsr FLOPPYSEK
            jsr READF                   öläser på skiva


            movl #FLOTEXT,a1
            jsr OUTPUT
            movl #SAVE,a1
            jsr OUTPUT                  öskriver ut data på skärm

            movw #/0391,FLOPCTR         öfloppy motor stannar
            movl #/7ffff,d1
            jsr DELAY1

            movl #85,d1
            jsr SJUSEG1                 ösjuseg åtta
            rts

        ö*************************************************************
        ö*************************************************************
              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN FLOPRES            +.
              ö+gör reset på floppyn ock startar       +.
              ö+motorn                                 +.
              ö+++++++++++++++++++++++++++++++++++++++++.

FLOPRES:    movw #/0000,FLOPCTR       ögör reset på floppykontrollen
                                            ö (min 50u)
            movl #10,d1
            jsr DELAY1                 öger tidsfördröjning för RESET
            movw #/0c19,FLOPCTR       öinitierar FLOPPYN (motorn startas)
            movl #2,d1
            jsr DELAY1
            movw #/0c1d,FLOPCTR        öHEADLOAD TIME
            clrl d4
FLOP:       addl #1,d4
            cmpl #/ffff,d4
            bne FLOP1
            jsr STATUS                 öom FLOPPY ej isatt
FLOP1:      movb FLOPSTA,d3
            movb   WDOG,d7             ökick DOG
            andb #/80,d3
            cmpb #/80,d3
            beq FLOP
            movb #/0c,FLOPCMD          öRESTORE floppy
            movl #/fff,d1       öungefär 25ms
            jsr DELAY1                 öger tidsfördröjning för RESET
    rts
```

```
      ö**********************************************************************
      ö**********************************************************************
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN WRITEF             +.
            ö+Skriver ut på floppy INDATA:a2         +.
            ö+en sekltor i taget                     +.
            ö+++++++++++++++++++++++++++++++++++++++++.


WRITEF:     movb #0,d3
            jsr DIOD1
            movl a2,a4
            movb   WDOG,d7        ökick DOG
            movb #/ae,FLOPCMD     öatt vi skall WRITE floppy TRACK
            movl #FLOPDATA,a5
            movl #1,d1
            jsr DELAY1


WRI:        movb FLOPSTA,d3
            rorb #1,d3  ötestar busy biten
            bcc WRI12

            rorb #1,d3  öom data request
            bcc WRI

            movb (a2)+,a5@        ölägger data i dataregistret
            bra WRI               öSKRIVER nästa tecken


WRI12:      movb FLOPSTA,d3       öskriver ut FELKOD om finnes
    cmpb #0,d3
            beq WRI14
            movl #FLOFEL3  ,a1
            jsr OUTPUT            ögör omförsök att skriva rätt
            movl #/ffff,d1
            jsr DELAY1
            movl a4,a2
            jmp WRITEF
WRI14:      movl #0,d3
            jsr DIOD1             önollställer lysdioderna
            rts

      ö**********************************************************************
      ö**********************************************************************
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN READF              +.
            ö+Läser data från floppy.UTDATA SAVE     +.
            ö+en sektor i taget                      +.
            ö+++++++++++++++++++++++++++++++++++++++++.


READF:      movl #SAVE,a2
            movb   WDOG,d7        ökick DOG

            movb #/8e,FLOPCMD     öatt vi skall READ floppy
            movl #1,d1
            jsr DELAY1



            movb   WDOG,d7        ökick DOG
REA11:      movb FLOPSTA,d3
            btst #0,d3            ö BUSY ?
```

```
            beq  REA12
            btst #1,d3              ö DATAREQUEST ?
            beq REA11

            movb FLOPDATA,(a2)+  ölägger data i a2@
            bra REA11            öhämtar nästa tecken


REA12:      movb FLOPSTA,d3
     cmpb #0,d3
     beq REA57
     jsr DIOD1          öskriver ut felkod på lysdiod
     movl #FLOREERR,a1     ödet har blivit fel på läsning
     jsr OUTPUT
     movl #/ffff,d1
     jsr DELAY1
     jmp READF             ögör omförsök


REA57:      addl #1,a2
            movb #94,a2@
            movl #SAVE,a1
            movb   WDOG,d7        ökick DOG
            movl #0,d3
     jsr DIOD1       önollställer lysdioder
            rts

        ö*************************************************************
        ö*************************************************************
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN STATUS           +.
              ö+Kollar att dator inte är upptagen    +.
              ö+++++++++++++++++++++++++++++++++++++++.

STATUS:     clrl d4
STA:        movb FLOPSTA,d3
            movb   WDOG,d7        ökick DOG
            andb #/80,d3
            cmpb #/80,d3
            beq  STAFEL           öfloppy är inte klar
            movb FLOPSTA,d3
            andb #/01,d3
            cmpb #/01,d3
            beq STA               öfloppykontrollen upptagen
            rts

STAFEL:     addl #1,d4
            cmpl #5,d4
            bne STA
            movl #FLOFEL1,a1      öskriver ut att floppyn ej ansluten
                                  ökan även vara fel på kontroller
            jsr OUTPUT
            movl #/ffff,d1
            jsr DELAY1
            jmp FLOP2             ögör omtest

        ö*************************************************************
        ö*************************************************************
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+          SUBRUTIN FLOPPYSEK         +.
              ö+Väljer sektor INDATA SEK             +.
              ö+++++++++++++++++++++++++++++++++++++++.
```

```
FLOPPYSEK: jsr STATUS
           movb FLOPSTA,d3
           btst #1,d3
           bne FLOPPYSEK          öATT INTE CONTROLLEN ÄR UPPTAGEN
           movb SEK,FLOPSEK       ösektor noll
         rts

       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+          SUBRUTIN TRACKTEST         +.
             ö+Testar om vi har stegat till rätt spår+.
             ö+++++++++++++++++++++++++++++++++++++++.


TRACKTEST: jsr STATUS
           movb FLOPTRACK,d4          öom rätt spår är stegat till
           cmpb TRAC,d4               öom rätt spår
           beq FLOT1
           movl #FLOFEL2,a1     öTRACKREGISTRET FEL floppykontroll FEL
           jsr OUTPUT                 öeller floppyn ej formaterad
           movl #/fff,d1
           jsr DELAY1
           jmp FLOP2                   ögör omtest
FLOT1:     rts

       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+          SUBRUTIN TRACK             +.
             ö+  flyttar huvudet till spår TRAC     +.
             ö+++++++++++++++++++++++++++++++++++++++.


TRACK:     jsr STATUS             ökollar om kontrollen upptagen
           movb TRAC,FLOPDATA       övilket spår i dataregistret
           movb #/1d,FLOPCMD      östegar tiil spår d3
           movl #1,d1             öfördröjning
           jsr DELAY1             öinnan vi kan läsa i statusreg
           rts

       ö*************************************************************
       ö*************************************************************

             ö+++++++++++++++++++++++++++++++++++++++.
             ö+          SUBRUTIN NVRAM             +.
             ö+         testar NVRAMMET             +.
             ö+++++++++++++++++++++++++++++++++++++++.


NVRAM:  movl #TEST,a1
        jsr OUTPUT

        movl #18,d1
        jsr SJUSEG1                 ösjuseg 26
        movb   WDOG,d7              ökick DOG
 jsr CIOENABLE              öinitierar CIO


 movb #/06,CIK
 jsr DELA
 movb #/00,CIK              öalla utgångar
 jsr DELA                   öfördröjning
        movb   WDOG,d7              ökick DOG
```

```
   movb #/00,REGI     ötestar register noll
NV1: movb #/01,DATA            ödata som skrivs in VANDRANDE ETTA
NV: movb DATA,d3


  jsr ERASE            önVRAM +ENABLE PROGRAMMING MODE
 jsr RAMWRITE               öskriver testmönster i rammet
 jsr RAMSAVE                öspara NVRAMMETS innehåll

 rolb #1,d3
 cmpb #/01,d3
 beq NV2
 movb d3,DATA
 jmp NV
NV2: addb #1,REGI
 cmpb #15,REGI
 bne NV1      önästa register


        jsr NOENRAM     öskrivskyddar NVRAMMET


 movl #RAMOK,a1
 jsr OUTPUT
 movl #/7ffff,d1
 jsr DELAY1


 movl #82,d1
 jsr SJUSEG1     ösjuseg 8 "meny"
 rts


       ö*****************************************************************
T: movl #SAVE,a4               övar data skall ligga,
T1: movb a4@,d3
 addl #1,a4
 jsr DIOD1
 movl #/7fff,d1
 jsr DELAY1
 clrb d3
 jsr DIOD1
 movl #/7fff,d1
 jsr DELAY1
 cmpl #/60120,a4
 bne T1
 rts

       ö*****************************************************************
       ö*****************************************************************
             ö+++++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN NOENRAM            +.
             ö+skrivskyddar NVRAMMET                  +
             ö+++++++++++++++++++++++++++++++++++++++++.

NOENRAM:movb #/06,CIK           ödata riktningen ställs in
 jsr DELA
 movb #/00,CIK            öalla utgångar
 jsr DELA                 öfördröjning
 movb #/04,CIC
 jsr DELA  öNO ENABLE
```

```
öERASE/WRITE DISABLE

 movb #/0c,CIC            öchip select
 jsr DELA
 movb #/0d,CIC          öklockar ut den
 jsr DELA

 jsr ONE

 jsr ZERO

 jsr ZERO

 jsr ZERO

 jsr ZERO


 jsr ZERO

 jsr ZERO

 jsr ZERO

 jsr ZERO

 movb #/04,CIC            öklocka låg + NOSELECT
 jsr DELA

 rts

       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN RAMWRITE          +.
             ö+skriver in testmönster i NVRAMMET     +
      ö+INDATA : REGI,DATA                          +
             ö+++++++++++++++++++++++++++++++++++++++.


RAMWRITE:movb #/06,CIK          öställa in datariktningen
 jsr DELA
 movb #/00,CIK            öalla utgångar
 jsr DELA                 öfördröjning

 jsr ZERO  öenable NVRAM


       movb #/04,OPCODE         öopcode WRITE
 jsr RAM1                 övilken OPKOD

       jsr RAM2  öskriver ut vilket REGISTER


 movb #0,d6  ödata BITS räknare

WRNV: movb DATA,d4
 rorb #6,d4
 jsr WRN5

 movb DATA,d4
 rorb #5,d4
 jsr WRN5
```

```
 movb DATA,d4
 rorb #4,d4
 jsr WRN5

 movb DATA,d4
 rorb #3,d4
 jsr WRN5

 movb DATA,d4
 rorb #2,d4
 jsr WRN5

 movb DATA,d4
 rorb #1,d4
 jsr WRN5

 movb DATA,d4
 jsr WRN5

 movb DATA,d4
 rolb #1,d4
 jsr WRN5

 addb #1,d6
 cmpb #3,d6  öom vi har skrivit in 16 BITS
 bne WRNV

 movb #/04,CIC  öklocka låg + noenable
 jsr DELA

 movl #/600,d1
 jsr DELAY1


 rts
      ö*****************************************************************
WRN5: andb #/02,d4
 cmpb #/02,d4              öom "1" eller "0"
 bne WRN1
 jsr ONE   öskriver ut "1"
 jmp WRN2
WRN1: jsr ZERO

WRN2: rts
      ö*****************************************************************
      ö*****************************************************************
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN ERASE              +.
            ö+RESETAR nvrammet ,det läggs ettor i   +.
      ö+ alla register                             +.
            ö+++++++++++++++++++++++++++++++++++++++++.


ERASE: movb #/06,CIK           ö ställer in data riktningen
 jsr DELA
 movb #/00,CIK           öalla utgångar
 jsr DELA                öfördröjning
 movb #/04,CIC
 jsr DELA  öNO ENABLE

öERASE/WRITE ENABLE
ögör programming enable på chip
```

```
 movb #/0c,CIC            öchip select
 jsr DELA
 movb #/0d,CIC          öklockar ut den
 jsr DELA

 jsr ONE

 jsr ZERO

 jsr ZERO

 jsr ONE

 jsr ONE

 jsr ZERO

 jsr ZERO

 jsr ZERO

 jsr ZERO

 movb #/04,CIC            öklocka låg + NOSELECT
 jsr DELA


öSTART BIT
 movb #/0c,CIC   öchip select
 jsr DELA
 movb #/0d,CIC
 jsr DELA

 jsr ONE


öOPCODEN
 jsr ZERO

 jsr ZERO

 jsr ONE

 jsr ZERO


öADRESS
 jsr ZERO

 jsr ZERO

 jsr ZERO

 jsr ZERO



öEND
 movb #/0c,CIC  öklocka låg
 jsr DELA
 movb #/0d,CIC
 jsr DELA
 movb #/04,CIC  öklocka låg
```

```
 jsr DELA  öavslutar med en klockpuls

        movl #/600,d1  öungefär 25ms fördröjning
  jsr DELAY1              öfördröjning efter NO SELECT

 rts


       ö*****************************************************************
ZERO: movb #/0c,CIC            öladdar "0"
 jsr DELA
 movb #/0d,CIC             öklockar ut den
 jsr DELA
 rts
       ö*****************************************************************
ONE: movb #/0e,CIC  öladdar "1"
 jsr DELA
 movb #/0f,CIC  öklockar ut den
 jsr DELA
 rts
       ö*****************************************************************

       ö*****************************************************************
       ö*****************************************************************
             ö+++++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN RAMSAVE            +.
             ö+ Lagrar undan det som ligger i NVRAMET+.
      ö+ Det skall läggas tillbaka sedan      +.
             ö+++++++++++++++++++++++++++++++++++++++++.



RAMSAVE:clrl d6
 movb #/04,CIC  öklocka låg start clocka NOENBLE
 jsr DELA

 movl #/fff,d1
 jsr DELAY1

 movl #0,d6                öadressregister
 movl #SAVE,a4             övar datat skall ligga
 movb #/08,OPCODE öread NVRAM

RAM31: jsr RAM21                ölagrar register REGI
        movb   WDOG,d7             ökick DOG

 movb #/0c,CIC
 jsr DELA
 movb #/0d,CIC
 jsr DELA
 movb #/0c,CIC
 jsr DELA  öavslutar med en klockpuls


 rts


       ö*****************************************************************
       ö*****************************************************************

             ö+++++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN RAM21              +.
             ö+ INITIERAR vad NVRAMMET skall lagra   +
```

```
      ö+ i minnet
            ö+++++++++++++++++++++++++++++++++++++++.


RAM21: movb #/06,CIK           öställa in datariktningen
 jsr DELA
 movb #/00,CIK            öalla utgångar
 jsr DELA                 öfördröjning

 movb #/0c,CIC            öenable nwram
 jsr DELA
 movb #/0d,CIC  öklocka hög
 jsr DELA

 jsr RAM1                 övilken opkod

        jsr RAM2  öskriver ut vilket register

 jsr RAM3   öhämtar data och lagrar i SAVE

        movb   WDOG,d7           ökick DOG

 rts


       ö**************************************************************************
       ö**************************************************************************

             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN RAM1              +.
             ö+ Skriver ut OPCODE till NVRAM         +
      ö+ INDATA  OPCODE             +
             ö+++++++++++++++++++++++++++++++++++++++.

RAM1:
öSTARTBIT
        movb #/0e,CIC  öklocka låg + data hög
 jsr DELA
 movb #/0f,CIC            öklocka hög +"1" ut startbit
 jsr DELA
öOPCODE

 movb OPCODE,d4
 rorb #2,d4
 jsr RAM10                öklockar ut första kod biten

 movb OPCODE,d4
 rorb #1,d4
 jsr RAM10   öklockar ut andra kod biten


 movb OPCODE,d4
 jsr RAM10  öklockar ut tredje kodbiten


 movb OPCODE,d4
  rolb #1,d4
 jsr RAM10        öklockar ut fjärde kodbiten
        movb   WDOG,d7             ökick DOG



 rts
```

```
RAM10: andb #/02,d4
 orb #/0c,d4       öklocka låg + data in låg
 movb d4,CIC
 jsr DELA
 orb #/01,d4
 movb d4,CIC              öklocka hög + data ut
 jsr DELA

 rts

       ö*************************************************************
       ö*************************************************************

            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN RAM2               +.
            ö+ Skriver ut vilket register till NVRAM+
      ö+ INDATA REGI                 +
            ö+++++++++++++++++++++++++++++++++++++++++.


RAM2:   movb REGI,d6
 rorb #2,d6
 jsr NVADD              öA3

 movb REGI,d6
 rorb #1,d6
 jsr NVADD  öA2

 movb REGI,d6
 jsr NVADD  öA1

 movb REGI,d6
 rorl #1,d6
 jsr NVADD  öA0


       movb   WDOG,d7            ökick DOG
       rts


NVADD:  andb #/02,d6        ötar fram bit som skall sändas över
 orb  #/0c,d6             ögör klocka låg+  data ut
 movb d6,CIC
 jsr DELA
 orb #/01,d6
 movb d6,CIC              öklocka hög klockar ut data
 jsr DELA
 rts

       ö*************************************************************
       ö*************************************************************

            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN RAM3               +.
            ö+ HÄMTAR data BYTES vis och lagrar det +
      ö+ i minne INDATA a4                    +
            ö+++++++++++++++++++++++++++++++++++++++++.


RAM3: movb #/06,CIK  öatt vi skall ändra på data riktningen
 jsr DELA
 movb #/02,CIK            öen ingång
```

```
 jsr DELA                öfördröjning

 movb #1,d6              öbyteräknare

RAM333: clrl d3                  ödata ordet som bitarna lagras i

RAM33: jsr ZERO
        movb CIC,d4
 andb #/02,d4

 rolb #6,d4
 jsr OR

 rolb #5,d4
 jsr OR

 rolb #4,d4
 jsr OR

 rolb #3,d4
 jsr OR

 rolb #2,d4
 jsr OR

 rolb #1,d4
 jsr OR

 jsr OR
 rorb #1,d4
 orb d4,d3


 movb d3,(a4)+           ölagrar ut ett BYTE

 movb DATA,d4
       cmpb d3,d4
 bne RAM65

 addb #1,d6
 cmpb #3,d6  ökollar om ett WORD är lagrat
 bne RAM333

 movb #/0c,CIC           öklocka går låg
 movb #/04,CIC           öNO SELECT
 movl #/fff,d1
 jsr DELAY1  öfördröjning efter no select
 rts


      ö*****************************************************************
RAM65:  eorb d4,d3              ötar fram felaktig bit
        jsr DIOD1               öskriver ut felaktig bit
        movl #RAMFEL,a1  öskriver ut att NVRAMMET ej okey
 jsr OUTPUT
 movl #/ffff,d1
 jsr DELAY1
 jmp NVRAM  ögör omtest

      ö*****************************************************************
OR: orb d4,d3
 jsr ZERO
 movb CIC,d4
 andb #/02,d4
```

```
 rts

        ö**********************************************************************
        ö**********************************************************************
              ö++++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN DMA               +.
              ö+         testar DMA                   +.
              ö++++++++++++++++++++++++++++++++++++++++.

DMAM:    movl #TEST,a1
         jsr OUTPUT

         movb   WDOG,d7        ökick DOG
         movl #16,d1
         jsr SJUSEG1           ösjuseg 24

         movw #/0391,FLOPCTR   öfloppy motor stannar vid omstart
         movl #2,d1
         jsr DELAY1
         clrl d3
         jsr DIOD1             öresetar lysdioderna

 movl #/80,d3           öresetar 256K och framåt
 jsr PAGTRAN1
 movl #/20000,a0
DMA93:   clrb (a0)+
         movb   WDOG,d7        ökick DOG
 cmpl #/60000,a0
         bne DMA93


         movb   WDOG,d7        ökick DOG
 movl #/40,d3
 jsr PAGTRAN2       ölogisk adress 0-32K,fysisk 128+32K
         movl #/20000,a0

DMA10:   clrb (a0)+            öRESETAR SKRIVAREAN
         cmpl #/40000,a0       ö28000 resetar 128k
 bne DMA10
         movb   WDOG,d7        ökick DOG

ö---------------------------------------------------------------------
 öskriver in data på floppydriver
ö---------------------------------------------------------------------

DMA14:   movl #SAVE,a0         övar data skall ligga
         clrl d2

DMA36:   cmpw #256,d2          öom 0-255 byte inskrivet
 beq DMA37
         movb d2,(a0)+
         addw #1,d2
 bra DMA36

DMA37:   jsr FLOPRES

         movb #0,TRAC          öspår NOLL
 jsr TRACK


         movb #1,SEK           ösektor ETT
        jsr FLOPPYSEK
         movb   WDOG,d7        ökick DOG
```

```
 movl #SAVE,a2
 jsr WRITEF           öskriver in data på spår NOLL
        movb   WDOG,d7       ökick DOG


ö-------------------------------------------------------------------
öläsning från floppy till minnesområde med DMA
ö-------------------------------------------------------------------

DMA39:  movb #/0f,SPCTROL    öset bit SYSFS för INTERN hämtning  av DMA
                             öfrån floppy


        movb #/02,DMAMAP0    ö128K och uppåt DMA 0
 movb #/9f,DMAMAP0+1 ö floppyn

 movb #/a3,DMA0         öreset and disable
 movb #/83,DMA0         ödisable DMA


        movb #/c3,DMA0         öRESET
        movb #/c3,DMA0         öRESET
        movb #/c3,DMA0         öRESET
        movb #/c3,DMA0         öRESET
        movb #/c3,DMA0         öRESET

ö---- WR0-GROUP-----------

        movb #/79,DMA0      öinitiera DMA port a till b TRANSFER

        movb #/00,DMA0
  movb #/00,DMA0      östartadress 128K


        movb #/ff,DMA0
        movb #/00,DMA0      öhur stort blocket är 256BYTE

ö---- WR1-GROUP----------

 movb #/54,DMA0      öVAD port a är för nått
  movb #/0d,DMA0      öCYCKELLÄNGD TRE

ö---- WR2-GROUP----------

 movb #/68,DMA0      övad port b är för nått
 movb #/0d,DMA0      öCYCKELLÄNGD TRE

ö---- WR3-GROUP----------

        movb #/80,DMA0      ögroup 3

ö---- WR4-GROUP----------

        movb #/cd,DMA0      ö4 group BURST
        movb #/06,DMA0
        movb #/f0,DMA0      öport b:s startadress

ö---- WR5-GROUP----------

 movb #/92,DMA0      ö5 group CE/WAIT AKTIV LÅG .READY AKTIV LÅG

ö---------------------
 movb #/0d,SPCTROL    ösignalen DMADIS
ö---------------------
```

```
ö---- WR6-GROUP---------

        movb #/cf,DMA0        öload A and reset counter
  movb #/87,DMA0        öDMA START

ö----------------------


 movb #1,SEK           ösektor ett
        jsr FLOPPYSEK
        movb #/8e,FLOPCMD     öatt vi skall READ floppy GER RDY signal
        ötill DMA
        movl #1,d1
 jsr DELAY1

DMA35:  movb FLOPSTA,d3
        btst #0,d3            ö BUSY ? hoppar ej ur lopp då
        bne DMA35
 cmpb #0,d3
 beq DMA76

        movl #FLOREERR,a1     öskriver ut FELstatus på lysdioderna READ
        jsr OUTPUT
        jsr DIOD1
 movl #/ffff,d1
   jsr DELAY1
 jmp DMA39             ögör omtest



DMA76: movb #/bf,DMA0        öläser statusbyte
        movb DMA0,d3

 movb d3,d4
 cmpb #/1b,d4
  beq DMA38
 movl #DMAFEL1,a1      öDMA överföringen ej genomförd
 jsr OUTPUT

 movl #DMAVI,a1        öskriver ut status för DMA
 jsr OUTPUT
        jsr DIOD1
 movl #/ffff,d1
 jsr DELAY1


 movb d3,d4
 andb #/20,d4
 cmpb #0,d4
        beq DMA38
 movl #DMAFEL2,a1      öEND of block not found
 jsr OUTPUT
 jmp DMAM              ögör omförsök


DMA38: movl #/20000,a0
  clrl d4
ö--------------------------
DMA40:  movb (a0)+,d3
 cmpb d4,d3
```

```
 bne DMA63              ömönstret FEL överfört
 addb #1,d4
        cmpl #/20101,a0      ötestar block om 256byte
  bne DMA40

ö---------------------------
DMA47: movb (a0)+,d3
 cmpb #0,d3
 bne DMA64              öfelaktigt kopierad
        movb   WDOG,d7        ökick DOG
        movl a0,d5
        cmpb #/80,d5
        bne DMA48
        addl #1,a0            ötar bort felaktiga adresser
DMA48: cmpl #/40000,a0
 bne DMA47
ö---------------------------
        movl #/80,d3          ötestar 256K och framåt skall vara nollat
 jsr PAGTRAN1
 movl #/20000,a0
DMA97:  movb (a0)+,d3
        cmpb #0,d3
 bne DMA64
        movb   WDOG,d7        ökick DOG
 movl a0,d5
        cmpb #/80,d5
        bne DMA98             öhoppar över felaktiga adresser
        addl #1,a0            ötar bort felaktiga adresser
DMA98: cmpl #/60000,a0
        bne DMA97
 jmp DMA65
ö---------------------------

DMA63:  movl #DMAFEL3,a1      ömönstret har blivit felaktigt överfört
 jsr OUTPUT
 jsr DIOD1             öskriver ut felaktig bit
 movl #/ffff,d1
        jsr DELAY1
        jmp DMAM              ögör omförsök
ö---------------------------

DMA64: movl #DMAFEL4,a1      öskriver ut att det har blivit kopiering
        jsr OUTPUT             ötill annat block
 jsr DIOD1             öskriver ut felaktig bit
 movl #/ffff,d1
        jsr DELAY1
        jmp DMAM              ögör omförsök
ö---------------------------

DMA65:  movl #AMD0,a1          öfloppy till DMA till minne fungerar
 jsr OUTPUT
        movw #/0391,FLOPCTR   öfloppy motor stannar
        movl #/ffff,d1
        jsr DELAY1
        clrl d3
        jsr DIOD1             öresetar lysdioderna
        movl #84,d1
        jsr SJUSEG1           ösjuseg åtta
        rts
       ö*************************************************************
       ö*************************************************************
             ö++++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN BUSS               +.
             ö+         testar busskortet             +.
```

```
              ö+++++++++++++++++++++++++++++++++++++++.

BUSS:  clrl d3
 jsr DIOD1
 jsr CRNUM              ö vilken kanal XEBECkontrollen sitter i

        movb #0,CONRESET      ögenererar en reset till controllen
 movl #/ff,d1
 jsr DELAY1

 jsr BUSSTEST           ötestar  kommunikation med kontrollen
        clrl d3
        jsr DIOD1               öresetar lysdioderna
        movl #83,d1
        jsr SJUSEG1             ösjuseg åtta

 rts

       ö**************************************************************************
       ö**************************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN CRNUM            +.
             ö+         testar var busskortet       +.
             ö+++++++++++++++++++++++++++++++++++++++.

CRNUM:   movl #17,d1
        jsr SJUSEG1           ösjuseg 25


BU1: movl #BUSS1,a1        övar kortet sitter
        jsr OUTPUT

       öläser databuss genom att generera stroben RCSB*

        movb BUSSSTA,d3       ötar reda på var XEBEC KONTROLL sitter
 notb d3
 jsr DIOD1
        cmpb #/01,d3
 bne BU2
        movl #BUS2,a1         öBUSS 2
        jsr OUTPUT
 jmp BU5

BU2:    cmpb #/02,d3
        bne BU3
        movl #BUS1,a1         öbuss 1
 jsr OUTPUT
 jmp BU5

BU3:    cmpb #/40,d3
 bne BU4
 movl #BUS0E,a1        öbuss 0 extern
 jsr OUTPUT
 jmp BU5

BU4:    cmpb #/80,d3
        bne HU
        movl #BUS0,a1         öbuss 0
        jsr OUTPUT
 jmp BU5

HU:     cmpl #/04,d3
 beq HU5
```

```
 cmpl #/08,d3
 beq HU5
 cmpl #/10,d3
 beq HU5
 cmpl #/20,d3
 beq HU5

FELBU:  movl #CARDFEL,a1      öhittar inte rätt kort
        jsr OUTPUT
        movl #/7ffff,d1
        jsr DELAY1
        jmp BUSS              öGÖR OMTEST



HU5: movl #CARDRAT,a1
 jsr OUTPUT     öskriver ut att kortvalet okey extern buss
BU5: movl #/7ffff,d1
 jsr DELAY1
  clrl d3
 jsr DIOD1             öresetar lysdioderna

 rts

       ö**************************************************************
       ö**************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN BUSSTEST         +.
             ö+    testar bussKANAL mot WINCHESTER  +.
             ö+++++++++++++++++++++++++++++++++++++++.

öselectar kortet först genom att skicka /01 och signalen C1*

BUSSTEST:movb #/00,BYTE       öinitiering av commando test drive ready
 movb #/00,BYTE+1      ödrive 0
        clrb BYTE+2
        clrb BYTE+3
        clrb BYTE+4
        clrb BYTE+5

        movb   WDOG,d7        ökick DOG

 jsr SELCNTR           öselectar kortet och genererar "SEL"-puls
 jsr WCOM              öskriver till controllen vilket kommando
 jsr GETSTAT           ökollar  commandot blivit rätt överfört

        rts

       ö**************************************************************
       ö**************************************************************
ö---------------------------
ösänder över kommand
ödet består av ett sex-byte block DCB
öbyte 0  765:class command 43210:opcode of the command
öbyte 1  765:LUN logical number 43210 logical diskadress 2
öbyte 2 7-0 logikal disk adress 1
öbyte 3 7-0 logical disk adress 0 (LSB)
öbyte 4 7-0 interleav or blockcount
öbyte 5 7:noll vid normal operation
ö 6:noll vid normal operation
ö 5-3 set to ZERO
ö 2:BUFFERSTEP OPTION (200 MIKROSEK PER STEP)
ö 1:half step option for tandon drives
```

```
ö 0:halfstep option of SEAGATE and INSTRUMENT drives
ö---------------------------
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN WCOM              +.
              ö+      skriver COMMANDO controllen     +.
       ö+          INDATA:BYTE-BYTE+5              +.
              ö+++++++++++++++++++++++++++++++++++++++.

WCOM:  jsr REQWC              öcontrollen vill ha kommand
         ö(controller request)
 movb BYTE,  COMDAT
 movl #1,dl
 jsr DELAYl

        jsr REQWC             öcontrollen vill ha kommand
         ö(controller request)
 movb BYTE+1,COMDAT    öskickar ut data och genererar ACK "0"
      movl #1,dl
 jsr DELAYl

        jsr REQWC             öcontrollen vill ha kommand
         ö (controller request)
 movb BYTE+2,COMDAT
 movl #1,dl
 jsr DELAYl

        jsr REQWC             öcontrollen vill ha kommand
         ö(controller request)
 movb BYTE+3,COMDAT
 movl #1,dl
 jsr DELAYl

        jsr REQWC             öcontrollen vill ha kommand
         ö(controller request)
 movb BYTE+4,COMDAT
 movl #1,dl
 jsr DELAYl

        jsr REQWC             öcontrollen vill ha kommand
         ö(controller request)
 movb BYTE+5,COMDAT
 movl #1,dl
 jsr DELAYl

        movb   WDOG,d7        ökick DOG
 rts

       ö**************************************************************
       ö**************************************************************
öfår tillbaka STATUSBYTE (2ST)
öbit 5 logical numberof driver 0 or 1
öbit 1 om set har det hänt något FEL i commandot. HOPPAR till felrutin
öandra bytet är ett nollbyte som talar om att kommandot är klart

              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN GETSTAT           +.
              ö+    testar bussKANAL mot WINCHESTER   +.
              ö+++++++++++++++++++++++++++++++++++++++.


GETSTAT:jsr REQRD             ödatorn vill sända tillbaka status
        movb COMDAT,d3
 andb #/02,d3
 cmpb #2,d3
```

```
        beq ERROR              öhoppar snabbt ut för att hämta FELKOD
 movl #CONOKEY,a1      ökontrollen okey
 jsr OUTPUT
 movl #/fff,d1
 jsr DELAY1
 rts        öhämtar komplett felkod

       ö**********************************************************************
       ö**********************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN ERROR             +.
             ö+  Hämtar error cod från controllen 4st+.
             ö+++++++++++++++++++++++++++++++++++++++.

öhämtar komplett felstatus från kontrollen
öcontrollen RETUNERAR fyra st byten tillbaka

ERROR:  clrl d3
        clrl REL               öför errtype utskrift
 jsr DIOD1       öresetar lysdioderna

        movb #/03,BYTE         öinitiering commando REQEST SENSE STATUS
 movb #/00,BYTE+1      ödrive 0
        clrb BYTE+2
        clrb BYTE+3
        clrb BYTE+4
        clrb BYTE+5

        jsr SELCNTR            ögenererar SELPULS till CONTROLLER O
        ö(byglad)
 jsr WCOM               öskriver ut kommandot

 movl #ERRCODE,a1       öERRCODE OCH ERRTYPE PÅ DIOD OCH SKÄRM
 jsr OUTPUT


 jsr REQRD1             öom kontrollen vill sända något
        movb ERRDAT,d3
 movl d3,a0             öskriver ut felcod på skärm
 jsr ADD
 jsr DIOD1

 jsr REQRD1              öom kontrollen vill sända något
        movb ERRDAT,d3

 jsr REQRD1              öom kontrollen vill sända något
        movb ERRDAT,d3

 jsr REQRD1             öom kontrollen vill sända något
        movb ERRDAT,d3


 movl #BUSSFEL4,a1
 jsr OUTPUT
 movl #/7ffff,d1
        jsr DELAY1             öfelbyte fyra

 jmp BUSS               ögör omtest


       ö**********************************************************************
       ö**********************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN REQWC             +.
```

```
            ö+ kollar om kontrollen vill ha COMMAND +.
            ö+++++++++++++++++++++++++++++++++++++++.


REQWC:  movl #0,d2
REQ2: addb #1,d2
 cmpb #/ff,d2
 beq REQ1              öfel i kommunikationen
 movb CARDSTA,d3
        movb   WDOG,d7        ökick DOG
 andb #/0f,d3
 cmpb #/0d,d3        ö d0:"1",d1:"0",d2:"1",d3:"1"
 bne REQ2

 rts


REQ1:   movb CARDSTA,d3
        jsr DIOD1
 movl #COMERR,a1    öcontrollen inte vill ha commando
 jsr OUTPUT
 movl #/ffff,d1
 jsr DELAY1
        movb   WDOG,d7        ökick DOG
 jmp BUSS       ögör omtest



       ö**************************************************************
       ö**************************************************************
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN REQWD            +.
            ö+ kollar om kontrollen vill ha DATA OBS+.
            ö+++++++++++++++++++++++++++++++++++++++.


REQWD: movb CARDSTA,d3
        movb   WDOG,d7        ökick DOG
 andb #/07,d3
 cmpb #/0a,d3        öbit I-/O "1",C-/D "0"
 bne REQWD
 rts

       ö**************************************************************
       ö**************************************************************
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN REQR             +.
            ö+    kollar om controllen vill SÄNDA  +.
            ö+++++++++++++++++++++++++++++++++++++++.


REQRD:  movl #/00,d2
RRD:    addl #1,d2            öom kontrollen inte vill sända kommando
 cmpl #/fff,d2
 beq RRD1
 movb CARDSTA,d3

        movb   WDOG,d7        ökick DOG
 andb #/0f,d3          öom controllen vill sända över errorkod
 cmpb #/05,d3          öC-/D är "1" I-/O är "0"
 bne RRD

 rts
```

```
REQRD1: movl #/00,d2          öKOLLAR OM CONTROLLEN VILL SÄNDA
                ö ERROR STATUS (DATA)
RRD2:   addl #1,d2            öom kontrollen inte vill sända kommando
 cmpl #/fff,d2
 beq RRD1
 movb CARDSTA,d3

        movb   WDOG,d7        ökick DOG
 andb #/0f,d3          ökollar om controllen vill sända
        ööver errorkod
 cmpb #/07,d3          öC-/D är "1" I-/O är "0"
 bne RRD2

 rts



RRD1:  movb CARDSTA,d3
        jsr DIOD1       öskriver ut statusregistret
 movl #SENDERROR,a1   öcontrollen inte vill sända status
 jsr OUTPUT
 movl #/ffff,d1
 jsr DELAY1
 jmp BUSS              ögör omtest

       ö**************************************************************
       ö**************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+            SUBRUTIN WBUSY           +.
             ö+     kollar om kontrollen är AKTIV   +.
             ö+++++++++++++++++++++++++++++++++++++++.


WBUSY:  movl #0,d2
WBU1: addb #1,d2
 cmpb #/ff,d2
 beq WBU2       öom controllen inte svarar på SELECT
  movb CARDSTA,d3
        movb   WDOG,d7        ökick DOG
 andb #/04,d3          ökollar busy biten "inverterad"
 cmpb #/04,d3
 bne WBU1

 rts

WBU2: movb CARDSTA,d3
        jsr DIOD1
 movl #OBUSY,a1        ökontrollen inte vill bli aktiv
 jsr OUTPUT
 movl #/7ffff,d1
 jsr DELAY1
 jmp BUSS              ögör omtest
       ö**************************************************************
       ö**************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+            SUBRUTIN NOWBUSY         +.
             ö+     kollar om kontrollen är ledig   +.
             ö+++++++++++++++++++++++++++++++++++++++.

NOWBUSY:movl #0,d2
OBU2: addl #1,d2
```

```
 cmpl #/fff,d2
 beq OBU1              ökontrollen blir inte ledig
 movb CARDSTA,d3
       movb   WDOG,d7        ökick DOG
 andb #/04,d3          ökollar busy biten om inverterad
 cmpb #/00,d3       öhoppar ut på aktiv "0" (inverterad)
 bne OBU2

 rts

OBU1: movb CARDSTA,d3
       jsr DIOD1
 movl #NOBUSY,a1
 jsr OUTPUT       ökontrollen vill inte bli ledig
 movl #/7ffff,d1
 jsr DELAY1
 jmp BUSS          ögör omtest

       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN SELCNTR          +.
             ö+         selectar controllen         +.
             ö+++++++++++++++++++++++++++++++++++++++.

ökollar att controllen inte är upptagen
ösänder en selectpuls till controllen

SELCNTR:jsr NOWBUSY       öatt kontrollen inte är upptagen
 movb #/01,DATASTROBE  ölatchar ut #/01 till controllen
 movb #/01,SELSTROB    ögenerera SELECTSTROBE aktiv låg
 jsr WBUSY
 rts

       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN SCC              +.
             ö+          testar SCC                 +.
             ö+++++++++++++++++++++++++++++++++++++++.

SCC:
          rts
       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN MENY             +.
             ö+          SKRIVER UT MENY            +.
             ö+++++++++++++++++++++++++++++++++++++++.

MENY:     movb #0,d3
            jsr DELAY1         önollställer sjuseg
            movl #TEXT3,a1
            jsr OUTPUT
            rts

       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+        SUBRUTIN   HIGH              +.
             ö+ Tar hand om bus interupt när vi     +.
             ö+kollar om det finns 1M primärminne   +
             ö+ i maskin                            +
             ö+++++++++++++++++++++++++++++++++++++++.
```

```
HIGH:   movb WDOG,d7     ökick watchdog
 movl #BUSSERROR,a1
 jsr OUTPUT
 movl #/ffff,a1
 jsr DELAY
        jmp WE1          ögår till MENY

       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+              RESETPROGRAM           +.
             ö+ hoppar hit vid reset av cpu         +.
             ö+ kollar i couse registret om det är  +.
             ö+paritetsfel eller vanlig reset       +.
             ö+Återhopps adressen läggs i a5        +.
             ö+GÄLLER VID DOM FÖRSTA 200 MASKINERNA +.
             ö+++++++++++++++++++++++++++++++++++++++.

RESET:       movb WDOG,d7     öhämtar causregistret
             andb #/02,d7     ötar fram paritetsbiten
             cmpb #0,d7
             beq MAIN         öom inte paritetsfel hoppartill mainprogram

             movl FEL33,a1
             jsr OUTPUT       öskriverut att det blivit fel     ,CAUSEREG
             jmp RESET        öprogrammet stoppar


       ö*************************************************************
       ö*************************************************************
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+              PARITETSINTERRUPT      +.
             ö+ hoppar hit vid PARITETSFEL av cpu   +.
             ö+ kollar i couse registret om det är  +.
             ö+ paritetsfel                         +.
             ö+ Återhopps adressen läggs i a5       +.
             ö+ GÄLLER VID DOM > 200 MASKINERNA     +.
             ö+++++++++++++++++++++++++++++++++++++++.

NMI:    movb WDOG,d3     öhämtar causregistret och skriver
     öut det på DIOD
   jsr DIOD1
   movb d3,d4
           andb #/02,d4     ötar fram paritetsbiten
           cmpb #/02,d4
           beq MI           öom paritetstesten fungererar riktigt

           movl#FEL3,a1
           jsr OUTPUT       öfelaktiga paritetskretsar eller annat fel
     ösom ger interupt

   movl #FE3,a1     öskriver ut CAUSE registret
   jsr OUTPUT
   jsr DIOD1


   cmpb #/ff,d6      öom paritetstest
   bne MI2
   jmp MI6


MI:    movl #RT3,a1
   jsr OUTPUT     ö PARITESKRETSARNA fungerer
```

```
    cmpb #/ff,d6      öom paritetstest
    bne MI2
MI6:    movl #PARFEL,a1  ödet har blivit fel vid paritetstesten
    jsr OUTPUT

MI2:    movl #PARTEST,a1 ötest av parit gjord
    jsr OUTPUT
    movl #/7ffff,d1

    jsr DELAY1

    movb #/0c,PARIT ösläcker PARITETSFELLAMPAN
     öoch sätter PARST

    movl #87,d1
    jsr SJUSEG1
    jmp WE1     öhoppar till meny

           rte

        ö*****************************************************************
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+              SUBRUTIN EST             +.
              ö+ testar att lådans sjuseg och dioder  +.
         ö+ fungerar
              ö+++++++++++++++++++++++++++++++++++++++.

EST:    movb #/01,d3
EST2:       jsr DIOD1
    movl #/ffff,d1
    jsr DELAY1
    cmpb #/80,d3
    beq EST3
    rolb #1,d3
    jmp EST2

EST3:       clrb d3
    jsr DIOD1
    movl #92,d1
    jsr SJUSEG1
    clrl d4

EST4:    movl #1,d1
    jsr SJUSEG1
           movl #/ffff,d1
    jsr DELAY1
    addb #1,d4
    cmpb #10,d4
    bne EST4
           clrl d4

EST5:       movl #10,d1
           jsr SJUSEG1
           movl #/ffff,d1
    jsr DELAY1
    addb #1,d4
    cmpb #9,d4
    bne EST5

    movl #8,d1
    jsr SJUSEG1
    rts
```

```
        ö**************************************************************
              ö++++++++++++++++++++++++++++++++++++++++.
              ö+              SUBRUTIN EXTERN          +.
              ö+ Laddar in program från externa eprom +.
              ö++++++++++++++++++++++++++++++++++++++++.

EXT:          movl #/300,dlölogiska adresser 0000-8000 = 32K (20000-28000)
              jsr PAGTRAN2
      movl #/180000,a4
EXT2:     movb (a4)+,d3
     jsr DIOD1
     movb #/ffff,d1
     jsr DELAY1
     jmp EXT2



              rts
        ö**************************************************************
        ö**************************************************************
              ö++++++++++++++++++++++++++++++++++++++++.
              ö+              MAINPROGRAM              +.
              ö+                                       +.
              ö++++++++++++++++++++++++++++++++++++++++.

MAIN:         movl #MAI1,a6
              movb #/00,d3          önollställer dioder
              jmp DIOD

MAI1:         movl #MAI2,a7
              jmp  SEGRAMM          ötest segram

MAI2:         movl #MAI5,a7
              jmp  PAGERAM          ötest pageram


MAI5:         movl #MAI6,a5         ötest av primärminnet
              jmp PRIMRAM


MAI6:         clrl d3
              jsr DIOD1             önollställer dioderna
              jsr DART              ötestar DART

WE1:          clrl d3
              jsr DIOD1             önollställer dioderna
              jsr MENY              öskriver ut test meny
              movb WDOG,d7          ökick watchdog
              jsr INPUT             öhämtar svaret
              movb IN,d1
              cmpb #49,d1
              bne WE2
              jsr  SJU
WE2:          cmpb #50,d1
              bne WE3
              jsr NIO
WE3:          cmpb #51,d1
              bne WE4
              jsr ELVA
WE4:          cmpb #52,d1
              bne WE5
              jsr TRETTON
WE5:          cmpb #53,d1
              bne WE6
```

```
            jsr EST                 ötester test lådan
WE6:        cmpb #54,d1
            bne WE7
            jsr CIO
WE7:        cmpb #55,d1
            bne WE8
            jsr FLOPPY
WE8:        cmpb #56,d1
            bne WE9
    jsr NVRAM

WE9:        cmpb #57,d1
            bne WE10
            jmp KOLL                ökollar paritetskretsarna

WE10:       cmpb #97,d1
            bne WE11
            jsr DMAM

WE11:       cmpb #98,d1             ötestar busskortet
    bne WE12
            jsr BUSS
            jmp WE1


WE12:       cmpb #42,d1
    bne WE1
    jsr KODE
    jmp WE1


        ö*************************************************************************
        ö*************************************************************************
KODE:       movl #KOD,a1
    jsr OUTPUT
    movl #/fffff,d1
    jsr DELAY1
    rts
        ö*************************************************************************
        ö*************************************************************************

TRA:     movl #trace,a1
         jsr OUTPUT
         jsr INPUT              öhämtar svaret
         movb IN,d1
         cmpb #106,d1
         beq DMAM
 rts


stopp:      movb WDOG,d7         ökick watchdog
           jmp stopp
        ö*************************************************************************
        ö*************************************************************************

TEXT4:    .ascii    "SKALL KLOCKAN TESTAS  (j/n) ^"
TEXT5:    .ascii    "DET SKRIVS UT (YYDDMMDDHHMMSS) '993112235956' PÅ SK
          .byte 13,10
          .ascii    "Värdet skall räkna upp tills det slår om till 00000
TEXT6:    .ascii    "SKALL TIDEN ÄNDRAS (j/n)^ "
TEXT7:    .ascii    "SKRIV IN (YYDDMMDDHHMMSS) PÅ TANGENTBORDET.^"
TEXT9:    .ascii    "ack^"
RTT1:    .ascii   "MINNES-AREAN ÄR BITTESTAD^"
```

```
RTT2:     .ascii    "MINNES-AREAN ÄR ADRESSTESTAD^"
RTT3:     .ascii    "MINNES-AREAN ÄR PARITETS TESTAD^"
FEL1:     .ascii    "SKRIV ELLER BIT-FEL I MINNES-AREAN^"
FEL2:     .ascii    "ADRESS-FEL I MINNES-AREAN^"
FEL3:     .ascii    "KONTROLL KRETSARNA FELAKTIGA FOR PARITETSTEST (ELLER
FE3:      .ascii    "SKRIVER UT CAUSE REGISTRET PÅ DIODERNA^"
RT3:      .ascii    "PARITETS KRETSARNA FUNGERAR (CAUSEREGISTRET SKRIVS U
FEL33:    .ascii    "FEL PA PARITETSKRETSARNA^"
PARFEL:   .ascii    "DET HAR BLIVIT FEL I PARITETSTEST^"
PARTEST:.ascii      "PARITETS TEST GJORD^"


FEL4:     .ascii    "PARITETSFEL I PARITETSKRETS^"
RI1:      .ascii    "SKALL MINNES-AREAN PARITETSTESTAS (j/n)^"
RI2:      .ascii    "SKALL MINNES-AREAN BITTESTAS (j/n)^"
RI3:      .ascii    "SKALL MINNES-AREAN ADRESSTESTAS (j/n)^"

FLOTEXT1:.ascii     "SKRIV IN TEXT FOR FLOPPY SKRIVNING/LÄSNING^"
FLOTEXT:.ascii      "OM SKRIVNING/ LÄSNING FUNGERAR SKRIVS TEXTEN UT PÅ S

FLOFEL1:.ascii      "FLOPPYN EJ ANSLUTEN ELLER FEL PÅ KONTROLLEN^"
FLOFEL2:.ascii      "STEGNING TILL FEL SPÅR/ KONTROLLEN FELAKTIG/ FLOPPYN
FLOFEL3:.ascii      "FELAKTIG SKRIVNING(STATUS PÅ LYSDIOD) .GOR OMSKRIVNI
FLOPWR :.ascii      "FLOPPY STATUS 'SKRIVNING' PÅ LYSDIODERNA^"
FLOPRD :.ascii      "FLOPPY STATUS 'LÄSNING' PÅ LYSDIODERNA^"
FLOREERR:.ascii     "LÄSNINGEN FUNGERAR EJ RIKTIGT ,STATUS PÅ LYSDIODERNA

DMAFEL1:.ascii      "DMA OVERFORINGEN EJ GENOMFORD (GOR OMTEST)^"
DMAFEL2:.ascii      "NOT END OF BLOCK^"
DMAFEL3:.ascii      "TEST MONSTRET FELAKTIGT OVERFORT (GOR OMTEST)^"
DMAFEL4:.ascii      "OVERFORINGEN HAR BLIVIT KOPIERAD TILL ANNAT BLOCK (G
AMD:      .ascii    "DMA SVARAR PÅ REDY-SIGNALEN ^"
AMD0:     .ascii    "DMA0 FRÅN FLOPPY TILL MINNE FUNGERAR .INGA FEL HITTA
DMAVI:     .ascii   "DMA STATUS  PÅ LYSDIODERNA^"

RAMFEL:   .ascii    "DET AR FEL PÅ NVRAM ELLER OMGIVNING (GOR OMTEST)^"
RAMOK:    .ascii     "NVRAMMET OKEY^"

BUSS1:    .ascii    "ANGER VAR KORTET SITTER PÅ LYSDIODERNA^"
BUSS2:    .ascii    "STATUSEN FOR ISATT KORT^"
BUSSFEL4:.ascii     "GOR OMTEST ( RESET PÅ SIGNAL C3*)^"

SENDERROR:.ascii    "CONTROLLEN VILL INTE SÄNDA STATUS,STATUS PÅ LYSDIODE
COMERR:     .ascii   "CONTROLLEN TAR INTE KOMMANDO,STATUS PÅ LYSDIODERNA
OBUSY:    .ascii   "CONTROLLEN SVARA INTE PÅ SEL STROBE.STATUS VISAS PÅ
NOBUSY:   .ascii    "CONTROLLEN VILL INTE BLI LEDIG ,STATUS PÅ LYSDIODER

BUS0:     .ascii    "XEBEC KONTROLLEN INKOPPLAD PÅ BUSS0^"
BUS0E:    .ascii    "XEBEC KONTROLLEN INKOPPLAD PÅ BUSS0 EXTERN^"
BUS1:     .ascii    "XEBEC KONTROLLEN INKOPPLAD PÅ BUSS1^"
BUS2:     .ascii    "XEBEC KONTROLLEN INKOPPLAD PÅ BUSS2^"
CARDFEL:.ascii      "FEL PÅ KORTVALET.OMTEST GORS^"
CARDRAT:.ascii      "KORTVALET OKEY (EXTERN BUSS)^"
ACK:      .ascii    "SLÅ OM BRYTAREN FOR ACK-SIGNALEN^"
ADDFEL:   .ascii    "SKRIVSVÅRIGHETER I MINNET^"
CONOKEY:.ascii      "CONTROLLEN ÄR OKEY (INGA FEL HITTADE)^"
ERRCODE:.ascii      "ERRCODE OCH ERRTYPE PÅ DIODERNA  OCH SKÄRM (SE DATA


VEKTOR:   .ascii    "ETT VEKTORAVBROTT HAR SKETT ,VEKTORN SKRIVS UT PÅ
CIOINT:   .ascii    "CIO HAR GJORT ETT INTERRUPT^"
DARTINT:.ascii      "DARTEN HAR GJORT ETT INTERRUPT^"
ADRINT:   .ascii   "DET HAR SKETT ETT ADRESSERROR^"
```

```
TEST:    .ascii  "TESTNING ..................^"
TET1:    .ascii  "TESTNING JÄMN PARITET.......^"
TET2:    .ascii  "TESTNING UDDA PARITET.......^"

BUSSERROR:.ascii "DET FINNS INTE 1M MINNE ,ELLER BUSS ERROR INTERUPT (

TEXT:     .ascii   "(* DARTEN TESTAS HÄR *)^"
TEXT1:    .ascii   "MATA IN NÅGRA TECKEN ,OM TECKNEN EKAS FUNGERAR DA
TEXT2:    .ascii   "    ^"
TEXT3:    .ascii   "(* MENY *)"
          .byte 13,10
          .ascii   "VAD SOM REDAN ÄR TESTAT:"
          .byte 13,10
          .ascii   "MACCEN  (SEGRAM ,PAGERAM) SAMT INITIERAD"
          .byte 13,10
          .ascii   "PRIMÄRMINNE 0K-2K (EJ ADRESS) BITTESTAD"
          .byte 13,10
          .ascii   "STACK INITIERAD "
          .byte 13,10
          .ascii   "VAL AV NEDAN :"
          .byte 13,10
          .ascii   "PRIMÄRMINNE  2K-256K       (1)"
          .byte 13,10
          .ascii   "PRIMÄRMINNE 256K-512K      (2)"
          .byte 13,10
          .ascii   "PRIMÄRMINNE 512K-768K      (3)"
          .byte 13,10
          .ascii   "PRIMÄRMINNE 768K-1M        (4)"
          .byte 13,10
          .ascii   "TEST AV LÅDA (SJUSEG+DIOD) (5)"
          .byte 13,10
          .ascii   "CIO+KLOCKA                 (6)"
          .byte 13,10
          .ascii   "FLOPPY                     (7)"
          .byte 13,10
          .ascii   "NVRAM                      (8)"
          .byte 13,10
          .ascii   "TEST AV PARITETSKONTROLL   (9)"
          .byte 13,10
          .ascii   "DMA                        (a)"
          .byte 13,10
          .ascii   "TEST AV BUSSKORT           (b)"
          .byte 13,10
          .byte 13,10
          .ascii   "VAl AV OVAN:^"


KOD:      .ascii   "HÅKAN HELDMANN 850516   24.S ^"


trace:   .ascii   "TRACE^"

test:    .ascii   "HÄR^"

        ö**************************************************************
```

```
.text


DIODLATCH       =       /0280
SEGCLOCK        =       /0380
SYSSP           =       /60800
WDOG            =       /80007
BURK            =       /20000          öreset av burk

SEGRAM          =       /80003
PROCSTART       =       /40
TASKREG         =       /80005
SEGSTART        =       /80003
STARTBIT        =       /01

PAGESTART       =       /80000
SPCTROL         =       /7fe00
PARIT           =       /7fe00
IOSTART         =       /fe000
BURK            =       /7ff00

BUS             =       /40008          ö resetvektor 512K
ö-----------------------------------------------------------
WR0A            =       /7f204          ö kontrollport A
WR0B            =       /7f200          ö kontrollport B
DARTA           =       /7f206          ö ut/in port A
DARTB           =       /7f202          ö ut/in port B
ö-----------------------------------------------------------
CIC             =       /7f700          öcio port c
CIB             =       /7f702          öcio port b
CIA             =       /7f704          öcio port a
CIK             =       /7f706          öcio kontrollport
ö-----------------------------------------------------------
FLOPCMD         =       /7f000          öcommandoreg
FLOPSTA         =       /7f000          östatusreg
FLOPTRACK       =       /7f002          öspårregistret
FLOPSEK         =       /7f004          ösektorregister
FLOPDATA        =       /7f006          öDATA register

FLOPCTR         =       /7fb00          öfloppy TTl control register
ö-----------------------------------------------------------
DMAMAP0         =       /7fd06
DMAMAP1         =       /7fd04
DMAMAP2         =       /7fd00

DMA0            =       /7f300
DMA1            =       /7f400
DMA2            =       /7f500

ö-----------------------------------------------------------
BUSSSTA         =       /7eca0          övar korten sitter

ERRDAT          =       /7e4a0          ögenererar INP          (W)
COMDAT          =       /7e4a0          ögenererar INP          (W)
SELCTRL         =       /7e4a4          öSELECT PULS C1*        (W)
SELSTROB        =       /7e4a4          ögenererar selectstrobe(W)

DATASTROBE      =       /7e4a0          ögenererar LATCH till
```

```
controllen

CARDSTA         =           /7e4a2          östatus på controllen  (R)

CONRESET        =           /7e4a8          ögenererar C3* puls för reset

ö-------------------------------------------------------------------
EPROM1          =           /180000         öEXTRA EPROM (add efter mac)
EPROM2          =           /188000
EPROM3          =           /190000
EPROM4          =           /198000
ö-------------------------------------------------------------------


IN              =           /60000          ötangenbords text (lagring)
REF             =           /60080          öreferens till klocka
MODE            =           /60081          övad som skall testas
LOOP            =           /60082          öloopräknare
REL             =           /60084

OPCODE          =           /60094          öopcode till NVRAM
DATA            =           /60090          ödata som skall skrivas in i
NVRAM
REGI            =           /60088          övilket register NVRAM

TRAC            =           /60088          öSPÅR FLOPPY
SEK             =           /60090          öSEKTOR FLOPPY
BYTE            =           /60094
SAVE            =           /60100          öspar AREA

MEMTEST         =           /60090          övisar att det är minnes test

AUTO            =           /60090          öom auto test

org:
.long SYSSP,RESET,HIGH,QW,QW1,QW2,QW3,QW4,QW5,QW6,QW7
.long
QW8,QW9,QW10,QW11,QW12,QW13,QW14,QW15,QW16,QW17,QW18,QW19,QW20,QW21
.long QW22,QW23,QW24,QW25,QW26,QW27,QW28,QW29



ö*************************************************************************
*
ö*************************************************************************
*
QW:     movb #3,d3
        jsr DIOD1
        movl #ADRINT,a1
        jsr OUTPUT                              öadressinterupt
        jmp TRAP
QW1:    movb #4,d3
        jsr DIOD1
        jmp TRAP
QW2:    movb #5,d3
        jsr DIOD1
```

```
        jmp TRAP
QW3:    movb #6,d3
        jsr DIOD1
        jmp TRAP
QW4:    movb #7,d3
        jsr DIOD1
        jmp TRAP
QW5:    movb #8,d3
        jsr DIOD1
        jmp TRAP
QW6:    movb #9,d3
        jsr DIOD1
        jmp TRAP
QW7:    movb #10,d3
        jsr DIOD1
        jmp TRAP
QW8:    movb #11,d3
        jsr DIOD1
        jmp TRAP
QW9:    movb #12,d3
        jsr DIOD1
        jmp TRAP
QW10:   movb #13,d3
        jsr DIOD1
        jmp TRAP
QW11:   movb #14,d3
        jsr DIOD1
        jmp TRAP
QW12:   movb #15,d3
        jsr DIOD1
        jmp TRAP
QW13:   movb #16,d3
        jsr DIOD1
        jmp TRAP
QW14:   movb #17,d3
        jsr DIOD1
        jmp TRAP
QW15:   movb #18,d3
        jsr DIOD1
        jmp TRAP
QW16:   movb #19,d3
        jsr DIOD1
        jmp TRAP
QW17:   movb #20,d3
        jsr DIOD1
        jmp TRAP
QW18:   movb #21,d3
        jsr DIOD1
        jmp TRAP
QW19:   movb #22,d3
        jsr DIOD1
        jmp TRAP
QW20:   movb #23,d3
        jsr DIOD1
        jmp TRAP
QW21:   movb #24,d3
        jsr DIOD1
        jmp TRAP
QW22:   movb #25,d3
```

```
        jsr DIOD1
        jmp TRAP
QW23:   movb #26,d3
        jsr DIOD1
        movl #CIOINT,a1
        jsr OUTPUT                          ö CIOINTERRUPT
        jmp TRAP
QW24:   movb #27,d3
        jsr DIOD1
        jmp TRAP
QW25:   movb #28,d3
        jsr DIOD1
        jmp TRAP
QW26:   movb #29,d3
        jsr DIOD1
        movl #DARTINT,a1
        jsr OUTPUT                          öDART ELLER SCC INT
        jmp TRAP
QW27:   movb #30,d3
        jsr DIOD1
        jmp TRAP
QW28:   movb #31,d3
        jsr DIOD1
        jsr NMI                             öparitetsfel
        jmp TRAP
QW29:   movb #32,d3
        jsr DIOD1
        jmp TRAP


TRAP:   movl #VEKTOR,a1
        jsr OUTPUT
        jmp stopp
```

```
ö*****************************************************************************
*
ö*****************************************************************************
*
          ö++++++++++++++++++++++++++++++++++++++++++++++++
          ö+                  SUBRUTIN                    +.
          ö+  lysdioder. indata i d3.                     +.
          ö++++++++++++++++++++++++++++++++++++++++++++++++.

DIOD:     clrl    d0
DIO:      addql   #1,d0
          movb    d3,DIODLATCH              ö lysdioderna .
          cmpl    #4,d0
          bne     DIO
          jmp     aöé


DIOD1:    clrl    d0
DIO1:     addql   #1,d0
```

```
          movb    d3,DIODLATCH            ö lysdioderna .
          cmpl    #4,d0
          bne     DIO1
          rts
ö*************************************************************************
**.
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+                SUBRUTIN               +.
              ö+  sjusegment räknas upp med ett.       +.
              ö+++++++++++++++++++++++++++++++++++++++.

SJUSEG:   clrl    d0
SJU22:    addql    #1,d0
          movb    #/00,SEGCLOCK         öräknar upp sjusegment.
          cmpl    #4,d0
          bne     SJU22
          jmp     aóé

ö*************************************************************************
**.
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+                SUBRUTIN               +.
              ö+  sjusegment räknas upp med d1 gånger  +.
              ö+++++++++++++++++++++++++++++++++++++++.

SJUSEG1:  clrl    d2
SJU2:     clrl    d0
SJU1:     addql    #1,d0
          movb    #/00,SEGCLOCK         öräknar upp sjusegment.
          cmpl    #4,d0
          bne     SJU1
          addl #1,d2
          cmpl d1,d2
          bne SJU2
          rts

ö*************************************************************************
**.
ö*************************************************************************
.
               ö+++++++++++++++++++++++++++++++++++++++.
               ö+                SUBRUTIN               +.
               ö+    delay rutin  .indata d1           +.
               ö+  ungefär 15mikrosek per enhet        +.
               ö+++++++++++++++++++++++++++++++++++++++.

DELAY:    clrl    d2
DEL:      addql   #1,d2
          movb    WDOG,d7           ökick DOG
          cmpl    d1 ,d2
          bne     DEL
          jmp aóé

DELAY1:   clrl    d2
DEL1:     addql   #1,d2
          movb    WDOG,d7           ökick DOG
          cmpl    d1 ,d2
          bne     DEL1
          rts
```

```
ö*****************************************************************
.
ö*****************************************************************
******

                    ö*******************************************
                    ö*          SUBRUTIN SEGDAFEL              *
                    ö*  om det blir BITfel i segment rammet   *
                    ö*                                         *
                    ö*******************************************

SEGDAFEL:       movl #H3,a6
                movb d3,d5
                movb d3,d4
                clrl d3
                andb #/0f,d4
                cmpb #0,d4
                beq H4
                orb #/01,d3
H4:             andb #/f0,d5
                cmpb #0,d5
                beq H5
                orb #/02,d3
H5:             jmp DIOD
H3:             jmp AC                  ö testar på nytt (stopp)

ö*****************************************************************
******
ö*****************************************************************
******

                    ö*******************************************
                    ö*          SUBRUTIN SEGADFEL              *
                    ö*  om det är fel på adressbussen         *
                    ö*                                         *
                    ö*******************************************

SEGADFEL:   movb #/ff,d3            ötänder åtta lysdioder som
indekerar
            movl #FELADD2,a6        öfel på adressbussen
            jmp DIOD
FELADD2:    jmp AB                  ötestar på nytt  (stopp)

ö*****************************************************************
******

                    ö*******************************************
                    ö*          SUBRUTIN SEGRES                *
                    ö*  resetar alla adresser i segram        *
                    ö*  indata a0                              *
                    ö*******************************************


SEGRES:     movb #/40,TASKREG
            movl #/40,d2                        öprocess ETT
            movl #SEGSTART,a0                   östart adress segram
```

```
A11:        movb #/00,a0é
            addl #/100,a0
            movb #/00,a0é
            addl #/7f00,a0
            cmpl #/100003,a0                    ösista adressen
            bne  A11                            önästa process
            cmpb #/4f,d2                        öom process 15
            beq  A2
            addb #1,d2                          önästa process
            movb d2,TASKREG
            movl #SEGSTART,a0
            movb WDOG,d7                        ökick watchdog
            jmp A11
A2:         jmp a6é

ö*************************************************************************
******
ö*************************************************************************
******

                    ö*****************************************
                    ö*          SUBRUTIN SEGDATA             *
                    ö*  testar skrivbarhet i segram         *
                    ö*  för alla processer                  *
                    ö*****************************************


SEGDATA:    movb   WDOG,d7                      ökick DOG
            clrl d3                             öfel register
            movb #/01,d4                        östart mönstret
B1:         movb #PROCSTART,d0                  öprocess NOLL
B3:         movl #SEGSTART,a0
            movb d0,TASKREG                     öprocess x
            movb d4,d2                          ötestmönster
B0:         movb d2,a0é
            addl #/100,a0
            movb d2,a0é
            addl #/7f00,a0
            cmpl #/100003,a0                    öom sista adressen
            bne  B0
            addb #1,d0
            movb   WDOG,d7                      ökick DOG
            cmpb #/50,d0                        öom sista processen
            bne B3                              öMAGICbiten ej satt

ö-------------------------------------------------------------------ö
ö testar att det är inskrivet rätt på alla bitar ,i alla processer.  ö
ö-------------------------------------------------------------------ö

test1:      movb   WDOG,d7                      ökick DOG
            movb #/40,d0                        öprocess noll
            movl #SEGSTART,a0                   östart adress
B2:         movb a0é,d2                         öhämtar
            eorb d4,d2                          öom någon felaktig bit
            orb d2,d3                           öläggs den i d3
            addl #/100,a0
            movb a0é,d2
            eorb d4,d2                          öom rätt bit
            orb d2,d3
```

```
            addl#/7f00,a0
            cmpl #/100003,a0            öom sista adressen
            bne B2
            movb   WDOG,d7              ökick DOG
            lslb #1,d4                  öskiftar testmönstret ett HACK
            cmpb #/80,d4                öom alla bitar är testade
            bne B1                      öMAGICbiten testas ej här
            cmpb #0,d3
            bne SEGDAFEL                öom bitfel

            jmp a5é

ö*************************************************************************
******
ö*************************************************************************
******
                  ö*****************************************
                  ö*          SUBRUTIN SEGADD              *
                  ö*  testar adressbus till MAC           *
                  ö*  indata a0 (adressen det är skrivet  *
                  ö*  på.                                 *
                  ö*****************************************

SEGADD:
            movb WDOG,d7                ökick watchdog
            movb #/40,TASKREG           öprocess NOLL
            movb #/7f,d1
            movb d1,a0é                 ölägger ut testmönster
            movl #SEGSTART,a1           ötestar alla andra adresser
A6:         cmpl a1,a0                  ösä ingen kopiering har skett
            beq A4                      öi process NOLL
A7:         movb a1é,d2
            andb #/7f,d2                ömaskar av magic biten
            cmpb #0,d2
            bne SEGADFEL                öadressbussen troligen felaktig
A4:         addl #/100,a1
            cmpl a1,a0
            beq A5
            movb a1é,d2
            andb #/7f,d2                ömaskar av magic biten
            cmpb #0,d2
            bne SEGADFEL                öadressbussen troligen felaktig
A5:         addl #/7f00,a1
            cmpl #/100003,a1            ösista adressen i processen NOLL
            beq  A50
            cmpl a1,a0
            beq A4
            jmp A7
ö-------------------------------------------------------------------------
-----ö
ö testar i övriga processer på adress a0 att inte /7f är skrivet.
     ö
ö processnummret pekas ut av d0
     ö
ö adressen pekas ut av a0
     ö
ö-------------------------------------------------------------------------
-----ö
```

```
A50:        movb WDOG,d7              ökick watchdog
            movb #/41,d4              öprocess ETT
A9:         movb d4,TASKREG
            movb a0é,d5
            cmpb #0,d5
            bne SEGADFEL              ödata har blivit skrivit i fel
process
            addb #1,d4
            cmpb #/50,d4              öom process 15 är testad
            bne A9
            movb #/40,TASKREG
            movb #/00 ,a0é            öskriver noll på adressen i
process NOLL
A91:        jmp a6é

ö*************************************************************************
******
ö*************************************************************************
******
ö*************************************************************************
******
                  ö*****************************************
                  ö*          SUBRUTIN SEGRAM              *
                  ö*  testar MAC + sätter upp den         *
                  ö*****************************************

ö+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
SEGRAMM:    movl #AC,a6               ösjusegment ETT
            jmp SJUSEG                ösegmentramets bittestas
ö+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
ö---------------------------------------------------ö
ö                    bittestar segram               ö
ö---------------------------------------------------ö

AC:         movl #G8,a5
            jmp SEGDATA             ötestar bitmässigt i segram

G8:         movl #B11,a6
            jmp SEGRES              öresetar segram för adresstest


ö+++++++++++++++++++++++++++++++++++++++++++++++
B11:        movl #AB,a6             ösjusegment TVÅ
            jmp SJUSEG              ösegramets adressbuss skall testas
ö+++++++++++++++++++++++++++++++++++++++++++++++
ö-----------------------------------------------------ö
ö               testar adressbuss  segram             ö
ö-----------------------------------------------------ö
AB:         movl #SEGSTART,a0
A12:        movl #A23,a6
            jmp SEGADD              ötestar adressbuss ,segram
A23:        addl #/100,a0           öA8
            movl #A10,a6
            jmp SEGADD
A10:        addl #/7f00,a0          öA15-A18
            cmpl #/100003,a0         ösista adressen
            bne A12

ö+++++++++++++++++++++++++++++++++++++++++++++++
```

```
            movl #B14,a6            ösjusegment TRE
            jmp SJUSEG              ösegramets görs TRANSPARENT
ö++++++++++++++++++++++++++++++++++++++++

B14:        movl #B12,a6
            jmp SEGTRAN             ögör segmentrammet transparent

B12:        jmp a7é

ö*************************************************************************
******
ö*************************************************************************
******
                    ö*****************************************
                    ö*          SUBRUTIN SEGTRAN             *
                    ö*  gör segmentramet transparent 512K   *
                    ö*****************************************

SEGTRAN:    movb #/40,TASKREG       öprocess NOLL
            movl #/80003,a0
            clrl d2                 östart adress
C2:         movb d2,a0é             öskriver in det i segram
            addb #1,d2
            movb WDOG,d7            ökick watchdog
            addl #/8000,a0          önästa adress i SEGRAM
            cmpl #/100003,a0        öom sista adress A19=0 (A8=0)
            bne  C2
            jmp a6é

ö*************************************************************************
******
ö*************************************************************************
******
                    ö*****************************************
                    ö*          SUBRUTIN PAGRES              *
                    ö*  resetar pageram                     *
                    ö*****************************************

PAGERES:    movb #/00,d2                östart segram adress 80003
E2:         movb d2,SEGSTART            öläggs på första adressen
            movl #PAGESTART,a0          östart pageram
E1:         movw #/0000,a0é             öskriver NOLLOR
            addl #/800,a0
            cmpl #/88000,a0             ösista adressen skriven i pageram
 15sidor
            bne E1
            addb #1,d2                  önästa adress från segram
            cmpb #/40,d2                öom sista adressen i segram 64st
block
            bne E2
            movb WDOG,d7                ökick watchdog
            jmp a5é

ö*************************************************************************
******
                    ö*****************************************
                    ö*          SUBRUTIN PAGADFEL            *
                    ö*  om det är felö i adressavkodning     *
                    ö*****************************************
```

```
PAGEADFEL:  movl #H1,a6
            movb #/ff,d3
            jmp DIOD
H1:         jmp C8                  ögör ett nytt försök (stopp)

ö*************************************************************************
******
ö*************************************************************************
******
                    ö*****************************************
                    ö*          SUBRUTIN PAGEDAFEL           *
                    ö*  om det är felö i bittesten          *
                    ö*****************************************

PAGEDAFEL:    clrl d6
              movw d3,d5
              andw #/000f,d5
              cmpw #0,d5
              beq H6
              orb #/01,d6           öfel i pageram 18G
H6:           movw d3,d5
              andw #/00f0,d5
              cmpw #0,d5
              beq H7
              orb #/02,d6           öfel i pageram 22G
H7:           movw d3,d5
              andw #/0f00,d5
              cmpw #0,d5
              beq H9
              orb #/04,d6           öfel i pageram 20G
H9:           movw d6,d3
              movl #H2,a6
              jmp DIOD
H2:           jmp C10               ögör ny test (stopp )

ö*************************************************************************
******
ö*************************************************************************
******
                    ö*****************************************
                    ö*          SUBRUTIN PAGEDATA            *
                    ö*          bitestar pageramet           *
                    ö*****************************************

PAGEDATA:   movb WDOG,d7            ökick watchdog
            clrl d3
            movw #/0001,d5          ö/0200???  starttestord
G4:         movb #/00,d4
G2:         movb d4,SEGSTART        östartadress till segram
            movl #PAGESTART,a0      östartadress PAGERAM
G1:         movw d5,a0é
            addl #/800,a0
            cmpl #/88000,a0         ösista adressen
            bne G1
            movb WDOG,d7            ökick watchdog
            addb #1,d4
            cmpb #/40,d4            öom sista segadress
            bne G2
```

```
            movb WDOG,d7            ökick watchdog

            movb #/00,d4            ötestar att rätt bitfel
G10:        movb d4,SEGSTART
            movl #PAGESTART,a0
G3:         movw a0é,d6
            eorw d5,d6              öger felaktiga bitar
            orw d6,d3               öläggs i d3
            addl #/800,a0
            cmpl #/88000,a0         ösista adressen
            bne G3
            movb WDOG,d7            ökick watchdog
            addb #1,d4
            cmpb #/40,d4
            bne G10
            rorw #1,d5              öroterar vänster av testmönstret
            cmpw #/4000,d5
            bne G4                  önästa testmönster
            andw #/c3ff,d3
            cmpw #0,d3
            bne PAGEDAFEL           öom bitfel
            jmp a6é
ö*************************************************************************
******
ö*************************************************************************
******
                    ö*****************************************
                    ö*          SUBRUTIN PAGEADD             *
                    ö*  testar adressbuss hos PAGERAM       *
                    ö*****************************************

PAGEADD:    movb WDOG,d7                ökick watchdog
            movb #/00,d4
F5:         movb d4,SEGSTART            östartadress till segram
            movw #/c3ff,d5              ötestord
F3:         movl #PAGESTART,a0          östartadress PAGERAM
F6:         movw d5,a0é
            movl #PAGESTART,a1          ötest i övriga adresser
F2:         cmpl a1,a0
            beq F1
            movw a1é,d6                 öläser minnesinnehållet
            andw #/c3ff,d6
            movb WDOG,d7                ökick watchdog
            cmpw #/0000,d6              öskall inte stå nått
            bne  PAGEADFEL
F1:         addl #/800,a1
            cmpl #/88000,a1             ösista adressen
            bne F2
            movb WDOG,d7                ökick watchdog
            movw #/0000,a0é
            addl #/800,a0               önästa adress i pageramet
            cmpl #/88000,a0
            bne F6
            movb #/00,d6                ötest på a0 i alla segadresser
F8:         movb d6,SEGSTART            öförsta segadress
            cmpb d4,d6
            beq F7
            movw a0é,d2
            andw #/c3ff,d2
```

```
            cmpw #/0000,d2
            bne PAGEADFEL
            movb WDOG,d7                    ökick watchdog
F7:         addb #1,d6
            cmpb #/40,d6
            bne F8

F9:         movb WDOG,d7                    ökick watchdog
            addb #1,d4
            cmpb #/40,d4                    öom sista adress i segram 64st
block
            bne F5                          öny testadress


            jmp a5é
ö*************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+          SUBRUTIN PAGERAM             +.
              ö+ testar pagerammet                     +.
              ö+++++++++++++++++++++++++++++++++++++++++.

PAGERAM:

ö+++++++++++++++++++++++++++++++++++++++++++++++
            movl #C16,a6            ösjusegment FYRA
            jmp SJUSEG              öpagerammet bittestas
ö+++++++++++++++++++++++++++++++++++++++++++++++

C16:        movl #C10,a5
            jmp PAGERES

ö-----------------------------------------------------ö
ö         testar skrivbarheten i pagerammet           ö
ö-----------------------------------------------------ö

C10:        movl #C11,a6
            jmp PAGEDATA

C11:       movl #C5,a5
           jmp PAGERES

ö+++++++++++++++++++++++++++++++++++++++++++++++
C5:         movl #C8,a6             ösjusegment FEM
            jmp SJUSEG              öpagerammet adressbuss test
ö+++++++++++++++++++++++++++++++++++++++++++++++
ö-----------------------------------------------------ö
ö  testar att adresseringen fungerar samt att ingen   ö
ö feladressering sker                                 ö
ö-----------------------------------------------------ö

C8:         movl #C12,a5
            jmp PAGEADD

C12:        jmp a7é


ö*************************************************************************
******
```

```
ö*****************************************************************
******
                ö*************************************
                ö*          SUBRUTIN PAGTRAN         *
                ö*  gör pagetramet transparent       *
                ö* för process NOLL ändast          *
                ö* INDATA d3                        *
                ö*************************************

PAGTRAN:        movb #/40,TASKREG           öprocess NOLL
                movl #/a0000,a0             östart adress pageram
                orl  #/4000,d3              ögör all minnesceller skriv
och
H11:            movw d3,a0é                 öläsbara
                addl #1,d3                  önästa inadress
                addl #/800,a0
                cmpl #/e0000,a0             öom 256K ,man räknar upp
128steg
                bne H11
                jmp a7é
ö---------------------------------------------------------------
PAGTRAN1:       movb #/40,TASKREG           öprocess NOLL
                movl #/a0000,a0             östart adress pageram
                orl  #/4000,d3              ögör all minnesceller skriv
och
H111:           movw d3,a0é                 öläsbara
                addl #1,d3                  önästa inadress
                addl #/800,a0
                cmpl #/e0000,a0             öom 256K ,man räknar upp
128steg
                bne H111
                rts
ö---------------------------------------------------------------

PAGTRAN2:       movb #/40,TASKREG           öprocess NOLL
                movl #/a0000,a0             östart adress pageram
                orl  #/4000,d3              ögör all minnesceller skriv
och
H112:           movw d3,a0é                 öläsbara
                movb WDOG,d7                ökick watchdog
                addl #1,d3                  önästa inadress
                addl #/800,a0
                cmpl #/a8000,a0             öom 32K ,man räknar upp
128steg
                bne H112
                rts

ö*****************************************************************
******
ö*****************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++.
                ö+             PRIMADD                  +.
                ö+     primärminnet adress testas       +.
                ö+++++++++++++++++++++++++++++++++++++++.

PRIMADD1:       movb AUTO,d1                ökollar om auto test
                cmpb #36,d1                 öom ¤
```

```
                beq JM1

                movl #RI3,a1
                jsr OUTPUT              öfrågar om ADRESSTEST SKALL
GöRAS
                jsr INPUT               ökollar svaret
                movb IN,d1
                cmpb #106,d1            öom j-tangenten
                bne TI79

RT12:           movl #TEST,a1
                jsr OUTPUT              ötestning utföres
                movl #/fff,d1
                jsr DELAY1

JM1:            movl #/20000,a0         östart adress primärminne
                clrl d6
PRI111:         clrl d3
T21:            cmpb #9,d3              öom NIO skrivningar =FEL
                beq PRI911
                addb #1,d3              öANTAL SKRIVNINGAR
                movl d6,a0é             ölägger ut testmönstret två
gånger
                movl d6,a0é
                movl a0é,d5
                movb d1,BURK
                movb WDOG,d7            ökick watchdog

                eorl d6,d5              ötar fram felaktigt ord (ej
inskrivet)
                cmpl #0,d5              öom inskrivet ord felaktigt
                bne T21                 ökorrekt läsning

T23:            addl #4,a0
                addl #1,d6

                movb WDOG,d7            ökick watchdog
                cmpl #/60000,a0         öom sista adressen 256K
                bne PRI111              ölägger ut ett långt word

ö---------------------------------------------------------------
        öläsning av utskrivet mönster

PRI31:          movl #/20000,a0         östart adress primärminne
                clrl d6
PRI53:          clrl d7
PRI51:          movl a0,d5
                cmpb #/280,d5
                beq RR1
                cmpb #/380,d5
                beq RR1
                movl a0é,d3
                addl #1,d7
                cmpl #9,d7
                beq PRI91               öOM NIO FELLÄSNINGAR HOPP
                eorl d6,d3              ötar fram felbit
                cmpl #0,d3
                bne PRI51               öadressfel
RRR:            movb WDOG,d7            ökick watchdog
```

```
RR1:            addl #4,a0
                addl #1,d6
                cmpl #/60000,a0          öom sista adressen 256K
                bne PRI53
                movb AUTO,d1             ökollar om auto test
                cmpb #36,d1              öom ¤
                beq TI79                 öskriver ej ut skrift
                movl #RTT2,a1            öadresstesten klar
                jsr OUTPUT
TI79:           rts

PRI91:          movl #FEL2,a1            öskriver ut adressfel
                jsr OUTPUT
                jsr ADD                  öskriver ut feladress på
skärm
                movl #/ffff,d1
                jsr DELAY1
                jmp RRR


PRI911:         movl #ADDFEL,a1          ösvårt att skriva
                jsr OUTPUT
                jmp RT12

ö-----------------------------------------------------------------
ö-----------------------------------------------------------------

PRIMADD2:       movb AUTO,d1             ökollar om auto test
                cmpb #36,d1              öom ¤
                beq JM2


                movl #RI3,a1
                jsr OUTPUT               öfrågar om ADRESSTEST SKALL
GöRAS
                jsr INPUT                ökollar svaret
                movb IN,d1
                cmpb #106,d1             öom j-tangenten
                bne TI35

RT21:           movl #TEST,a1            öatt vi testar
                jsr OUTPUT
                movl #/fff,d1
                jsr DELAY1

JM2:            movl #/20000,a0          öTESTAR DOM 2K-256K ADRESS
                clrl d6                  öförsta k:na
RI111:          clrl d3
TI21:           cmpb #9,d3               öom 9 skrivningar =FEL
                beq RI911
                addb #1,d3               öANTAL SKRIVNINGAR
                movl d6,a0é              ölägger ut testmönstret två
gånger
                movl d6,a0é
                movl a0é,d5
                movb d1,BURK
                movb WDOG,d7             ökick watchdog

                eorl d6,d5               ö felaktigt ord (ej
inskrivet)
```

```
                cmpl #0,d5                      öom inskrivet ord felaktigt
                bne TI21                        ökorrekt läsning

                addl #1,d6
                addl #4,a0
                movb WDOG,d7                    ökick watchdog
                cmpl #/5f800,a0                 öom sista adressen 256K
(??38000????)
                bne RI111                       ölägger ut ett långt word

ö---------------------------------------------------------
            öutläsning av inskrivet mönster

RI31:           movl #/20000,a0                 östart adress primärminne
                clrl d6
RI53:           clrl d7
RI51:           movl a0,d5
                cmpb #/280,d5
                beq RI96
                cmpb #/380,d5
                beq RI96
                movl a0é,d3
                addl #1,d7
                cmpl #9,d7
                beq RI91                        öOM NIO FELLÄSNINGAR HOPP
                eorl d6,d3                      ötar fram felbit
                cmpl #0,d3
                bne RI51                        öadressfel
RI94:           movb WDOG,d7                    ökick watchdog
RI96:           addl #4,a0
                addl #1,d6
                cmpl #/5f800,a0                 öom sista adressen 256K
                bne RI53

                movb AUTO,d1                    ökollar om auto test
                cmpb #36,d1                     öom $
                beq TI35                        öskriver ej ut skrift
                movl #RTT2,a1                   öadresstesten klar
                jsr OUTPUT

TI35:           rts

RI91:           movl #FEL2,a1                   öskriver ut adressfel
                jsr OUTPUT
                jsr ADD                         öskriver ut feladress på skärm
                movl #/ffff,d1
                jsr DELAY1
                jmp RI94



RI911:          movl #ADDFEL,a1                 ösvårt att skriva
                jsr OUTPUT
                jmp RT21                        ögor omtest

ö*****************************************************************
******
```

```
ö*****************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++
              ö+              PRIMDATA               +
              ö+       bitfel i primärminnet         +
              ö+ vissar felet och står kvar och      +
              ö+ snurrar på felaktig adress.         +
              ö+++++++++++++++++++++++++++++++++++++++

PRIMDATA1:        movb AUTO,d1          ökollar om auto test
                  cmpb #36,d1           öom ¤
                  beq JM3               öÄNDRING 850703

                  movl #RI2,a1
                  jsr OUTPUT            öfrågar om BITtest skall göras
                  jsr INPUT             ökollar svaret
                  movb IN,d1
                  cmpb #106,d1          öom j-tangenten
                  bne DA79

                  movl #TEST,a1
                  jsr OUTPUT

JM3:              movl #/01,d4          ötestmönter
                  clrl d3
DA81:             movl #/20000,a0       östart adress 256 första k:na


DA31:             jsr BIT               ölägger ut testmönster
                  cmpl #/60000,a0
                  bne DA31

DA71:             lslb #1,d4
                  cmpb #/00,d4
                  bne DA81
                  movb AUTO,d1              ökollar om auto test
                  cmpb #36,d1               öom ¤
                  beq DA79                  öskriver ej ut skrift
                  movl #RTT1,a1
                  jsr OUTPUT

DA79:             rts

ö*****************************************************************
******
ö*****************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+              PRIMRAM                +.
              ö+ här testar vi primärminnet          +.
              ö+++++++++++++++++++++++++++++++++++++++.


ö+++++++++++++++++++++++++++++++++++++++++++
PRIMRAM:  movl #PR01,a6       ösjusegment SEX
          jmp SJUSEG          öPARITETSBITEN sätts
ö+++++++++++++++++++++++++++++++++++++++++++
```

```
PR01:       movl #IOSTART,a0        öadress 504K-512K
            movl #/43fc,d3          östartadress FYSISK
P0:         movw d3,a0é             öpageram upsatt för I/O
            addl #1,d3
            addl #/800,a0
            cmpl #/100000,a0
            bne P0

ö+++++++++++++++++++++++++++++++++++++++++++++++++
ö vi sätter specialkontroll registret
öADRESS 7fe00 med 0c ,det gör att paritetsfel
öej gör  RESET på cpu:n  samt att paritetsbiten
öskrivs med fel paritet (den kan kollas då)
ö+++++++++++++++++++++++++++++++++++++++++++++++++

            movb #/0c,PARIT         öskriver i paritetsbiten

ö+++++++++++++++++++++++++++++++++++++++++++++++
            movl #PR0,a6        ösjusegment SJU
            jmp SJUSEG          öprimärminnet testas för
                                ö STACKEN 2K
ö+++++++++++++++++++++++++++++++++++++++++++++++




ö---------------------------------------------------------
ötestar 2K av primärminnet och sätter upp stacken
ö logisk adress stack 60800-60000
ö---------------------------------------------------------

PR0:        movl #PPR02,a7
            movl #/0000,d3          östart noll i primärminnet
            jmp  PAGTRAN            ögör pageram transparent  första 256K

PPR02:      movl #/01,d4            ötestmönter
            clrl d3
PDA8:       movl #/20000,a0         östart adress 2 första k:na

PDA3:       movw a0,d7              öKOLLAR SÅ ATT VI INTE FÅR SVAR FRÅN
BURKEN
            andw #/2ff,d7
            cmpw #/280,d7
            beq PTE60
ö    cmpw #/300,d7
ö    beq PTE60
            cmpw #/380,d7
            beq PTE60

PTE7:       movb d4,a0é             ölägger ut testmönstret
PTE3:       movb a0é,d5
            movb d3,BURK            öFÖR ATT RESETA BURKEN
            movb WDOG,d7            ökick watchdog


            eorb d4,d5              ötar fram felaktigt ord (ej inskrivet)
            cmpb #/00,d5            öom inskrivet ord felaktigt
            beq PTE4
            movl #PTE,a6
```

```
            movb d3,BURK            öFöR ATT RESETA BURKEN

            movb d5,d3
            jmp DIOD
PTE:        movl #PTE2,a6
            movl #/ffff,d1
            jmp DELAY
PTE2:       movl #PDA3,a6           öom skrivning på ord
            clrl d3
            jmp DIOD
PTE4:       addl #1,a0
            movb WDOG,d7            ökick watchdog
            cmpl #/20801,a0         ötest av 2K primärminne
            blt PDA3

PDA7:       lslb #1,d4
            cmpb #/00,d4
            bne PDA8

            movl #/e0000,a0         ö sätter upp MACCEN
            movl #/4000,d3          öför stacken 2K (60000-60800)
PPR35:      movw d3,a0é
            addl #1,d3
            orl #/4000,d3           öminnet skriv och läsbart
            movb WDOG,d7            ökick watchdog
            addl #/800,a0
            cmpl #/e0800,a0
            bne PPR35

            movl #SYSSP,a7          östart adress för stacken
            jmp a5é

ö+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
öOm farlig adress 280,380
ö+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


PTE60:      clrl d3                 ötest åtta gånger innan utskrift
PTE70:      addb #1,d3
            cmpb #9,d3
            beq PTE80
            movb d4,a0é
            movb d4,a0é
            movb a0é,d5
            movb d3,BURK            öresetar burk
            eorb d4,d5
            cmpb #0,d5
            bne PTE70
            jmp PTE4                öom det hela är rätt

PTE80:      movb d5,d3
            movl #PTE85,a6
            jmp DIOD
PTE85:      movl #PTE86,a6
            movl #/ffff,d1
            jmp DELAY
PTE86:      movl #PTE90,a6          öom skrivning på ord
            clrl d3
```

```
            jmp DIOD

PTE90:      jmp PTE60               öomförsök


ö-------------------------------------------------------------------------
-------

 ö+++++++++++++++++++++++++++++++++++++++++++
   öprimärminnet bittestas 2K-256K (1)
 ö+++++++++++++++++++++++++++++++++++++++++++

SJU:        movl #1,d1
            jsr SJUSEG1             ösjuseg 9
            movl #/01,d3
            jsr PAGTRAN1            ösätter upp MACEN för 2K-256K (1)
            movl #/01,d4            ötestmönter
            clrl d3
            movb AUTO,d1            ökollar om auto test
            cmpb #36,d1             öom ø
            beq PPDA8

            movl #RI2,a1
            jsr OUTPUT              öfrågar om BITtest skall göras
            jsr INPUT               ökollar svaret
            movb IN,d1
            cmpb #106,d1            öom j-tangenten
            bne PD23

PD233:      movl #TEST,a1
            jsr OUTPUT

PPDA8:      movl #/20000,a0         östart adress 32K och upp till 256K
            movl #/fffe0800,REL     öanger att adressen börjar från 2K
fffe8000
PPDA3:      jsr BIT
            cmpl #/5f800,a0         öom sista adressen 254K
            bne PPDA3

            lslb #1,d4
            cmpb #/00,d4
            bne PPDA8
            movb AUTO,d1            ökollar om auto test
            cmpb #36,d1             öom ø
            beq PD23                öskriver ej ut skrift
            movb WDOG,d7            ökick watchdog
            movl #RTT1,a1           öskriver ut att minnesarean är
bittestad

            jsr OUTPUT


 ö+++++++++++++++++++++++++++++++++++++++++++
            öprimärminnet adresstestas
 ö+++++++++++++++++++++++++++++++++++++++++++
PD23:       movl #1,d1
            jsr SJUSEG1             ösjuseg 10
            jsr PRIMADD2            ötestar adresseringen i primärminnet
```

```
            movb WDOG,d7            ökick watchdog

 ö++++++++++++++++++++++++++++++++++++++++++++
            öparitetstest
 ö++++++++++++++++++++++++++++++++++++++++++++

            movl #1,d1
            jsr SJUSEG1             ösjuseg 11
            jsr PARITS              öparitetsbit test
            movl #97,d1
            jsr SJUSEG1             ösjuseg 8 MENY
PD239:      rts

ö-------------------------------------------------------------------------
-------
 ö++++++++++++++++++++++++++++++++++++++++++++
NIO:      öprimärminnet  256K ANDRA
 ö++++++++++++++++++++++++++++++++++++++++++++

            movl #4,d1
            jsr SJUSEG1             ösjuseg 12
            movl #/80,d3
            jsr PAGTRAN1            ö256K (2)
 ö++++++++++++++++++++++++++++++++++++++++++++
            ö 256K (2)
 ö++++++++++++++++++++++++++++++++++++++++++++
            movl #/20000,REL        öadressen börjar på 256K
            jsr PRI                 öBIT ,ADRESS ,PARITETSTEST
            movl #94,d1
            jsr SJUSEG1             ösjuseg 8 MENY
PD234:      rts


ö-------------------------------------------------------------------------
-------


ö-------------------------------------------------
ötest om primärminnet är större än 512K
ö-------------------------------------------------
ELVA:       movl #/100,d3               ö sätter upp MACCEN för 256K (3)
            movb WDOG,d7                ökick watchdog
            jsr PAGTRAN1
            movl #/20000,a0
            movb #/ff,a0é               ötestar om det går att skriva i övre
512K

ö-------------------------------------------------------
öom ej 1M ges buss error här
öprogrammet hoppar till rutinen HIGH
ö-------------------------------------------------------

 ö++++++++++++++++++++++++++++++++++++++++++++
            ö 256K (3)
 ö++++++++++++++++++++++++++++++++++++++++++++

            movl #7,d1
            jsr SJUSEG1             ösjuseg 14
            movb WDOG,d7            ökick watchdog
```

```
          movl #/60000,REL     öanger adress 512K
          jsr PRI              öBIT ,ADRESS ,PARITETSTEST
          movl #91,d1          ösjuseg 8 MENY
          jsr SJUSEG1
PD235:    rts

ö-------------------------------------------------------------------
-------

ö-------------------------------------------------
ötest om primärminnet är större än 512K
ö-------------------------------------------------
TRETTON:  movl #/180,d3            ö sätter upp MACCEN för 256K (3)
          movb WDOG,d7             ökick watchdog
          jsr PAGTRAN1
          movl #/20000,a0
          movb #/ff,a0é            ötestar om det går att skriva i övre
512K

ö-------------------------------------------------
öom ej 1M ges buss error här
öprogrammet hoppar till rutinen HIGH
ö-------------------------------------------------


 ö+++++++++++++++++++++++++++++++++++++++++++
          ö 256K (4)
 ö+++++++++++++++++++++++++++++++++++++++++++

          movl #10,d1
          jsr SJUSEG1          ösjuseg 17
          movl #/180,d3        ö sätter upp MACCEN för 768K (4)
          movb WDOG,d7         ökick watchdog
          jsr PAGTRAN1
          movl #/a0000,REL     öadressen från 768K
          jsr PRI              öBIT ,ADRESS ,PARITETSTEST
          movl #88,d1
          jsr SJUSEG1          ösjuseg 8 MENY
PD236:    rts

ö*******************************************************************
******
ö*******************************************************************
******
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+          SUBRUTIN PRI                 +.
             ö+ testar bit ,adress och paritetsbit    +.
             ö+ i primärminnet för 256K               +.
             ö+++++++++++++++++++++++++++++++++++++++.

 ö+++++++++++++++++++++++++++++++++++++++++++
           öbittest 256K
 ö+++++++++++++++++++++++++++++++++++++++++++


PRI:      movb WDOG,d7         ökick watchdog
          jsr PRIMDATA1
```

```
 ö++++++++++++++++++++++++++++++++++++++++++++
            öadresstest 256K
 ö++++++++++++++++++++++++++++++++++++++++++++

          movl #1,d1
          jsr SJUSEG1          ösjuseg
          movb WDOG,d7         ökick watchdog
          jsr PRIMADD1         öadresstestar 256K


 ö++++++++++++++++++++++++++++++++++++++++++++
          öparitetstest
 ö++++++++++++++++++++++++++++++++++++++++++++

          movl #1,d1
          jsr SJUSEG1          ösjuseg
          jsr PARITS           öparitetsbit test
          movb WDOG,d7         ökick watchdog
          rts

ö*****************************************************************************
******
ö*****************************************************************************
******
              ö++++++++++++++++++++++++++++++++++++++++
              ö+           SUBRUTIN BIT               +
              ö+ Lägger ut bit mönster och testar     +
              ö+ så att det är rätt                   +
              ö+ INDATA d4,a0                         +
              ö++++++++++++++++++++++++++++++++++++++++


BIT:      movl a0,d7           öKOLLAR SÅ ATT VI INTE FÅR SVAR FRÅN
BURKEN
          andw #/03ff,d7       öom farlig adress hoppar vi till egen
rutin
          cmpw #/280,d7
          beq BIT20
          cmpw #/380,d7
          beq BIT20

          movb d4,a0é          ölägger ut testmönstret
          movb a0é,d5
          movb WDOG,d7         ökick watchdog


          eorb d4,d5           ötar fram felaktigt ord (ej inskrivet)
          cmpb #/00,d5         öom inskrivet ord felaktigt
          beq BIT4

BIT22:    jsr ADD              öskriver ut adress om BITfel
          movb d3,BURK         öresetar burk
          movl #FEL1,a1
          jsr OUTPUT           öskriver ut skriv eller bitfel

          movb d5,d3
          jsr DIOD1            öskriver ut på dioder
```

```
            movl #/ffff,d1
            jsr DELAY1
            clrl d3
            jsr DIOD1
            jmp BIT                 ögör om-försök med LÄSNING STÄR OCH
LOOPAR


BIT4:       addl #1,a0
            movb WDOG,d7            ökick watchdog
            rts


öOm vi adresserar på "burkens " adresser 280,300,380


BIT20:      clrl d3
BIT21:      cmpb #9,d3               öom NIO skrivningar =FEL
            beq BIT22
            addb #1,d3               öANTAL SKRIVNINGAR
            movb d4,a0é              ölägger ut testmönstret två gånger
            movb d4,a0é
            movb a0é,d5
            movb d1,BURK
            movb WDOG,d7             ökick watchdog

            eorb d4,d5               ötar fram felaktigt ord (ej
inskrivet)
            cmpb #/00,d5             öom inskrivet ord felaktigt
            bne BIT21                ökorrekt läsning

            jmp BIT4
ö*************************************************************************
******
ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++.
                ö+           SUBRUTIN AUTOTEST           +.
                ö+    testar minnet automatiskt          +.
                ö+++++++++++++++++++++++++++++++++++++++.


AUTOTEST:   movl #RTT4,a1
            jsr OUTPUT
            movl #RTT5,a1
            jsr OUTPUT
AUTOTEST1:  movb #36,AUTO           öanger att det är autotest
            jsr SJU
            jsr NIO
            jsr ELVA
            jsr TRETTON
            jmp AUTOTEST1           óSTÄR OCH LOPPAR I DENNA TEST TILLS
RESET


ö-------------------------------------------------------------------------
-------
ö*************************************************************************
******
```

```
ö*****************************************************************
******
              ö++++++++++++++++++++++++++++++++++++++++
              ö+          SUBRUTIN  ADD                +
              ö+ skriver ut adress på skärm om bitfel+
              ö+ utdata HEX på skärm.indata a0       +
              ö++++++++++++++++++++++++++++++++++++++++

ADD:      movl #IN,a1             övar feldata skall ligga
          movl a0,a5              öRELATIVT A4
          addl REL,a5              öger absolut adress i primärminnet
vid fel
          movl a5,d7              öfeladress
          swap d7
          rorw #8,d7
          rorb #4,d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA
          jsr DEC                 ö 1:a
          jmp HA2
HA:       jsr HEX

HA2:      movl a5,d7
          swap d7
          rorw #8,d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA3
          jsr DEC                 ö 2:a
          jmp HA4
HA3:      jsr HEX

HA4:      movl a5,d7
          swap d7
          rorb #4,d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA5
          jsr DEC                 ö 3:a
          jmp HA6
HA5:      jsr HEX

HA6:      movl a5,d7
          swap d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA7
          jsr DEC                 ö 4:a
          jmp HA8
HA7:      jsr HEX


HA8:      movl a5,d7
          rorw #8,d7
          rorb #4,d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA9
```

```
          jsr DEC                   ö 5:a
          jmp HA10
HA9:       jsr HEX


HA10:     movl a5,d7
          rorw #8,d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA11
          jsr DEC                   ö 6:a
          jmp HA12
HA11:     jsr HEX

HA12:     movl a5,d7
          rorb #4,d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA13
          jsr DEC                   ö 7:a
          jmp HA14
HA13:     jsr HEX

HA14:     movl a5,d7
          andb #/0f,d7
          cmpb #9,d7
          bgt HA15
          jsr DEC                   ö 8:a
          jmp HA16
HA15:     jsr HEX


HA16:     movb #94,a1é              öavslutar med Ü
          movl #IN,a1
          jsr OUTPUT                öskriver ut feladress på skärm

          rts


ö*************************************************************************
*******
DEC:      addb #48,d7               öger ascii värde
          movb d7,a1é               öskriver in ascii värde 0-9
          addl #1,a1
          rts
ö*************************************************************************
*******
HEX:      addb #55,d7               öger hex A-F
          movb d7,a1é               ölägger i minne
          addl #1,a1
           rts
ö*************************************************************************
*******
```

```
ö*****************************************************************************
******
ö*****************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN PARITS           +.
              ö+ testar paritetsbiten för minnesarea  +.
              ö+ om fel hoppas det till RESETADRESS   +.
              ö+ där cause registret läses av         +.
              ö+ om paritesfel visas det på skärm och +.
              ö+ lysdioder (FF) (PROGRAMMET STANNAR   +.
              ö+ FöR DOM 200 FöRSTA APPARATERNA       +.
              ö+ för efterföljande apparater hoppas   +.
              ö+ det till subrutinen NMI              +.
              ö+++++++++++++++++++++++++++++++++++++++.

PARITS:         movb AUTO,d1            ökollar om auto test
                cmpb #36,d1             öom ¤
                beq PAR333

                movl #RI1,a1
                jsr OUTPUT              öfrågar om paritetstest skall
göras
                jsr INPUT               ökollar svaret
                movb IN,d1
                cmpb #106,d1            öom j-tangenten
                bne PAR5

                movl #TET1,a1
                jsr OUTPUT

PAR333:         movb #/ff,d6            öatt vi gör paritetstest (NMI)
                movb #/04,PARIT         ötar bort paritets flaggan
                movl #/20000,a0         östart adress 256 första k:na
PAR1:           movb #/ff,d4            ötestmönter
                movb #/ff,d3
                movb d4,a0é

                movl a0,d5
                cmpb #/280,d5            öom farlig adress
                beq PAR14
                cmpl #/380,d5
                beq PAR14
                movb a0é,d4             öger paritetsfel om
paritetskrets FEL
PAR14:          movb #0,d3
                movb WDOG,d7            ökick watchdog
                addl #1,a0
                cmpl #/60000,a0
                bne PAR1


PAR2:           movb #/0c,PARIT         ösätter paritetsflaggan igen

                movb AUTO,d1            ökollar om auto test
                movb #/04,PARIT         ötar bort paritets flaggan
                cmpb #36,d1             öom ¤
```

```
                beq JM5

                movl #TET2,a1
                jsr OUTPUT

JM5:            movl #/20000,a0         östart adress 256 första k:na
PAR10:          movb #/07,d4            ötestmönter
                movb #/07,d3
                movb d4,a0é
                movl a0,d5
                cmpb #/280,d5
                beq PAR15               öom farlig adress
                cmpb #/380,d5
                beq PAR15
                movb a0é,d4             öger paritetsfel om
paritetskrets FEL
PAR15:          movb #0,d3
                movb WDOG,d7            ökick watchdog
                addl #1,a0
                cmpl #/60000,a0
                bne PAR10


                movb #/0c,PARIT         ösätter paritetsflaggan igen

                movb AUTO,d1            ökollar om auto test
                cmpb #36,d1             öom $
                beq PAR5                öskriver ej ut skrift
                movl #RTT3,a1           öparitetstesten klar
                jsr OUTPUT

PAR5:           rts

                movb #/00,d6            öatt paritetstest är klar (NMI)
                movb AUTO,d1            ökollar om auto test
                cmpb #36,d1
                beq PAR53
                bra WE1                 öhoppar direkt till meny
(stackproblem)

PAR53:          rts

ö*************************************************************************
******
ö*************************************************************************
******
 ö++++++++++++++++++++++++++++++++++++++++
 öparitets kretsarna testas
 öskrivs först med fel paritet sedan ändras
 öparitetsbiten varvid fel skall uppkomma
 ötesten görs på stacktoppen
 ö++++++++++++++++++++++++++++++++++++++++

KOLL:       movl #13,d1
            jsr SJUSEG1
            movb #/0c,PARIT         öändrar paritetsflaggan
            movb #/ff,/60000        öskriver på stacktoppen ger etta i
paritet
            movb WDOG,d7            ökick watchdog
```

```
          movb #/04,PARIT        öändrar paritetsflaggan
          movl PPR36,a6          öåterhoppsadress från reset
          movb /60000,d1         öSKALL GE PARITETS FEL SKRIVS UT PÅ
SKÄRM
          movl FEL3,a1
          jsr OUTPUT             öSKRIVER UT PÅ SKÄRM ATT FEL I
KONTROLLKRETS
          movl #/0f,d3           ötänder alla lysdioder vid fel i
paritetskrets
          jsr DIOD1
          jmp KOLL               öom test ger fel

PPR36:    movb #/0c,PARIT        öändrar paritetsflaggan
          movb #/ef,/60000       öger nolla paritetbit
          movb WDOG,d7           ökick watchdog
          movb #/04,PARIT        öändrar paritetsflagga
          movl PPR37,a6          öåterhoppsadress från reset
          movb /60000,d1         öSKALL GE PARITETS FEL SKRIVS UT PÅ
SKÄRM
          movl #/f0,d3           ötänder alla lysdioder vid fel i
paritetskrets
          jsr DIOD1
          jmp PPR36              öom test ger fel

PPR37:    movl #87,d1
          jsr SJUSEG1            öom återhopp FEL
          rts

ö*************************************************************************
******
ö*************************************************************************
******
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+          SUBRUTIN OUTPUT            +.
             ö+ skriver ut textsträng.INDATA a1     +.
             ö+avsluta texten med Ü (ascii dec 94)  +.
             ö+++++++++++++++++++++++++++++++++++++++.

OUTPUT:   movb a1é,d0              ötecken från sträng
          cmpb #94,d0
          beq LI9

          jsr TX                   ökollar om transmit BUFFERTEN är tom
          movb d0,DARTB            öskriver ut tecken på skärm
          addl #1,a1
          jmp OUTPUT               ögår att skriva en bit till


LI9:      jsr TX                   ökollar om transmit BUFFERTEN är tom
          movb #10,DARTB           öline feed
          jsr TX                   ökollar om transmit BUFFERTEN är tom
          movb #13,DARTB           öreturn
          rts
ö*************************************************************************
******
ö*************************************************************************
******

TX:       movb WR0B,d1             ökollar om sändnings BUFFERTEN är
```

```
tom
          movb WDOG,d7              ökick watchdog
          andb #/04,d1              ömaskar av Tx BUFFERT EMTY
          cmpb #/04,d1
          bne TX
          rts
ö*************************************************************************
******
RX:       movb #/00,WR0B
          movb WR0B,d0
          movb WDOG,d7              ökick watchdog
          andb #/01,d0
          cmpb #/01,d0              ökollar om något i inbufferten
          bne RX
          rts
ö*************************************************************************
******
ö*************************************************************************
******
             ö+++++++++++++++++++++++++++++++++++++++.
             ö+           SUBRUTIN INPUT             +.
             ö+ utdata i sträng IN.Hoppar ut vid     +.
             ö+ return. tecknet ekas på skärm.       +.
             ö+ första tecknet läggs i d1            +.
             ö+++++++++++++++++++++++++++++++++++++++.

INPUT:    movl #IN,a1
IN3:      jsr RX                    öom inbuffert tom
          movb DARTB,d2             ölägger ascii värdet i register d2
          jsr TX                    ökollar om transmitBUFFERT tom
          movb d2,DARTB             öekar tecknet
          cmpb #13,d2               öom return hoppa ut
          beq IN1

          movb d2,a1é               öanvänder toppen (botten) på stacken
          addl #1,a1
          jmp IN3                   önästa tecken

IN1:      jsr TX                    ökollar om transmitBUFFERT tom
          movb #10,DARTB            öläger ut linefeed
          movb #94,a1é              ösluttecken
          rts

ö*************************************************************************
******
ö*************************************************************************
******

 ö+++++++++++++++++++++++++++++++++++++++++++
              ösjusegment 8
 ö+++++++++++++++++++++++++++++++++++++++++++


DART:     movl #1,d1
          jsr SJUSEG1                  öDMA testas
          movb WDOG,d7              ökick watchdog
          movb #/18,WR0B            ö kanalreset B
          movb #/01,WR0B            ö  REGISTER 1
          movb #/00,WR0B            ö reset
```

```
            movb #/03,WR0B          ö  REGISTER 3
            movb #/41,WR0B          ö 7bits/character
            movb #/04,WR0B          ö  REGISTER 4
            movb #/47,WR0B          ö paritet osv
            movb #/05,WR0B          ö  REGISTER 5
            movb #/28,WR0B          ö bitar vid sändning

            jsr TX
            movl #TEXT,a1
            jsr OUTPUT
            movl #TEXT1,a1
            jsr OUTPUT
            jsr INPUT
            movl #TEXT2,a1
            jsr OUTPUT

            rts

ö*************************************************************************
******
ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++.
                ö+           SUBRUTIN CIO                +.
                ö+         testar cio:n                  +.
                ö+++++++++++++++++++++++++++++++++++++++.

CIO:        movl #14,d1
            jsr SJUSEG1         ösjuseg 22

            jsr CIOENABLE       ögör enable på cio

CIO3:       jsr CLOCKVISI
            movl #TEXT4,a1
            jsr OUTPUT          öfrågar om klockan skall testas
            jsr INPUT           öhämtar svaret
            movl #IN,a1
            movb a1é,d1
            cmpb #106,d1
            bne CIO4
            jsr CLOCKTEST       ötestar clockan
            movl #TEXT6,a1
            jsr OUTPUT          öfrågar om klockan skall andras
            jsr INPUT           öhämtar svaret
            movl #IN,a1
            movb a1é,d1
            cmpb #106,d1
            bne CIO4
            movl #TEXT7,a1
            jsr OUTPUT
            jsr INPUT
            movl #IN,a1         öhämtar data
            jsr CLOCK           öändrar tiden på klockan


CIO4:       movl #86,d1
            jsr SJUSEG1         ösjuseg 8 MENY
            rts
ö*************************************************************************
```

```
******
ö*****************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++.
                ö+           SUBRUTIN CIOENABLE          +.
                ö+ gör enable på CIO                     +.
                ö+++++++++++++++++++++++++++++++++++++++.

CIOENABLE: movb #/00,CIK       ökontrollreg
           jsr DELA
           movb #/01,CIK       öreset port c
           jsr DELA
           movb #/00,CIK       ökontrollreg
           jsr DELA
           movb #/02,CIK       öåterställer reset port c
           jsr DELA
           movb #/07,CIK       öspecialcontrol register normal input or
output
           jsr DELA
           movb #/00,CIK
           jsr DELA

           movb #/05,CIK
           jsr DELA
           movb #/00,CIK       öpolarity register
           jsr DELA

           movb #/1e,CIK       ömode specifikation reg
           jsr DELA
           movb #/00,CIK
           jsr DELA
           movb #/04,CIC       önot enable clock+nvram  ++???++?? 0c
????
           jsr DELA

           movb #/01,CIK       öport c SELECT
           jsr DELA
           movb #/10,CIK
           jsr DELA

           movb #/04,CIC       önot enable clock+nvram ++???+++
0c+++++?+
           jsr DELA

           rts

ö*****************************************************************
******
ö*****************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++.
                ö+           SUBRUTIN CLOCKVISI          +.
                ö+ skriver ut (YYDDMMDDHHMMSS) på skärm  +.
                ö+ räknar åtta sekunder                  +.
                ö+++++++++++++++++++++++++++++++++++++++.

CLOCKVISI:movb #0,LOOP
VIS1:      jsr SECPUT          öhämtar data från klockan
           movl #/60000,a1      övar talen skall ligga
```

```
            jsr OUTPUT

            movl #120000,d1
            jsr DELAY1          öväntar en sek

            addb #1,LOOP
            cmpb #8,LOOP
            bne VIS1
            rts

ö*************************************************************************
******
ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++++.
                ö+           SUBRUTIN SEC                  +.
                ö+ tar fram data ur  klockan              +.
                ö+++++++++++++++++++++++++++++++++++++++++.


SECPUT:     movl #6,d6          ötitta på ÅR först
            movl #IN,a1
TTT:        jsr CIOREADC
            jsr ASCIIOUT
            addl #1,a1
            subl #1,d6
            cmpb #/ff,d6
            bne TTT
ö:          addl #1,a1
            movb #94,a1é        öavslutar med Ü
            rts

ö*************************************************************************
******
ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++++.
                ö+           SUBRUTIN CLOCK                +.
                ö+ ändrar tiden i klockan                 +.
                ö+ indata a1 :var data ligger             +.
                ö+ avsluta med Ü                          +.
                ö+++++++++++++++++++++++++++++++++++++++++.

CLOCK:      movb #6,MODE
CLOCK1:     movb a1é,d3
            cmpb #94,d3
            beq CLOCK2
            rolb #4,d3
            andb #/f0,d3
            addl #1,a1
            movb a1é,d4
            cmpb #94,d4
            beq CLOCK2
            andb #/0f,d4
            orb d4,d3
            movb MODE,d6
            jsr CIOWRITE        öskriver ut data i klockan
            addl #1,a1
            subb #1,MODE        önästa mode
```

```
          jmp CLOCK1
CLOCK2:   rts

ö*************************************************************************
******
ö*************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN CLOCKTEST          +.
              ö+ skriver ut (YYDDMMDDHHMMSS)          +.
              ö+ 990712312359S6 som sedan räknas upp  +.
              ö+ till 000101010010O4 om klockan fung  +.
              ö+++++++++++++++++++++++++++++++++++++++.


CLOCKTEST:jsr SECPUT
          movl #IN,a1         ösparar undan nuvarande tid
          movl #SAVE,a2       övar det skall lagras
          movb #0,LOOP
TEST1:    movb a1é,a2é
          addl #1,a1
          addl #1,a2
          addb #1,LOOP
          cmpb #16,LOOP       öallt inläst
          bne TEST1


          movl #6,d6          öår 99
          movl #/99,d3
          jsr CIOWRITE
          movl #5,d6          öday of week
          movl #/07,d3
          jsr CIOWRITE
          movl #4,d6          ömonth
          movl #/12,d3        ödec
          jsr CIOWRITE
          movl #3,d6          ödate
          movl #/31,d3        ö31:a
          jsr CIOWRITE
          movl #2,d6          öhour
          movl #/23,d3        ö23 timman
          jsr CIOWRITE
          movl #1,d6
          movl #/59,d3        ö59 minuter
          jsr CIOWRITE
          movl #0,d6          ösek
          movl #/56,d3        ö56 sekunder
          jsr CIOWRITE


          jsr CLOCKVISI       öräknar upp den nya tiden 8SEK

          movl #SAVE,a1
          jsr CLOCK           öskriver tillbaka den gamla tiden
          rts
ö*************************************************************************
******
ö*************************************************************************
******
```

```
DELA:       movl #/01,d1        ö20mikrosek fördröjning
            jsr DELAY1
            rts

ö*************************************************************************
******
ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++.
                ö+          SUBRUTIN CIOWRITE          +.
                ö+ skriver ut på port c.indata d6:vad  +.
                ö+ d3:värde till vad                   +.
                ö+++++++++++++++++++++++++++++++++++++++.


CIOWRITE: movb #/06,CIK
          jsr DELA
          movb #/00,CIK      ödatarikningen alla UTGÅNGAR

          movb #/05,CIC       önoselect + clock HöG
          jsr DELA
          movb #/00,CIC       öselect +clock låg
          jsr DELA

          jsr CIOADD          öadress (vad som skall skrivas in
sec-year)

          movb #/00,CIC       öwrite mod läggs ut på port
          jsr DELA
          movb #/01,CIC       öklockas ut
          jsr DELA

          rorb #6,d3          öindata till adress (läggs på bit två)
          movb d3,d4
          jsr WRITE           ödataord 1

          movb #0,d5          öloop räknare för 7dataord
CIO1:     rolb #1,d3
          movb d3,d4
          jsr WRITE

          addb #1,d5
          cmpb #7,d5
          bne CIO1            öom inte sista dataordet
          movb #/01,CIC       önollpuls på slutet
          movb #/04,CIC       öno enable chip SET DATA

          rts

ö*************************************************************************
******
ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++.
                ö+          SUBRUTIN CIOREADC          +.
                ö+ läser från port c.indata d6:vad     +.
                ö+ utdata d3  BCD-format:värde på vad  +.
                ö+++++++++++++++++++++++++++++++++++++++.
```

```
CIOREADC:movb #/06,CIK
         jsr DELA
         movb #0,CIK        ödatarikningen ALLA ut
         jsr DELA
         movb #/05,CIC      öclockpuls + no chip enable
         jsr DELA
        movb #/00,CIC       öclock+chip select
         jsr DELA
        clrl d3
        jsr CIOADD          övad som skall tittas på
        movb #/02,CIC       ölägger ut READ mod
         jsr DELA
        movb #/03,CIC       öklockar ut
         jsr DELA
        movb #/00,CIC       öclocka låg
         jsr DELA
        movb #/01,CIC       öklocka hög för HIHG IMPEDANSBIT
         jsr DELA
        movb #/00,CIC       öclocka låg
         jsr DELA
        movb #/06,CIK
         jsr DELA
         movb #/02,CIK      ödatarikningen en INGÅNG ut
         jsr DELA
         clrl d3
         jsr READ
        rorl #1,d4
        orb d4,d3
        jsr READ
        orb d4,d3
        movl #1,d5          öloop räknare för bit 3-7
CIO2:   jsr READ
        rolb d5,d4
        orb d4,d3
        addb #1,d5
        cmpb #7,d5
        bne CIO2
        movb #/04,CIC       öno enable chip
        rts
ö*************************************************************************
******
ö*************************************************************************
******
             ö++++++++++++++++++++++++++++++++++++++++.
             ö+          SUBRUTIN ASCII               +.
             ö+ tar fram ascii-värdet                 +.
             ö+ indata d3 BCD .som läggs i adress a1 +.
             ö++++++++++++++++++++++++++++++++++++++++.

ASCIIOUT:movb d3,d4
        rorb #4,d4
        andb #/0f,d4
        addb #48,d4
        movb d4,a1é
         movb d3,d4
        andb #/0f,d4
        addb #48,d4
```

```
        addl #1,a1
        movb d4,a1é
        rts


ö*****************************************************************************
******
ö*****************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN CIOADD               +.
            ö+ skriver till klockan vad som skall     +.
            ö+ skrivas eller tittas på                +.
            ö+++++++++++++++++++++++++++++++++++++++++.

CIOADD:  movb d6,d4
         rorb #1,d4
         jsr WRITE
         movb d6,d4
         jsr WRITE
         movb d6,d4
         rolb #1,d4
         jsr WRITE            öadress för vad som skal göras (sec-year)
         andb #/02,d4
          movb d4,CIC         öklocka låg,gör klart för read/write
biten
         rts

ö*****************************************************************************
******
ö*****************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN WRITE                +.
            ö+ skriver ut data till klocka            +.
            ö+ indata d4.  BIT TVÅ                    +.
            ö+ gör klocka låg först                   +.
            ö+++++++++++++++++++++++++++++++++++++++++.

WRITE:   andb #/02,d4
         orb #/00,d4          öno select nvram
         movb d4,CIC          ölägger ut data +no enable nvram
          jsr DELA
         orb #/01,d4
         movb d4,CIC          öklockar ut data
          jsr DELA
         rts
ö*****************************************************************************
******
ö*****************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN READ                 +.
            ö+ läser ut data från klocka              +.
            ö+ utdata d4.  BIT TVÅ                    +.
            ö+++++++++++++++++++++++++++++++++++++++++.

READ:    clrl d4
```

```
        jsr DELA
        movb #/01,CIC           öklocka på utgång
        movb CIC,d4             öhämtar data
        jsr DELA
        movb #/00,CIC           öklocka låg
        andb #/02,d4
        rts

ö*************************************************************************
******
ö*************************************************************************
******

            ö+++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN FLOPPY           +.
            ö+         testar floppykontroller     +.
            ö+++++++++++++++++++++++++++++++++++++++.

FLOPPY:     movl #15,d1
            jsr SJUSEG1                 ösjuseg 23

FLOP2:      jsr FLOPRES                 öresetar floppyn och startar
motorn

            movb #45,TRAC               öinitiera 45 TRACK (sista spåret)
FLO:        jsr TRACK                   öSöKER EFTER SPÅR

            movb   WDOG,d7              ökick DOG
            jsr TRACKTEST               ötestar att det är rätt spår

            movb #1,SEK
            jsr FLOPPYSEK               övilken sektor

            movl #FLOTEXT1,a1           ötest text
            jsr OUTPUT
            jsr INPUT                   övad som skall skrivas på floppy

             movl #IN,a2                öskriver in text på floppyn
            jsr WRITEF                  öskriver in data


            movb #1,SEK                 ösektor ett
            jsr FLOPPYSEK
            jsr READF                   öläser på skiva


            movl #FLOTEXT,a1
            jsr OUTPUT
            movl #SAVE,a1
            jsr OUTPUT                  öskriver ut data på skärm

            movw #/0391,FLOPCTR         öfloppy motor stannar
            movl #/7ffff,d1
            jsr DELAY1

            movl #85,d1
            jsr SJUSEG1                 ösjuseg åtta
            rts
```

```
ö*****************************************************************************
******
ö*****************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+          SUBRUTIN FLOPRES             +.
              ö+gör reset på floppyn ock startar       +.
              ö+motorn                                 +.
              ö+++++++++++++++++++++++++++++++++++++++++.

FLOPRES:      movw #/0000,FLOPCTR        ögör reset på floppykontrollen
(min 50u)
              movl #10,d1
              jsr DELAY1                 öger tidsfördröjning för RESET
              movw #/0c19,FLOPCTR        öinitierar FLOPPYN (motorn
startas)
              movl #2,d1
              jsr DELAY1
              movw #/0c1d,FLOPCTR        öHEADLOAD TIME
              clrl d4
FLOP:         addl #1,d4
              cmpl #/ffff,d4
              bne FLOP1
              jsr STATUS                 öom FLOPPY ej isatt
FLOP1:        movb FLOPSTA,d3
              movb   WDOG,d7             ökick DOG
              andb #/80,d3
              cmpb #/80,d3
              beq FLOP
              movb #/0c,FLOPCMD          öRESTORE floppy
              movl #/fff,d1              öungefär 25ms
              jsr DELAY1                 öger tidsfördröjning för RESET
              rts

ö*****************************************************************************
******
ö*****************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+          SUBRUTIN WRITEF              +.
              ö+Skriver ut på floppy INDATA:a2         +.
              ö+en sekltor i taget                     +.
              ö+++++++++++++++++++++++++++++++++++++++++.


WRITEF:       movb #0,d3
              jsr DIOD1
              movl a2,a4
              movb   WDOG,d7         ökick DOG
              movb #/ae,FLOPCMD      öatt vi skall WRITE floppy TRACK
              movl #FLOPDATA,a5
              movl #1,d1
              jsr DELAY1


WRI:          movb FLOPSTA,d3
              rorb #1,d3             ötestar busy biten
              bcc WRI12
```

```
            rorb #1,d3              öom data request
            bcc WRI

            movb (a2)+,a5é          ölägger data i dataregistret
            bra WRI                 öSKRIVER nästa tecken


WRI12:      movb FLOPSTA,d3         öskriver ut FELKOD om finnes
            cmpb #0,d3
            beq WRI14
            movl #FLOFEL3  ,a1
            jsr OUTPUT              ögör omförsök att skriva rätt
            movl #/ffff,d1
            jsr DELAY1
            movl a4,a2
            jmp WRITEF
WRI14:      movl #0,d3
            jsr DIOD1               önollställer lysdioderna
            rts

ö*************************************************************************
******
ö*************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN READF              +.
              ö+Läser data från floppy.UTDATA SAVE    +.
              ö+en sektor i taget                     +.
              ö+++++++++++++++++++++++++++++++++++++++.


READF:      movl #SAVE,a2
            movb   WDOG,d7          ökick DOG

            movb #/8e,FLOPCMD       öatt vi skall READ floppy
            movl #1,d1
            jsr DELAY1


            movb   WDOG,d7          ökick DOG
REA11:      movb FLOPSTA,d3
            btst #0,d3              ö BUSY ?
            beq  REA12
            btst #1,d3              ö DATAREQUEST ?
            beq REA11

            movb FLOPDATA,(a2)+     ölägger data i a2é
            bra REA11               öhämtar nästa tecken


REA12:      movb FLOPSTA,d3
            cmpb #0,d3
            beq REA57
            jsr DIOD1               öskriver ut felkod på lysdiod
            movl #FLOREERR,a1       öskriver ut att det har blivit fel på
läsning
            jsr OUTPUT
            movl #/ffff,d1
```

```
            jsr DELAY1
            jmp READF               ögör omförsök


REA57:      addl #1,a2
            movb #94,a2é
            movl #SAVE,a1
            movb   WDOG,d7          ökick DOG
            movl #0,d3
            jsr DIOD1               önollställer lysdioder
            rts

ö*************************************************************************
******
ö*************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+          SUBRUTIN STATUS            +.
              ö+Kollar att dator inte är upptagen    +.
              ö+++++++++++++++++++++++++++++++++++++++.

STATUS:     clrl d4
STA:        movb FLOPSTA,d3
            movb   WDOG,d7          ökick DOG
            andb #/80,d3
            cmpb #/80,d3
            beq  STAFEL             öfloppy är inte klar
            movb FLOPSTA,d3
            andb #/01,d3
            cmpb #/01,d3
            beq STA                 öfloppykontrollen upptagen
            rts

STAFEL:     addl #1,d4
            cmpl #5,d4
            bne STA
            movl #FLOFEL1,a1        öskriver ut att floppyn ej ansluten
                                    ökan även vara fel på kontroller
            jsr OUTPUT
            movl #/ffff,d1
            jsr DELAY1
            jmp FLOP2               ögör omtest

ö*************************************************************************
******
ö*************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+          SUBRUTIN FLOPPYSEK         +.
              ö+Väljer sektor INDATA SEK             +.
              ö+++++++++++++++++++++++++++++++++++++++.

FLOPPYSEK:  jsr STATUS
            movb FLOPSTA,d3
            btst #1,d3
            bne FLOPPYSEK           öATT INTE CONTROLLEN ÄR UPPTAGEN
            movb SEK,FLOPSEK        ösektor noll
            rts
```

```
ö*****************************************************************************
******
ö*****************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN TRACKTEST          +.
              ö+Testar om vi har stegat till rätt spår+.
              ö+++++++++++++++++++++++++++++++++++++++++.


TRACKTEST:  jsr STATUS
            movb FLOPTRACK,d4          ökollar så att det är stegat till
rätt
            cmpb TRAC,d4               öom rätt spår
            beq FLOT1
            movl #FLOFEL2,a1           öTRACKREGISTRET FEL
floppykontroll FEL
            jsr OUTPUT                 öeller floppyn ej formaterad
            movl #/fff,d1
            jsr DELAY1
            jmp FLOP2                   ögör omtest
FLOT1:      rts

ö*****************************************************************************
******
ö*****************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN TRACK              +.
              ö+  flyttar huvudet till spår TRAC      +.
              ö+++++++++++++++++++++++++++++++++++++++++.


TRACK:      jsr STATUS              ökollar om kontrollen upptagen
            movb TRAC,FLOPDATA        övilket spår i dataregistret
            movb #/1d,FLOPCMD       östegar till spår d3
            movl #1,d1              öfördröjning
            jsr DELAY1              öinnan vi kan läsa i statusreg
            rts

ö*****************************************************************************
******
ö*****************************************************************************
******

              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+            SUBRUTIN NVRAM             +.
              ö+          testar NVRAMMET              +.
              ö+++++++++++++++++++++++++++++++++++++++++.


NVRAM:    movl #TEST,a1
          jsr OUTPUT

          movl #18,d1
          jsr SJUSEG1                 ösjuseg 26
          movb   WDOG,d7              ökick DOG
          jsr CIOENABLE               öinitierar CIO
```

```
        movb #/06,CIK
        jsr DELA
        movb #/00,CIK               öalla utgångar
        jsr DELA                    öfördröjning
        movb   WDOG,d7              ökick DOG


        movb #/00,REGI              ötestar register noll
NV1:    movb #/01,DATA              ödata som skrivs in VANDRANDE ETTA
NV:     movb DATA,d3


        jsr ERASE                   ögör reset NVRAM +ENABLE PROGRAMMING
MODE
        jsr RAMWRITE                öskriver testmönster i rammet
        jsr RAMSAVE                 öspara NVRAMMETS innehåll

        rolb #1,d3
        cmpb #/01,d3
        beq NV2
        movb d3,DATA
        jmp NV
NV2:    addb #1,REGI
        cmpb #15,REGI
        bne NV1                     önästa register


        jsr NOENRAM                 öskrivskyddar NVRAMMET


        movl #RAMOK,a1
        jsr OUTPUT
        movl #/7ffff,d1
        jsr DELAY1


        movl #82,d1
        jsr SJUSEG1                 ösjuseg 8 "meny"
        rts


ö*************************************************************************
******
T:      movl #SAVE,a4               övar data skall ligga, läser ett
register
T1:     movb a4é,d3
        addl #1,a4
        jsr DIOD1
        movl #/7fff,d1
        jsr DELAY1
        clrb d3
        jsr DIOD1
        movl #/7fff,d1
        jsr DELAY1
        cmpl #/60120,a4
        bne T1
```

```
        rts

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN NOENRAM           +.
            ö+skrivskyddar NVRAMMET                +
            ö+++++++++++++++++++++++++++++++++++++++.

NOENRAM:movb #/06,CIK           öatt vi skall ställa in data
riktningen
        jsr DELA
        movb #/00,CIK           öalla utgångar
        jsr DELA                öfördröjning
        movb #/04,CIC
        jsr DELA                öNO ENABLE

öERASE/WRITE DISABLE
ögör programming NO enable på chip ,det komms ihåg tills det görs
enable

        movb #/0c,CIC           öchip select
        jsr DELA
        movb #/0d,CIC           öklockar ut den
        jsr DELA

        jsr ONE

        jsr ZERO

        jsr ZERO

        jsr ZERO

        jsr ZERO


        jsr ZERO

        jsr ZERO

        jsr ZERO

        jsr ZERO

        movb #/04,CIC           öklocka låg + NOSELECT
        jsr DELA

        rts

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN RAMWRITE          +.
            ö+skriver in testmönster i NVRAMMET    +
```

```
                ö+INDATA : REGI,DATA                              +
                ö+++++++++++++++++++++++++++++++++++++++++++++++.

RAMWRITE:movb #/06,CIK          öställa in datariktningen
         jsr DELA
         movb #/00,CIK          öalla utgångar
         jsr DELA               öfördröjning

         jsr ZERO               öenable NVRAM


         movb #/04,OPCODE       öopcode WRITE
         jsr RAM1               övilken OPKOD

         jsr RAM2               öskriver ut vilket REGISTER


         movb #0,d6             ödata BITS räknare

WRNV:    movb DATA,d4
         rorb #6,d4
         jsr WRN5

         movb DATA,d4
         rorb #5,d4
         jsr WRN5

         movb DATA,d4
         rorb #4,d4
         jsr WRN5

         movb DATA,d4
         rorb #3,d4
         jsr WRN5

         movb DATA,d4
         rorb #2,d4
         jsr WRN5

         movb DATA,d4
         rorb #1,d4
         jsr WRN5

         movb DATA,d4
         jsr WRN5

         movb DATA,d4
         rolb #1,d4
         jsr WRN5

         addb #1,d6
         cmpb #3,d6             öom vi har skrivit in 16 BITS
         bne WRNV

         movb #/04,CIC          öklocka låg + noenable
         jsr DELA

         movl #/600,d1
```

```
        jsr DELAY1


        rts
ö*****************************************************************************
******
WRN5:   andb #/02,d4
        cmpb #/02,d4            öom "1" eller "0"
        bne WRN1
        jsr ONE                 öskriver ut "1"
        jmp WRN2
WRN1:   jsr ZERO

WRN2:   rts
ö*****************************************************************************
******
ö*****************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN ERASE             +.
            ö+RESETAR nvrammet ,det läggs ettor i  +.
            ö+ alla register                       +.
            ö+++++++++++++++++++++++++++++++++++++++.



ERASE:  movb #/06,CIK           öatt vi skall ställa in data
riktningen
        jsr DELA
        movb #/00,CIK           öalla utgångar
        jsr DELA                öfördröjning
        movb #/04,CIC
        jsr DELA                öNO ENABLE

öERASE/WRITE ENABLE
ögör programming enable på chip ,det komms ihåg tills det görs no
enable

        movb #/0c,CIC           öchip select
        jsr DELA
        movb #/0d,CIC           öklockar ut den
        jsr DELA

        jsr ONE

        jsr ZERO

        jsr ZERO

        jsr ONE

        jsr ONE

        jsr ZERO

        jsr ZERO

        jsr ZERO
```

```
        jsr ZERO

        movb #/04,CIC           öklocka låg + NOSELECT
        jsr DELA


öSTART BIT
        movb #/0c,CIC           öchip select
        jsr DELA
        movb #/0d,CIC
        jsr DELA

        jsr ONE


öOPCODEN
        jsr ZERO

        jsr ZERO

        jsr ONE

        jsr ZERO


öADRESS
        jsr ZERO

        jsr ZERO

        jsr ZERO

        jsr ZERO


öEND
        movb #/0c,CIC           öklocka låg
        jsr DELA
        movb #/0d,CIC
        jsr DELA
        movb #/04,CIC           öklocka låg
        jsr DELA                öavslutar med en klockpuls

        movl #/600,d1           öungefär 25ms fördröjning
        jsr DELAY1              öfördröjning efter NO SELECT

        rts


ö*************************************************************************
******
ZERO:   movb #/0c,CIC           öladdar "0"
        jsr DELA
        movb #/0d,CIC           öklockar ut den
        jsr DELA
        rts
ö*************************************************************************
******
```

```
ONE:    movb #/0e,CIC           öladdar "1"
        jsr DELA
        movb #/0f,CIC           öklockar ut den
        jsr DELA
        rts
ö*************************************************************************
******

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN RAMSAVE           +.
            ö+ Lagrar undan det som ligger i NVRAMET+.
            ö+ Det skall läggas tillbaka sedan     +.
            ö+++++++++++++++++++++++++++++++++++++++.


RAMSAVE:clrl d6
        movb #/04,CIC           öklocka låg start clocka NOENBLE
        jsr DELA

        movl #/fff,d1
        jsr DELAY1

        movl #0,d6              öadressregister (vilket minnes
innehåll)
        movl #SAVE,a4           övar datat skall ligga
        movb #/08,OPCODE        öread NVRAM

RAM31:  jsr RAM21               ölagrar register REGI
        movb   WDOG,d7            ökick DOG

        movb #/0c,CIC
        jsr DELA
        movb #/0d,CIC
        jsr DELA
        movb #/0c,CIC
        jsr DELA                öavslutar med en klockpuls


        rts

ö*************************************************************************
******
ö*************************************************************************
******

            ö+++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN RAM21             +.
            ö+ INITIERAR vad NVRAMMET skall lagra  +
            ö+ i minnet
            ö+++++++++++++++++++++++++++++++++++++++.
```

```
RAM21:  movb #/06,CIK           öställa in datariktningen
        jsr DELA
        movb #/00,CIK           öalla utgångar
        jsr DELA                öfördröjning

        movb #/0c,CIC           öenable nwram
        jsr DELA
        movb #/0d,CIC           öklocka hög
        jsr DELA

        jsr RAM1                övilken opkod

        jsr RAM2                öskriver ut vilket register

        jsr RAM3                öhämtar data och lagrar i SAVE

        movb   WDOG,d7          ökick DOG

        rts

ö*************************************************************************
******
ö*************************************************************************
******

              ö+++++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN RAM1               +.
              ö+ Skriver ut OPCODE till NVRAM         +
              ö+ INDATA  OPCODE                       +
              ö+++++++++++++++++++++++++++++++++++++++++.

RAM1:
öSTARTBIT
        movb #/0e,CIC           öklocka låg + data hög
        jsr DELA
        movb #/0f,CIC           öklocka hög +"1" ut startbit
        jsr DELA
öOPCODE

        movb OPCODE,d4
        rorb #2,d4
        jsr RAM10               öklockar ut första kod biten

        movb OPCODE,d4
        rorb #1,d4
        jsr RAM10               öklockar ut andra kod biten


        movb OPCODE,d4
        jsr RAM10               öklockar ut tredje kodbiten


        movb OPCODE,d4
        rolb #1,d4
        jsr RAM10               öklockar ut fjärde kodbiten
        movb   WDOG,d7            ökick DOG
```

```
        rts

RAM10:  andb #/02,d4
        orb #/0c,d4             öklocka låg + data in låg
        movb d4,CIC
        jsr DELA
        orb #/01,d4
        movb d4,CIC             öklocka hög + data ut
        jsr DELA

        rts

ö*************************************************************************
******
ö*************************************************************************
******

            ö++++++++++++++++++++++++++++++++++++++++.
            ö+          SUBRUTIN RAM2                +.
            ö+ Skriver ut vilket register till NVRAM+
            ö+ INDATA REGI                           +
            ö++++++++++++++++++++++++++++++++++++++++.


RAM2:   movb REGI,d6
        rorb #2,d6
        jsr NVADD               öA3

        movb REGI,d6
        rorb #1,d6
        jsr NVADD               öA2

        movb REGI,d6
        jsr NVADD               öA1

        movb REGI,d6
        rorl #1,d6
        jsr NVADD               öA0


        movb   WDOG,d7            ökick DOG
        rts


NVADD:  andb #/02,d6            ötar fram bit som skall sändas över
        orb  #/0c,d6            ögör klocka låg+  data ut
        movb d6,CIC
        jsr DELA
        orb #/01,d6
        movb d6,CIC             öklocka hög klockar ut data
        jsr DELA
        rts

ö*************************************************************************
******
ö*************************************************************************
******
```

```
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN RAM3                +.
            ö+ HÄMTAR data BYTES vis och lagrar det  +
            ö+ i minne INDATA a4                     +
            ö+++++++++++++++++++++++++++++++++++++++++.


RAM3:       movb #/06,CIK           öatt vi skall ändra på data riktningen
            jsr DELA
            movb #/02,CIK           öen ingång
            jsr DELA                öfördröjning

            movb #1,d6              öbyteräknare

RAM333:     clrl d3                 ödata ordet som bitarna lagras i

RAM33:      jsr ZERO
            movb CIC,d4
            andb #/02,d4

            rolb #6,d4
            jsr OR

            rolb #5,d4
            jsr OR

            rolb #4,d4
            jsr OR

            rolb #3,d4
            jsr OR

            rolb #2,d4
            jsr OR

            rolb #1,d4
            jsr OR

            jsr OR
            rorb #1,d4
            orb d4,d3


            movb d3,(a4)+           ölagrar ut ett BYTE

            movb DATA,d4
            cmpb d3,d4
            bne RAM65

            addb #1,d6
            cmpb #3,d6              ökollar om ett WORD är lagrat
            bne RAM333

            movb #/0c,CIC           öklocka går låg
            movb #/04,CIC           öNO SELECT
            movl #/fff,d1
            jsr DELAY1              öfördröjning efter no select
```

```
        rts

ö*****************************************************************
******
RAM65:  eorb d4,d3              ötar fram felaktig bit
        jsr DIOD1               öskriver ut felaktig bit
        movl #RAMFEL,a1         öskriver ut att NVRAMMET ej okey
        jsr OUTPUT
        movl #/ffff,d1
        jsr DELAY1
        jmp NVRAM               ögör omtest

ö*****************************************************************
******
OR:     orb d4,d3
        jsr ZERO
        movb CIC,d4
        andb #/02,d4
        rts

ö*****************************************************************
******
ö*****************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN DMA                +.
            ö+         testar DMA                    +.
            ö+++++++++++++++++++++++++++++++++++++++.

DMAM:   movl #TEST,a1
        jsr OUTPUT

        movb   WDOG,d7          ökick DOG
        movl #16,d1
        jsr SJUSEG1             ösjuseg 24

        movw #/0391,FLOPCTR     öfloppy motor stannar vid omstart
        movl #2,d1
        jsr DELAY1
        clrl d3
        jsr DIOD1               öresetar lysdioderna

        movl #/80,d3            öresetar 256K och framåt
        jsr PAGTRAN1
        movl #/20000,a0
DMA93:  clrb (a0)+
        movb   WDOG,d7          ökick DOG
        cmpl #/60000,a0
        bne DMA93


        movb   WDOG,d7          ökick DOG
        movl #/40,d3
        jsr PAGTRAN2            ölogisk adress 0-32K,fysisk 128+32K
        movl #/20000,a0

DMA10:  clrb (a0)+              öRESETAR SKRIVAREAN
        cmpl #/40000,a0         ö28000 resetar 128k
```

```
        bne DMA10
        movb   WDOG,d7        ökick DOG

ö-------------------------------------------------------------------
        öskriver in data på floppydriver
ö-------------------------------------------------------------------

DMA14:  movl #SAVE,a0         övar data skall ligga
        clrl d2

DMA36:  cmpw #256,d2          öom 0-255 byte inskrivet
        beq DMA37
        movb d2,(a0)+
        addw #1,d2
        bra DMA36

DMA37:  jsr FLOPRES

        movb #0,TRAC          öspår NOLL
        jsr TRACK


        movb #1,SEK           ösektor ETT
        jsr FLOPPYSEK
        movb   WDOG,d7        ökick DOG

        movl #SAVE,a2
        jsr WRITEF            öskriver in data på spår NOLL
        movb   WDOG,d7        ökick DOG


ö-------------------------------------------------------------------
öläsning från floppy till minnesområde med DMA
ö-------------------------------------------------------------------

DMA39:  movb #/0f,SPCTROL     öset bit SYSFS för INTERN hämtning  av DMA

                              öfrån floppy


        movb #/02,DMAMAP0     ö128K och uppåt DMA 0
        movb #/9f,DMAMAP0+1   ö floppyn

        movb #/a3,DMA0         öreset and disable
        movb #/83,DMA0         ödisable DMA


        movb #/c3,DMA0         öRESET
        movb #/c3,DMA0         öRESET
        movb #/c3,DMA0         öRESET
        movb #/c3,DMA0         öRESET
        movb #/c3,DMA0         öRESET

ö---- WR0-GROUP-----------

        movb #/79,DMA0       öinitiera DMA port a till b TRANSFER

        movb #/00,DMA0
        movb #/00,DMA0       östartadress 128K
```

```
        movb #/ff,DMA0
        movb #/00,DMA0      öhur stort blocket är 256BYTE

ö---- WR1-GROUP----------

        movb #/54,DMA0      öVAD port a är för nått
        movb #/0d,DMA0      öCYCKELLÄNGD TRE

ö---- WR2-GROUP----------

        movb #/68,DMA0      övad port b är för nått
        movb #/0d,DMA0      öCYCKELLÄNGD TRE

ö---- WR3-GROUP----------

        movb #/80,DMA0      ögroup 3

ö---- WR4-GROUP----------

        movb #/cd,DMA0      ö4 group BURST
        movb #/06,DMA0
        movb #/f0,DMA0      öport b:s startadress

ö---- WR5-GROUP----------

        movb #/92,DMA0      ö5 group CE/WAIT AKTIV LÅG .READY AKTIV LÅG

ö----------------------
        movb #/0d,SPCTROL     ösignalen DMADIS
ö----------------------


ö---- WR6-GROUP---------

        movb #/cf,DMA0        öload A and reset counter
        movb #/87,DMA0        öDMA START

ö----------------------


        movb #1,SEK           ösektor ett
        jsr FLOPPYSEK
        movb #/8e,FLOPCMD     öatt vi skall READ floppy GER RDY signal
till DMA

        movl #1,d1
        jsr DELAY1

DMA35:  movb FLOPSTA,d3
        btst #0,d3            ö BUSY ? hoppar ej ur lopp då
        bne DMA35
        cmpb #0,d3
        beq DMA76

        movl #FLOREERR,a1     öskriver ut FELstatus på lysdioderna READ
        jsr OUTPUT
        jsr DIOD1
```

```
        movl #/ffff,d1
        jsr DELAY1
        jmp DMA39               ögör omtest



DMA76:  movb #/bf,DMA0          öläser statusbyte
        movb DMA0,d3

        movb d3,d4
        cmpb #/1b,d4
ö        andb #/01,d4
ö        cmpb #/01,d4
        beq DMA38
        movl #DMAFEL1,a1        öDMA överföringen ej genomförd
        jsr OUTPUT

        movl #DMAVI,a1          öskriver ut status för DMA
        jsr OUTPUT
        jsr DIOD1
        movl #/ffff,d1
        jsr DELAY1


        movb d3,d4
        andb #/20,d4
        cmpb #0,d4
        beq DMA38
        movl #DMAFEL2,a1        öEND of block not found
        jsr OUTPUT
        jmp DMAM                ögör omförsök


DMA38:  movl #/20000,a0
        clrl d4
ö-----------------------------
DMA40:  movb (a0)+,d3
        cmpb d4,d3
        bne DMA63               ömönstret FEL överfört
        addb #1,d4
        cmpl #/20101,a0         ötestar block om 256byte
        bne DMA40

ö-----------------------------
DMA47:  movb (a0)+,d3
        cmpb #0,d3
        bne DMA64               öfelaktigt kopierad
        movb   WDOG,d7          ökick DOG
        movl a0,d5
        cmpb #/80,d5
        bne DMA48
        addl #1,a0              ötar bort felaktiga adresser
DMA48:  cmpl #/40000,a0
        bne DMA47
ö-----------------------------
        movl #/80,d3            ötestar 256K och framåt skall vara nollat
        jsr PAGTRANi
```

```
        movl #/20000,a0
DMA97:  movb (a0)+,d3
        cmpb #0,d3
        bne DMA64
        movb   WDOG,d7          ökick DOG
        movl a0,d5
        cmpb #/80,d5
        bne DMA98               öhoppar över felaktiga adresser
        addl #1,a0              ötar bort felaktiga adresser
DMA98:  cmpl #/60000,a0
        bne DMA97
        jmp DMA65
ö---------------------------

DMA63:  movl #DMAFEL3,a1        ömönstret har blivit felaktigt överfört
        jsr OUTPUT
        jsr DIOD1               öskriver ut felaktig bit
        movl #/ffff,d1
        jsr DELAY1
        jmp DMAM                ögör omförsök
ö---------------------------

DMA64:  movl #DMAFEL4,a1        öskriver ut att det har blivit kopiering
        jsr OUTPUT              ötill annat block
        jsr DIOD1               öskriver ut felaktig bit
        movl #/ffff,d1
        jsr DELAY1
        jmp DMAM                ögör omförsök
ö---------------------------

DMA65:  movl #AMD0,a1            öfloppy till DMA till minne fungerar
        jsr OUTPUT
        movw #/0391,FLOPCTR     öfloppy motor stannar
        movl #/ffff,d1
        jsr DELAY1
        clrl d3
        jsr DIOD1               öresetar lysdioderna
        movl #84,d1
        jsr SJUSEG1             ösjuseg åtta
        rts
ö*************************************************************************
******
ö*************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN BUSS               +.
              ö+         testar busskortet             +.
              ö+++++++++++++++++++++++++++++++++++++++.


BUSS:   clrl d3
        jsr DIOD1
        jsr CRNUM               ökollar vilken kanal XEBECkontrollen
sitter i

        movb #0,CONRESET        ögenererar en reset till controllen
        movl #/ff,d1
        jsr DELAY1
```

```
        jsr BUSSTEST            ötestar om vi får kommunikation med
kontrollen
        clrl d3
        jsr DIOD1               öresetar lysdioderna
        movl #83,d1
        jsr SJUSEG1             ösjuseg åtta

        rts

ö*************************************************************************
******
ö*************************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN CRNUM              +.
              ö+         testar var busskortet        +.
              ö+++++++++++++++++++++++++++++++++++++++.

CRNUM:   movl #17,d1
        jsr SJUSEG1             ösjuseg 25


BU1:    movl #BUSS1,a1          övar kortet sitter
        jsr OUTPUT
                                öläser databuss genom att generera
stroben RCSB*
        movb BUSSSTA,d3         ötar reda på var XEBEC KONTROLL sitter
        notb d3
        jsr DIOD1
        cmpb #/01,d3
        bne BU2
        movl #BUS2,a1           öBUSS 2
        jsr OUTPUT
        jmp BU5

BU2:    cmpb #/02,d3
        bne BU3
        movl #BUS1,a1           öbuss 1
        jsr OUTPUT
        jmp BU5

BU3:    cmpb #/40,d3
        bne BU4
        movl #BUS0E,a1          öbuss 0 extern
        jsr OUTPUT
        jmp BU5

BU4:    cmpb #/80,d3
        bne HU
        movl #BUS0,a1           öbuss 0
        jsr OUTPUT
        jmp BU5

HU:     cmpl #/04,d3
        beq HU5
        cmpl #/08,d3
        beq HU5
        cmpl #/10,d3
        beq HU5
```

```
        cmpl #/20,d3
        beq HU5

FELBU:  movl #CARDFEL,a1        öhittar inte rätt kort
        jsr OUTPUT
        movl #/7ffff,d1
        jsr DELAY1
        jmp BUSS                öGöR OMTEST



HU5:    movl #CARDRAT,a1
        jsr OUTPUT              öskriver ut att kortvalet okey extern
buss
BU5:    movl #/7ffff,d1
        jsr DELAY1
        clrl d3
        jsr DIOD1               öresetar lysdioderna

        rts

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN BUSSTEST           +.
            ö+    testar bussKANAL mot WINCHESTER    +.
            ö+++++++++++++++++++++++++++++++++++++++.

öselectar kortet först genom att skicka /01 och signalen C1*

BUSSTEST:movb #/00,BYTE          öinitiering av commando test drive
ready
        movb #/00,BYTE+1        ödrive 0
        clrb BYTE+2
        clrb BYTE+3
        clrb BYTE+4
        clrb BYTE+5

        movb   WDOG,d7          ökick DOG

        jsr SELCNTR             öselectar kortet och genererar
"SEL"-puls
        jsr WCOM                öskriver till controllen vilket kommando
        jsr GETSTAT             ökollar att commandot blivit rätt
överfört

        rts

ö*************************************************************************
******
ö*************************************************************************
******
ö--------------------------
ösänder över kommand
ödet består av ett sex-byte block DCB
öbyte 0  765:class command 43210:opcode of the command
```

```
öbyte 1  765:LUN logical number 43210 logical diskadress 2
öbyte 2 7-0 logikal disk adress 1
öbyte 3 7-0 logical disk adress 0 (LSB)
öbyte 4 7-0 interleav or blockcount
öbyte 5 7:noll vid normal operation
ö       6:noll vid normal operation
ö       5-3 set to ZERO
ö       2:BUFFERSTEP OPTION (200 MIKROSEK PER STEP)
ö       1:half step option for tandon drives
ö       0:halfstep option of SEAGATE and INSTRUMENT drives
ö-----------------------------
              ö++++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN WCOM               +.
              ö+     skriver COMMANDO controllen       +.
              ö+          INDATA:BYTE-BYTE+5           +.
              ö++++++++++++++++++++++++++++++++++++++++.

WCOM:    jsr REQWC              öcontrollen vill ha kommand (controller
request)
         movb BYTE,  COMDAT
         movl #1,d1
         jsr DELAY1

         jsr REQWC              öcontrollen vill ha kommand (controller
request)
         movb BYTE+1,COMDAT     öskickar ut data och genererar ACK "0"

         movl #1,d1
         jsr DELAY1

         jsr REQWC              öcontrollen vill ha kommand (controller
request)
         movb BYTE+2,COMDAT
         movl #1,d1
         jsr DELAY1

         jsr REQWC              öcontrollen vill ha kommand (controller
request)
         movb BYTE+3,COMDAT
         movl #1,d1
         jsr DELAY1

         jsr REQWC              öcontrollen vill ha kommand (controller
request)
         movb BYTE+4,COMDAT
         movl #1,d1
         jsr DELAY1

         jsr REQWC              öcontrollen vill ha kommand (controller
request)
         movb BYTE+5,COMDAT
         movl #1,d1
         jsr DELAY1

         movb   WDOG,d7         ökick DOG
         rts

ö*************************************************************************
******
```

```
ö*****************************************************************
******
öfår tillbaka STATUSBYTE (2ST)
öbit 5 logical numberof driver 0 or 1
öbit 1 om set har det hänt något FEL i commandot. HOPPAR till felrutin
öandra bytet är ett nollbyte som talar om att kommandot är klart

              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN GETSTAT          +.
              ö+    testar bussKANAL mot WINCHESTER  +.
              ö+++++++++++++++++++++++++++++++++++++++.


GETSTAT:jsr REQRD             ödatorn vill sända tillbaka status
        movb COMDAT,d3
        andb #/02,d3
        cmpb #2,d3
        beq ERROR             öhoppar snabbt ut för att hämta FELKOD
        movl #CONOKEY,a1      ökontrollen okey
        jsr OUTPUT
        movl #/fff,d1
        jsr DELAY1
        rts                   öhämtar komplett felkod

ö*****************************************************************
******
ö*****************************************************************
******
              ö+++++++++++++++++++++++++++++++++++++++.
              ö+           SUBRUTIN ERROR            +.
              ö+  Hämtar error cod från controllen 4st+.
              ö+++++++++++++++++++++++++++++++++++++++.

öhämtar komplett felstatus från kontrollen
öcontrollen RETUNERAR fyra st byten tillbaka

ERROR:  clrl d3
        clrl REL              öför errtype utskrift
        jsr DIOD1             öresetar lysdioderna

        movb #/03,BYTE        öinitiering av commando REQEST SENSE
STATUS
        movb #/00,BYTE+1      ödrive 0
        clrb BYTE+2
        clrb BYTE+3
        clrb BYTE+4
        clrb BYTE+5

        jsr SELCNTR           ögenererar SELPULS till CONTROLLER 0
(byglad)
        jsr WCOM              öskriver ut kommandot

        movl #ERRCODE,a1      öERRCODE OCH ERRTYPE PÅ DIOD OCH SKÄRM
        jsr OUTPUT


        jsr REQRD1            öom kontrollen vill sända något
        movb ERRDAT,d3
        movl d3,a0            öskriver ut felcod på skärm
```

```
        jsr ADD
        jsr DIOD1

        jsr REQRD1              öom kontrollen vill sända något
        movb ERRDAT,d3

        jsr REQRD1              öom kontrollen vill sända något
        movb ERRDAT,d3

        jsr REQRD1             öom kontrollen vill sända något
        movb ERRDAT,d3


        movl #BUSSFEL4,a1
        jsr OUTPUT
        movl #/7ffff,d1
        jsr DELAY1              öfelbyte fyra

        jmp BUSS                ögör omtest

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN REQWC              +.
            ö+ kollar om kontrollen vill ha COMMAND +.
            ö+++++++++++++++++++++++++++++++++++++++.


REQWC:  movl #0,d2
REQ2:   addb #1,d2
        cmpb #/ff,d2
        beq REQ1                öfel i kommunikationen
        movb CARDSTA,d3
        movb   WDOG,d7          ökick DOG
        andb #/0f,d3
        cmpb #/0d,d3            ö d0:"1",d1:"0",d2:"1",d3:"1"
        bne REQ2

        rts


REQ1:   movb CARDSTA,d3
        jsr DIOD1
        movl #COMERR,a1     öskriver ut att controllen inte vill ha
commando
        jsr OUTPUT
        movl #/ffff,d1
        jsr DELAY1
        movb   WDOG,d7          ökick DOG
        jmp BUSS                ögör omtest


ö*************************************************************************
******
ö*************************************************************************
```

```
******
                ö+++++++++++++++++++++++++++++++++++++++++.
                ö+            SUBRUTIN REQWD             +.
                ö+ kollar om kontrollen vill ha DATA OBS+.
                ö+++++++++++++++++++++++++++++++++++++++++.


REQWD:   movb CARDSTA,d3
         movb   WDOG,d7         ökick DOG
         andb #/07,d3
         cmpb #/0a,d3           öbit I-/O "1",C-/D "0"
         bne REQWD
         rts

ö*************************************************************************
******
ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++++.
                ö+            SUBRUTIN REQR              +.
                ö+     kollar om controllen vill SÄNDA   +.
                ö+++++++++++++++++++++++++++++++++++++++++.


REQRD:   movl #/00,d2
RRD:     addl #1,d2             öom kontrollen inte vill sända kommando
         cmpl #/fff,d2
         beq RRD1
         movb CARDSTA,d3

         movb   WDOG,d7         ökick DOG
         andb #/0f,d3           ökollar om controllen vill sända över
errorkod
         cmpb #/05,d3           öC-/D är "1" I-/O är "0"
         bne RRD

         rts



REQRD1:  movl #/00,d2           öKOLLAR OM CONTROLLEN VILL SÄNDA ERROR
STATUS (DATA)
RRD2:    addl #1,d2             öom kontrollen inte vill sända kommando
         cmpl #/fff,d2
         beq RRD1
         movb CARDSTA,d3

         movb   WDOG,d7         ökick DOG
         andb #/0f,d3           ökollar om controllen vill sända över
errorkod
         cmpb #/07,d3           öC-/D är "1" I-/O är "0"
         bne RRD2

         rts


RRD1:    movb CARDSTA,d3
```

```
        jsr DIOD1               öskriver ut statusregistret
        movl #SENDERROR,a1      öskriver att controllen inte vill sända
status
        jsr OUTPUT
        movl #/ffff,d1
        jsr DELAY1
        jmp BUSS                ögör omtest

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN WBUSY              +.
            ö+     kollar om kontrollen är AKTIV     +.
            ö+++++++++++++++++++++++++++++++++++++++++.


WBUSY:  movl #0,d2
WBU1:   addb #1,d2
        cmpb #/ff,d2
        beq WBU2                öom controllen inte svarar på SELECT
        movb CARDSTA,d3
        movb   WDOG,d7          ökick DOG
        andb #/04,d3            ökollar busy biten "inverterad"
        cmpb #/04,d3
        bne WBU1

        rts

WBU2:   movb CARDSTA,d3
        jsr DIOD1
        movl #OBUSY,a1          öskriver ut att kontrollen inte vill bli
aktiv
        jsr OUTPUT
        movl #/7ffff,d1
        jsr DELAY1
        jmp BUSS                ögör omtest
ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN NOWBUSY            +.
            ö+     kollar om kontrollen är ledig     +.
            ö+++++++++++++++++++++++++++++++++++++++++.

NOWBUSY:movl #0,d2
OBU2:   addl #1,d2
        cmpl #/fff,d2
        beq OBU1                ökontrollen blir inte ledig
        movb CARDSTA,d3
        movb   WDOG,d7          ökick DOG
        andb #/04,d3            ökollar busy biten om inverterad
        cmpb #/00,d3            öhoppar ut på aktiv "0" (inverterad)
        bne OBU2

        rts
```

```
OBU1:   movb CARDSTA,d3
        jsr DIOD1
        movl #NOBUSY,a1
        jsr OUTPUT              ökontrollen vill inte bli ledig
        movl #/7ffff,d1
        jsr DELAY1
        jmp BUSS                ögör omtest

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN SELCNTR          +.
            ö+         selectar controllen         +.
            ö+++++++++++++++++++++++++++++++++++++++.

ökollar att controllen inte är upptagen ;sänder en selectpuls till
controllen

SELCNTR:jsr NOWBUSY             öatt kontrollen inte är upptagen
        movb #/01,DATASTROBE    ölatchar ut #/01 till controllen
        movb #/01,SELSTROB      ögenerera SELECTSTROBE aktiv låg
        jsr WBUSY
        rts

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+           SUBRUTIN MENY             +.
            ö+          SKRIVER UT MENY            +.
            ö+++++++++++++++++++++++++++++++++++++++.

MENY:       movb #0,d3
            jsr DIOD1           önollställer sjuseg
            movl #TEXT3,a1
            jsr OUTPUT
            rts

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+       SUBRUTIN   HIGH               +.
            ö+ Tar hand om bus interupt när vi     +.
            ö+kollar om det finns 1M primärminne   +
            ö+ i maskin                            +
            ö+++++++++++++++++++++++++++++++++++++++.

HIGH:   movb WDOG,d7                ökick watchdog
        movl #BUSSERROR,a1
        jsr OUTPUT
        movl #/7fff,d1
        jsr DELAY1
        movb AUTO,d1                ökollar om auto test
        cmpb #36,d1                 öom ¤
```

```
        beq AUTOTEST1               öskriver ej ut skrift
        jmp WE1                     ögår till MENY

ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+              RESETPROGRAM             +.
            ö+ hoppar hit vid reset av cpu           +.
            ö+ kollar i couse registret om det är    +.
            ö+paritetsfel eller vanlig reset         +.
            ö+Återhopps adressen läggs i a5          +.
            ö+GÄLLER VID DOM FÖRSTA 200 MASKINERNA   +.
            ö+++++++++++++++++++++++++++++++++++++++.

RESET:      movb WDOG,d7    öhämtar causregistret
            andb #/02,d7    ötar fram paritetsbiten
            cmpb #0,d7
            beq MAIN        öom inte paritetsfel hoppartill mainprogram

            movl FEL33,a1
            jsr OUTPUT      öskriverut att det blivit fel     ,CAUSEREG
            jmp RESET       öprogrammet stoppar


ö*************************************************************************
******
ö*************************************************************************
******
            ö+++++++++++++++++++++++++++++++++++++++.
            ö+            PARITETSINTERRUPT          +.
            ö+ hoppar hit vid PARITETSFEL av cpu     +.
            ö+ kollar i couse registret om det är    +.
            ö+ paritetsfel                           +.
            ö+ Återhopps adressen läggs i a5         +.
            ö+ GÄLLER VID DOM > 200 MASKINERNA       +.
            ö+++++++++++++++++++++++++++++++++++++++.

NMI:        movb WDOG,d3            öhämtar causregistret och skriver ut
det på DIOD
            jsr DIOD1
            movb d3,d4
            andb #/02,d4            ötar fram paritetsbiten
            cmpb #/02,d4
            beq MI                  öom paritetstesten fungererar riktigt

            movl#FEL3,a1
            jsr OUTPUT              öfelaktiga paritetskretsar eller annat
fel som                                     öger interupt

            movl #FE3,a1            öskriver ut CAUSE registret
            jsr OUTPUT
            jsr DIOD1


            cmpb #/ff,d6            öom paritetstest
            bne MI2
            jmp MI6
```

```
MI:         movl #RT3,a1
            jsr OUTPUT              ö PARITESKRETSARNA fungerer

            cmpb #/ff,d6            öom paritetstest
            bne MI2
MI6:        movl #PARFEL,a1         ödet har blivit fel vid paritetstesten
            jsr OUTPUT

MI2:        movl #PARTEST,a1        ötest av parit gjord
            jsr OUTPUT

            movl #/7ffff,d1
            jsr DELAY1

            movb #/0c,PARIT         ösläcker PARITETSFELLAMPAN och sätter
PARST

            movl #87,d1
            jsr SJUSEG1


            movb AUTO,d1
            cmpb #36,d1             öom autotest
            beq AUTOTEST

            jmp WE1                 öhoppar till meny

            rte

ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++.
                ö+              SUBRUTIN EST             +.
                ö+ testar att lådans sjuseg och dioder  +.
                ö+ fungerar
                ö+++++++++++++++++++++++++++++++++++++++.

EST:        movb #/01,d3
EST2:       jsr DIOD1
            movl #/ffff,d1
            jsr DELAY1
            cmpb #/80,d3
            beq EST3
            rolb #1,d3
            jmp EST2

EST3:       clrb d3
            jsr DIOD1
            movl #92,d1
            jsr SJUSEG1
            clrl d4

EST4:       movl #1,d1
            jsr SJUSEG1
            movl #/ffff,d1
            jsr DELAY1
            addb #1,d4
```

```
            cmpb #10,d4
            bne EST4
            clrl d4

EST5:       movl #10,d1
            jsr SJUSEG1
            movl #/ffff,d1
            jsr DELAY1
            addb #1,d4
            cmpb #9,d4
            bne EST5

            movl #8,d1
            jsr SJUSEG1
            rts

ö*************************************************************************
******
               ö+++++++++++++++++++++++++++++++++++++++.
               ö+              SUBRUTIN EXTERN          +.
               ö+ Laddar in program från externa eprom +.
               ö+++++++++++++++++++++++++++++++++++++++.

EXT:        movl #/300,d3    ölogiska adresser 0000-8000 = 32K
(20000-28000)
            jsr PAGTRAN2
            movl #/20000,a4
EXT2:       movb (a4)+,d3
            jsr DIOD1
            movl #/7fff,d1
            jsr DELAY1
            cmpl #/20010,a4    öräknar 16 stycken adresser i eprommet
            bne EXT2

            jsr TRA

            movl #/310,d3    ölogiska adresser 0000-8000 = 32K
(20000-28000)
            jsr PAGTRAN2
            movl #/20000,a4
EXT22:      movb (a4)+,d3
            jsr DIOD1
            movl #/7fff,d1
            jsr DELAY1
            cmpl #/20010,a4    öräknar 16 stycken adresser i eprommet
            bne EXT22

            jsr TRA


            movl #/320,d3    ölogiska adresser 0000-8000 = 32K
(20000-28000)
            jsr PAGTRAN2
            movl #/20000,a4
EXT3:       movb (a4)+,d3
            jsr DIOD1
            movl #/7fff,d1
            jsr DELAY1
            cmpl #/20010,a4    öräknar 16 stycken adresser i eprommet
```

```
            bne EXT3

            jsr TRA

            movl #/330,d3    ölogiska adresser 0000-8000 = 32K
(20000-28000)
            jsr PAGTRAN2
            movl #/20000,a4
EXT4:       movb (a4)+,d3
            jsr DIOD1
            movl #/7fff,d1
            jsr DELAY1
            cmpl #/20010,a4    öräknar 16 stycken adresser i eprommet
            bne EXT4

            jsr TRA


            rts
ö*************************************************************************
******
ö*************************************************************************
******
                ö+++++++++++++++++++++++++++++++++++++++++.
                ö+              MAINPROGRAM                +.
                ö+                                         +.
                ö+++++++++++++++++++++++++++++++++++++++++.

MAIN:       movl #MAI1,a6
            movb #/00,d3          önollställer dioder
            jmp DIOD

MAI1:       movl #MAI2,a7
            jmp  SEGRAMM          ötest segram

MAI2:       movl #MAI5,a7
            jmp  PAGERAM          ötest pageram


MAI5:       movl #MAI6,a5         ötest av primärminnet
            jmp PRIMRAM


MAI6:       clrl d3
            jsr DIOD1             önollställer dioderna
            jsr DART              ötestar DART

WE1:        clrl d3
            jsr DIOD1             önollställer dioderna
            clrb AUTO             övi skall inte ha autotest på
primärminnet
            jsr MENY              öskriver ut test meny
            movb WDOG,d7          ökick watchdog
            jsr INPUT             öhämtar svaret
            movb IN,d1
            cmpb #49,d1
            bne WE2
            jsr  SJU
            clrl d1
```

```
WE2:        cmpb #50,d1
            bne WE3
            jsr NIO
            clrl d1
WE3:        cmpb #51,d1
            bne WE4
            jsr ELVA
            clrl d1
WE4:        cmpb #52,d1
            bne WE5
            jsr TRETTON
            clrl d1
WE5:        cmpb #53,d1
            bne WE6
            jsr EST                 ötester test lådan
WE6:        cmpb #54,d1
            bne WE7
            jsr CIO
WE7:        cmpb #55,d1
            bne WE8
            jsr FLOPPY
WE8:        cmpb #56,d1
            bne WE9
            jsr NVRAM

WE9:        cmpb #57,d1
            bne WE10
            jmp KOLL                ökollar paritetskretsarna ock stannar
                                    öRESET MÅSTE GöRAS IGEN FöR ATT FÅ IGÅNG
TEST

WE10:       cmpb #97,d1
            bne WE11
            jsr DMAM

WE11:       cmpb #98,d1             ötestar busskortet
            bne WE112
            jsr BUSS
            jmp WE1

WE112:      cmpb #99,d1
            bne WE12
            jsr AUTOTEST            östår och looptestar minnet
            jmp WE1

WE12:       cmpb #42,d1
            bne WE14
            jsr KODE
            jmp WE1

WE14:       cmpb #94,d1
            bne WE1
            jsr EXT                   ötestar dom xeterna programprom
            jmp WE1

ö*************************************************************************
*****
ö*************************************************************************
```

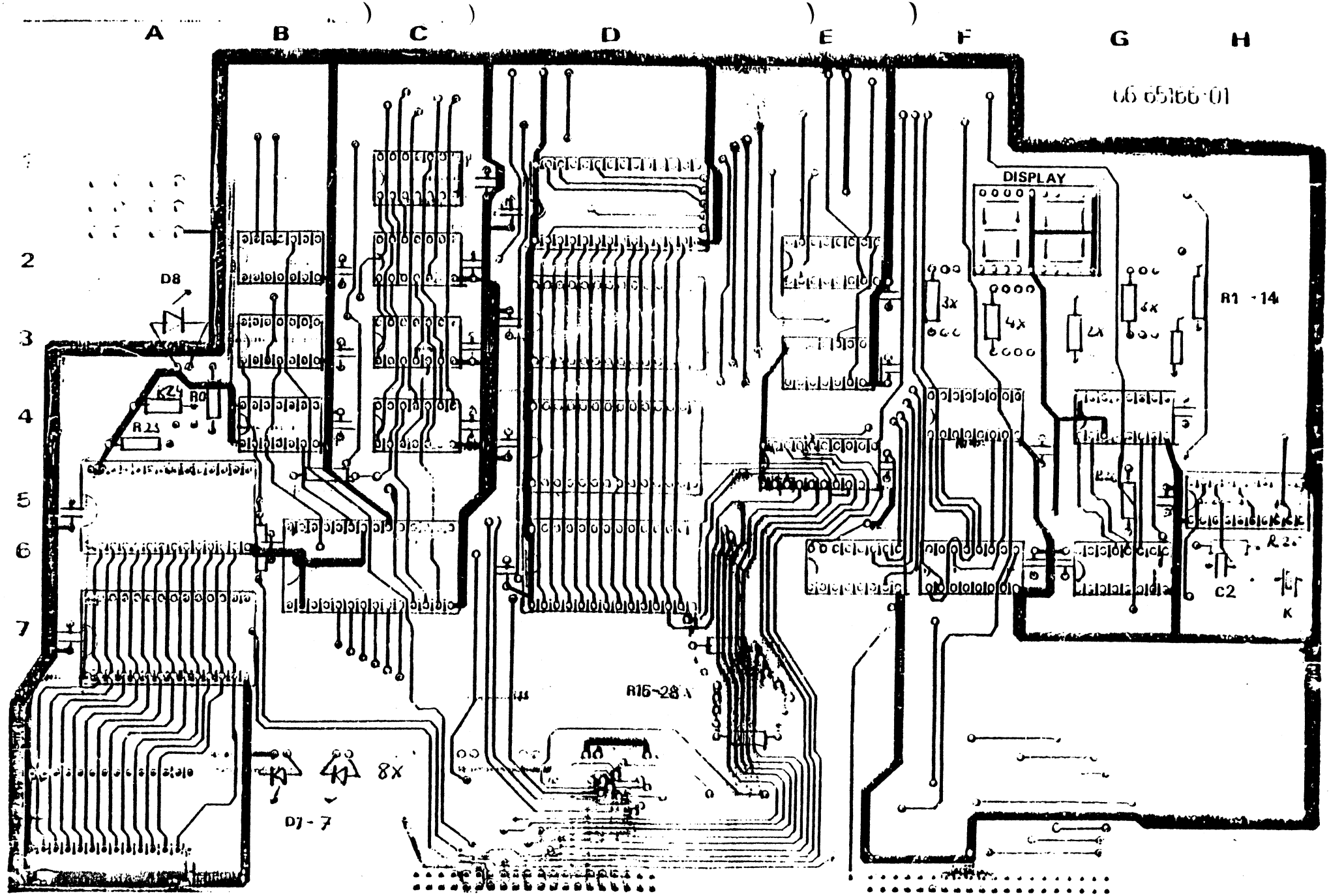
```
*****
KODE:       movl #KOD,a1
            jsr OUTPUT
            movl #/ffff,d1
            jsr DELAY1
            rts
ö*************************************************************************
*****
ö*************************************************************************
*****

TRA:      movl #trace,a1
          jsr OUTPUT
TRA1:     jsr INPUT              öhämtar svaret
          movb IN,d1
          cmpb #106,d1
          bne TRA
          rts


stopp:      movb WDOG,d7         ökick watchdog
            jmp stopp

ö*************************************************************************
*****
ö*************************************************************************
*****

TEXT4:    .ascii    "SKALL KLOCKAN TESTAS  (j/n) Ü"
TEXT5:    .ascii    "DET SKRIVS UT (YYDDMMDDHHMMSS) '993112235956' PÅ
SKÄRMÜ"
          .byte 13,10
          .ascii    "Värdet skall räkna upp tills det slår om till
00000000000"
TEXT6:    .ascii    "SKALL TIDEN ÄNDRAS (j/n)Ü "
TEXT7:    .ascii    "SKRIV IN (YYDDMMDDHHMMSS) PÅ TANGENTBORDET.Ü"
TEXT9:    .ascii    "ackÜ"
RTT1:    .ascii    "MINNES-AREAN ÄR BITTESTADÜ"
RTT2:    .ascii    "MINNES-AREAN ÄR ADRESSTESTADÜ"
RTT3:    .ascii    "MINNES-AREAN ÄR PARITETS TESTADÜ"
RTT4:    .ascii    "MINNESRADERNA TESTAS AUTOMATISKT (STOPP VID RESET)Ü"
RTT5:    .ascii    "VAR TESTEN ÄR VISAS PÅ SJUSEGMENTÜ"
FEL1:    .ascii    "SKRIV ELLER BIT-FEL I MINNES-AREANÜ"
FEL2:    .ascii    "ADRESS-FEL I MINNES-AREANÜ"
FEL3:    .ascii    "KONTROLL KRETSARNA FELAKTIGA FÖR PARITETSTEST (ELLER
NÅGOT SOM GER NMIÜ"
FE3:     .ascii    "SKRIVER UT CAUSE REGISTRET PÅ DIODERNAÜ"
RT3:     .ascii    "PARITETS KRETSARNA FUNGERAR (CAUSEREGISTRET SKRIVS
UT PÅ DIODERNA)Ü"
FEL33:   .ascii    "FEL PÅ PARITETSKRETSARNAÜ"
PARFEL:  .ascii    "DET HAR BLIVIT FEL I PARITETSTESTÜ"
PARTEST:.ascii    "PARITETS TEST GJORDÜ"


FEL4:    .ascii    "PARITETSFEL I PARITETSKRETSÜ"
RI1:     .ascii    "SKALL MINNES-AREAN PARITETSTESTAS (j/n)Ü"
RI2:     .ascii    "SKALL MINNES-AREAN BITTESTAS (j/n)Ü"
RI3:     .ascii    "SKALL MINNES-AREAN ADRESSTESTAS (j/n)Ü"
```

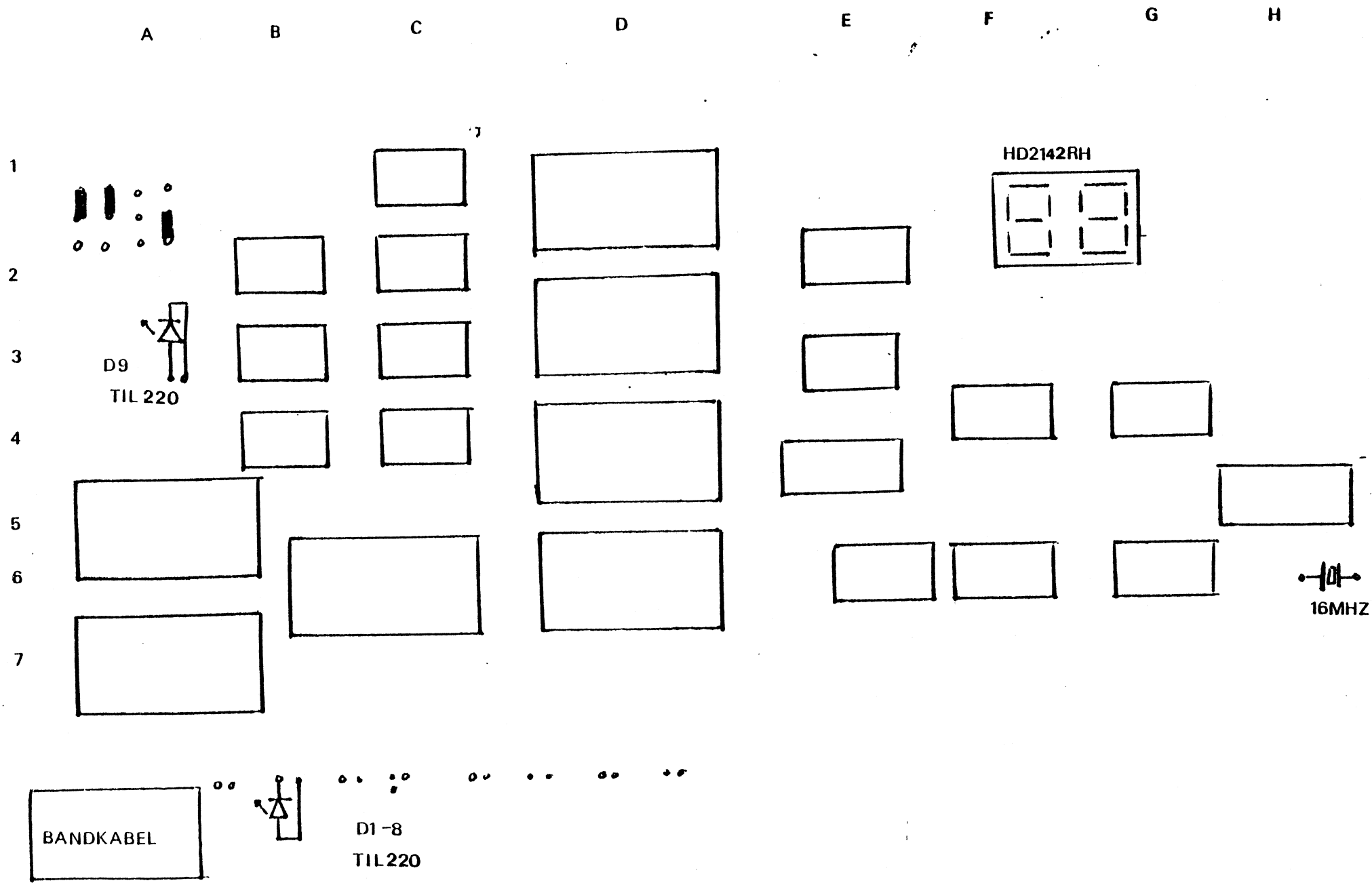
```
FLOTEXT1:.ascii   "SKRIV IN TEXT FOR FLOPPY SKRIVNING/LÄSNINGÜ"
FLOTEXT:.ascii    "OM SKRIVNING/ LÄSNING FUNGERAR SKRIVS TEXTEN UT PÅ
SKÄRMENÜ"

FLOFEL1:.ascii    "FLOPPYN EJ ANSLUTEN ELLER FEL PÅ KONTROLLENÜ"
FLOFEL2:.ascii    "STEGNING TILL FEL SPÅR/ KONTROLLEN FELAKTIG/ FLOPPYN
EJ FORMATERADÜ"
FLOFEL3:.ascii    "FELAKTIG SKRIVNING(STATUS PÅ LYSDIOD) .GÖR
OMSKRIVNINGÜ"
FLOPWR :.ascii    "FLOPPY STATUS 'SKRIVNING' PÅ LYSDIODERNAÜ"
FLOPRD :.ascii    "FLOPPY STATUS 'LÄSNING' PÅ LYSDIODERNAÜ"
FLOREERR:.ascii   "LÄSNINGEN FUNGERAR EJ RIKTIGT ,STATUS PÅ LYSDIODERNA
(GÖR OMTEST)Ü"

DMAFEL1:.ascii    "DMA ÖVERFÖRINGEN EJ GENOMFÖRD (GÖR OMTEST)Ü"
DMAFEL2:.ascii    "NOT END OF BLOCKÜ"
DMAFEL3:.ascii    "TEST MÖNSTRET FELAKTIGT ÖVERFÖRT (GÖR OMTEST)Ü"
DMAFEL4:.ascii    "ÖVERFÖRINGEN HAR BLIVIT KOPIERAD TILL ANNAT BLOCK
(GÖR OMTEST)Ü"
AMD:      .ascii  "DMA SVARAR PÅ REDY-SIGNALEN Ü"
AMD0:     .ascii  "DMA0 FRÅN FLOPPY TILL MINNE FUNGERAR .INGA FEL
HITTADEÜ"
DMAVI:    .ascii  "DMA STATUS  PÅ LYSDIODERNAÜ"

RAMFEL: .ascii    "DET ÄR FEL PÅ NVRAM ELLER OMGIVNING (GÖR OMTEST)Ü"
RAMOK:  .ascii     "NVRAMMET OKEYÜ"

BUSS1:  .ascii    "ANGER VAR KORTET SITTER PÅ LYSDIODERNAÜ"
BUSS2:  .ascii    "STATUSEN FÖR ISATT KORTÜ"
BUSSFEL4:.ascii   "GÖR OMTEST ( RESET PÅ SIGNAL C3*)Ü"

SENDERROR:.ascii "CONTROLLEN VILL INTE SÄNDA STATUS,STATUS PÅ
LYSDIODERNA (GÖROMFÖRSÖK)Ü"
COMERR:    .ascii  "CONTROLLEN TAR INTE KOMMANDO,STATUS PÅ LYSDIODERNA
(GÖR OMFÖRSÖK)Ü"
OBUSY:  .ascii  "CONTROLLEN SVARA INTE PÅ SEL STROBE.STATUS VISAS PÅ
LYSDIODER (GÖR OMFÖRSÖK)Ü"
NOBUSY: .ascii    "CONTROLLEN VILL INTE BLI LEDIG ,STATUS PÅ
LYSDIODERNA (GÖR OMFÖRSÖK)Ü"

BUS0:     .ascii  "XEBEC KONTROLLEN INKOPPLAD PÅ BUSS0Ü"
BUS0E:    .ascii  "XEBEC KONTROLLEN INKOPPLAD PÅ BUSS0 EXTERNÜ"
BUS1:     .ascii  "XEBEC KONTROLLEN INKOPPLAD PÅ BUSS1Ü"
BUS2:     .ascii  "XEBEC KONTROLLEN INKOPPLAD PÅ BUSS2Ü"
CARDFEL:.ascii    "FEL PÅ KORTVALET.OMTEST GÖRSÜ"
CARDRAT:.ascii    "KORTVALET OKEY (EXTERN BUSS)Ü"
ACK:      .ascii  "SLÅ OM BRYTAREN FÖR ACK-SIGNALENÜ"
ADDFEL: .ascii    "SKRIVSVÅRIGHETER I MINNETÜ"
CONOKEY:.ascii    "CONTROLLEN ÄR OKEY (INGA FEL HITTADE)Ü"
ERRCODE:.ascii    "ERRCODE OCH ERRTYPE PÅ DIODERNA  OCH SKÄRM (SE DATA
BLAD)Ü"


VEKTOR: .ascii    "ETT VEKTORAVBROTT HAR SKETT ,VEKTORN SKRIVS UT PÅ
DIODERNAÜ"
CIOINT: .ascii    "CIO HAR GJORT ETT INTERRUPTÜ"
DARTINT:.ascii    "DARTEN HAR GJORT ETT INTERRUPTÜ"
ADRINT: .ascii    "DET HAR SKETT ETT ADRESSERRORÜ"
```

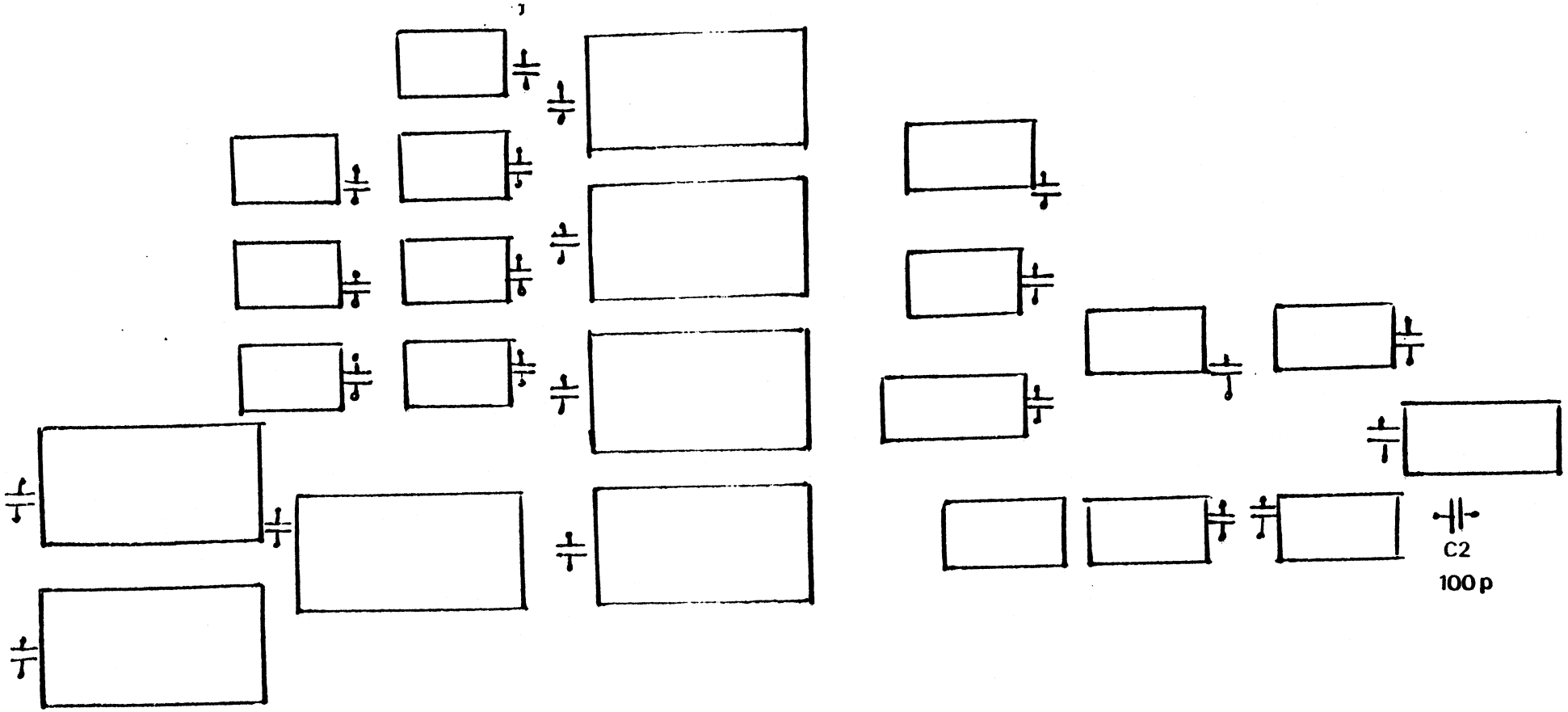
```
TEST:    .ascii  "TESTNING ..................Ü"
TET1:    .ascii  "TESTNING JÄMN PARITET.......Ü"
TET2:    .ascii  "TESTNING UDDA PARITET.......Ü"

BUSSERROR:.ascii "DET FINNS INTE 1M MINNE ,ELLER BUSS ERROR INTERUPT
(kolla minnesbyglingen)Ü"

TEXT:     .ascii    "(* DARTEN TESTAS HÄR *)Ü"
TEXT1:    .ascii    "MATA IN NÅGRA TECKEN ,OM TECKNEN EKAS FUNGERAR
DARTEN .Ü"
TEXT2:    .ascii    "    Ü"
TEXT3:    .ascii    "(* MENY *)"
          .byte 13,10
          .ascii    "VAD SOM REDAN ÄR TESTAT:"
          .byte 13,10
          .ascii    "MACCEN  (SEGRAM ,PAGERAM) SAMT INITIERAD"
          .byte 13,10
          .ascii    "PRIMÄRMINNE 0K-2K (EJ ADRESS) BITTESTAD"
          .byte 13,10
          .ascii    "STACK INITIERAD "
          .byte 13,10
          .ascii    "VAL AV NEDAN :"
          .byte 13,10
          .ascii    "PRIMÄRMINNE  2K-256K       (1)"
          .byte 13,10
          .ascii    "PRIMÄRMINNE 256K-512K      (2)"
          .byte 13,10
          .ascii    "PRIMÄRMINNE 512K-768K      (3)"
          .byte 13,10
          .ascii    "PRIMÄRMINNE 768K-1M        (4)"
          .byte 13,10
          .ascii    "TEST AV LÅDA (SJUSEG+DIOD)(5)"
          .byte 13,10
          .ascii    "CIO+KLOCKA                 (6)"
          .byte 13,10
          .ascii    "FLOPPY                     (7)"
          .byte 13,10
          .ascii    "NVRAM                      (8)"
          .byte 13,10
          .ascii    "TEST AV PARITETSKONTROLL   (9)"
          .byte 13,10
          .ascii    "DMA                        (a)"
          .byte 13,10
          .ascii    "TEST AV BUSSKORT           (b)"
          .byte 13,10
          .ascii    "AUTO TEST AV MINNE         (c)"
          .byte 13,10
          .byte 13,10
          .ascii    "VAL AV OVAN:Ü"


KOD:      .ascii    "HÅKAN HELDMANN 850719   TESTPROGRAM (25.s) FOR 1600
CPU-KORT Ü"
```

```
trace:  .ascii    "TRACEÜ"
test:   .ascii    "HÄRÜ"
          ö************** END MAINPROGRAM **************
```

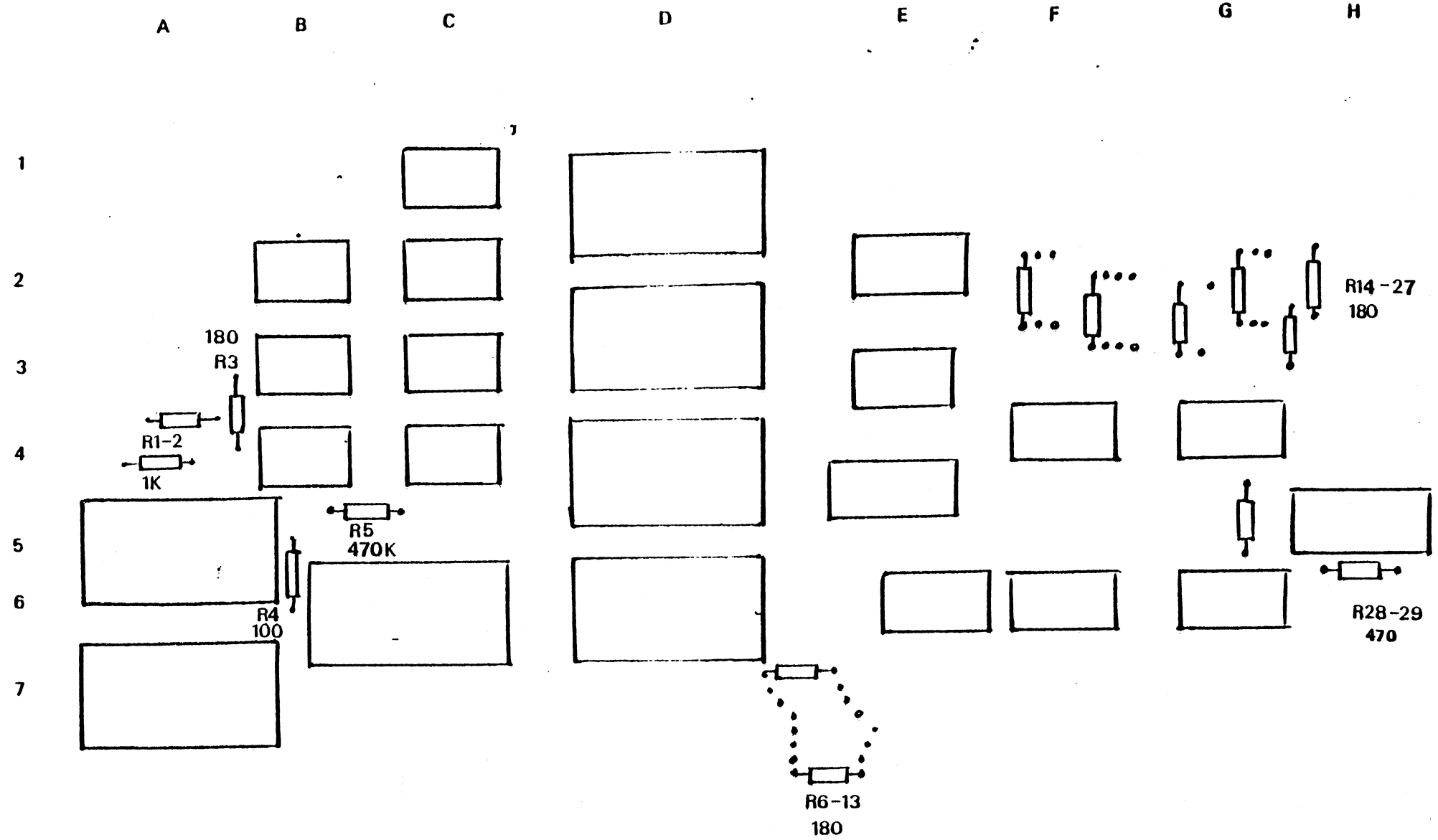
KOMPONENTSIDA



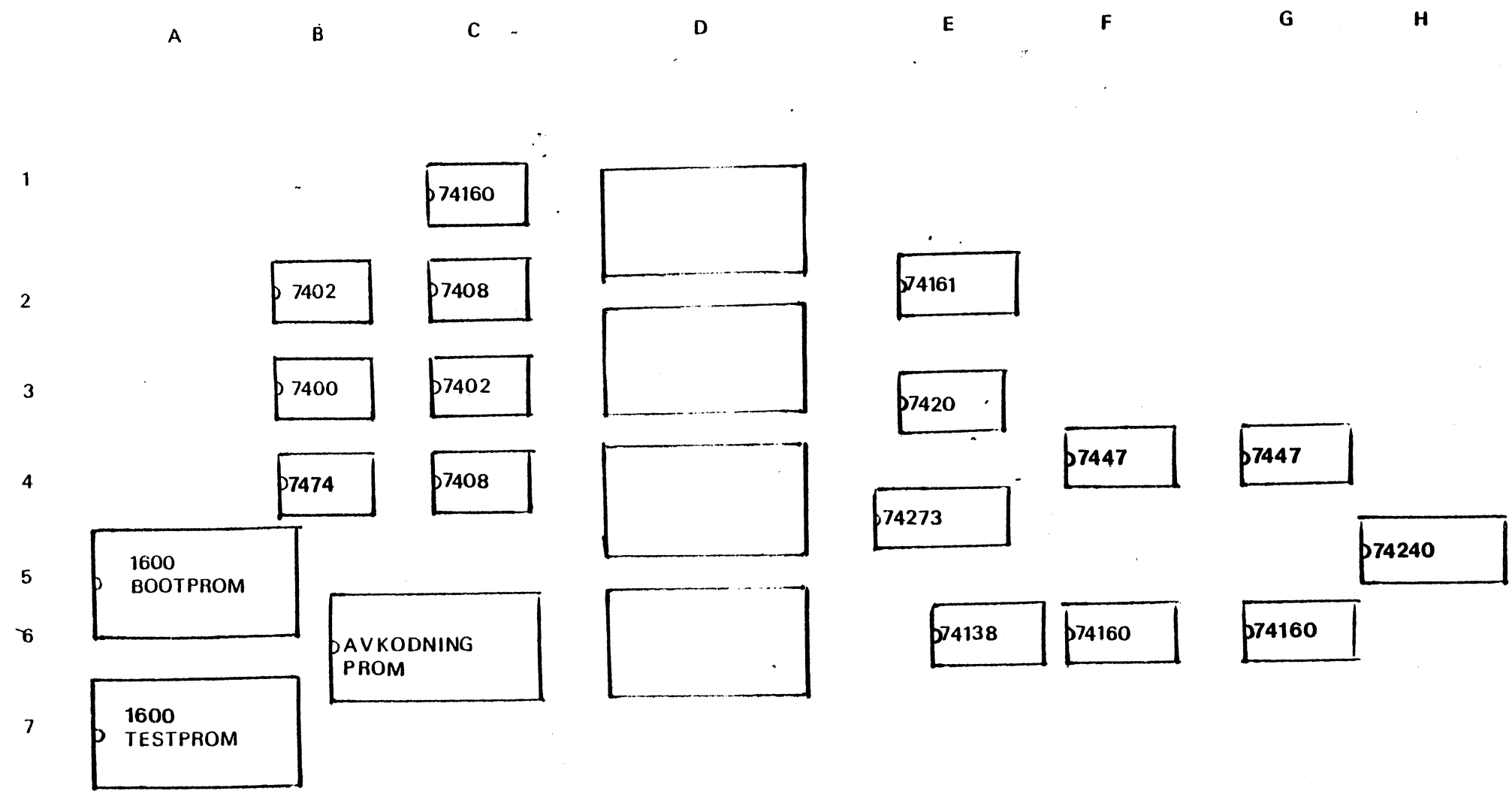

A
B
C
D
E
F
G
H
1
2
3
4
5
6
7
HD2142RH
D9
TIL 220
16MHZ
BANDKABEL
D1 -8
TIL220
ÖVRIGT



KONDENSATOR PLACERING



MOTSTÅND PLACERING



IC PLACERING

LUXOR

Dokumentnamn/Name of document

Blad nr/Sheet no ( )

Från/From	Datum/Date	Beteckning/Reference
	850820	HH 251L
Rubrik, ärende/Subject	**Ert datum/Your date**	**Er beteckning/Your reference**
Artikelnummer testlåda 1600		
	Gäller fr o m/Effective date	**Ersätter/Replaces**

Distribution

	värde	art nr	antal
kondensatorer	100 uF	62 00039-01	22
	100 p	62 00103-01	1
Motstånd	180 Ω	61 29541-01	23
	470 Ω	61 29256-01	2
	470 kΩ	61 29003-01	1
	100	61 29250-01	1
	1 kΩ	61 29258-01	2
IC kretsar	1600 boot		1
	Test Prom		1
	awk Prom		1
	7400	64 40089-01	1
	7402	64 40061-01	2
	7408	64 40032-01	2
	7420	64 40097-01	1
	7447		2
	7474	64 40035-01	1
	74160		3
	74161	64 40041-01	1
	74163	64 40100-01	1
	74240	64 40086-01	1
	74273	64 40048-01	1

LUXOR

Dokumentnamn/Name of document

Blad nr/Sheet no (2)

Från/From	Datum/Date	Beteckning/Reference
	850820	~~HH~~ 251C
Ruorik, ärende/Subject	Ert datum/Your date	Er beteckning/Your reference
Artikelnummer testlåda 1600	Gäller fr o m/Effective date	Ersätter/Replaces
	Distribution	

	bet	art nr	antal
sjusegment	HD2142RH		1
lysdiod	TIL220	63 40 057-01	9
kristall	16 MHZ	63 90065-01	1

bandkabel (kabel 40 ledar + 2st 40 poliga honor) 2

bandkabel (kabel 28 leda + 2st 28 poliga hona.) 1

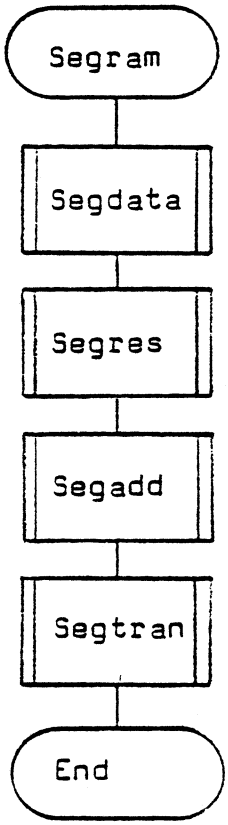
vippströmbrytare 3 vägs ej återfjädrande 1

—|— 1 väg återfjädrande 1

1st låda plåt

1st plexiglas röd (text på tryckt.)

Sjuseg ETT



Segmentramet testas

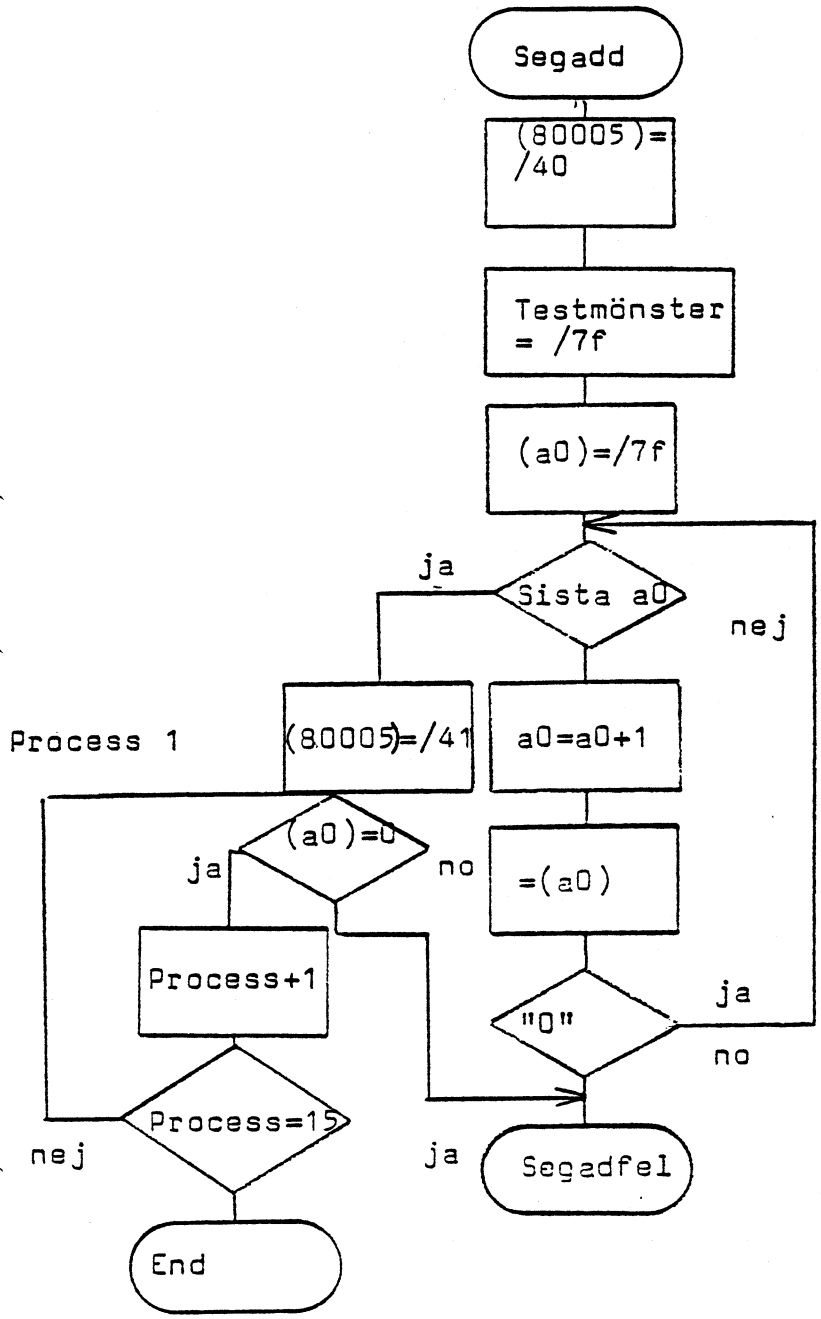
Bittest segram

Nollställning av segram

Adresstestar segram

Gör segram transparent

Ej återhopp via stack



Testar adressbussen hos segramen

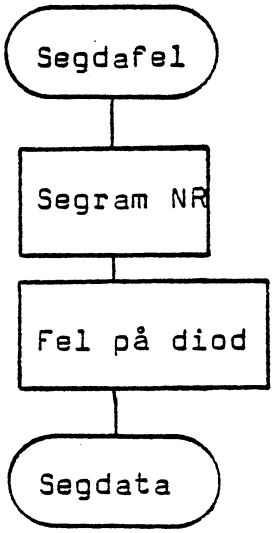
Process NOLL

Skriver in byte

Om sista adressen

Om kopiering

Troligen fel på adressbussen

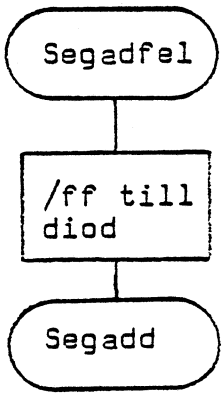


Bitfel i segram

Tar fram vilket segram

Lägger ut på dioder "lys"

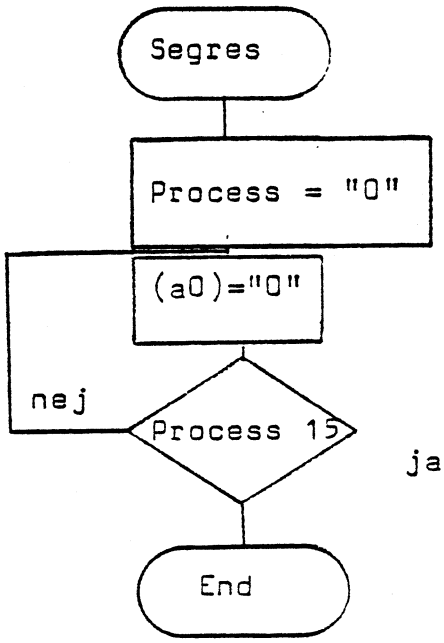
OMTEST



Adr[illegible]fel i segram

Tänder alla lys dioder

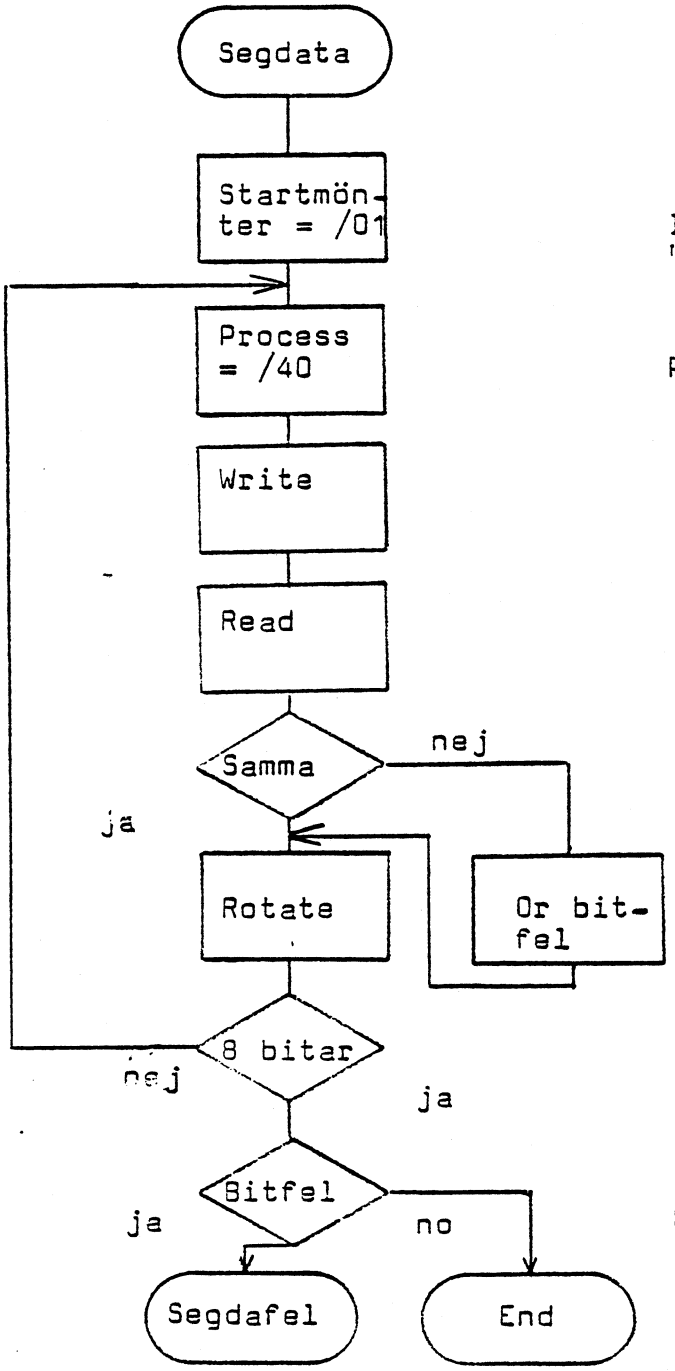
OMTEST



Nollställer segram

Skriver in noll

Om alla processer NOLL ställda



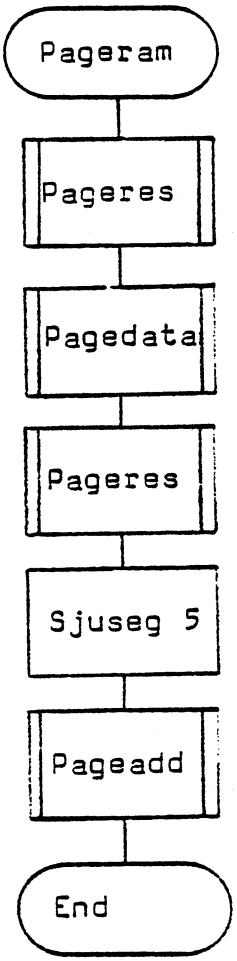
Initierar start-mönster "vandrande etta"

Process NOLL

Bildar felbyte

Om det har blivit bitfel

Sjuseg 4



Testar pageramen

Resetar pageram

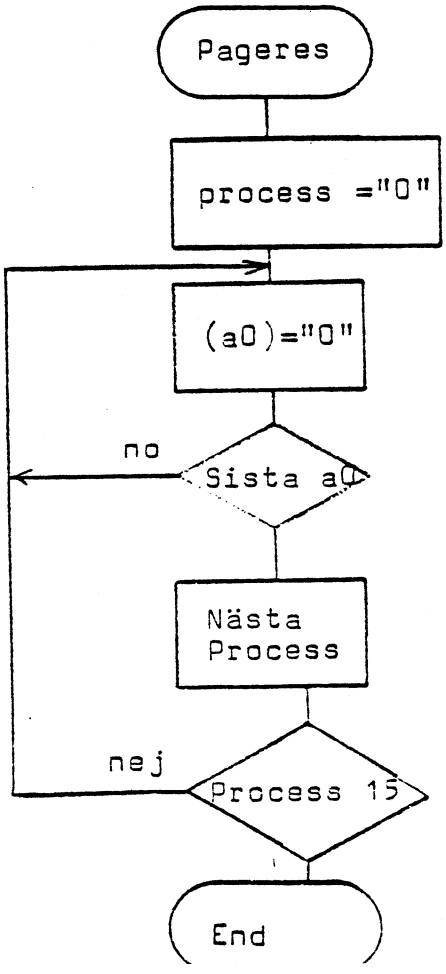
Bittestar pageram

Resetar pageram

Sjuseg 5

Räknar upp sjusegment till 5

Adresstestar pageram

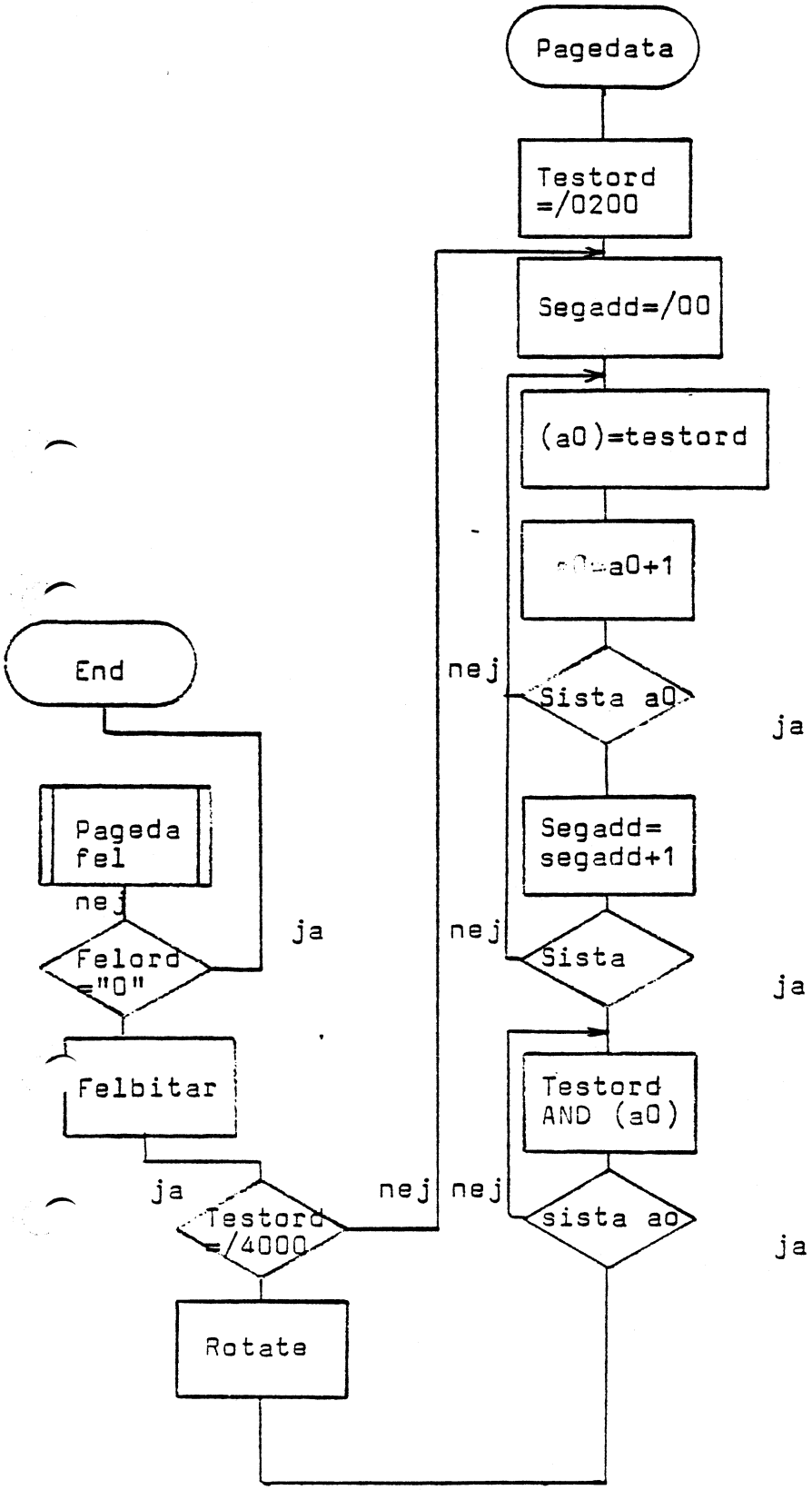


Resetar pageram

Skriver in NOLL

Om sista adress

Om sista processen



Bittestar pageram

Segram adress noll

Skriver in testord

Nästa adress i pageram

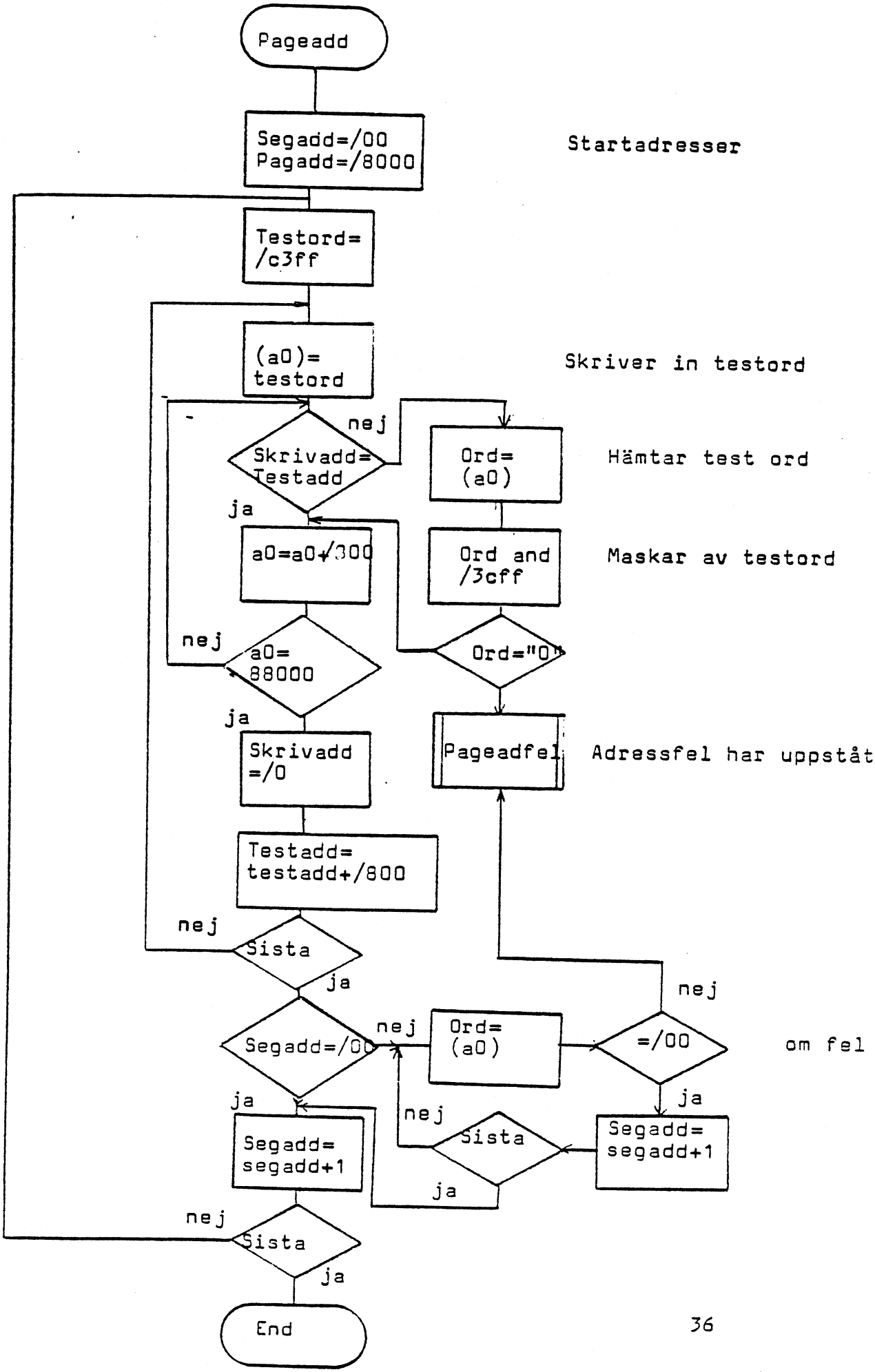
Sista testadress

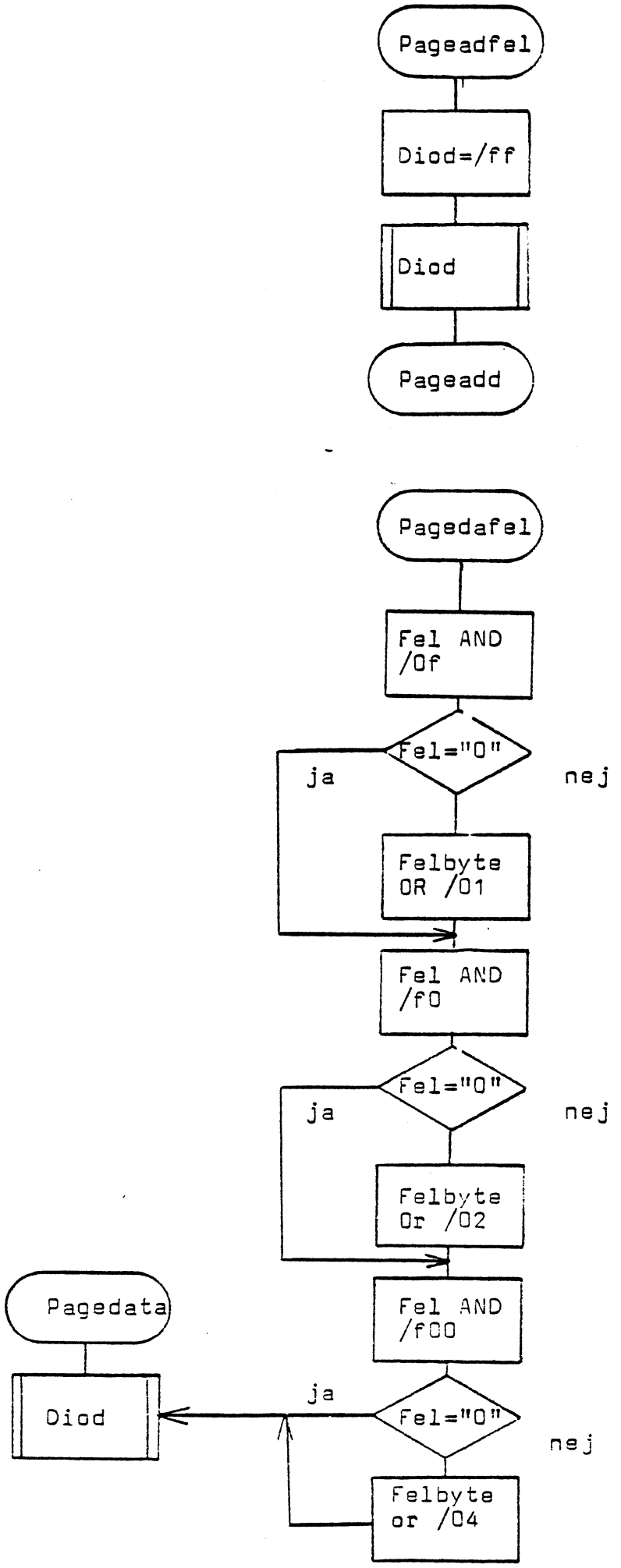
Nästa segadress

Sista adress i segram

Tar fram felaktigbit

om sista adress


Pageadd
Segadd=/00
Pagadd=/8000
Startadresser
Testord=
/c3ff
(a0)=
testord
Skriver in testord
nej
Skrivadd=
Testadd
Ord=
(a0)
Hämtar test ord
ja
a0=a0+/300
Ord and
/3cff
Maskar av testord
nej
a0=
88000
Ord="0"
ja
Skrivadd
=/0
Pageadfel
Adressfel har uppstått
Testadd=
testadd+/800
nej
Sista
ja
nej
Segadd=/00
nej
Ord=
(a0)
=/00
om fel
ja
ja
nej
Segadd=
segadd+1
Sista
Segadd=
segadd+1
ja
nej
Sista
ja
End



Adrsseringsfel vid pageramtest

Tänder alla lysdioder

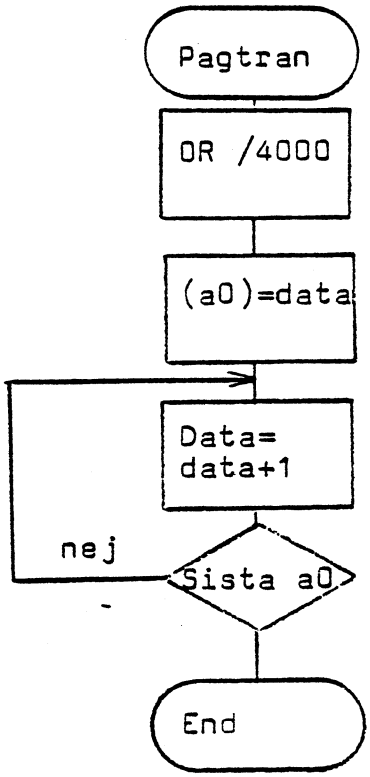
Gör omtest

Datafel vi bit-test

Om fel i pagram 18G

Om fel i pageram 22G

Om fel i pageram 20G



Gör pagtran

Alla celler skriv och läsbara

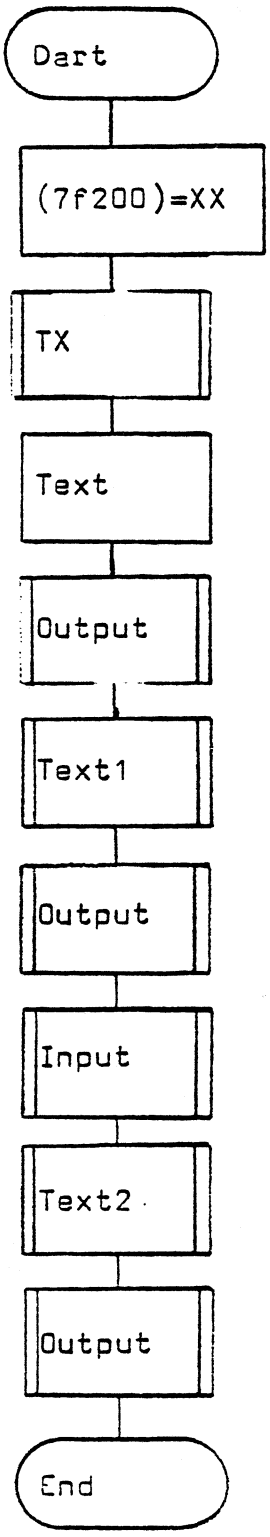
Skriver in data

Om sista adress

Pagtran 1: Används när stacken är testad

Pagtran 2: Sätter upp maccen för 32K

Sjuseg 8



Testar darten

Initierar DART

Kollar om transmit buffert tom

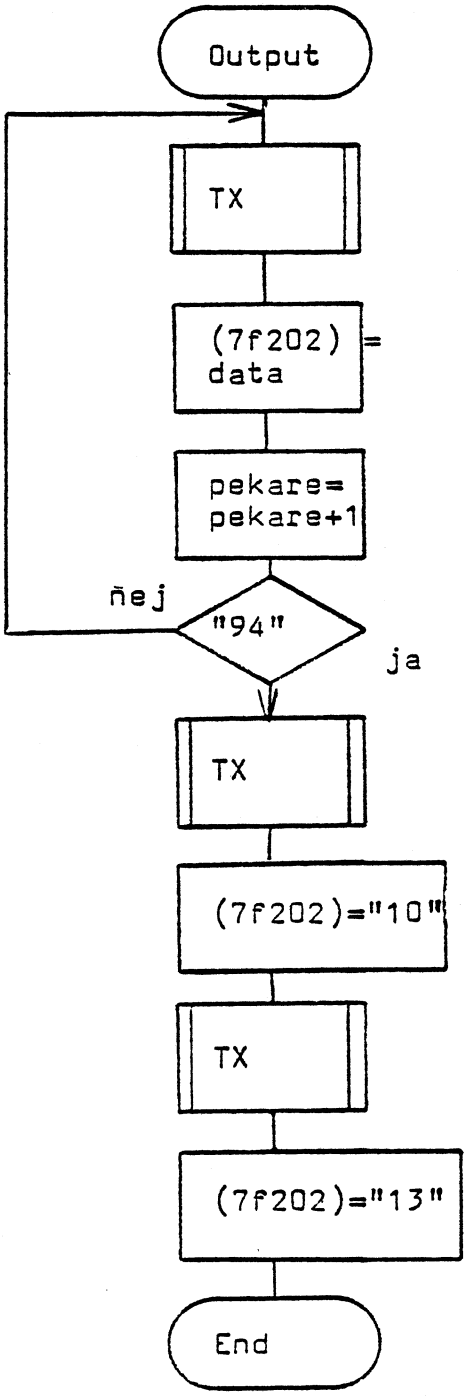
"Darten testas här"

Skriver ut på skärm

"Skriv några tecken"

Hämtar text från tangenbo

"MENY"



Skriver ut TEXT på skärm

Om transmitt-buffert tom

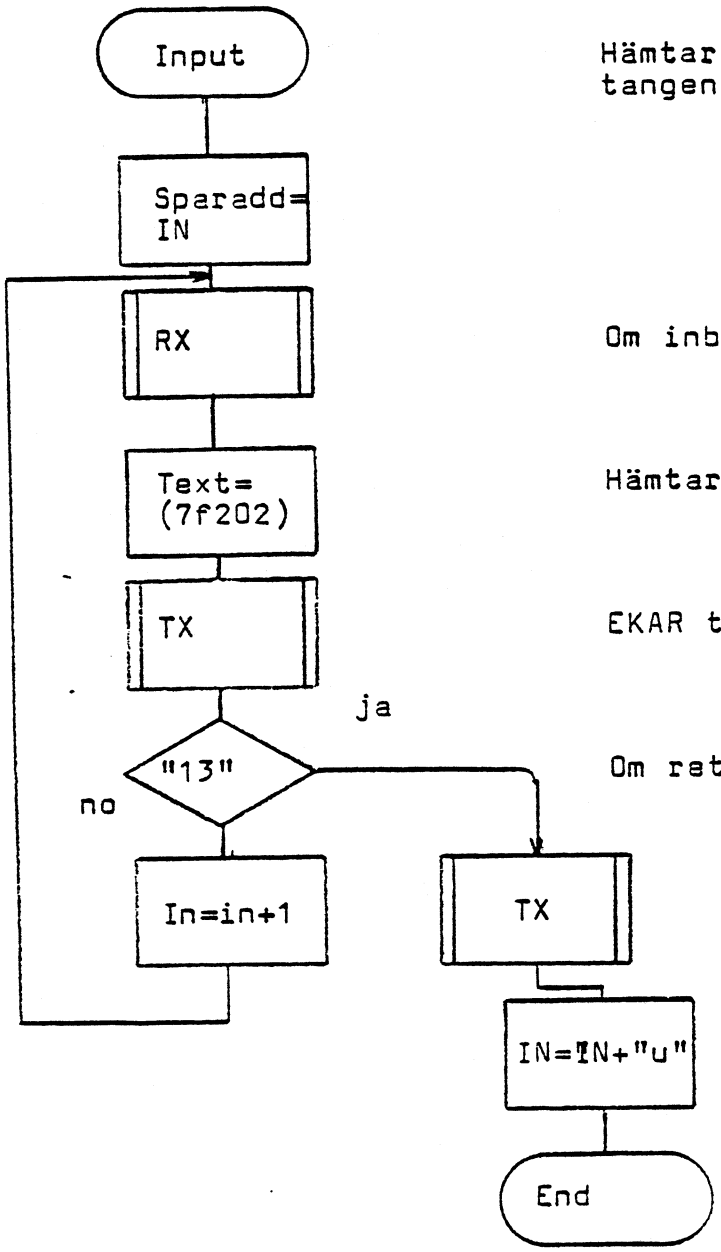
Skriver till dart

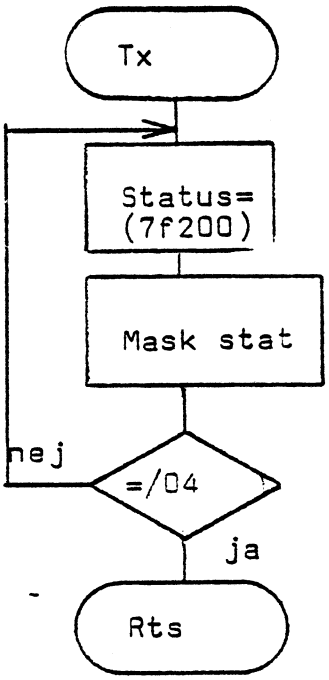
Räknar upp strängpekare

Om ascii "U"

Linefeed

Return

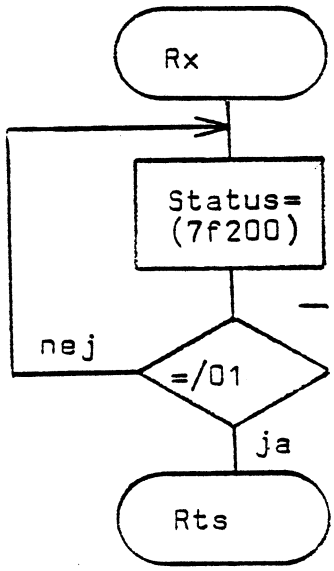

Input
Sparadd=
IN
RX
Text=
(7f202)
TX
"13"
ja
no
In=in+1
TX
IN=IN+"u"
End
Hämtar text från
tangentbord
Om inbuffert tom
Hämtar tecken
EKAR text
Om return
Lägger till
slut tecken



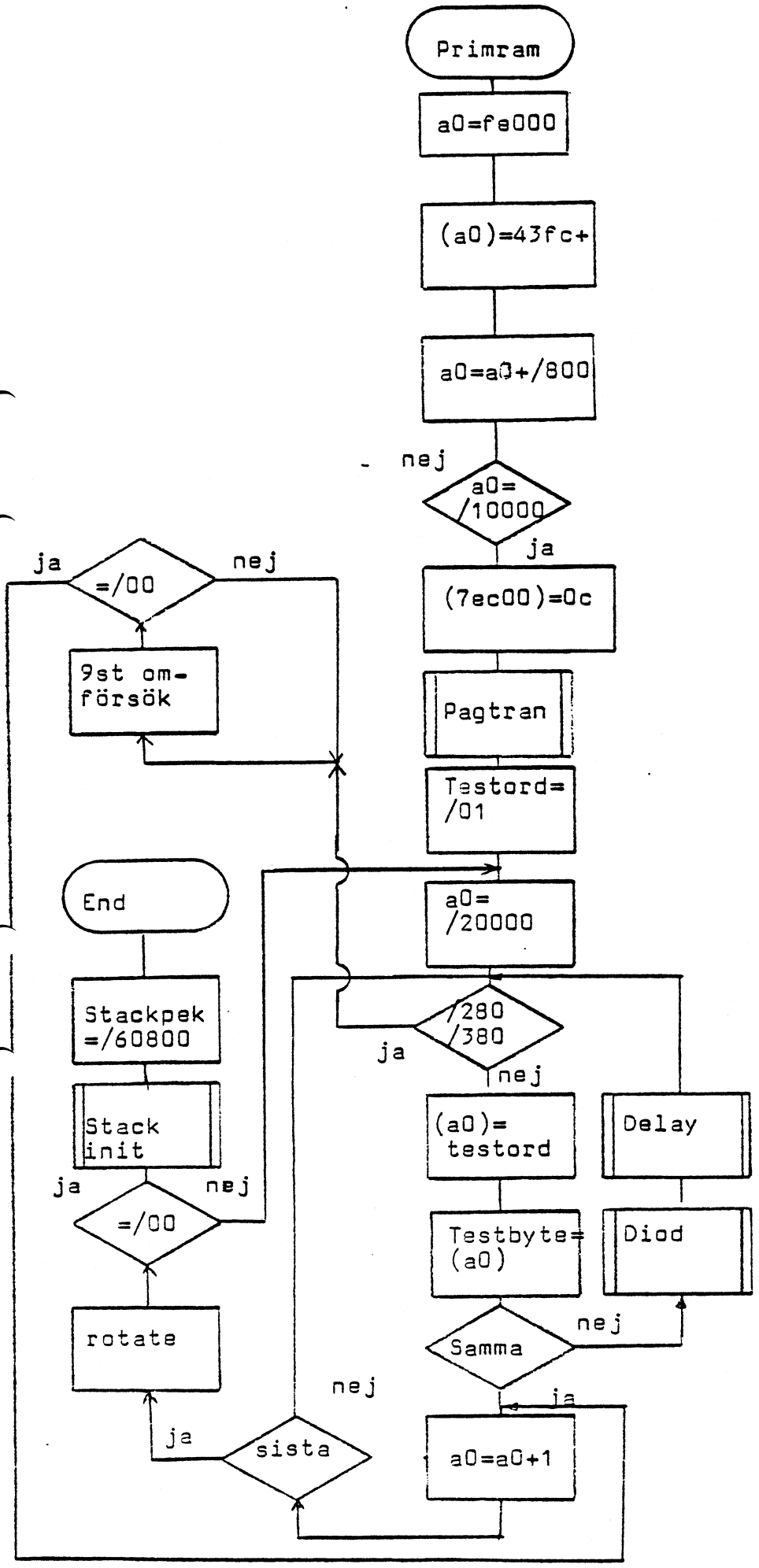
Om sändbuffert tom

Status dma

Tom?



Om tecken i buffert



Testar primärminne

Sätter upp mac för I/O page

SET partst biten

Sjuseg 7

Startadress

Omtest

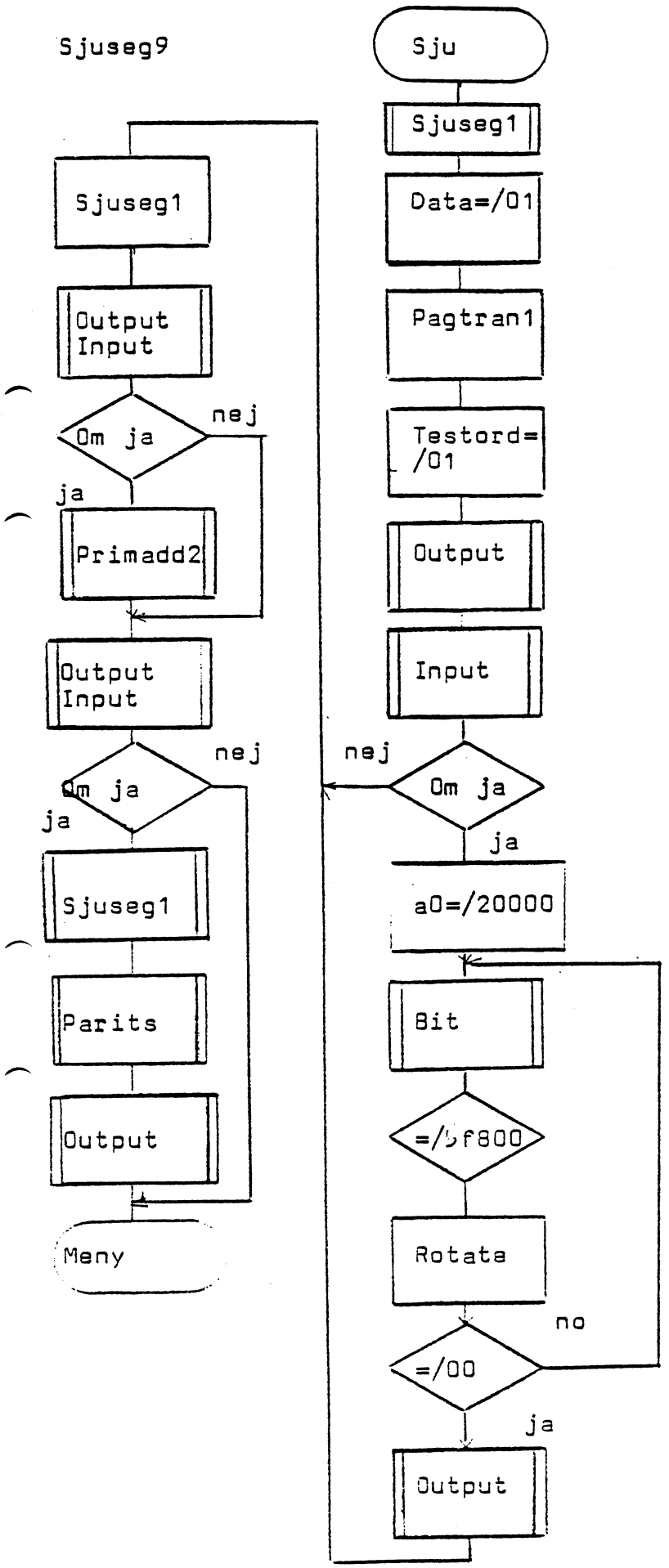
Om FARLIG adress

Skriver in biten

Hämtar inskrivet ord

Kollar om samma sak lagras

Nästa adress



Testar primärminnet
2K-256K
Bit,adress,paritet

Sätter upp maccen för block

"om bittest skall göras"

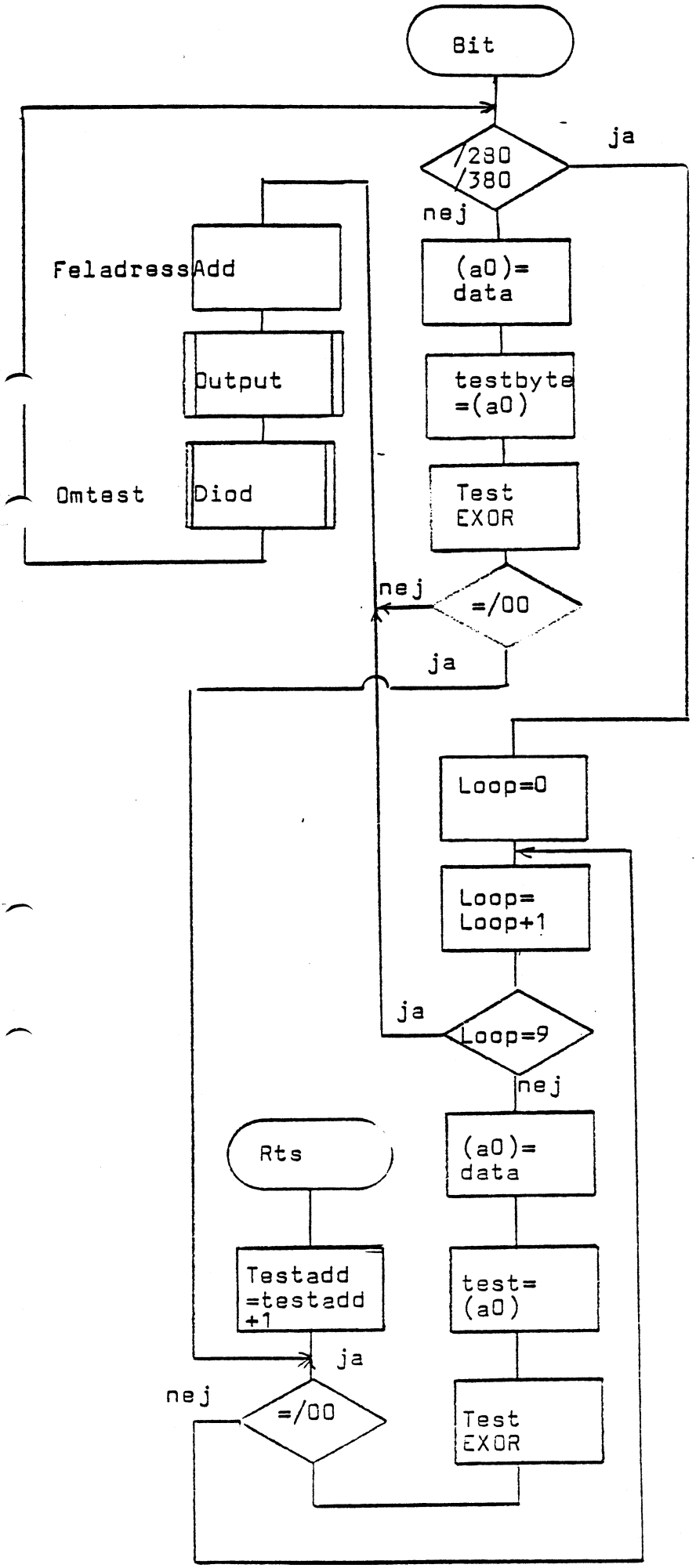
Hämtar svar från tangentbord

Start adress

Testar bit

Om 254K

"Bittesten klar"



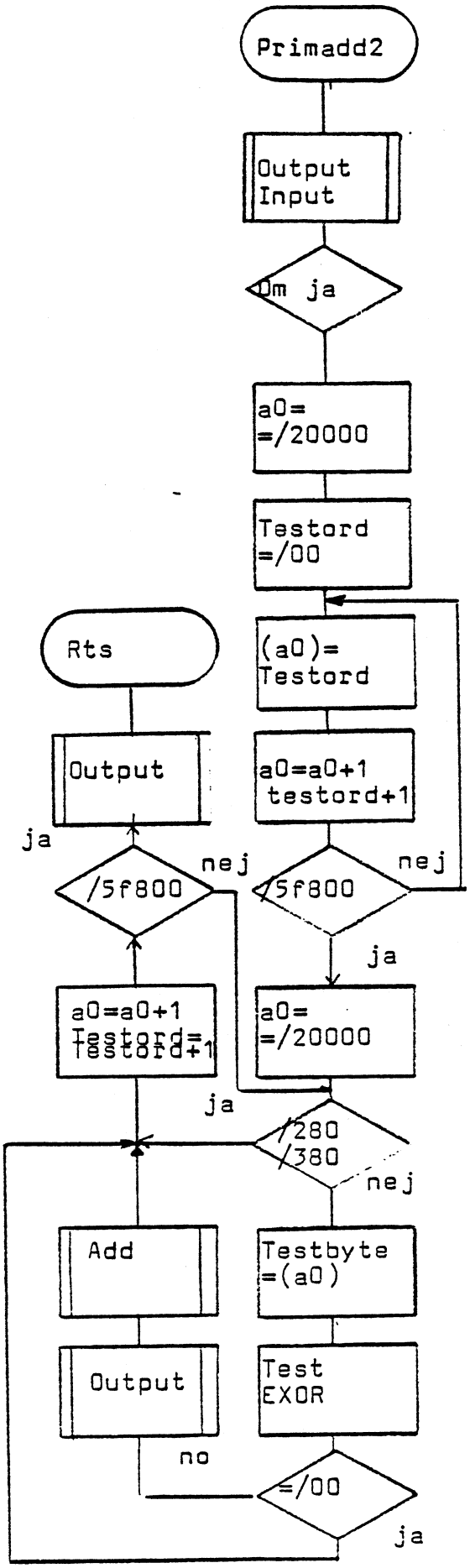
Lägger ut testmönster och testar om det är rätt

Skriver ut testmönster

Hämtar testmönster

Tar fram felbit

Ingen felbit



Adresstest
2K-256K

Om adresstest

Startadress

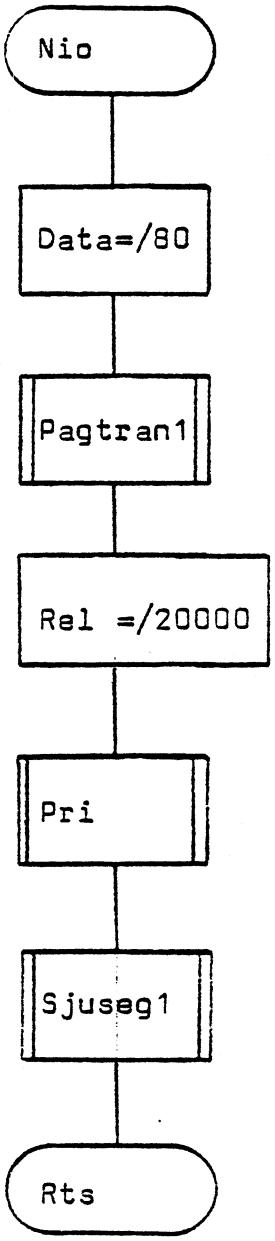
Skriver ut testmönster

Om sista adressen

Startadress

Hämtar mönster

Sjuseg 12



Testar primärminnet 256K-512K

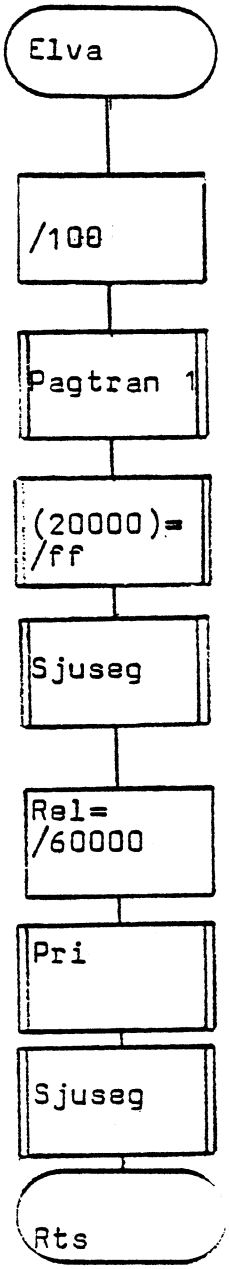
Sätter upp maccen

Relativ adress

Bti, adress och paritets-test

Sjuseg 8 + meny

Sjuseg 14



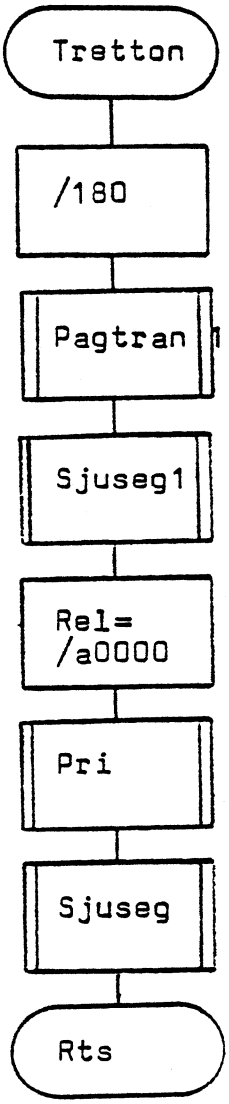
Testar om 1M i maskin

Sätter upp maccen för 512k-768K

Om det inte finns 1M ges BUSS ERROR

sjuseg 8 +meny

Sjuseg 17



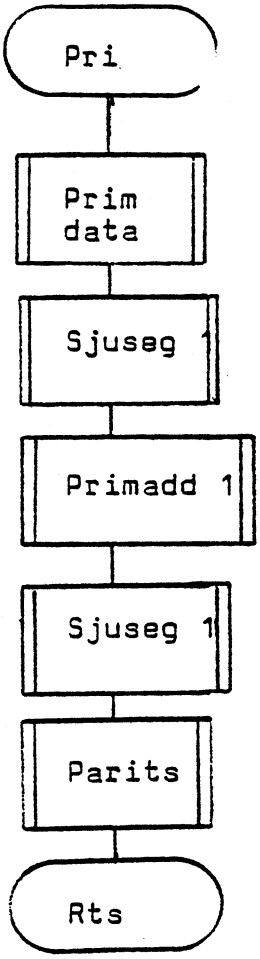
Testar 768K-1M

Sätter upp maccen för minnesblocket

Relativa adressen

Bit ,adress ooh paritets-test

Sjuseg 8 + meny

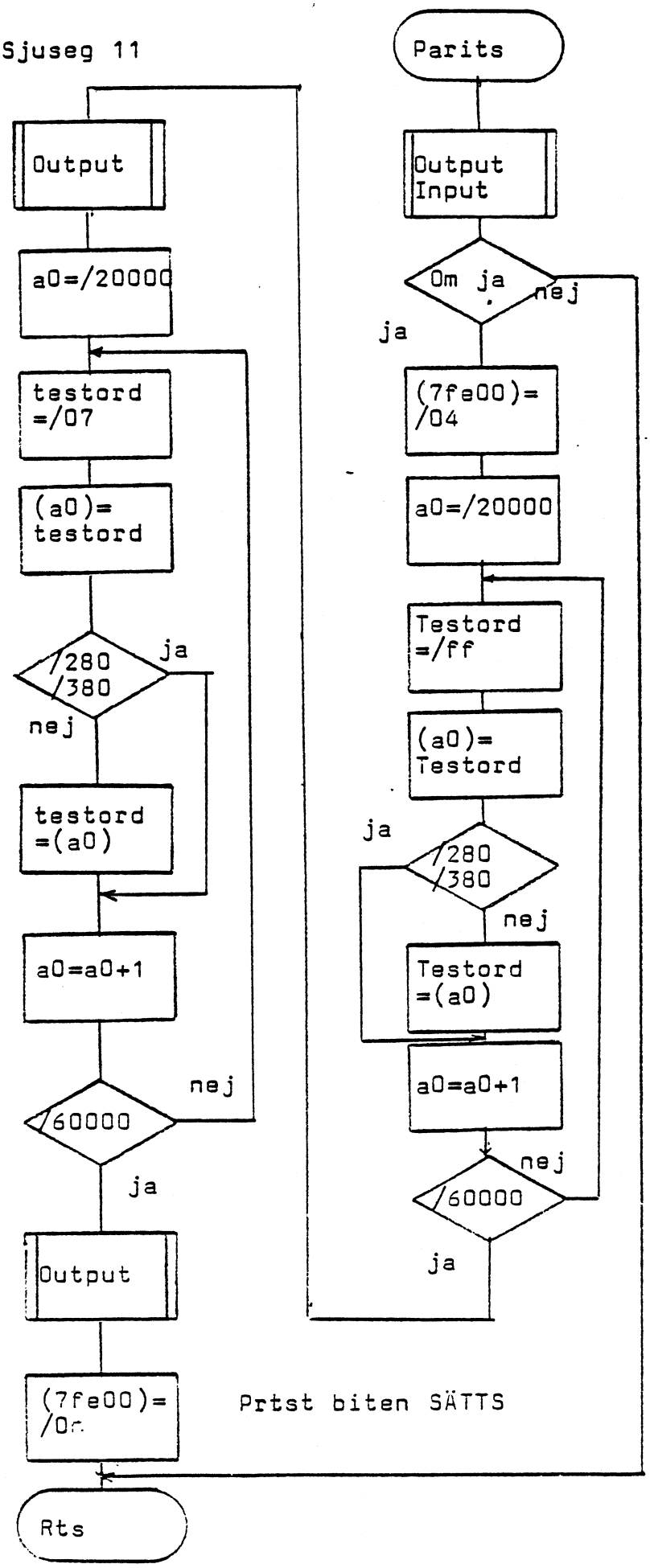


Testar bit
adress
paritet

Bit-test

Adress-test

Paritets-test



Testar paritetsbiten "krets"

"Om paritets test"

Tar bort paritetsflaggan
Partst biten nollställs

Startadress

Ger JÄMN paritet

Skriver in

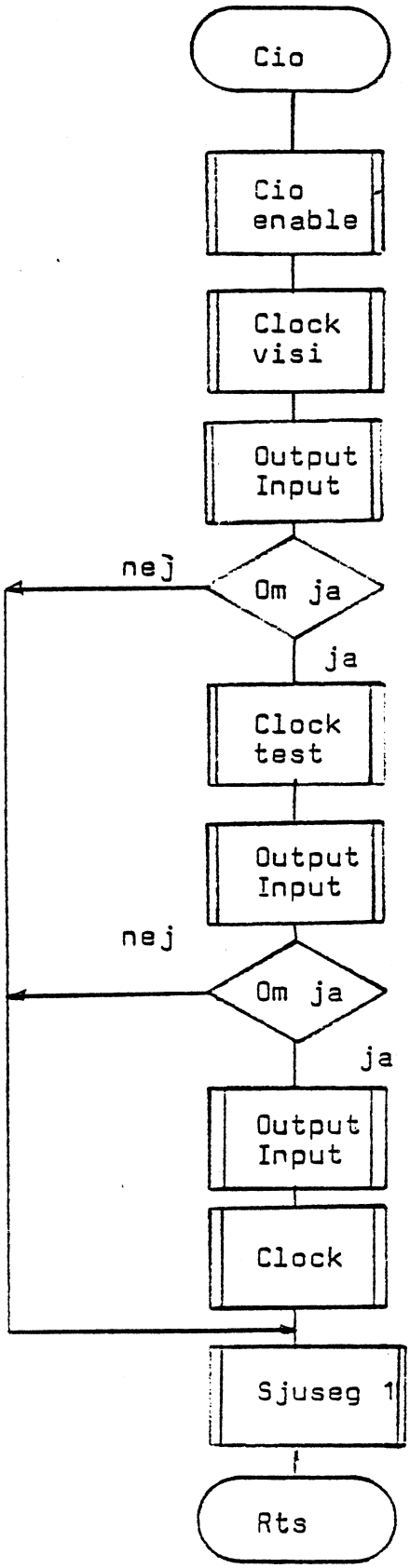
Om farlig adress

Om paritesfel görs
INTERUPT

Nästa adress

Sista adress

Sjuseg 22



Testar CIO+klocka

Gör enable

Visar klocka 8 sek

Om klockan skall testas

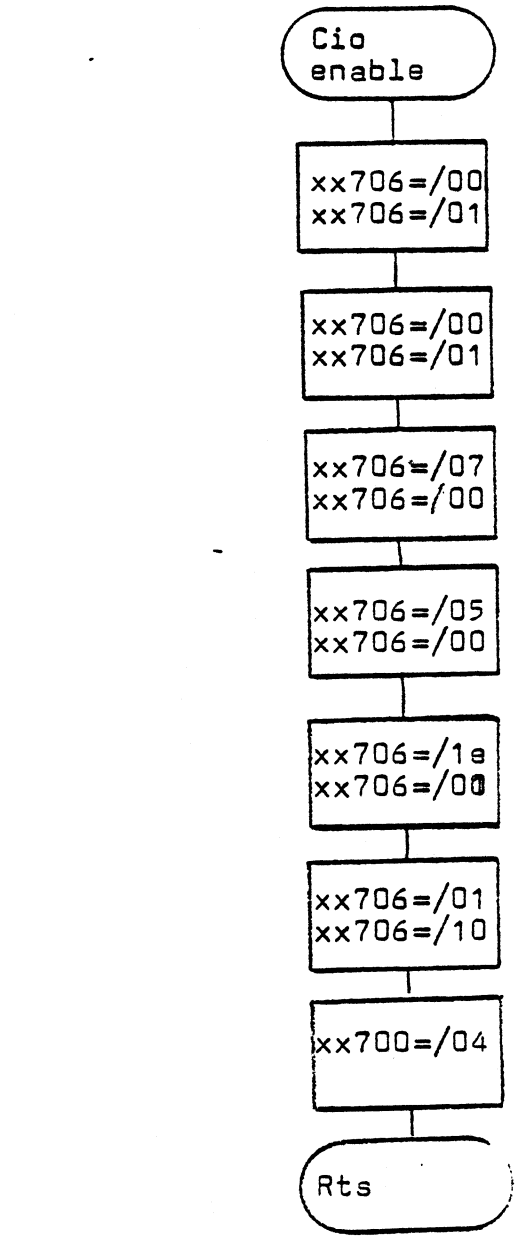
Testar klockan

Om tiden skall ändras

Till vad

Ändrar clockans innehåll

Sjuseg 8 + meny



Initierar port c

Resetar port c

Återställer reseten

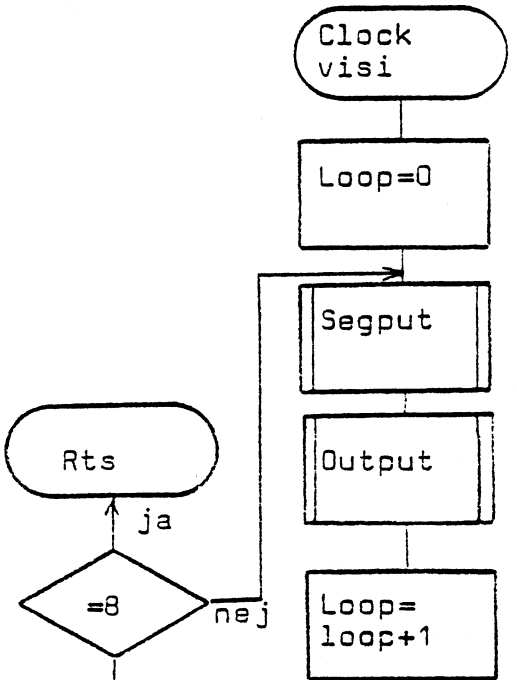
Alla utgångar

Polaritetsregister

Mode

Select port c

No select clock+nvram

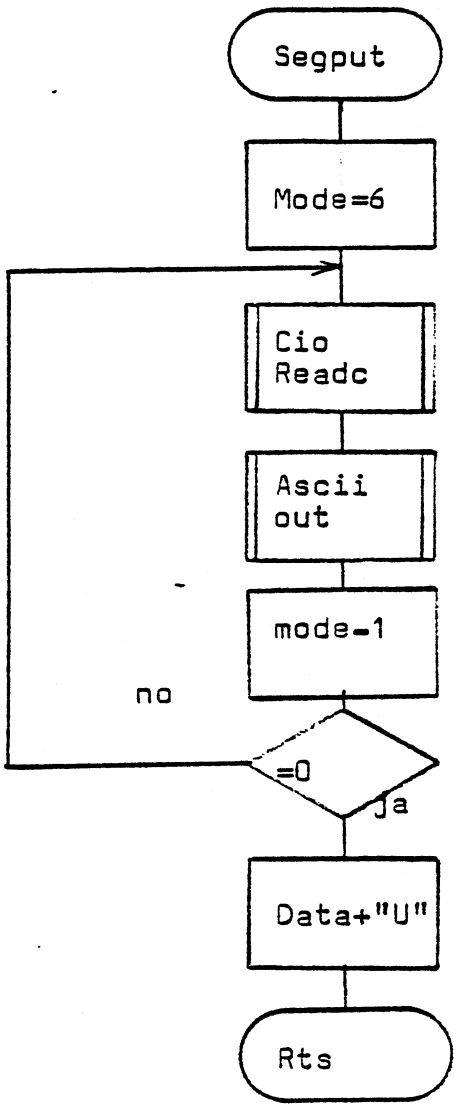


Skriver ut klockan under 8 sekunder

Antal sekunder

Hämtar data från klockan

Skriver ut tid



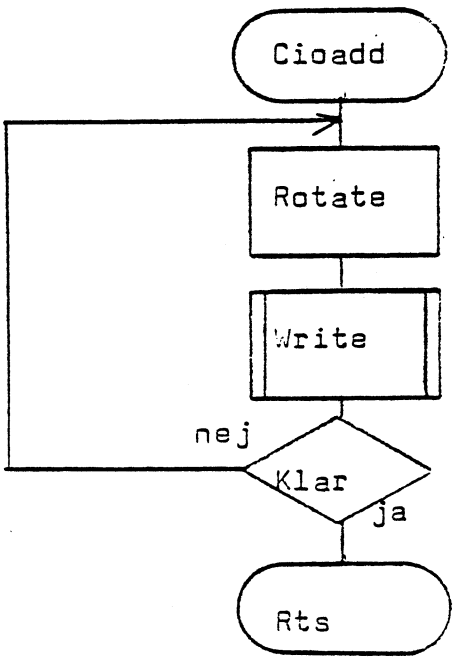
Tar fram data ur klockan

År först

Hämtar data

Tar fram ascii värdet för data

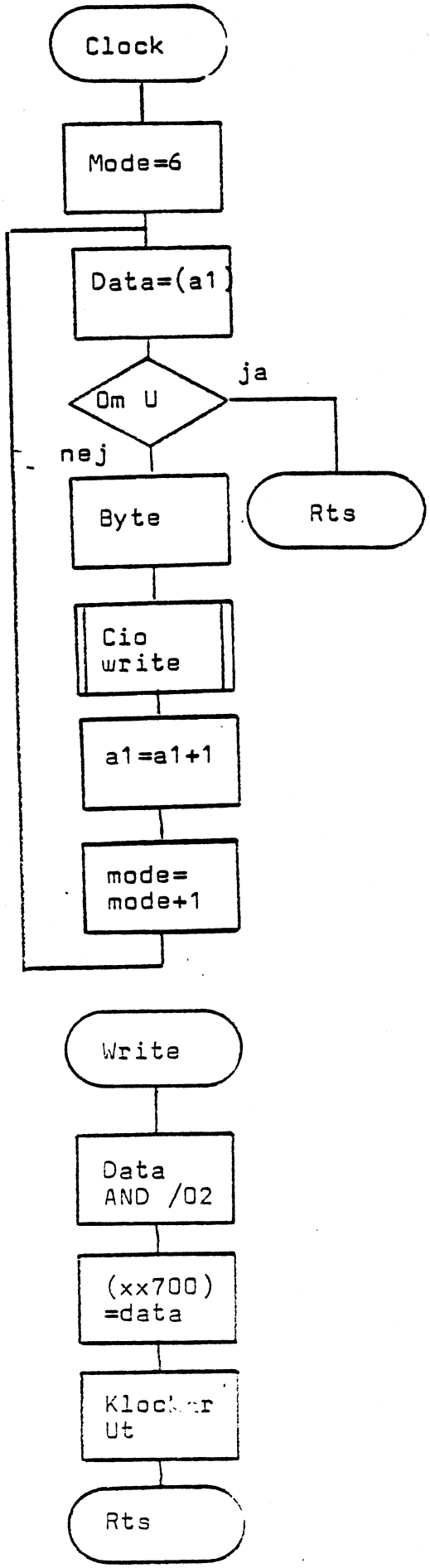
Avslutar data med "U"



Skriver kommando till klockan

Roterar indata

Lägger ut data till klocka



Ändrar tiden i klockan

Hämtar data

Om sista tecknet

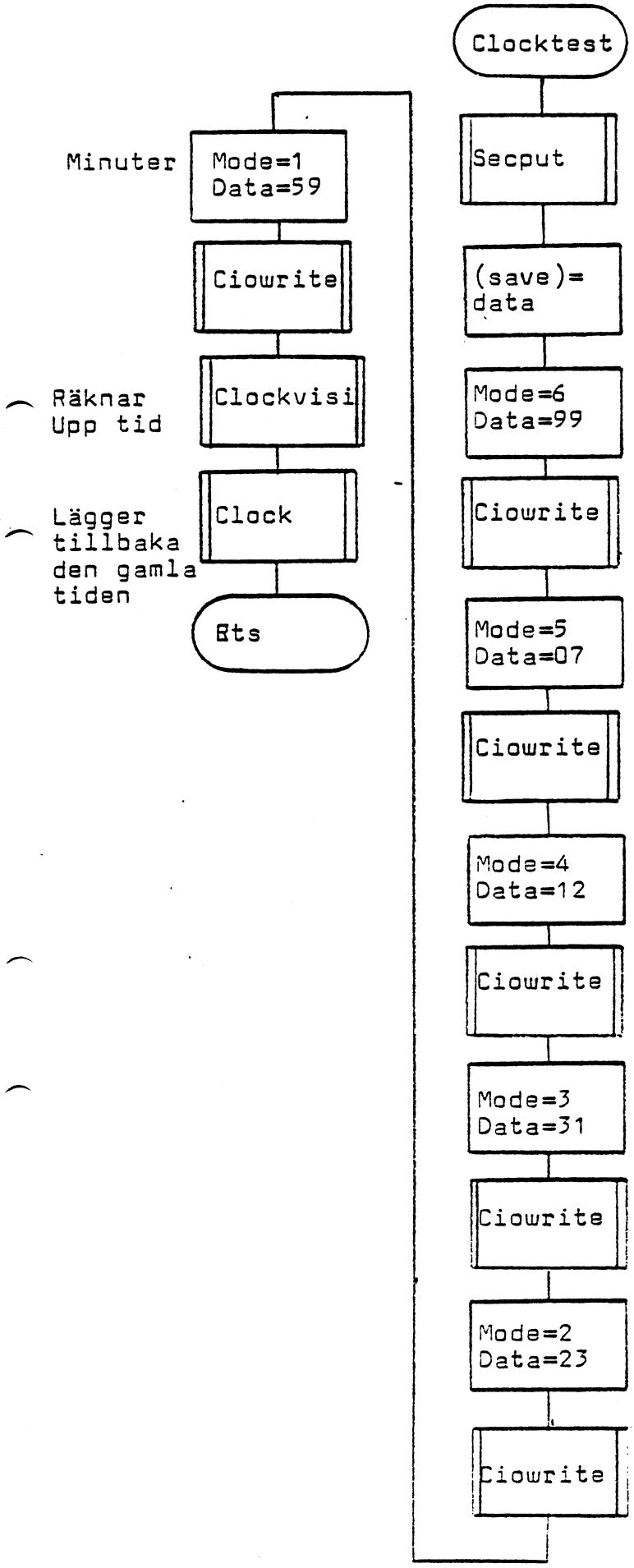
Lägger ihop becknen till BYTE

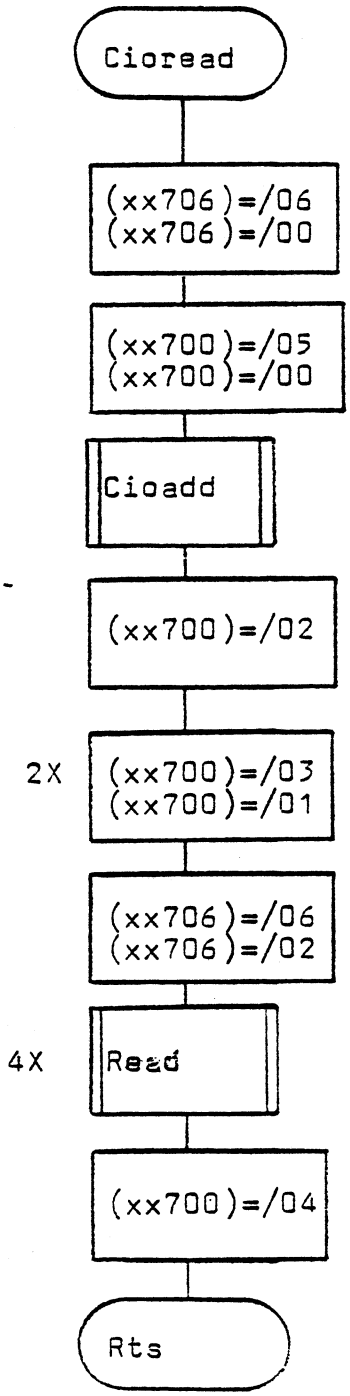
Skriver in i klockan

Skriver data till klocka

Tar fram data biten

Skriver ut data


Clocktest
Testar klockan. Stegar över 000000000 i tid
Secput
Hämtar data i klockan
(save)= data
Spara data i arean save
Mode=6 Data=99
År 99
Ciowrite
Mode=5 Data=07
Dag i veckan "söndag"
Ciowrite
Mode=4 Data=12
Månad "dec"
Ciowrite
Mode=3 Data=31
Dag "31"
Ciowrite
Mode=2 Data=23
Timmar "23"
Ciowrite
Minuter
Mode=1 Data=59
Ciowrite
Räknar Upp tid
Clockvisi
Lägger tillbaka den gamla tiden
Clock
Rts



Läser från port C

Alla portar utgångar

Clock PC0

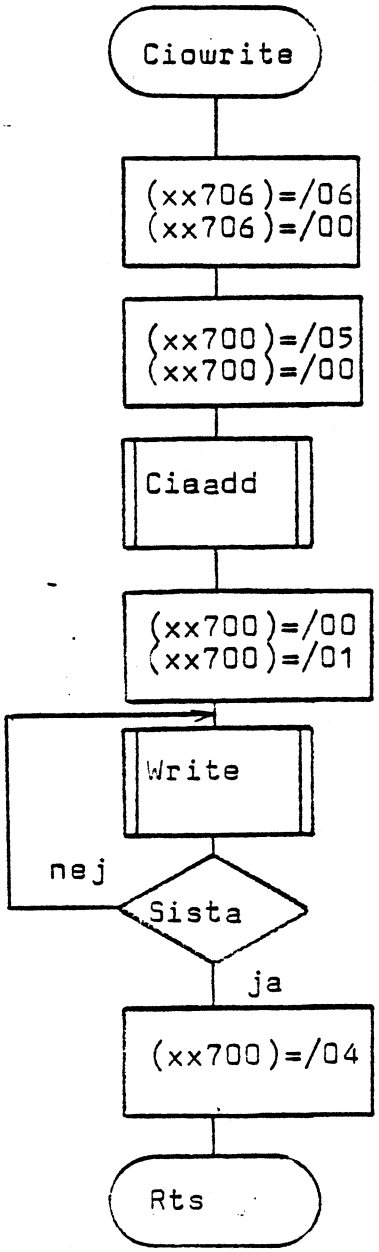
Vad som skall tittas på

Read mode

Clock hög-låg

En ingång

No enable PC2 "hög



Skriver till klocka
Från port C

Alla utgångar

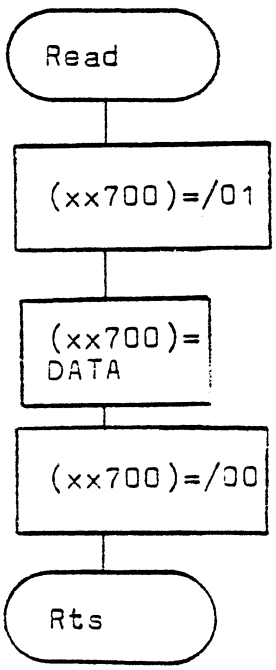
Clock hög-låg
No select-select

Hämtar vad som skall
skrivas in

Write mode

Lägger ut data ord

No select clocka



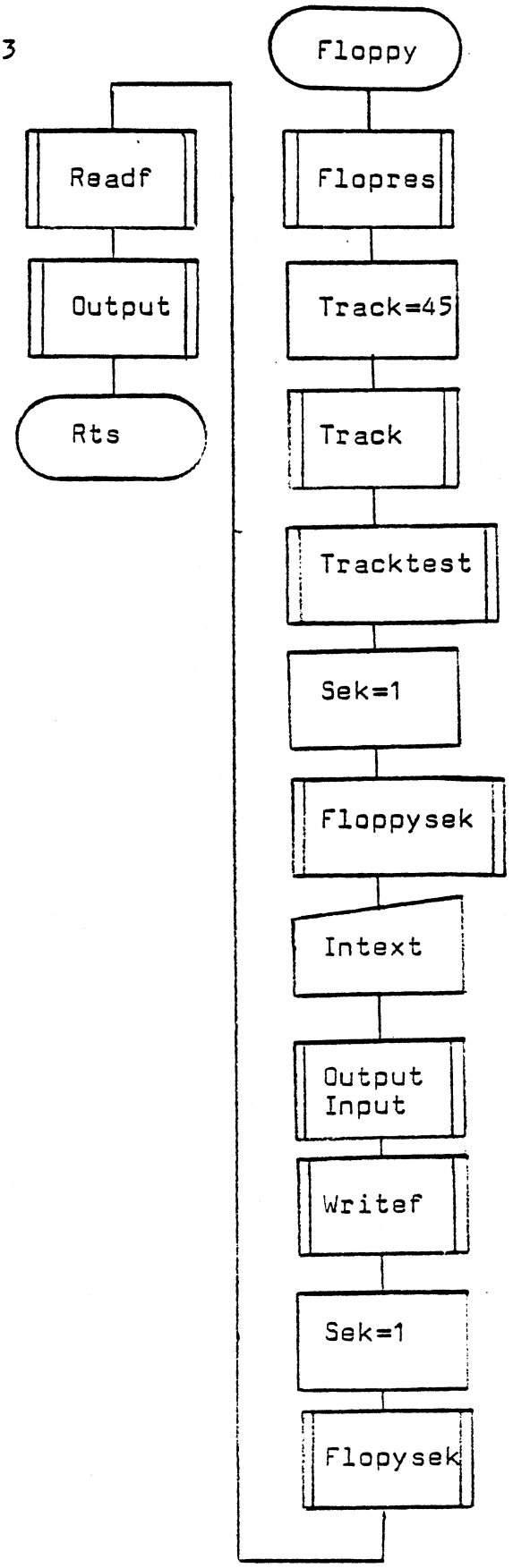
Läser data från klocka

klocka "hög" PC0

Hämtar data

Klocka låg

Sjuseg 23



Testar floppy controll

Resetar floppy controll

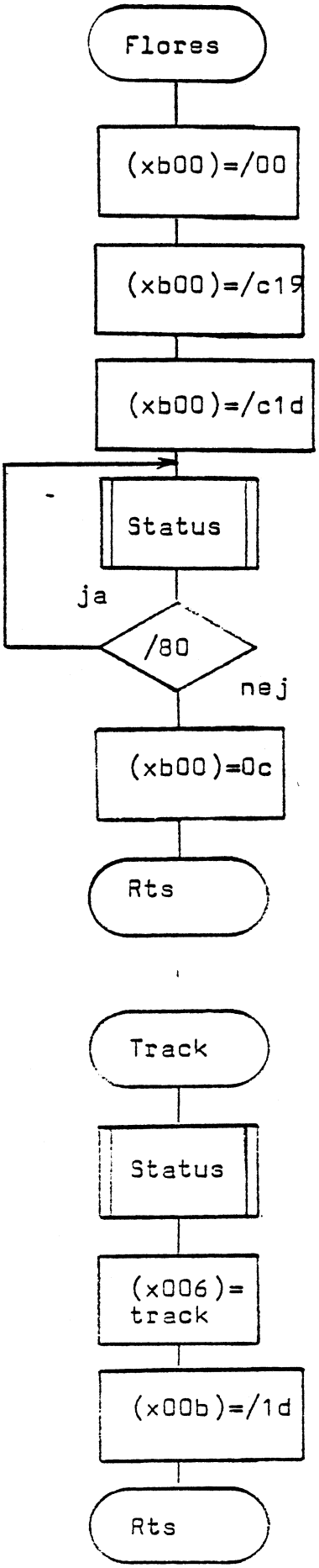
Läser på skiva

Stegar till spår 45

Kollar om rätt spår

Väljer sektor ett

Skriver in data på skiva



Resetar kontrollen

Startar motor och
initierar floppy

Head load time

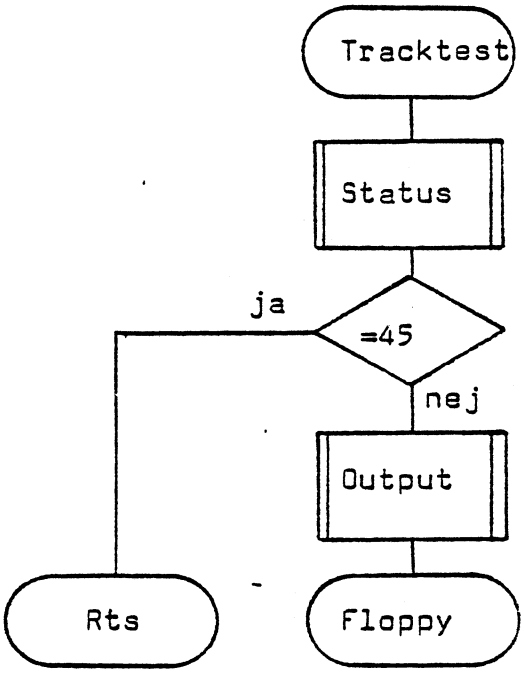
Kollar status

Drive not ready

Stegar till spår

Initierar spår i kontrol

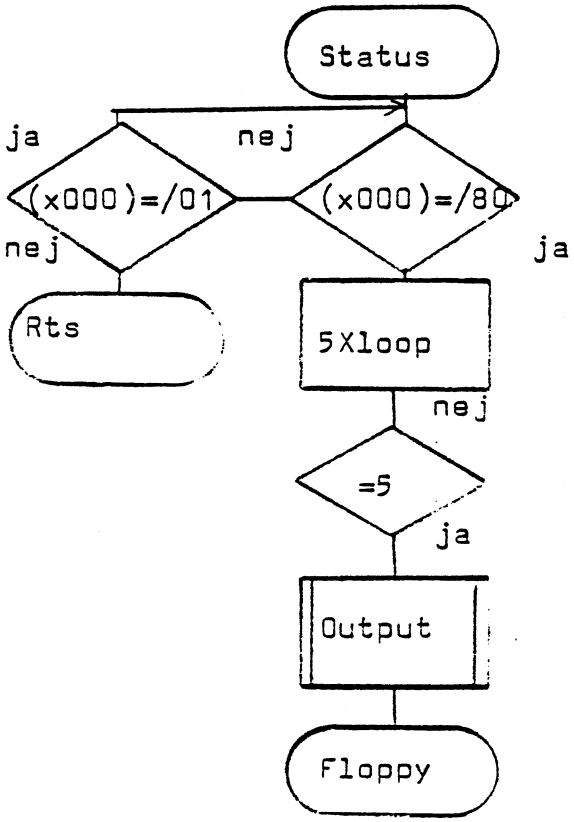
Stegar till spår



Testar att det är stegat till spår 45

Controllen felaktig
Skivan ej formaterad

Gör omtest

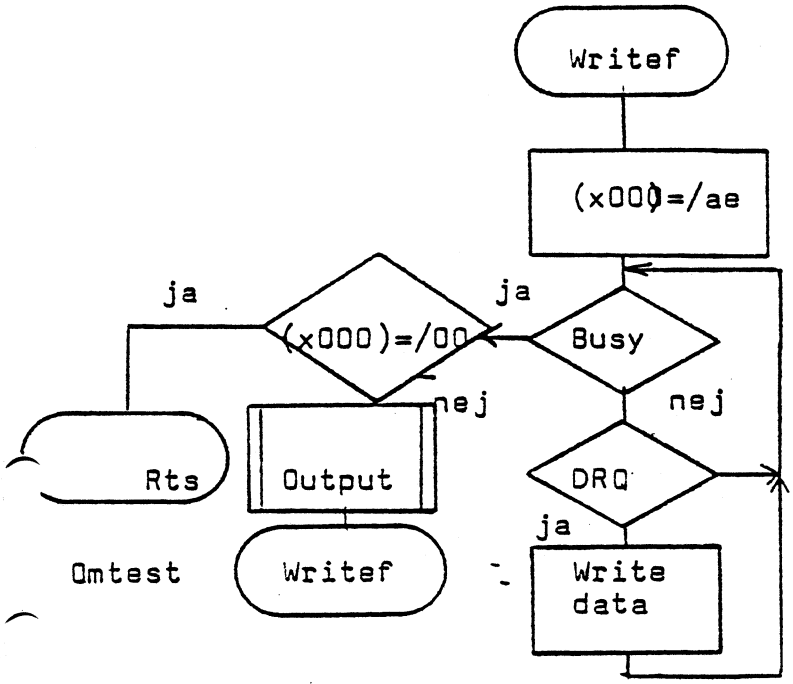


Kollar att kontrollen inte är upptagen

Kollar not ready

Floppyn ej ansluten

Gör omtest

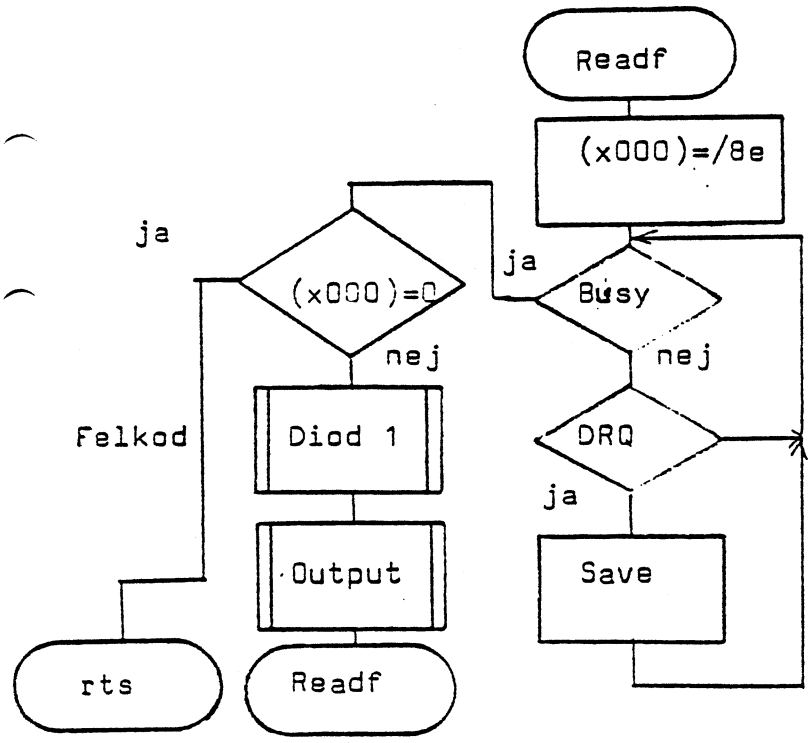


Skriver ut data på floppy

Om kontrollen busy

Om data request

Skriver till floppyn

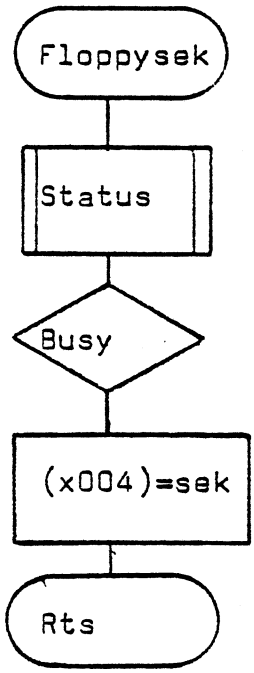


Läser data från floppy

Read mode

Lagrar data i minne

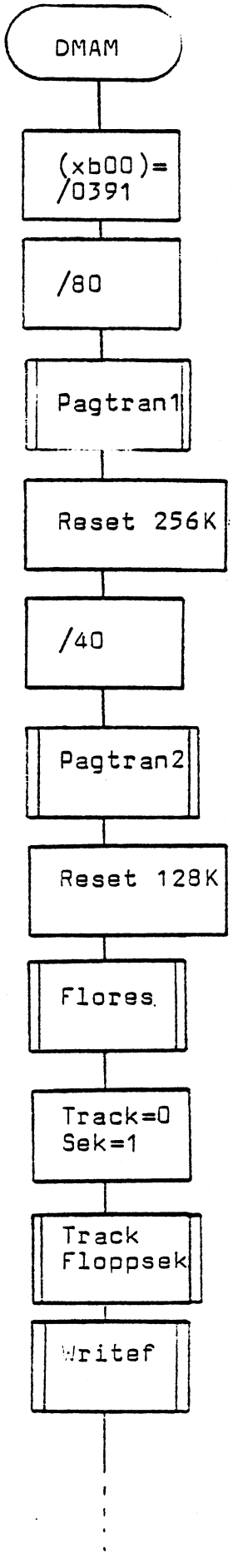
Omförsök



Initierar vilken sektor

Skriver sektor till kontrollen

Sjuseg 24



Tester dma

Stannar floppy motorn

Sätter upp maccen för 256K-

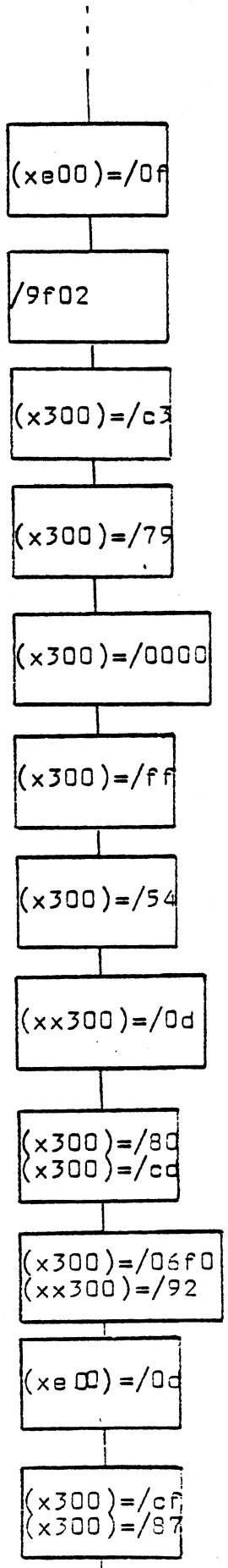
Sätter upp mac för Skrivarea 128K

Resetar floppyn

Initierar spår ock sektor

Skriver in data floppy

Omtest DMA39



SET bit SYSFS
"intern hämtning dma"

Skriver i DMA-MAP

Reset dma

Port a-portb

Startadress

Blockstorlek 256K

Vad port a är

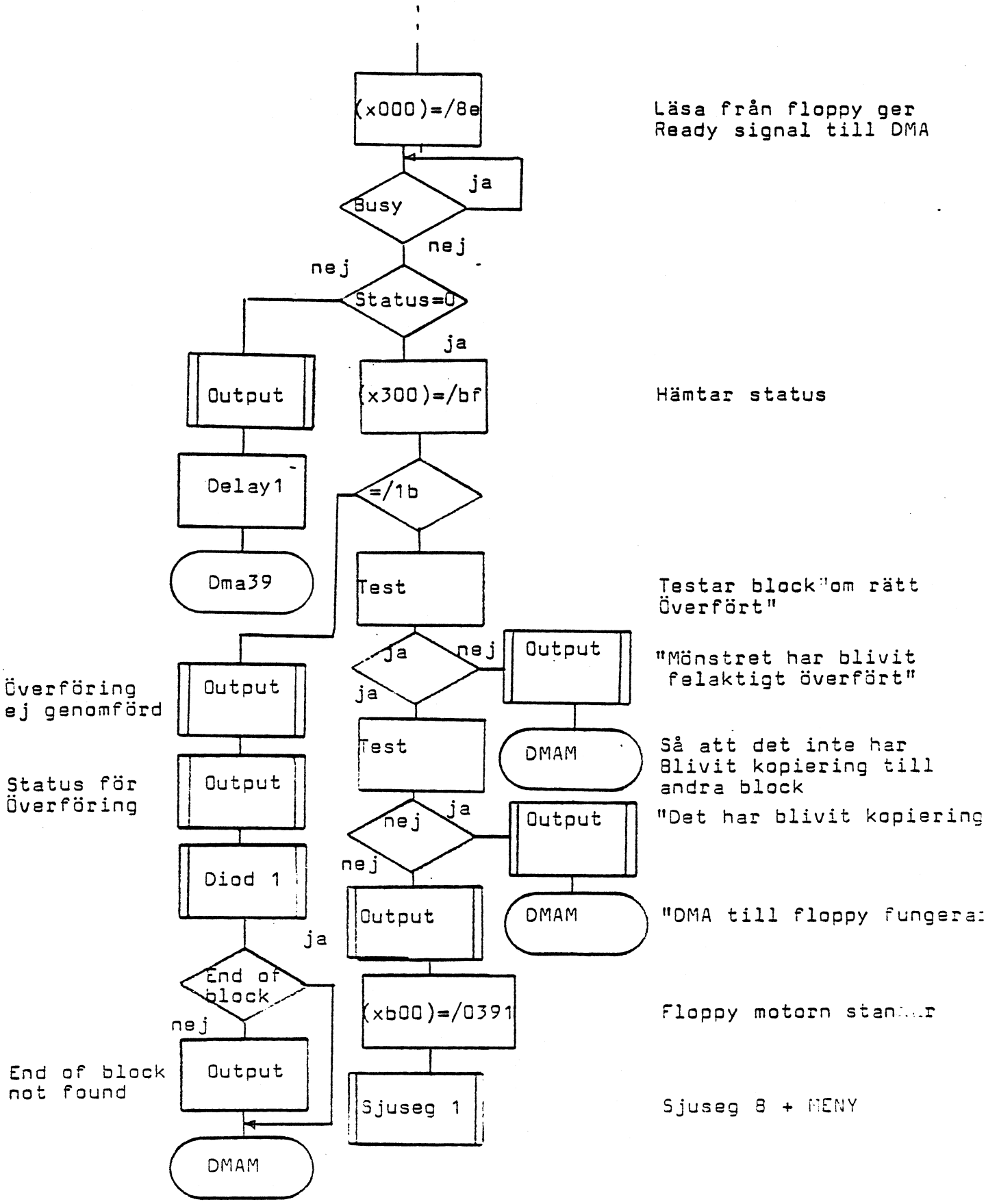
Cykellängd 3

Burst

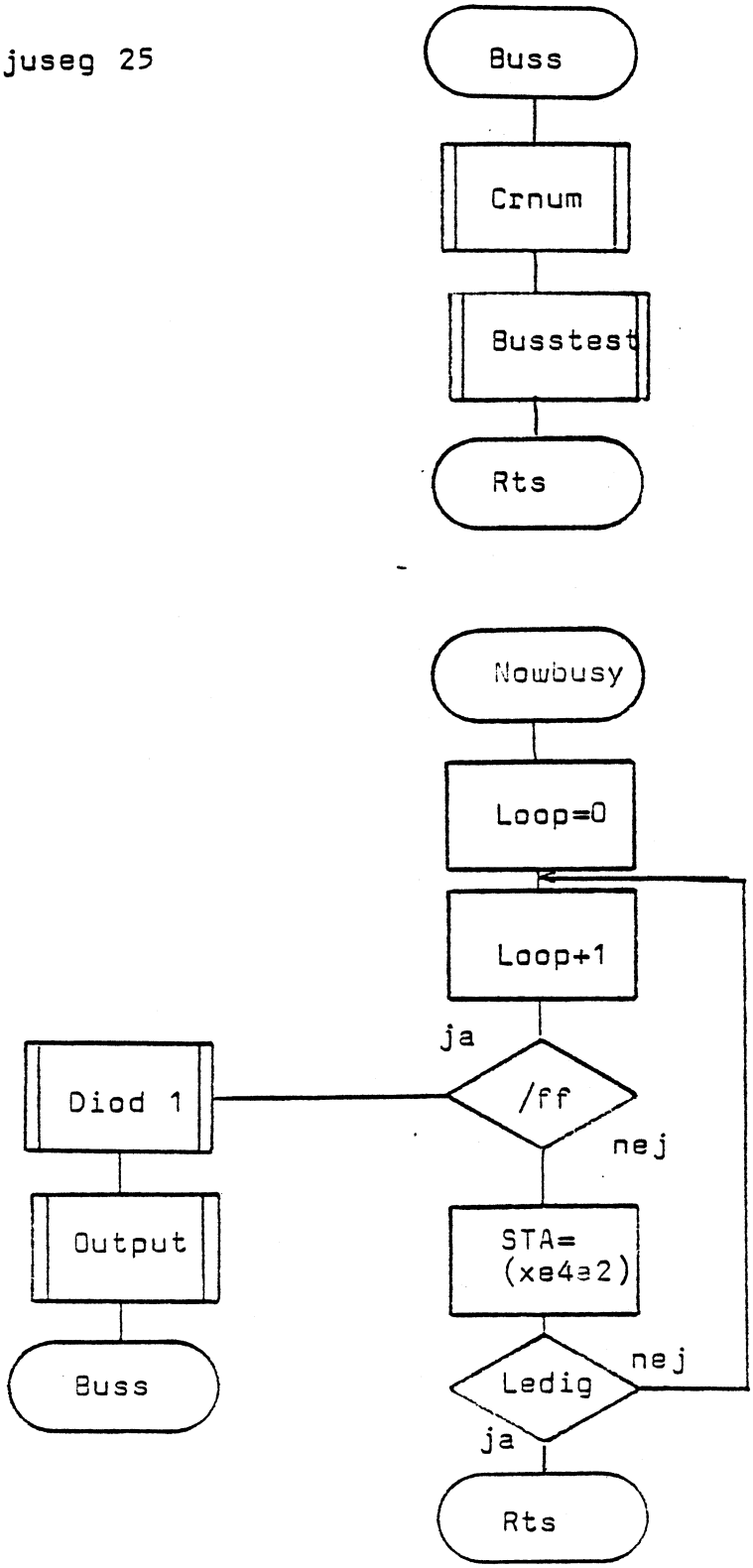
Startadress
READY aktiv låg

DMADIS signal

Reset and load
START DMA


(x000)=/8e
Busy
ja
nej
nej
Status=0
ja
Output
(x300)=/bf
Hämtar status
Delay1
=/1b
Dma39
Test
Testar block "om rätt Överfört"
ja
nej
Output
"Mönstret har blivit felaktigt överfört"
Överföring ej genomförd
Output
ja
Test
DMAM
Så att det inte har Blivit kopiering till andra block
Status för Överföring
Output
nej
ja
Output
"Det har blivit kopiering
Diod 1
nej
Output
DMAM
"DMA till floppy fungera
ja
End of block
(xb00)=/0391
Floppy motorn stan
nej
End of block not found
Output
Sjuseg 1
Sjuseg 8 + MENY
DMAM

Sjuseg 25



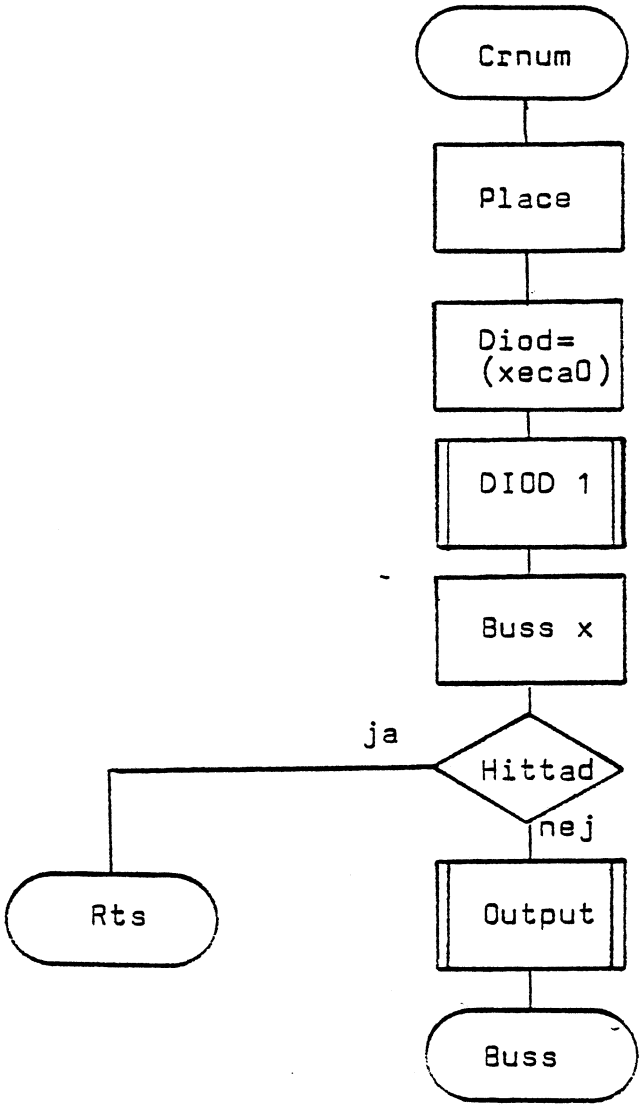
Testar busskortet

Kollar i vilken kanal interfacet sitter i

Testar om vi får kommunikation med

Om kontrollen är ledig

Kontrollen blir inte ledig

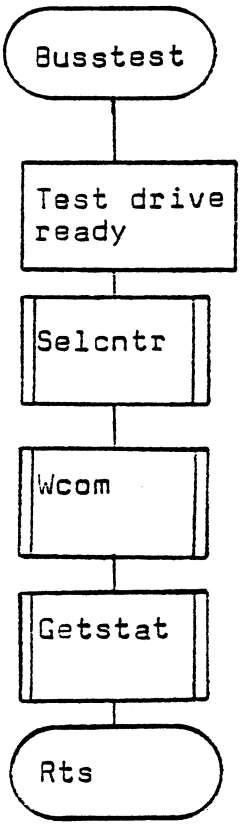


Tar reda på var interfacet sitter

Genererar RSCB signal

Kollar vilken buss

"hittar inte rätt kort"



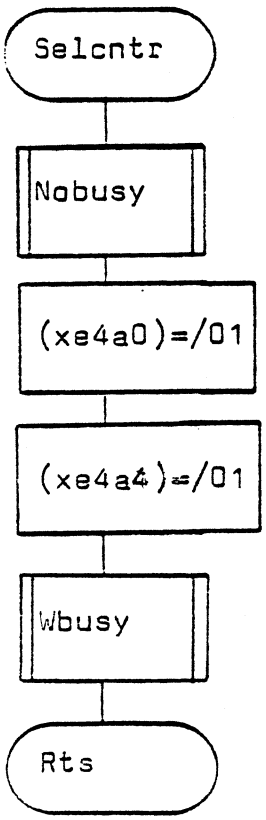
Testar om vi får kommunikation med interfacet

Test kommando

Genererar "select puls"

Skriver kommando till kontrollen

Hämtar status



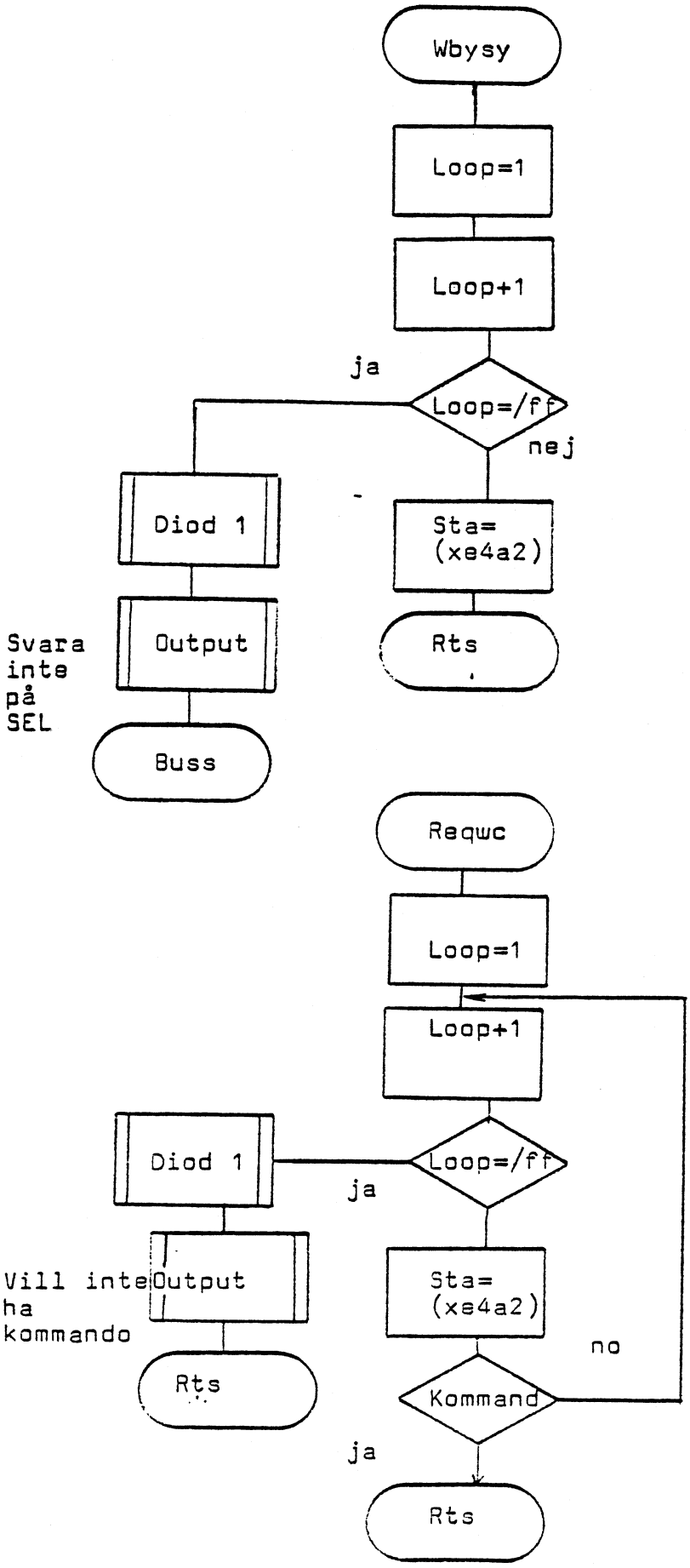
Genererar SELect puls

Om kontrollen upptagen

Genererar DATA STROBE "OUT"

Genererar selstrobe C1

Om konttollen ledig



Kollar om kontrollen aktiv

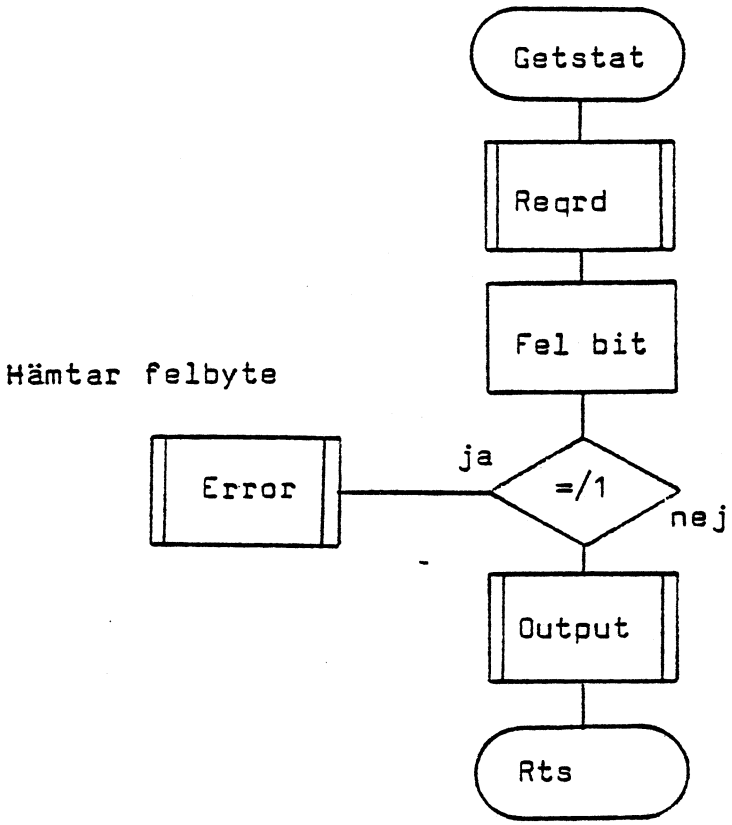
Kontrollen blir inte aktiv

Hämtar status

Om kontrollen vill ha kommando

Kontrollen vill inte kommando

Hämtar status



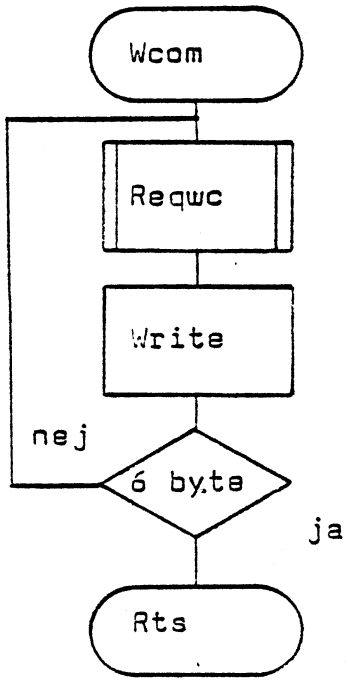
Hämtar status för utfört kommando

Om kontrollen vill sända Data

Tar fram felbit

Om error byte

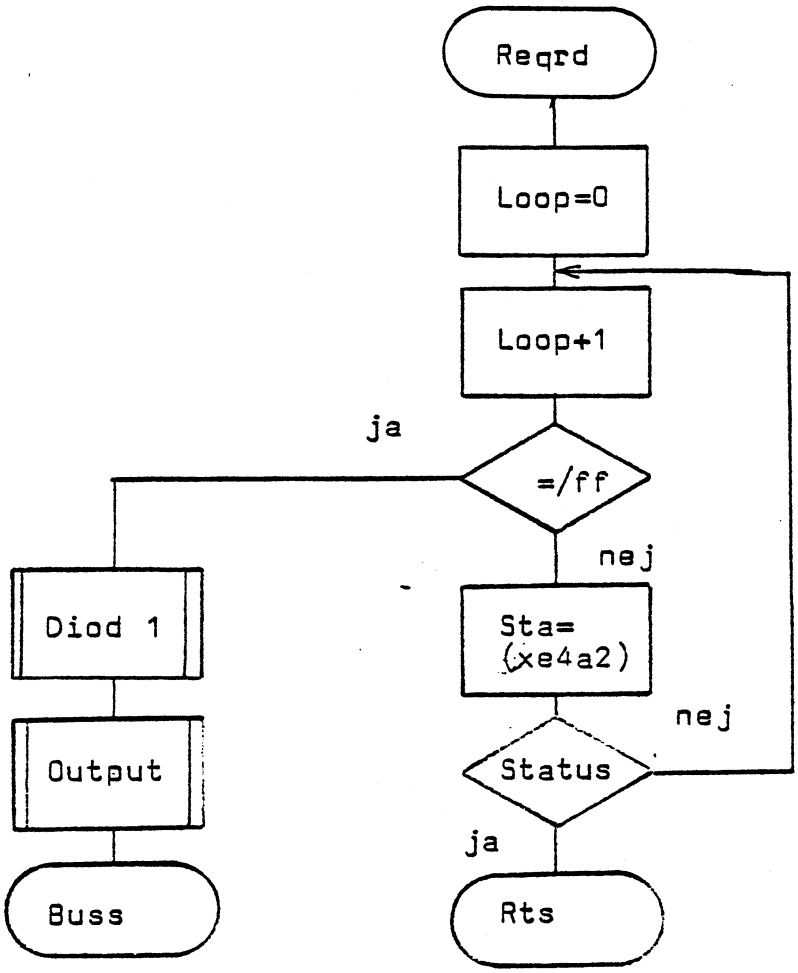
Drive is ready



Skriver kommando

Om kontrollen vill ha kommando

Skriver ut kommandot

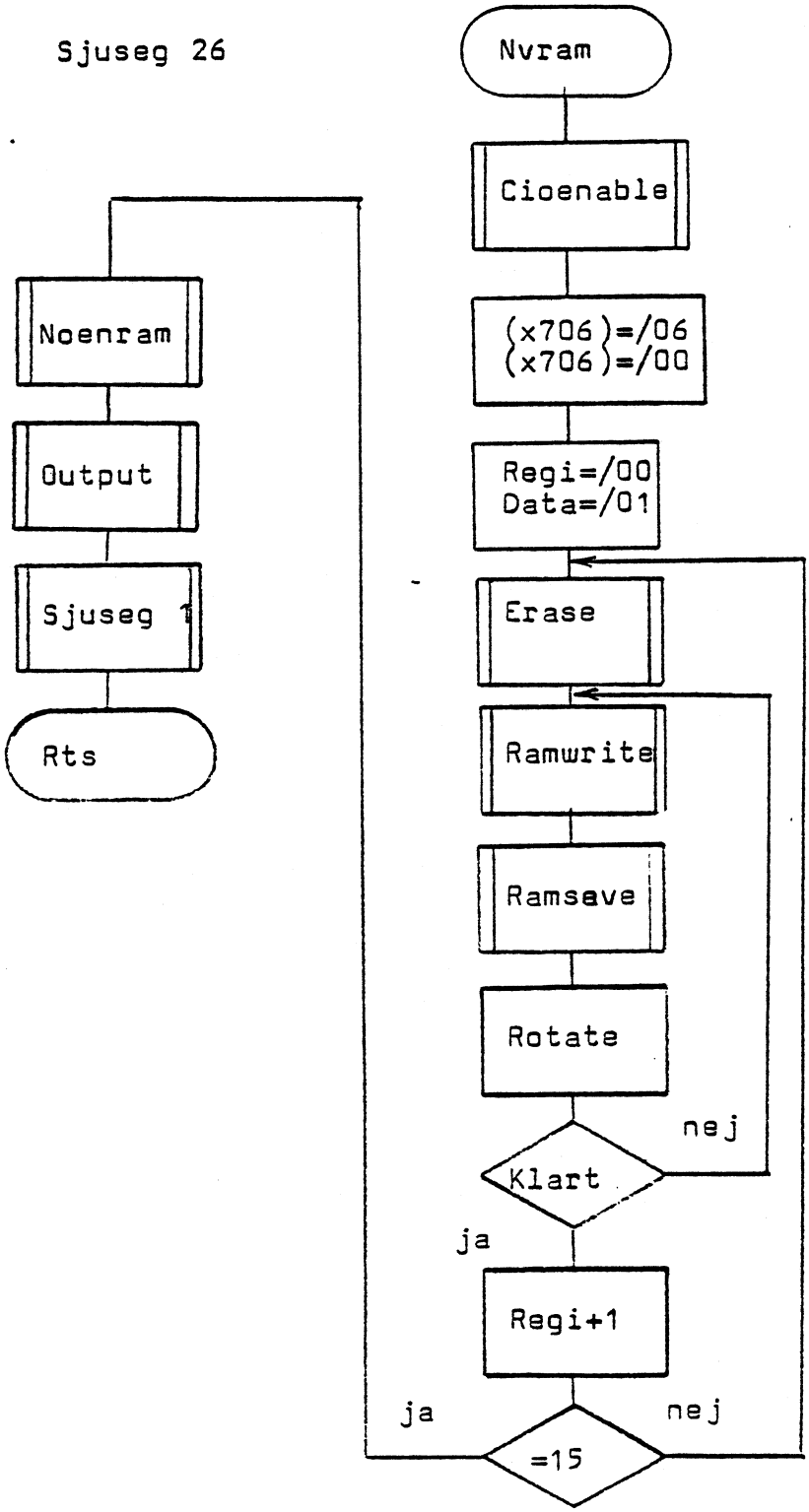


Om kontrollen vill sända status

Om kontrollen inte vill sända status

Hämtar status

Sjuseg 26



Initierar cio

Alla utgångar

Register noll
Vandrande etta

RADERAR nvramet

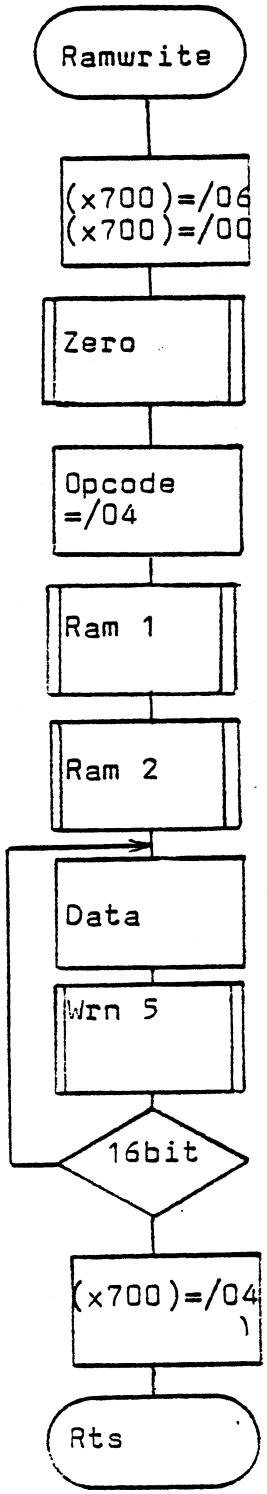
Skriver in testmönstret

Testar nvrammets innehåll

Roterar testmönstret

Nästa register

Om sista



Skriver in testmönstret i nvram

Alla utgångar

Enable nvram

Write mod

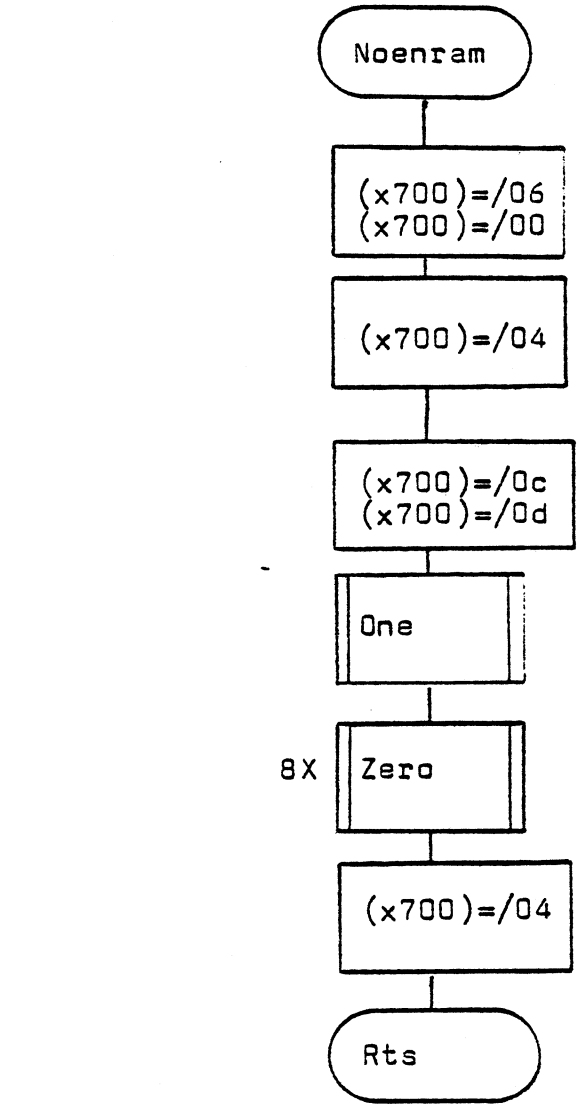
Skriver opcode

Vilket register

Skriver ut data

Om 16 bitar inskriv

Klocka låg + no enable



Alla utgångar

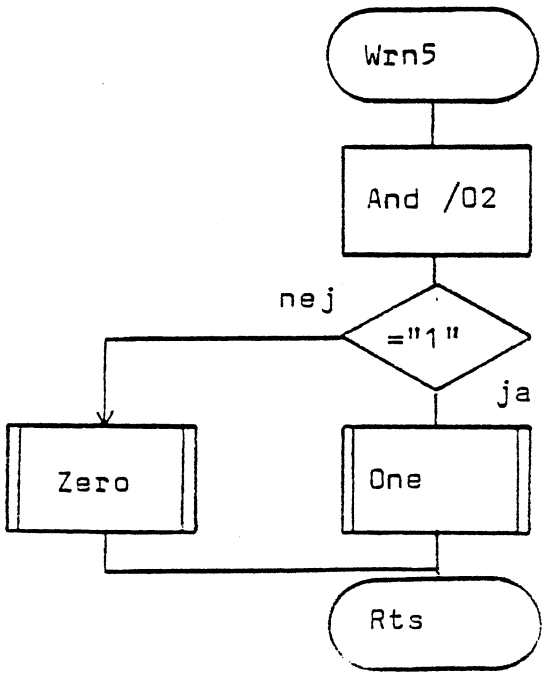
No enable clock

Chip select + klockpuls

Skriver ut ETTA

Skriver ut 8 nollor

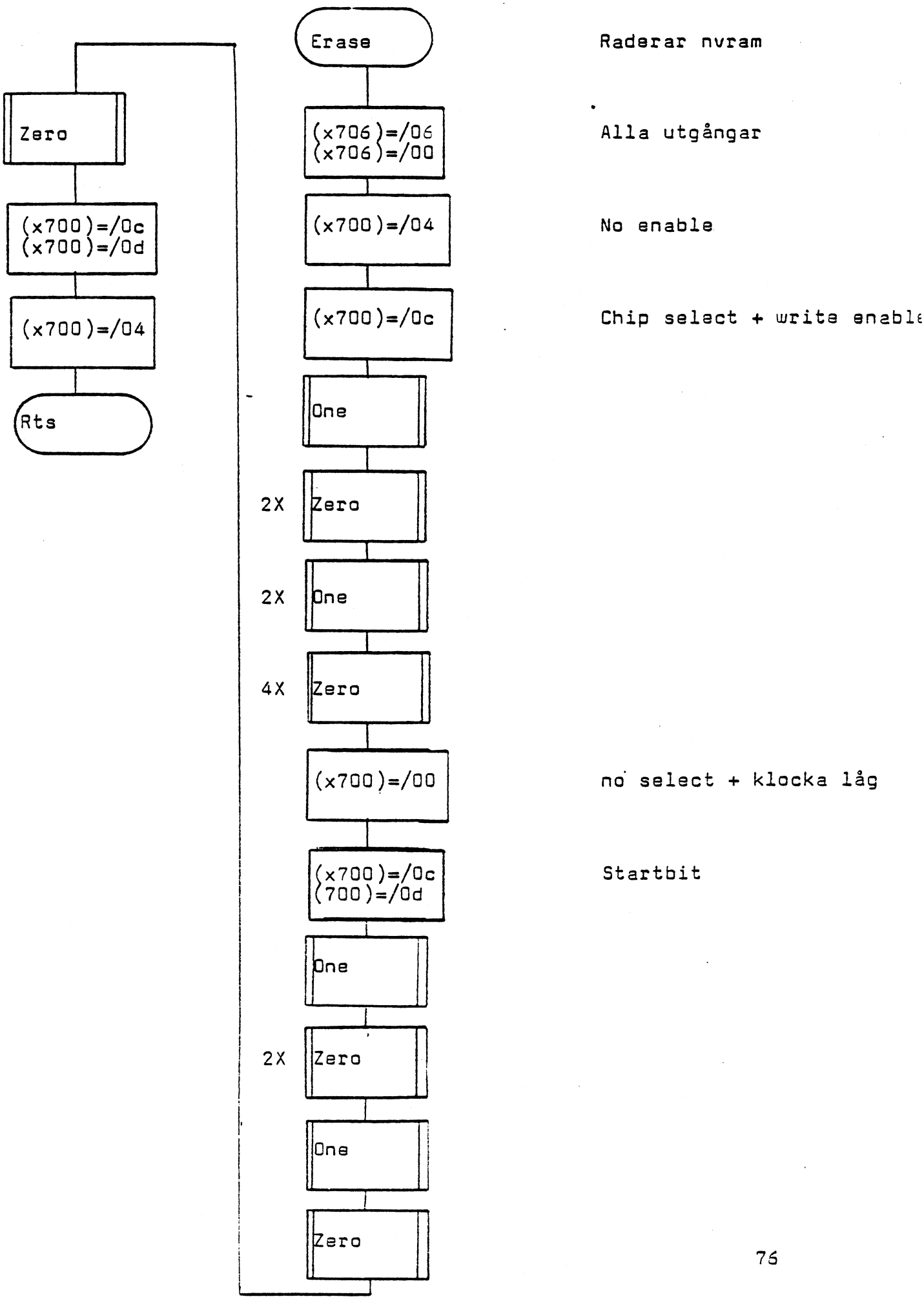
klockpuls+noselect

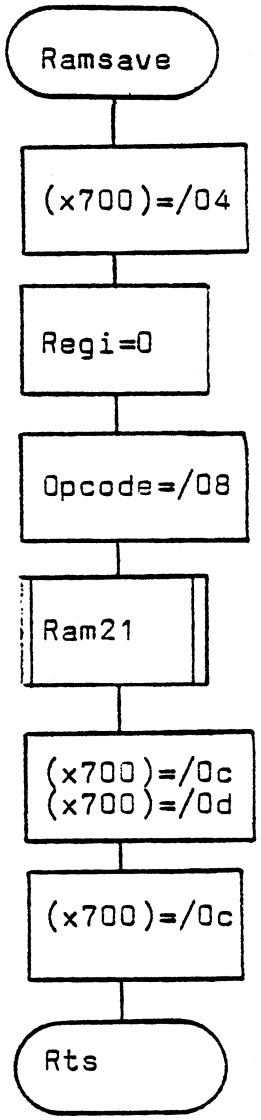


Kollar om etta eller noll
Skriver sedan in data

Tar fram data bit

Om etta


Erase
Raderar nvram
Zero
(x706)=/06
(x706)=/00
Alla utgångar
(x700)=/0c
(x700)=/0d
(x700)=/04
No enable
(x700)=/04
(x700)=/0c
Chip select + write enable
Rts
One
2X
Zero
2X
One
4X
Zero
(x700)=/00
no select + klocka låg
(x700)=/0c
(700)=/0d
Startbit
One
2X
Zero
One
Zero



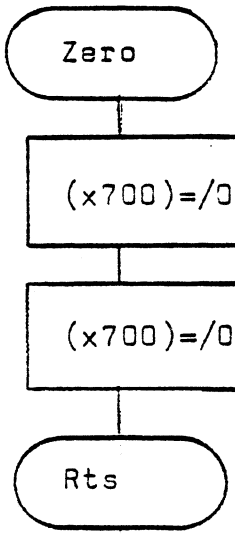
Spara undan innehållet

No enable

Vilket register

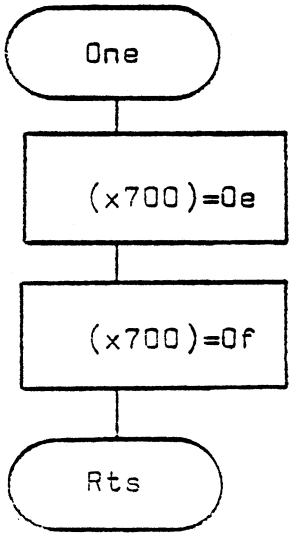
Läsa nvram

Lagrar innehåll



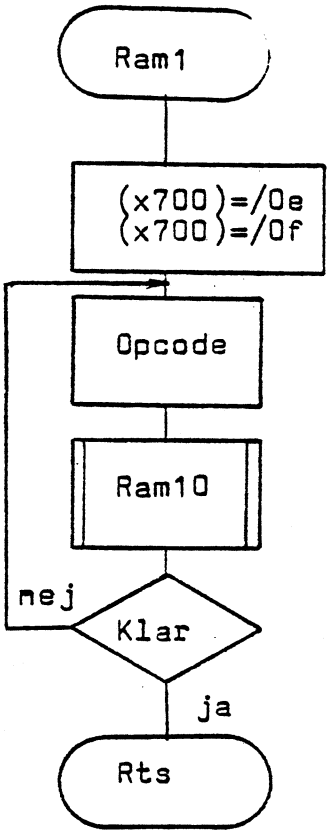
Laddar nolla

Klockar ut



Laddar etta

Klockar ut



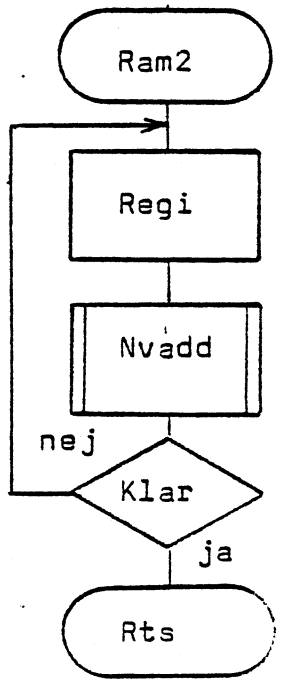
Skriver ut opcode

Etta som startbit

Vilken opcode

Skriver ut opcode

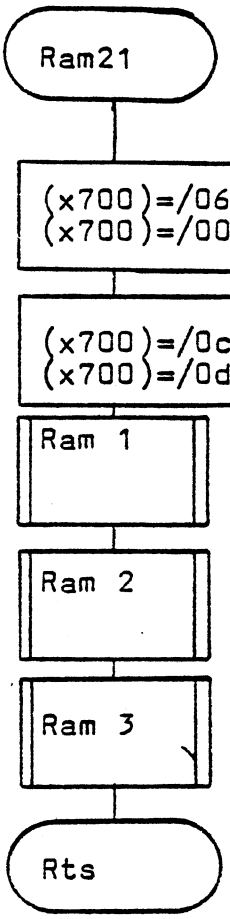
Om 4 bitar



Skriver ut vilket register

Vilket register

Klockar ut bit



Initierar nvram

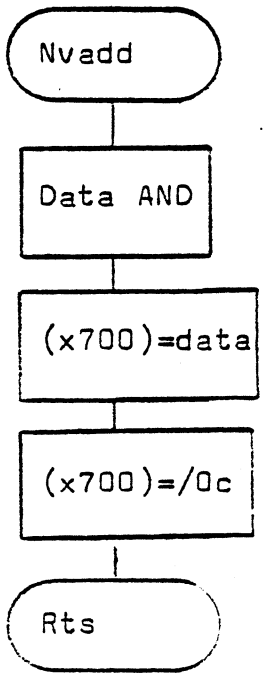
Alla utgångar

Enable nvram + klocka

Skriver in opcode

Vilket register

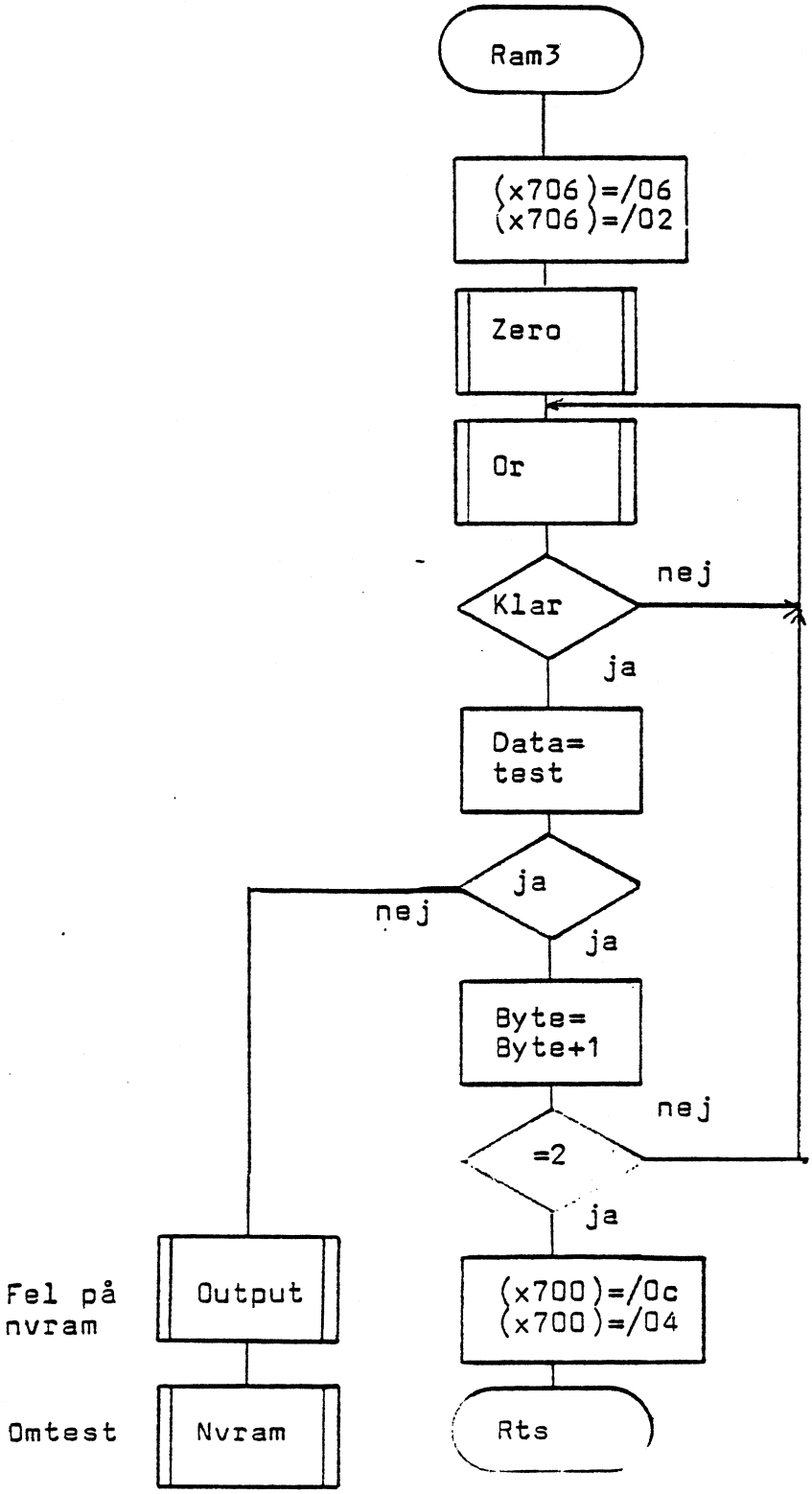
Hämtar dat och lagrar



Tar fram biten som skall
Sändas över

Klockar ut data

Klocka låg



Hämtar data och testar

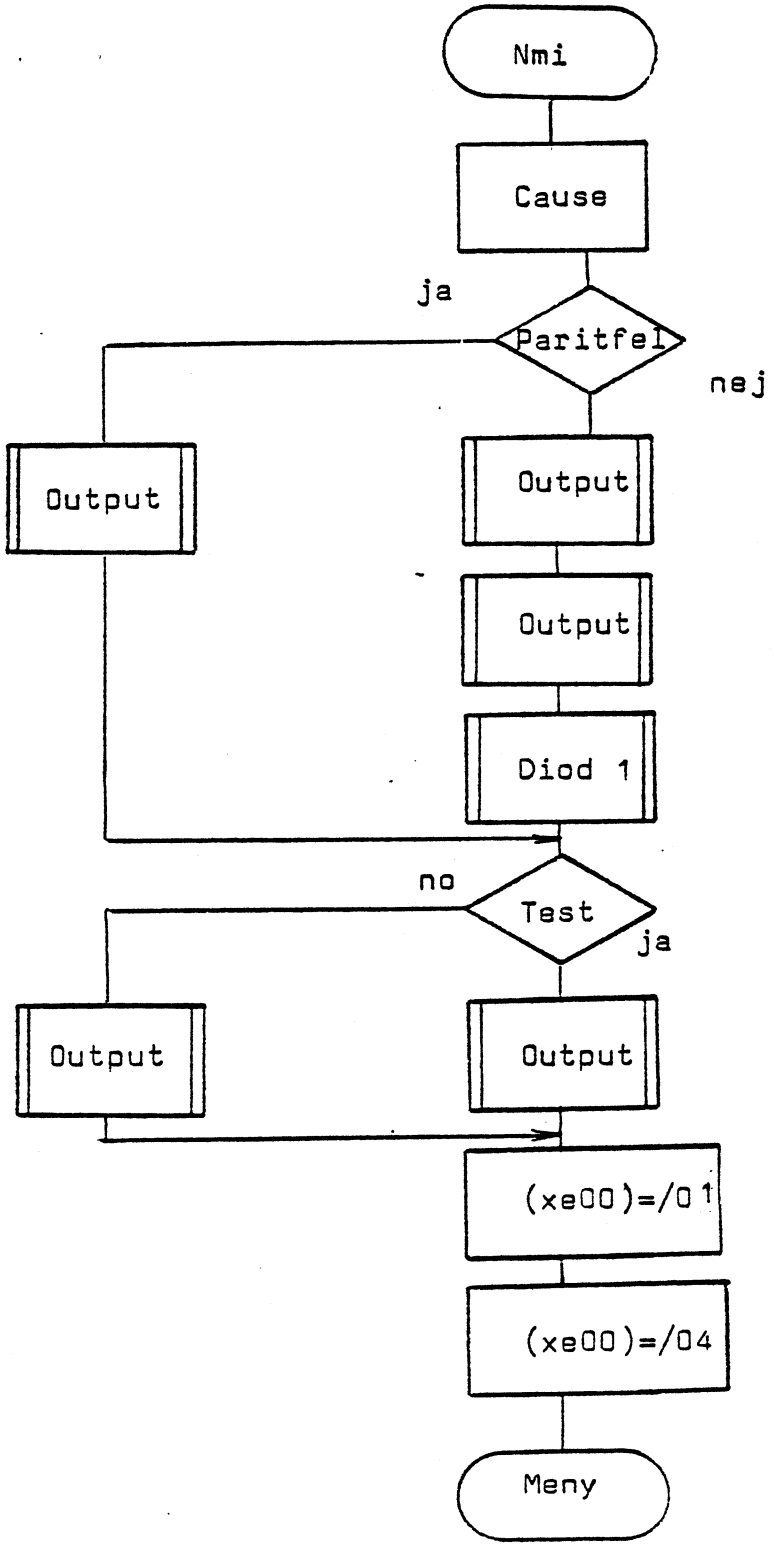
En ingång

Dummy bit

Hämtar data

Om byte okey

Klocka låg + no select



Hoppar hit vid Paritetsfel

Hämtar data från cause-registret

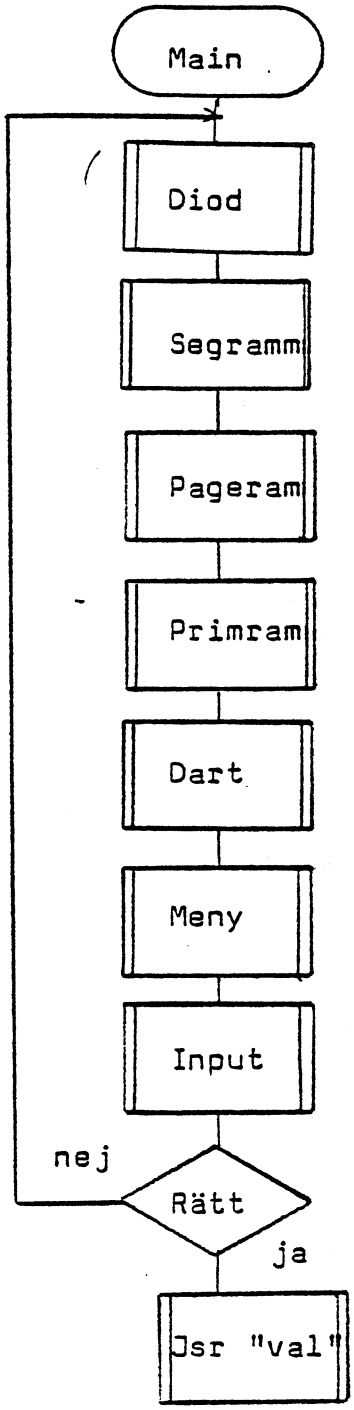
Något annat fel

Skriver ut cause register

Om paritetstest

Släcker paritetlampan

Noset PARTST "0"



Hoppar hit vid reset

Nollställer dioderna

Testar segmentram

Testar pageram

Testar 2K för stack

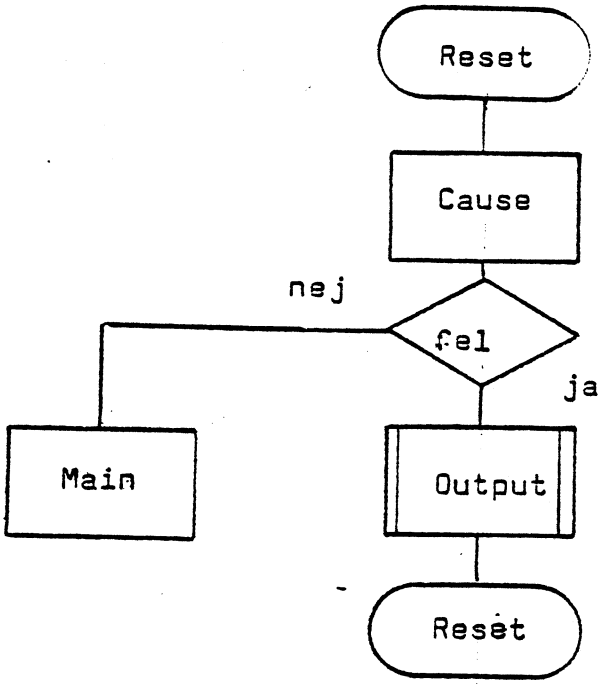
Testar darten för bild-
skärm och tangenbord

Skriver ut meny på skärm

Hämtar val

Om fel val

Hoppar till det utvalda

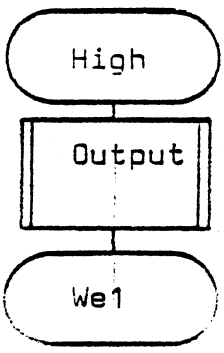


Hämtar causeregistret

Paritetsfel

Något fel på Cause regist

Gör omförsök



Om buss error

Det finns inte 1M minne i
eller annat fel

Skriver ut meny

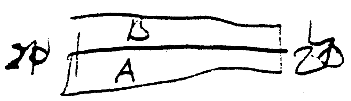
Kontakter mellan cpu-kort och video-kort (X35)

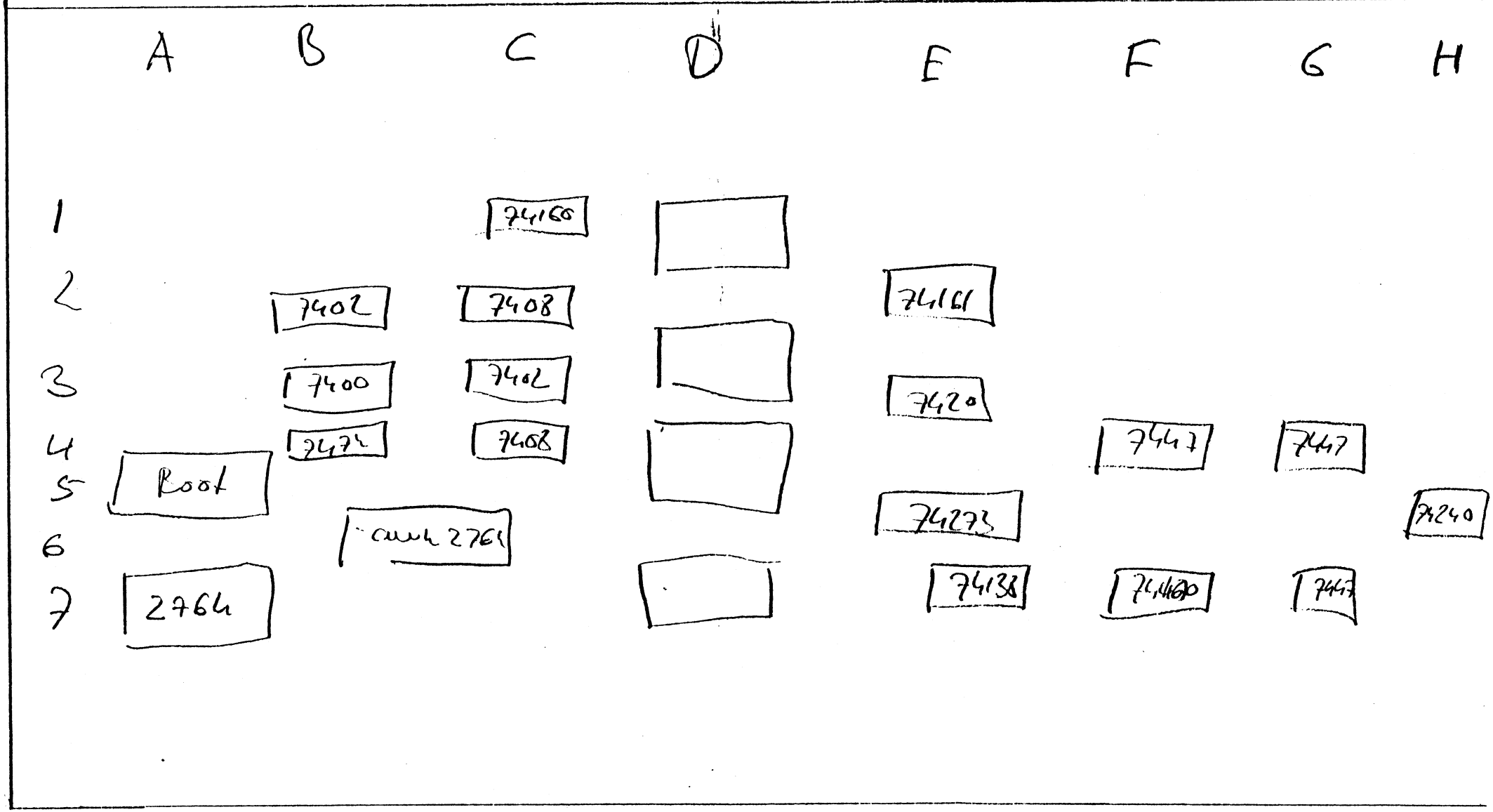
P10:

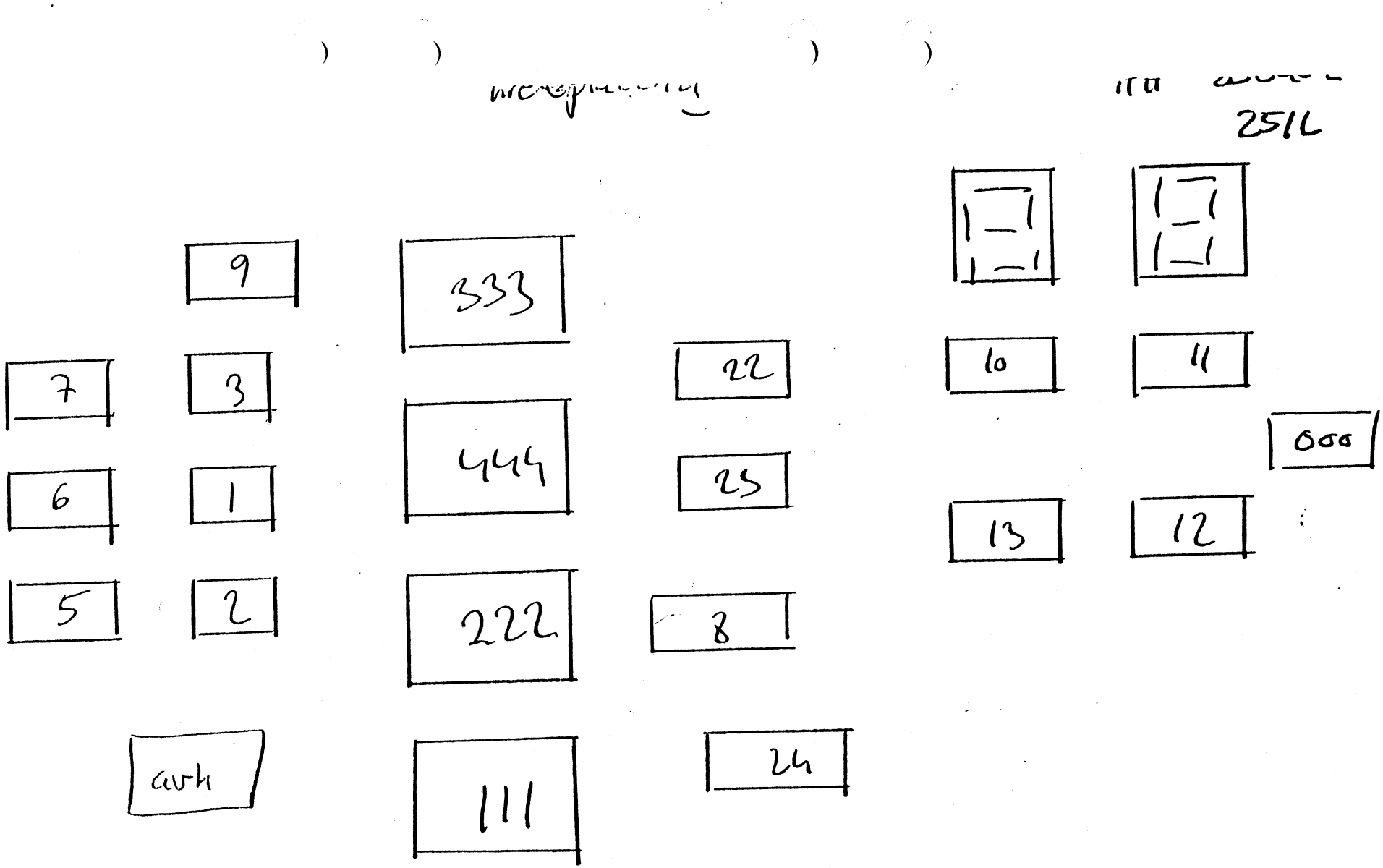
Signal A		A	Pin	B		Signal B
GND		o	1	o		GND
NODAS*	<---	o	2	o	--->	X15
X16	<---	o	3	o	--->	X17
X18	<---	o	4	o	--->	X14
X13	<---	o	5	o	--->	X12
X11	<---	o	6	o	--->	X19
X20	<---	o	7	o		GND
VSYNC	--->	o	8	o		GND
HSYNC	--->	o	9	o		GND
SCREENPOS	--->	o	10	o		GND
MINT	--->	o	11	o		GND
IOWR2*	<---	o	12	o		GND
IOWR1*	<---	o	13	o		GND
IOWR0*	<---	o	14	o		GND
2*CLK	--->	o	15	o		GND
IORD0*	<---	o	16	o		GND
IORQ*	<---	o	17	o		GND
W/R*	<---	o	18	o		GND
REFCLK*	--->	o	19	o	--->	FRST*
BDS*	<---	o	20	o		GND
		A		B		

P9:

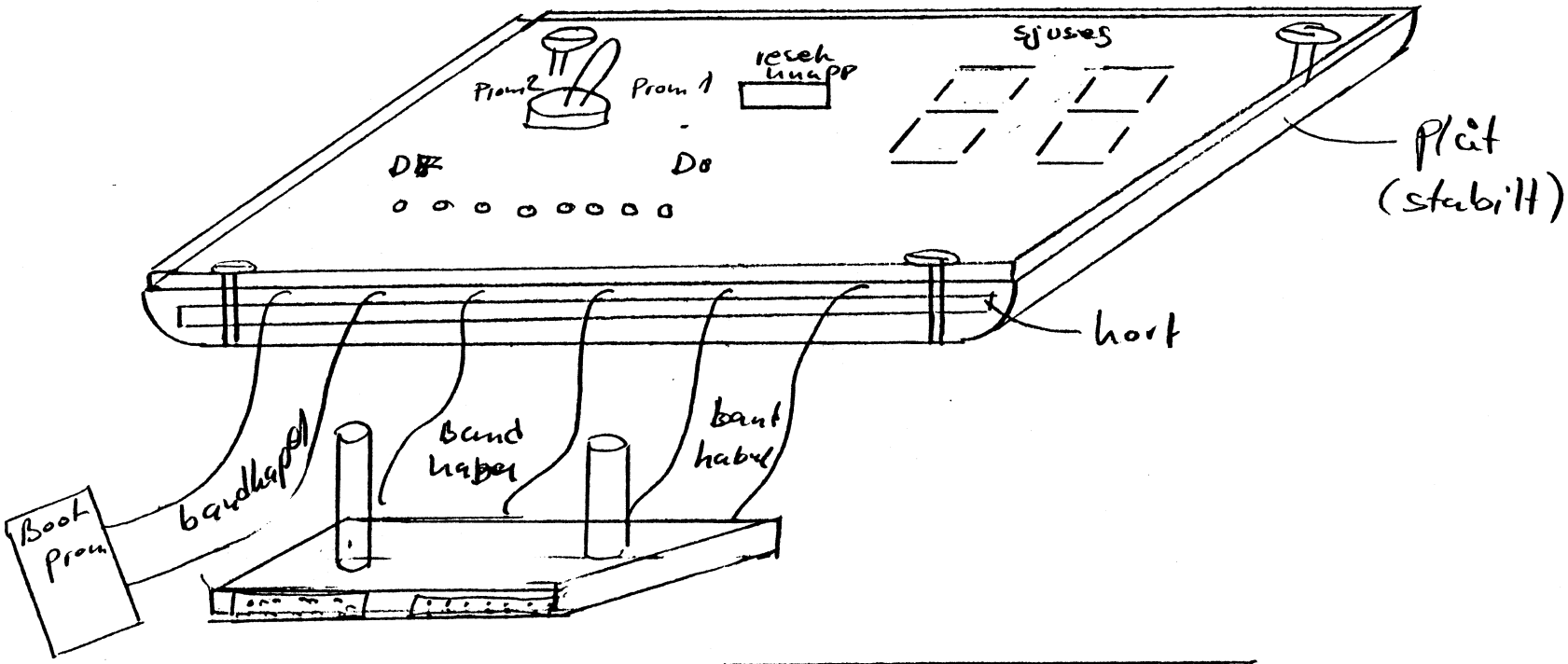
Signal A		A	Pin	B		Signal B
GND		o	1	o		GND
B8	<---	o	2	o	--->	B9
B10	<---	o	3	o	--->	B1
B3	<---	o	4	o	--->	B5
B6	<---	o	5	o	--->	B7
B4	<---	o	6	o	--->	B2
B0	<---	o	7	o		GND
DAS*	<---	o	8	o		GND
D0	<-->	o	9	o		GND
D2	<-->	o	10	o	<-->	D1
D4	<-->	o	11	o	<-->	D3
D6	<-->	o	12	o	<-->	D5
+5V		o	13	o	<-->	D7
PACK*	--->	o	14	o		GND
CRT*	<---	o	15	o		GND
RSTIN*		o	16	o		GND
GND		o	17	o		GND
GND		o	18	o		GND
GND		o	19	o		GND
GND		o	20	o		GND
		A		B		




A
B
C
D
E
F
G
H
1
2
3
4
5
6
7
74160
7402
7408
7400
7402
7404
Root
2764
74161
7420
7447
7447
74273
74138
74240


9
333
7
3
22
10
11
444
25
000
6
1
13
12
5
2
222
8
earth
111
24

HH 850214



tilläggg till föregående

Extra test prom: väljs med knapp (prom 1 eller 2) styr $\overline{OE}$ o $\overline{CS}$

2564 Eprom Direkt kort.