MÄT OCH STstyr med IBM PC och DATABOARD 4680

Utgåva 2 sept 1986

/Datum
Persondatorspecialisten

Copyright och ansvarsbegränsning PC4680

OBS!
Datum system förbehåller sig rätten att när som helst utföra förbättringar på produkten PC4680. Manualen kan innhålla tekniska felaktigheter eller typografiska fel. Sådana brister kan komma att vara rättade i senare upplagor eller versioner av manualen.

Datum System AB ställer inga garantier, vare sig direkta eller underförstådda, att produkten beskriven i denna manual motsvarar de specifika krav på kvalité, prestanda eller anpassning till speciellt syfte som användaren kan ha. Användaren står således själv risken för de direkta eller indirekta problem eller skador kan uppkomma.

Uppsala 1985-05-27

Datum System Uppsala AB

FELRAPPORT PC4680

Trots kontroll kan fel eller oklarheter finnas i manualen eller hårdvaran till PC4680. Om du råkar ut för sådana problem , ber vi Dig att fylla i den här felrapporten och skicka den till oss. Det är viktigt för oss att få kännedom om eventuella felaktigheter. Samtidigt får du chansen att erhålla en uppdaterad version av manualen.

Fyll i rapporten och skicka den till:

Datum System AB
Sågargatan 11
753 23 UPPSALA

Tack på förhand

Namn:_____

Företag:_____

Adress:_____

Postadress:_____

Telefon:_____

Lista över fel:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Beskrivning PC/4680

Innehåll

Kap 1 **Allmänt**

1.1 IBM PC/XT/AT
1.2 DOS 2.1/3.1
1.3 IBM BASIC och Basic Compiler
1.4 IBM 8088 Assembler, Interrupt
1.5 Skillnad mellan IBM PC BASIC och ABC 800 BASIC
1.6 BASIC II/PC

Kap 2 **PC4680**

2.1 Databoard 4680
2.2 Datums bussöversättare
2.3 Beskrivning av 4680-bussen
2.4 Expansionslådor till DataBoard 4680
2.5 Störningar i miljön
2.6 Översikt över DataBoard I/O-kort
- 2.6.1 Digitala I/O-kort
- 2.6.2 Analoga I/O-kort
- 2.6.3 Motorstyrkort
- 2.6.4 Kommunikationskort
- 2.6.5 Övriga kort

Kap 3 **Installation**

3.1 Datums bussöversättare
- 3.1.1 Installation av bussöversättare
- 3.1.2 Bussöversättarens switchar
- 3.1.3 Att ställa den batteriuppbackade klockan

3.2 DataBoard-korten
- 3.2.1 Montering av 4680-kort i expansionslåda
- 3.2.2 Val av kortadress
- 3.2.3 Programmering av 4680-korten
- 3.2.4 Anslutning av yttre enheter

Kap 4 **Hantering av data**

4.1 Olika talsystem
- 4.1.1 Decimala talsystemet

4.1.2 Binära talsystemet
4.1.3 Hexadecimala talsystemet
4.2 Hantering av olika talsystem i IBM BASIC
4.3 Logiska operationer

Kap 5 DataBoard för digitala signaler

5.1 DataBoard 4005 och 4006, TTL in/ut
5.1.1 DataBoard 4005, digitalt I/O-kort
5.1.1.1 Specifikationer
5.1.1.2 Beskrivning
5.1.1.3 Programmering
5.1.1.4 Installation

5.1.2 DataBoard 4006, Digitalt I/O-kort
5.1.2.1 Specifikationer
5.1.2.2 Beskrivning
5.1.2.3 Programmering
5.1.2.4 Installation
5.1.2.5 Programexempel

5.2 DataBoard 4008/11 och 4095
5.2.1 DataBoard 4008/11, Opto in
5.2.1.1 Specifikationer
5.2.1.2 Beskrivning
5.2.1.3 Programmering
5.2.1.4 Installation
5.2.1.5 Programexempel

5.2.2 DataBoard 4095, Opto ut
5.2.2.1 Specifikationer
5.2.2.2 Beskrivning
5.2.2.3 Programmering
5.2.2.4 Installation
5.2.2.5 Programexempel

5.3 DataBoard 4085, TTL in/ut
5.3.1 Specifikationer
5.3.2 Beskrivning
5.3.3 Programmering
5.3.4 Installation
5.3.5 Programexempel

5.4 DataBoard 4103, Relä
5.4.1 Specifikationer
5.4.2 Beskrivning
5.4.3 Programmering
5.4.4 Installation
5.4.5 Programexempel

5.5 DataBoard 4013, Mix I/O
5.5.1 Specifikationer
5.5.2 Beskrivning
5.5.3 Programmering
5.5.4 Installation
5.5.5 Programexempel

Kap 6 DataBoard för analoga signaler

6.1 DataBoard 4083, D/A omvandlare
6.1.1 Specifikationer
6.1.2 Beskrivning
6.1.3 Programmering
6.1.4 Installation
6.1.5 Programexempel

6.2 DataBoard 4084, D/A omvandlare
6.2.1 Specifikationer
6.2.2 Beskrivning
6.2.3 Programmering
6.2.4 Installation
6.2.5 Programexempel

6.3 DataBoard 4115, A/D omvandlare
6.3.1 Specifikationer
6.3.2 Beskrivning
6.3.3 Programmering
6.3.4 Installation
6.3.5 Programexempel

Kap 7 DataBoard för motorstyrning

7.1 DataBoard 4002, DC motor + positionering
7.1.1 Specifikationer
7.1.2 Beskrivning
7.1.3 Programmering
7.1.4 Installation
7.1.5 Programexempel

7.2 DataBoard stegmotorkort
7.2.1 DataBoard 4066, Stegmotorkontroller
7.2.1.1 Specifikationer
7.2.1.2 Beskrivning
7.2.1.3 Programmering
7.2.1.4 Installation
7.2.1.5 Användning

7.2.2. DataBoard 5085, drivsteg till 4066
7.2.2.1 Specifikationer
7.2.2.2 Beskrivning
7.2.2.3 Programmering
7.2.2.4 Installation

7.3 DataBoard 4014, DC motorstyrkort
7.3.1 Specifikationer
7.3.2 Beskrivning
7.3.3 Programmering
7.3.4 Installation
7.3.5 Programexempel

Kap 8 DataBoard kommunikationskort

8.1 DataBoard 4117 UART, V24 UART
8.1.1 Specifikationer
8.1.2 Beskrivning
8.1.3 Programmering
8.1.4 Installation

8.1.5 Programexempel

Kap 9 **DataBoard prototypkort**

9.1 DataBoard 5070
9.1.1 Specifikationer
9.1.2 Beskrivning

Appendix A Beskrivning av klocka/kalender 146818P

Appendix B Programmen på medföljande PC4680-disketten

Appendix C Hårdvaruspecifikation PC4680

Appendix D Interruptprogrammering

Appendix E Kända problem

Kapitel 1. Översikt

1.1 IBM PC-Familjen

IBM persondatorfamilj består för närvarande av 4 olika datorer, nämligen PCG, PCXT, PPC och PCAT.

PCG är standardmodellen med processorn INTEL 8088 och möjlighet till intern minnesutbyggnad till 640 KByte. Oftast med 2 st inbyggda diskettdriverar om vardera 360 Kbyte lagringskapacitet. På moderkortet finns IBM PC-bussen framdragen till 5 st "slots" för anslutning av olika typer av kretskort. Eftersom diskettdriverar, bildskärmskort och eventuellt kommunikationskort och minneskort kräver varsin kortplats blir det oftast bara en kortplats kvar för Datums bussöversättare till DataBoard 4680.

PCXT motsvarar IBM PCG men har en inbyggd winchester (hårddisk) på 10 Mbyte. Dessutom har PCXT ytterligare 3 kortplatser vilket kan underlätta utbyggnad av nya funktioner med extrakort.

PPC är IBMs bärbara persondator. Den har samma typ av moderkort som PCXT men ingen winchester. På grund av det kompakta formatet, är vissa av kortplatserna inte tillgängliga för utbygnadskort av fullängd. PPC kan vara praktisk att använda till DataBoard då det är önskvärt med ett litet format på datorn eller då man vill ha ett transportabelt mätsystem.

PCAT ÄR IBMs avancerade persondator med processorn INTEL 80286. Diskettdriverna är på 1.2 Mbyte och Winchesterminnet är på 20-40 Mbyte. Internminnet är 512 Kbyte på moderkortet och är utbyggbart till 3 MByte. OBS! PC-DOS hanterar endast 640 Kbyte. PCAT kan förses med operativsystemet Unix men alla exempel i den här bruksanvisningen förutsätter operativsystemet PCDOS. Assemblerprogram för processorn INTEL 80286 kan inte utan vidare köras på processorn 8088, vilken sitter i de tre andra PC-modellerna.

Alla modellerna kan förses med färggrafikkort och färgbildskärm. Samt asynkron och synkron kommunikation.

1.2 PC DOS

Operativsystemet PC DOS (MS-DOS) finns i olika utföranden och versioner. Operativsystemet är den programvara som användaren först kommer i kontakt med när man slår på strömmen till datorn. Med hjälp av operativsystemet kan man göra en applikation självstartande vid spänningspåslag. Med operativsystemet kan man kopiera disketter, program och datafiler. Viktiga funktioner i PC-DOS är:

DIR för att få reda på filnamnen på en diskett eller hårddisk. **FORMAT** för att göra iordning disketter. **COPY** för att kopiera filer. **EDLIN**, editeringsprogram för att göra iordning textfiler och systemfunktioner.

Vi rekommenderar att varje programmerare som skall skriva program till IBM PC, i första hand utnyttjar färdiga rutiner som finns tillgängliga i DOS. PCDOS finns för närvarande i tre versioner; ver 1, ver 2 och ver 3. För närmare beskrivning av alla möjligheterna i PCDOS hänvisas till IBMs beskrivning över operativsystemet i handböcker och tekniska referenser.

1.3 IBM BASIC och BASIC Compiler

Det finns 3 sorters IBM BASIC. Cassette-BASIC, Disc-BASIC och Advanced-BASIC. I exemplen i den här boken förutsätter vi att användaren har tillgång till Advanced-BASIC. IBM BASIC är en variant av Microsoft BASIC. En av funktionerna i Advanced-BASIC som saknas i de andra versionerna, är möjligheterna till avancerad grafik som kan utnyttjas tillsammans med IBM Color/Graphics Adapter. Vissa programexempel i den här manualen förutsätter tillgång till Color/Graphics Adapter.

Advanced-BASIC finns med på systemskivan till Operativsystemet. För att starta Advanced-BASIC skriver man helt enkelt BASICA när systemskivan är monterad i aktuell diskdrive. För närmare beskrivning hänvisar vi till IBMs manual över Microsoft BASIC.

DE funktioner I BASIC som är av speciellt intresse för DataBoard, är möjligheten att skriva och läsa data på in/ut-portarna på DataBoard-korten. Här kommer en sammanfattning av de kommandon och funktioner som är av vikt för hantering av data i samband med DataBoard.

Variabler
Eftersom den information som läses in och ut från I/O-korten består av max 8 bitar och endast kan anta värden mellan 0 och 255, är det lämpligast att variabler som innehåller data som skall till eller från I/O-korten är deklarerade som heltalsvariabler.

En heltalsvariabel kan endast innehålla heltal mellan -32786 och +32767 och deklareras genom att sätta ett %-tecken efter variabelnamnet.

Exempel: A% är namnet på en variabel som endast kan innehålla heltal.

Konstanter
Förutom decimalt kan konstanter behöva anges som hexadecimala alternativt binära tal.

Hexadecimala talsystemet
För mer avancerade programmerare kan det underlätta att ange vissa tal i det Hexadecimala talsystemet. I det Hexedecimala talsystemet finns det 16 siffror mellan 0 och 15. De 6 siffrorna ovanför 9 betecknas då med bokstäver: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. A=10 och F=15. En konstant som angivits i det hexadecimala talsystemet markeras med att ett **&H** sätts framför värdet.

Exempel: Talet 31 skrivs med det hexadecimala talet som &H1F. Ettan står i sextontals- positionen och är värd 16. F står i entals-positionen och är värd 15 (16+15=31).

Binära talsystemet
Varje signal som läses in från en port på I/O-korten, identifieras med hjälp av det binära talsystemet. Det är därför av största vikt att den som önskar skriva egna program för hantering av DataBoard-korten, sätter sig in i hur det binära talsystemet fungerar. Trots att IBM BASIC inte har speciella färdiga funktioner för att hantera det binära talsystemet har man ändå användning av en viss förståelse.

Se kapitel 4.1 för närmare förklaring.

I/O-funktioner
De viktigaste in- respektive utkommandon för att kommunicera med DataBoard kort är INP respektive OUT.

INP
Läser in 8 databitar från en I/O-port. Exempel:

V%=INP(&H303). Värdet av de 8 bitarna på port 303 (Hexadecimalt), läses in i heltalsvariabeln V%.

OUT
Sänder ut en byte (8 databitar) till en I/O-port. Exempel:

OUT P%,V%. Värdet i V% (0-255) sänds iväg till porten P%

För närmare beskrivning av de interna port-adresser i IBM-PC, hänvisar vi till IBM Personal Computer Technical Reference Manual. De portar som är aktuella i samband med DataBoard 4680 finns beskrivna i den här manualen.

Logiska operatorer

För att testa värdet av specifika bitar i talet som lästs av från en I/O-port, används ofta logiska operatorer. De vanligaste (och viktigaste) logiska operatorerna är : AND, OR samt XOR. Se närmare förklaring i kapitel 4.3.

IBM BASIC compiler
Med hjälp av IBM BASIC Compiler, kan man snabba upp ett program som skrivits och uttestats i IBM (Microsoft)-BASIC. Observera att vissa skillnader finns mellan ordinarie IBM-BASIC och BASIC Compiler. Skillnaderna framgår av bruksanvisningen till BASIC Compiler som följer med kompilatorn.

1.4 IBM 8088 Assembler

För den som vill utveckla program för mycket snabba förlopp, finns möjlighet att använda IBM 8088 ASSEMBLER. Med hjälp av assemblern kan man få fram program i maskinkod vilken utförs mycket snabbt av processorn. D v s i storleksordningen 1 miljon instruktioner i sekunden. Maskinkod kan t ex användas för att mäta längden på mycket korta pulser (eller antalet korta pulser om dom är fler). För att skriva program för assemblern måste man i viss mån sätta sig in i hur själva microprocessorn fungerar. Beskrivning av hur processorn fungerar finns beskrivet i INTELs handbok: IAPX86/88 user's manual and programmers referens.

Interrupt
Fördelen med att arbeta med interrupt (avbrottshantering) är att datorn kan bearbeta information i t ex BASIC, samtidigt som data samlas in i bakgrunden. För att göra detta måste det finnas en rutin skriven i assembler som tar hand om interrupt när det inträffar. PC 4680 kan generera 2 olika interrupt. Ett från klockan, och ett från DataBoardbussen. För att få reda på vilka interrupt som genererats via PC4680 läser man in en byte via port &H305 där varje interrupt har varsin bit.
En bit som är satt till "0" indikerar interrupt. Bit 0 = interrupt från klockan, bit 7 är RESIN och bit 6 är interrupt från DataBoardkorten. I IBM PC måste det finnas ett program (interruptrutin) inlagd för att man ska kunna ta hand om interruptet. Denna rutin skall söka av vilken enhet som gav interruptet och sen utföra olika saker beroende på vilken enhet som begärt interrupt.

Om ett interrupt har detekterats från klockan, skall register C i klockan läsas för att se vilken funktion som gav interrupt. Avläsning sker med hjälp av de stödrutiner som följer med disketten som hör till PC 4680-kortet. För ytterligare information se Appendix A. Om det var något DataBoard-kort som gav interrupt, så får man programmässigt söka igenom DataBoard-korten för att se vilket kort som gav interrupt.

Mer information om interrupt finns i DOS technical referens manual och IBM PC/XT technical referens manual.

1.5 Olikheter mellan PC BASIC och ABC 800 BASIC

Många tidigare tillämpningar för DataBoard-kort är skrivna för ABC 800 BASIC. Det kan därför vara av intresse vad som skiljer de olika BASIC-dialekterna från

varandra vad gäller hantering av DataBoard-korten.Här följer en sammanfattning av några olikheter.

En stor skillnad mellan de två dialekterna är hanteringen av funktioner. PC'n kan ej hantera flerradiga funktioner. Då får man istället använda sig av subrutiner och globala variabler.
De enradiga funktionerna måste i PC'n läggas först i programmet eftersom basictolken där börjar direkt att exekvera den första raden. Gör man inte det kommer man att få felmeddelandet funktionen ej definierad. I ABC800 går datorn först igenom programmet och tittar efter om det bla finns några funktioner.

Heltalshanteringen är väldigt strikt hos PC'n, man kan tex inte skriva `A%=B` om B som är ett flyttal innehåller ett högre värde än 32767. Man måste då räkna om talet enligt tvåkomplementsmetoden. Detta kan man lösa med följande funktion:

```
DEF FNHELTAL%(TAL)=TAL-65536
```

Innan man gör omvandlingen testar man om flyttalet är större än 32767, om så är fallet anropas funktionen annars är det ok.

```
IF B>32767 THEN A%=FNHELTAL%(B) ELSE A%=B
```

Likadant är det om man försöker göra bitorienterade logiska operationer (tex AND OR) på heltal. Är talet högre än 32767 fås felmeddelande.

Fullskärmseditorn i PC'n är mycket bättre än radeditorn i ABC800. Man tröttnar ganska snabbt på att skriva ED framför varje rad som skall ändras. Dessutom har PC'n ett utmärkt kommando som heter FILES, det visar innehållet på disketten.

Övriga skillnader är att funtionerna/instruktionerna heter lite olika ibland, även olika argument förekommer.
T ex att omvandla ett tal till en sträng heter hos ABC800 NUM$(A) och hos PC'n STR$(A).
Att plocka ut den högra delen från en sträng heter hos bägge RIGHT$(A$,A), men i ABC800 skall man räkna positionen från vänster och i PC'n från höger i strängen

Man måste även tänka på att det går inte att använda Å Ä eller Ö i variabel- och filnamn i PC-basicen.

1.6 BASIC II/PC

Denna BASICtolk är framtagen av DIAB för MS-DOS maskiner. Den är kompatibel med ABC800-basicen plus att den har ett antal nya instruktioner. Den instruktion som är mest intressant för användningen av PC4680 är OPTION IOBASE xx. OPTION IOBASE sätter grundadressen för OUT- och INP-kommandona vilket gör att man kan använda samma portnummer till PC4680 som i ABC800 miljö. Som exempel kan tas om PC4680 har grundadress &H300 eller 768 decimalt. Då ges instruktionen OPTION IOBASE 768 och sedan kan tex ett OUT 1,5 ges för att välja kort nr 5.

Kapitel 2. PC4680

2.1 DataBoard 4680

DataBoard 4680 är en generell buss för uppbyggnad av datorsystem. Bussen har 3 sorters anslutningar. Dels anslutningen för datorenheten dels för anslutning av minneskort och slutligen I/O-kort. Bussen är standardiserad och kortmodulerna konstrueras med i stort samma snitt mot bussen.

Korten ansluts parallellt till minnessidan respektive I/O-sidan. Observera att endast I/O-kort kan anslutas via PC4680. Minnessidan och minneskort kan inte alls användas tillsammans med IBM PC. De signaler som anslutes till IBM PC är endast de åtta dataledningarna D0-D7, portadressering och kontrolledningar för Card Select, strömmatning m m.

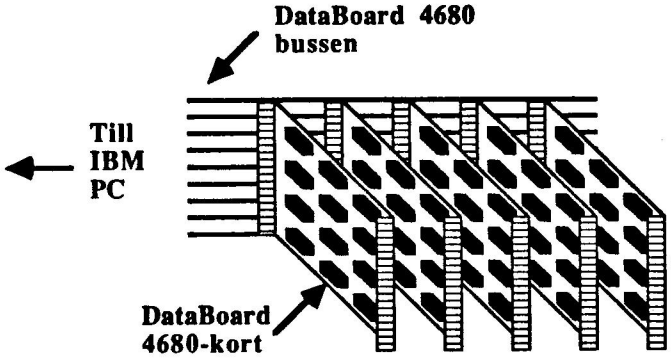


Fig 2.1 Schematisk beskrivning av DataBoard 4680-bussen

2.2 Datums Bussöversättare

PC 4680 består huvudsakligen av tre komponenter; 1: Den här manualen, 2: kabeln mellan IBM PC och DataBoard expansionslåda och slutligen 3: för att kunna ansluta IBM PC till DataBoard 4680, har Datum System AB utvecklat ett speciellt kretskort att montera i IBM PC (sk bussöversättare = konverter). Kortet innehåller förutom de speciella funktionerna för anslutning till DataBoard även en batteriuppbackad realtidsklocka och en hållare för ROM-kapsel. Den senare kan innehålla speciella drivrutiner i assembler som användaren vill använda i vissa tillämpningar. Rutinen läses då först in i minnet på IBM PC via en port på busskonvertern.

Det är inte självklart att alla I/O-kort i 4680-serien fungerar utan vidare till IBM PC De kort som finns beskrivna i den här manualen har kontrollerats och inga

funktionsproblem av vikt har upptäckts till dags dato. Om kort som skall utnyttja IORDY-signalen på DataBoard-bussen skall användas, rekommenderar vi att Du kontaktar din leverantör av PC4680-korten.

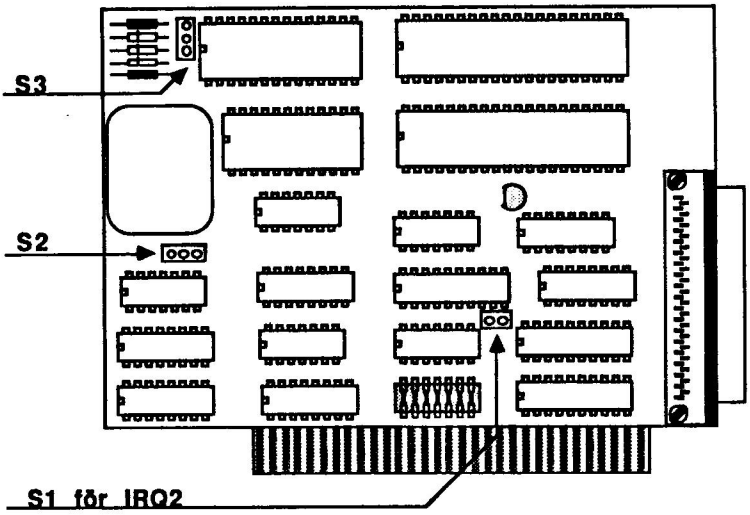


Fig 2.2 Datums bussöversättare

2.3 Beskrivning av DataBoard 4680-bussen

Signaleringen är asynkron medelst särskilda kontrollsignaler för respektive sidor. Databussen är dubbelriktad och kräver tri-state typ av drivsteg mot bussen. Alla I/O-kort anslutes parallellt mot bussen men endast ett kort är aktivt åt gången. Vilket kort som vid varje tillfälle är aktivt, bestämms av datorn genom instruktioner för kortval.

Fysiskt består 4680-bussen av dels en expansionslåda med strömförsörjning dels ett backplan med 64-poliga anslutningskontakter för anslutning av I/O-korten.

2.4 DataBoard expansionslåda

Den vanligaste typen av expansionslådor till DataBoard består av 3 platser för I/O-kort och 4 platser för minneskort. Det går inte att sätta ett I/O-kort på en plats avsedd för minneskort. Eftersom IBM PC inte kan utnyttja vanliga DataBoard minneskort är de fyra kortplatserna oanvändbara för den som använder en IBM PC.

Mellan de två olika typerna av kortplatser finns anslutningen av kabeln från IBM PC-bussen. Eftersom bussanslutningen innehåller samtliga signaler till DataBoard 4680, är det viktigt att inte ansluta kabeln till någon av kortplatserna.
Det finns även en variant av expansionslåda som innehåller platser för enbart anslutning av I/O-kort. Eftersom IBM endast kan utnyttja DataBoard I/O-kort är den senare typen av låda att rekommendera till IBM PC. En viktig fördel är då att man inte av misstag kan ansluta korten på fel plats.

I expansionslådan finns även ett kraftaggregat för strömförsörjning av de olika korten. Spänningen till expansionslådan bör vara avstängd när korten anslutes, men påslagen före uppstart av IBM PC.

För den som redan har en DataBoard expansionslåda för exempelvis LUXOR-datorer och vill byta till en IBM-dator, skall det normalt bara vara att köra direkt med hjälp av Datums bussöversättare och busskabel.

2.5 Störningar i miljön

För anslutning mellan IBM PC och DataBoard 4680 finns en speciell kabel framtagen. För den som önskar göra sin egen kabel, kan påpekas att längden normalt inte bör vara mer än 3 meter.

Det är viktigt att ansluta korten på rätt sätt till yttre enheter. Speciellt med tanke på att störningar kan uppkomma på vägen. Närhet till större strömförbrukare av typen elektriska motorer och kablage till dessa kan förorsaka störningar och felaktiga signaler på mätsidan. Skärmade eller kortare tilledare kan ibland vara lösningen. Vid mätning av analoga signaler kan även strömmar i jordledare förorsaka problem.

OBS !
Var försiktig vid anslutning av enheter som är galvaniskt anslutna till elnätet. Spartransformatorer kan exempelvis kortslutas från fas till jord via IBM PC's nätanslutning.

2.6 Översikt över DataBoard I/O-kort

2.6.1 Digitala I/O-kort

Digitala signaler i varierande nivåer förekommer nästan alltid i styr- och mätsystem. Galvaniskt isolerade signaler är att föredra vid industriella applikationer.

Exempel på användning av digitala kort är:

- Tryckknappar
- Gränslägesbrytare
- Magnetventiler
- Kontaktorer
- BCD-signaler till och från instrument

2.6.2 Analoga I/O-kort

Analoga signaler är mycket vanligt i datoriserade mät- och reglersystem. Analoga signaler används ofta för att mäta/styra:
- Tryck
- Vikt
- Hastighet
- Temperatur

2.6.3 Motorstyrkort

Med dessa kort kan olika typer av motorer styras, exempelvis för att utföra positionering av transportvagnar, traverser etc.

2.6.4 Kommunikationskort

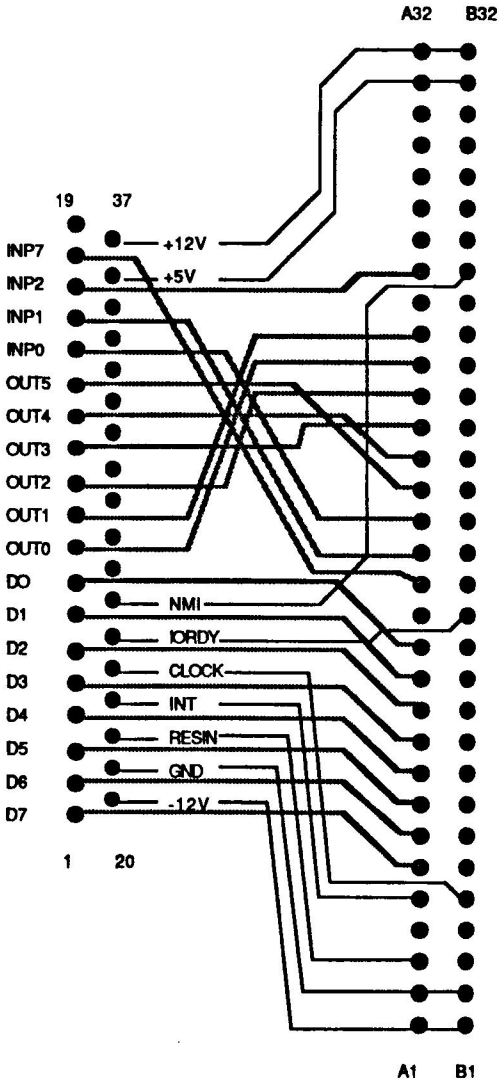
Detta är samlingsrubriken för olika standardiserade gränssnitt. Inom DataBoard 4680 finns ett flertal olika gränssnitt tillgängliga. Till IBM PC är följande av speciellt värde.

- V24/RS232C
- GPIB/IEC buss

2.6.5 Övriga kort

Utöver de tidigare angivna grupperna, finns ett flertal andra kort med specella funktioner. Till IBM PC kan bara garanteras funktionen hos de I/O-kort som speciellt redogörs för i den här manualen. Med egen programvara, och förståelse för kortens funktion, kan det finnas flera kort utöver ovanstående som kan anslutas till IBM PC.

Konfigurering av KA4680 - kabel me llan PC4680 och DataBoard



Kapitel 3. Installation

3.1 Datums bussöversättare

3.1.1 Bussöversättarens switchinställningar

För att undvika adresskollisioner med andra kort till IBM PC kan kortets grundadress ställas in med kortets switchar. Till de portadresser som gäller för kortets egna funktioner skall adderas kortets grundadress. Det är även möjligt at koppla in respektive ur interruptfunktionen och I/O ready funktionen från DataBoard-korten.

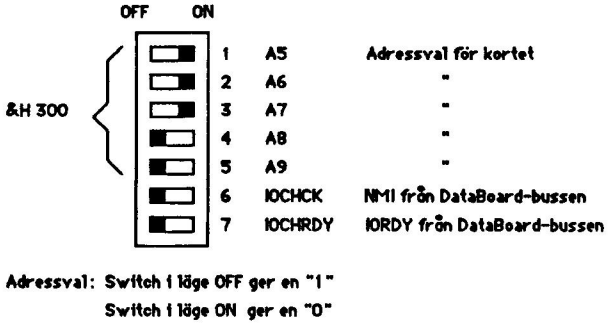


Fig 3.1 Bussöversättarens switchar. Bitvärden (Hexadecimalt): A5=&H20, A6=&H40, A7=&H80, A8=&H100, A9=&H200. Bilden visar läget &H300 inställt. Switch 6 och 7, läge off = funktionen urkopplad.

Bussöversättaren upptar 16 I/O-adresser med start från den inställda grundadressen och uppåt. De 8 första (&H300 till &H308 i programexemplen) är avkodade till DataBoard 4680-korten. De 8 sista adresserna (&H309 till &H30F) är avkodade till den elektronik som styr klockan och promhållaren. OBS! Försök inte att läsa eller skriva klockan utan att använda de medföljande drivrutinerna.

Byglingar
Om S1 är byglad kan kortet ge interrupt till IBM PC via IRQ2-signalen. Om S2 är i läge mot DataBoard-kontakten är klockfrekvensen ut till DataBoard-bussen = 2.35 MHz. I motsatt läge är frekvensen 4.7 MHz.

3.1.2 Installation av bussöversättaren

Bussöversättaren skall monteras inuti IBM PC centralenhet. Kontrollera först att datorns nätkabel inte är ansluten. Lossa skruvarna på baksidan och skjut höljet framåt. Längst bak på bottenkortet i centralenheten, finns kontaktdon (kortplatser) för anslutning till IBM PC interna CPU-buss. Bakom varje kortplats finns en plåtvinkel som täcker en öppning mot centralenhetens baksida.

Kontrollera att bussöversättaren har rätt grundadress. Grundadressen ställs in med switcharna på kortet enligt fig 3.1. Alla exempel i den här manualen förutsätter att grundadressen är &H300 (dvs 768 decimalt).

Välj ut en ledig kortplats och skruva loss plåtvinkeln bakom. Det är en fördel att placera ut olika typer av kort inuti, så att mesta möjliga tomrum finns mellan dem. Sätt korten så glest som möjligt och förslagsvis vartannat kort av den långa typen och vartannat av den korta typen. På så sätt undviks onödig överhettning om det finns många kort monterade i din PC.

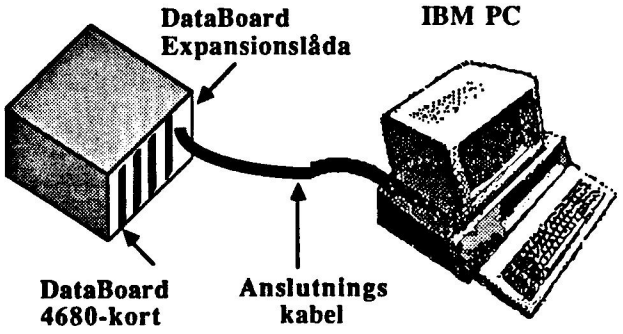


Fig 3.2 PC 4680-systemet

Tryck försiktigt ner kortet i hållaren och skruva fast bussöversättarens plåtvinkel.

Det är viktigt att kortet sitter fast ordentligt för att undvika att det kommer snett senare. Om kortet kommer snett finns det risk för kortslutning vilket kan allvarligt skada din PC.

Om allt ser ut att vara OK, skjut försiktigt tillbaka höljet. Kontrollera att inga kablar kommer i kläm. Skruva i de bakre fästskruvarna och montera bildskärm och tangentbord.

3.1.3 Att ställa den inbyggda batteri-uppbackade klockan

Den inbyggda klockan kan ställas med programmet DBTIME som finns med på disketten som medföljer kortet. När klockan väl är ställd skall den normalt inte behöva ställas nästa gång systemet startas. Vid nystart skall istället programmet PCTIME exekveras för att initiera den klocka som ingår som en del i IBM PC operativsystem. Den senare klockan nollställs dock var gång IBM PC varit avstängd. Samtidigt som man ställer klockan, får man en generell funktiontest av bussöversättaren.

3.2 DataBoardkorten

3.2.1 Montering av Databoard 4680-kort i expansionslåda

Det finns två sorters DataBoard expansionslådor. Den vanligaste typen är den med delad minnes- och I/O-sida. Datboardkort till IBM PC skall alltid monteras på I/O sidan. På varje I/O-kort finns en lysdiod som tänds när kortet är selekterat. Vänd kortet så att lysdioden vänds utåt och bort från expansionslådans bakplan. Passa in kortet i styrskenorna och för kortet försiktigt inåt. När kortet verkar ha fått anslutning till kontaktdonet på bakplanet, skall kortet tryckas fast ordentligt. Om kortet endast är inskjutet till hälften, finns det risk för glappkontakt och därmed förlorad tillförlitlighet. Var alltid försiktig vid monteringen. Om något stift i anslutningen kommit snett i motsvarande hylsa, kan det hända att kontakten på 4680-kortet blir helt förstörd om man brukar för stort våld vid monteringen.

3.2.2 Val av kortadress

För att kunna ansluta fler än ett I/O-kort till på IBM PC via 4680-bussen måste varje kort göras unikt på något sätt. Om man vill ansluta två exakt lika kort måste de se olika ut ur bussens synvinkel. Det sker genom att varje kort tilldelas en adress. Databoard-korten är utformade så, att de kan erhålla 63 olika adresser. DVS i teorin skulle det vara möjligt att ansluta 63 olika kort.

Valet av kortadress är mycket enkel att genomföra. På varje kort finns en hållare för bygling av kortadressen. När kortet levereras är det vanligt att hållaren är försedd med en färdig byglingsplugg med 7 st byglar. Av dessa 7 byglar är det bara de 6 första som räknas. För att välja adress skall man bryta upp en eller flera av byglingarna. När alla byglingar är opåverkade har kortet adressen 0. Om alla byglingar är brutna får kortet adressen 63. Värdet för alla andra kombinationer av byglingar får man genom att lägga ihop värdet för varje bygel enligt följande exempel.

Varje bygel har ett värde enligt det binära talsystemet. Genom att lägga ihop värdet för varje bruten bygel får man en totalsumma som då är kortet adress.

BYGLING AV I/O-KORTETS KORTADRESS

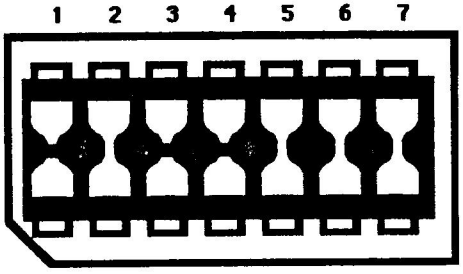


Fig 3.3 Bygel för att bestämma DataBoard-kortets adress

När instruktionen OUT &H301, A (där A står för kortadressen) utförs av datorn, aktiveras signalen CS (Card Select) på 4680 bussen. Det kort då som har en adress som motsvarar A, kommer då att bli utvalt samtidigt som alla andra kort förblir passiva. En lysdiod på det utvalda kortet tänds för att markera att kortet är utvalt. Det första som man bör göra när ett system kopplats upp, är att kontrollera att samtliga kort kan adresseras ett och ett.

Vid tillslag av strömförsörjning, händer det att ett eller flera kort slumpvis blir utvalda av sig själva. För att nollställa alla kortet på bussen låter man då (eller alltid för säkerhets skull), datorn utföra instruktionen INP(&H307) först av allt i alla program.
Efter Card Select blir in och utportar på det utvalda kortet tillgängliga för läsning eller skrivning. För att läsa data från en inport på kortet sker det genom instruktionen INP. Att skriva på porten sker genom instruktionen OUT. (Se mer om programmering av varje enskilt DataBoard-kort).

3.2.3 Programmering av 4680-korten

Till varje enskilt kort i DataBoard4680-serien finns det med en originalmanual från tillverkaren. Till de portadresser på kortet som anges där skall man addera

ett värde som läggs till av PC4680-kortet. Tilläggsvärdet är vid leverans 768 decimalt (&H300 hexadecimalt). Tänk därför på detta vid programmering av kort som inte finns med i den här bruksanvisningen. Genom att ställa om byglar på PC4680 kan man ställa om basadressen (tilläggsvärdet) på bussöversättaren ifall kortet skulle komma att kollidera med portadresserna på andra specialkort. Så vitt är känt upp till dags dato finns dock inget av IBM's originalkort med kolliderande portadresser.
Av speciellt intresse är nedanstående översättningstabell mellan kommandon i IBM PC BASIC och de hårdvarustrobar som genereras.

	signal	kommando
OUT-strobar	DATA	OUT &H300, X
	CS	OUT &H301, X
	C1	OUT &H302, X
	C2	OUT &H303, X
	C3	OUT &H304, X
	C4	OUT &H305, X
INP-strobar	DATA	X=INP (&H300)
	STAT	X=INP (&H301)
	OPS	X=INP (&H302)

Fig 3.4 Översättningstabell BASIC-kommandon och hårdvarustrobar

Vissa rutiner i programmen förekommer ofta i olika sammanhang. Det kan därför vara bra att ha tillgång till vissa fördefinierade generella subrutiner att lägga in i sitt BASIC program. Nedan följer exempel på sådana rutiner som kan vara användbara i olika sammanhang.

```
100   ,
110   , FUNKTION FÖR KONVERTERING AV FLYTTAL TILL HELTAL
120   ,
130   , In   Tal ! = Ett tal som skall omvandlas, OBS
      32768< tal !<65536
140   , Ut   Ett tal i intervallet -32768 - 0
150   ,
160   DEF FNHELTAL% (TAL) =TAL-65536
500   ,
510   , SUBRUTIN FÖR NOLLSTÄLLNING AV EN PORT
520   ,
530   , In   Port = Vilken port,   Sist% = Vad som sist
      skickades
540   , Ut   Ut% = Vad som skall skickas nu
550   ,
560   ,TAL=65535!-2^PORT::IF TAL>#"/&/THEN
      MASK%=FNHELTAL%(TAL) ELSEMASK%=TAL
```

```
570   UT%=SIST% AND MASK%
580   RETURN
600   ,
610   , SUBRUTIN FÖR ETTSTÄLLNING AV EN PORT
620   ,
630   , In   Port = Vilken port,  Sist% = Vad som
      skickades sist
640   , Ut   Ut% 0 Vad som skall skickas nu
650   ,
660   TAL=2^PORT  :: IF TAL>32767 THEN
      MASK%=FNHELTAL%(TAL) ELSE MASK%=TAL
670   UT%=SIST% OR MASK%
680   RETURN
700   ,
710   ,   SUBRUTIN FÖR INLÄSNING AV EN PORT
720   ,
730   ,   In   Port% kan vara 0-7
740   ,   Ut  Status% Läst värde antingen 1  (sant)
      eller  0  (falskt)
750   ,
760   MASK=^PORT%
770   IF MASK>#"/&/ THEN MASK%=FNHELTAL (MASK) ELSE
      MASK%=MASK' EV KONVERTERA
780   STATUS%=(INP (&H300)  AND MASK%) /MASK%
790   RETURN
```

3.2.4 Anslutning av yttre enheter

För att ansluta yttre enheter på I/O-kortet, måste alla aktuella stift på yttre I/O-kontakten anslutas enligt önskad funktion. Yttre I/O-kontakten sitter närmast lysdioden som markerar utvalt kort. Se upp så att kortet inte monteras baklänges i expansionslådan. Bäst är att börja med att tillverka en kabel med önskat antal ledar mellan DataBoard-kort och den enhet kortet skall styra eller läsa av. Det är viktigt att kabeln får rätt utformning med tanke på störningar som kan uppkomma i vissa miljöer. Speciellt vid användning av analoga I/O-kort är det viktigt med en väl genomtänkt uppkoppling. Genom att använda en kabelkontakt som passar till den "plastnyckel" som finns på I/O-sidan på många DataBoardkort, är det möjligt att dessutom förhindra förväxling av kabel vid anslutning.

Kapitel 4. Hantering av data

4.1 Olika talsystem

Det talsystem som vi använder oss av dagligen, kallas för det decimala talsystemet. Siffrorna som vi använder är 0 till 9. Ordet decimala kommer av att det talsystemet har 10 som bas. Det innebär att för varje position i ett tal, ökar siffrans värde 10 gånger (1-tal, 10-tal,100-tal, 1000-tal, 10000-tal o s v). I det Binära talsystemet finns endast siffrorna 0 och 1. I det Hexadecimala talsystemet finns det 16 siffror: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. Efter siffran nio har man fått ta till bokstäverna A-F för att symbolisera siffror.

För att arbeta med programutveckling till DataBoard I/O-kort är det i första hand det Binära talsystemet, men även det Hexadecimala talsystemet, som är av betydelse.

4.1.1 Decimala talsystemet

I det decimala talsystemet ökar siffrans värde med en faktor 10 för varje position i talet.

Exempel: 2074

TALET	0	0	2	0	7	4
POSITION	5	4	3	2	1	0
VIKT	10^5	10^4	10^3	10^2	10^1	10^0
VÄRDE	0*100000	0*10000	2*1000	0*100	7*10	4*1

+000000
+000000
+002000
+000000
+000070
+000004
=002074

Som vi ser får varje siffra sin vikt efter positionen i talet. Värdet av hela talet fås genom att multiplicera varje siffra i en position, med positionens vikt och sedan lägga ihop de olika deltalen.

4.1.2 Binära talsystemet

På samma sätt men med annan vikt, får varje siffra i det binära talsystemet sin vikt efter positionen i det binära talet:

Exempel: 01001110

TALET	0	1	0	0	1	1	1	0	
POSITION	7	6	5	4	3	2	1	0	
VIKT	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
VÄRDE	0*128 0+	1*64 64+	0*32 0+	0*16 0+	1*8 8+	1*4 4+	1*2 2+	0*1 0	=78

M a o det binära talet 01001110 motsvarar det decimala talet 78.

00101101=0+0+32+0+8+4+0+1=45
11010010=128+64+0+16+0+0+2+0=210
O s v.

Hur kommer nu det här in i samband med DataBoard? Jo, om man läser in en port där databitarna 7, 5,4 och 0 är nollor (0 volt) och databitarna 6,3,2 och 1 är ettor (5 volt) kommer vi att läsa in talet 78 från porten. Det är vi vanligen inte hjälpta av utan måste på något sätt identifiera varje databit och dess värde. Den identifieringen kan vi göra med hjälp av bitvikten och logiska operatorer enligt nedan (KAP 4.3).

4.1.3 Hexadecimala talsystemet

För att ange att ett tal skall tolkas som Hexadecimalt i IBM PC BASIC markeras det med &H framför siffrorna.

Exempel :	Decimalt	Hexadecimalt
	11	&H00B
	16	&H010
	32	&H020
	256	&H100
	768	&H300
	769	&H301

IN/UT portarna till Datums PC4680-kort, är adresserade med start från adress 768 och uppåt. Det blir med andra ord lite enklare om man anger adressen till varje port Hexadecimalt i stället för Decimalt. I flera exempel i den här boken, har vi anget de Hexadecimala talet för adressen till DataBoardkortet.

4.2 Hantering av olika talsystem i IBM BASIC

I IBM PC-BASIC finns inget sätt att arbeta med det Binära talsystemet. Däremot finns det färdiga funktioner för det Hexadecimala talsystemet och det Octala talsystemet (det senare går vi inte in på här).
Ett Hexadecimalt tal har symbolen &H framför talet. &H100 är omvandlat till Decimala talsystemet lika med 256 (16 upphöjt till 2 eftersom ettan står i tredje positionen).

För att omvandla ett decimalt tal, till en textsträng innehållande motsvarande

hexadecimala tal gör man så här i IBM PC BASIC:

```
10 INPUT X
20 A$=HEX$(X)
30 PRINT X "DECIMALT ÄR " A$ " HEXADECIMALT"
```

Talet X kan vara mellan -32768 till +65535.

Vid omvandling mellan binära tal och hexadecimala tal, börjar man med att dela upp det binära talet i grupper om 4 från höger. Vanligen består ett binärt tal av 8 st siffror (bitar). Tillsammans utgör dessa 8 siffror en BYTE. Uppdelningen till två grupper om fyra bitar, förenklar överföringen från binära tal till hexadecimala. Varje grupp om 4 siffror kalls en NIBBLE.
Fyra stycken binära siffror kan representera ett tal mellan 0 och 15 (decimalt), d v s 0-F hexadecimalt. Så här går det till:

01001011= 0100(=8) 1011(=B) = 8B (hexadecimalt)

4.3 Logiska operationer

Logiska operatorer, utför logiska (BOOLEAN) operationer på numeriska värden. Med en logisk operator jämför man två tal på bitnivå (som binära tal) och låter programmet utnyttja resultatet av jämförelsen. Vanligen används logiska operationer för att filtrera bort oönskad information från ett värde som lästs in från ett DataBoard-kort.

De viktigaste logiska operatorerna i IBM PC-BASIC är: **AND, NOT, OR** samt **XOR.** Varje Bit i resultatet bestäms av motsvarande bit i de två operander på vilka den logiska operatorn arbetar.

AND-operationen

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

Om man vill veta huruvida mest signifikanta biten i en inläst port från ett DataBoard-kort är en etta eller en nolla, kan man göra så här:

	DECIMALT	BINÄRT
INLÄST TAL	187	1011 1011
AND	128	1000 0000
RESULTAT	128	1000 0000

D v s eftersom resultatet blev 128 så måste det finnas en etta på efterfrågad insignal. Om vi i stället hade önskat testa biten med vikten 64, hade resultatet blivit som följer:

	DECIMALT	BINÄRT

INLÄST TAL	187	1011 1011
AND	64	0100 0000
RESULTAT	0	0000 0000

Eftersom resultatet nu blev 0, kan man konstatera att signalen på efterfrågad ingång på DataBoard-kortet är en nolla (logiskt sett).

NOT-operationen

X	NOT X
1	0
0	1

Observera att NOT endast arbetar på en operand till skillnad från de övriga operatorerna.

	DECIMALT	BINÄRT
INLÄST TAL	186	1011 1010
NOT		
RESULTAT	69	0100 0101

NOT är en funktion som används när alla bitar skall skifta värde. D v s när alla nollor i den inlästa byten skall vändas till ettor och vise versa. Funktionen är perfekt att använda då insignalerna till DataBoard-kortet är av fel (oönskad) polaritet.

OR-operationen

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

	DECIMALT	BINÄRT
INLÄST TAL	171	1010 1011
OR	15	0000 1111
RESULTAT	175	1010 1111

Med OR (eller) räcker det om en av bitarna i de två operanderna är en etta, för att resultatet skall bli en etta. Kan utnyttjas för att ettställa vissa bitar i en byte som skickas ut på en port.

XOR-operationen

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

	DECIMALT	BINÄRT
INLÄST TAL	187	1011 1011
XOR	171	1010 1011
RESULTAT	16	0001 0000

Intressant resultat får man vid användning av XOR(eXklusive OR). Vid upprepad avläsning av en port på DataBoard och jämförelse (XOR) med föregående avläsning blir resultatet 0 om inget har hänt på porten. Så fort någon av bitarna ändrar sig får man med XOR dessutom indikerat vilken signal som har ändrats.

Programexempel

Exempel: omvandling av hexadecimala tal skrivet i PC BASIC

```
90 ' P-J H/Datum  850110
100 ' OMVANDLING AV DECIMALTAL TILL BINÄRTAL OCH
    TVÄRTOM
110 CLS
120 PRINT "OMVANDLA TAL"
130 PRINT "1. FRÅN DECIMALTAL TILL BINÄRTAL"
140 PRINT "2. FRÅN BINÄRTAL TILL DECIMALTAL"
150 PRINT "3. AVSLUTA"
160 PRINT
170 PRINT "ANGE ALTERNATIV"; : INPUT SVAR
180 IF SVAR<1 OR SVAR>3 THEN 110
190 IF SVAR=3 THEN 900
200 ON SVAR GOSUB 400,600
210 LOCATE 1,1 : GOTO 120
400 '
410 ' DECIMALT TILL BINÄRT
420 '
430 LOCATE 10,1 : PRINT SPACE$(80) : LOCATE 10,1
440 INPUT "SKRIV ETT DECIMALTAL (MAX 65535)";DECIMAL
450 IF DECIMAL<0 OR DECIMAL>65535! THEN 400
455 TAL=DECIMAL : GOSUB 800 : DECIMAL%=TAL ' Omvandla
    till heltal
460 '
470 BIN$=""
480 FOR I=15 TO 0 STEP -1
485   TAL=2^I : GOSUB 800 : MASK%=TAL ' Omvandla till
    heltal
490   BIN$=BIN$+CHR$((DECIMAL% AND MASK%)/MASK%+48)
500 NEXT I
510 PRINT "BINÄRA TALET BLEV ";:COLOR 10 :PRINT BIN$ :
    COLOR 2
520 RETURN
600 '
610 ' BINÄRT TILL DECIMALT
620 '
630 LOCATE 15,1 : PRINT SPACE$(80) : LOCATE 15,1
640 PRINT "SKRIV ETT BINÄRTAL (MAX 16
    POSITIONER)";:INPUT BIN$
650 IF LEN(BIN$)>16 THEN 600
660 '
670 I=1 : ANTAL=LEN(BIN$)
680 DECIMAL=0 : PLATS=1
690 WHILE I<ANTAL+1
```

```
700    PLATS=INSTR(I,BIN$,"1") : IF PLATS=0 THEN 740
710    DECIMAL=DECIMAL+2Ü(ANTAL-PLATS)
720    I=PLATS+1
730 WEND
740 LOCATE 17,1 : PRINT SPACE$(80) : LOCATE 17,1
750 PRINT "DECIMALA TALET BLEV ";: COLOR 10: PRINT
    STR$(DECIMAL):COLOR 2
760 RETURN
800 '
810 ' Subrutin som omvandlar flyttal till heltal
820 '
830 IF TAL>-32769! AND TAL<32768! THEN 850
840 TAL=TAL-65536!
850 RETURN
900 '
910 ' AVSLUTA
920 '
930 LOCATE 19,1
940 END
```

Kapitel 5 DataBoard för digitala signaler

5.1.1	DataBoard 4005, digitalt I/O-kort TTL in/ut
5.1.2	DataBoard 4006, digitalt I/O-kort TTL in/ut
5.2.1	DataBoard 4008/11, opto in
5.2.1	DataBoard 4095, opto ut
5.3	DataBoard 4085, TTL in/ut
5.4	DataBoard 4103, relä
5.5	DataBoard 4013, mix I/O

Kapitel 5.1.1 DataBoard 4005
Digitalt I/O-KORT

5.1.1.1 Specifikationer 4005

Utgångar: 16 TTL OPEN-COLLECTOR
Ingångar: 8 TTL

5.1.1.2 Beskrivning 4005

Digitalt I/O-kort med 16 utgångar open collector. På utgångarna kan anslutas maximalt 24 V, och driva högst 125 mA. Utgångarna styrs i grupper om 8.

Ingångarna är av normalt TTL-snitt, vilket innebär att motsvarande bit nollställes om en spänning 0-1 V finns på ingången eller ettställes om en spänning 4-5 V finns på ingången. Alla bitar på utgångarna är nollställda efter reset (RST) signalen. Alla ingångar avläses samtidigt.

4005 kan användas till att styra mindre reläer, lampor, lysdioder samt liknande utrustning. 4005 kan även läsa av tumhjul, switchar etc. Man kan också ansluta TTL-kompatibel elektronik. Om man t ex låter utgångarna styra tumhjul så kan man avläsa upp till 16x2 = 32 st tumhjul.

4005 har open-collector utgångar. En vanlig TTL-krets har utgångar av totempåletyp, där utsignalen tas ut mellan två seriekopplade transistorer. Lägger TTL-kresten ut en logisk etta på utgången leder den övre transistorn och den undre spärrar, medan det motsatta förhållandet råder när kretsen lägger ut en logisk nolla.

En TTL-krets med open-collector utgång saknar den övre av dessa transistorer. En sådan utgång kan man således använda som en "elektronisk strömbrytare", genom att vi lägger ut logiska nollor och ettor på utgången.

När vi kopplar in kortet ligger nollor på alla utgångar vilket medför att transistorerna (="strömbrytarna") är ledande. Ansluter vi utgången till en lysdiod i serie med ett motstånd upp till +5 V kommer således lysdioden att lysa. För att öppna en strömbrytare, dvs göra utgångstransistorn spärrande skall man lägga ut en etta på den utgång man vill bryta.

Utgångarna sitter i grupper om åtta och styrs av binärtal. Detta innebär att man styr genom att skicka ut heltal som ligger mellan 0 och 255 med OUT-kommandot. För att veta vad man ettställer och nollställer måste man således göra om det decimala heltalet till ett binärt tal.

Exempel:

Ett kommando "OUT &H300,57" innebär att ett binärt tal, som motsvarar det decimala talet 57 läggs ut på grupp 1, utgång 0-7.

57 = 0011 1001 (= D7 D6 D5 D4 D3 D2 D1 D0)

Vi ser att utgång 0, 3, 4 och 5 blir ettställda medan övriga blir nollställda.

De åtta ingångarna avläses samtidigt och bildar att heltal mellan 0 och 255.
De inledningar som har en spänning 0-1 V ger en logisk nolla medan de som har en spänning 4-5 V ger en logisk etta. De ledningar som inte är anslutna ger logiska ettor.

Exempel på användning: BCD-signaler till och från instrument, tryckknappar, gränslägesbrytare etc.

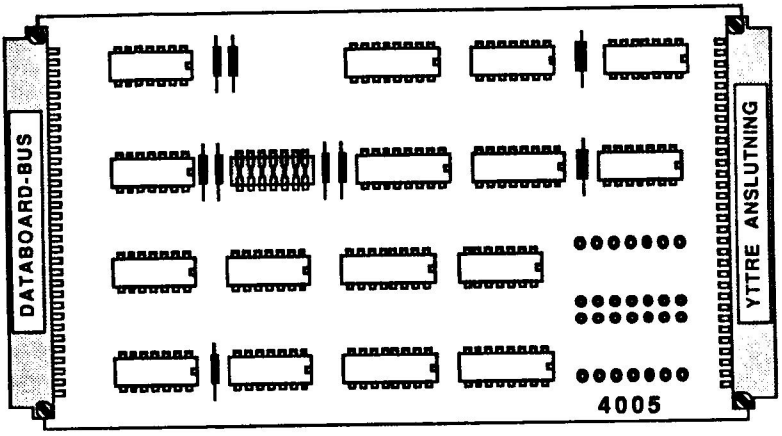


Fig 5.1.1.1 DataBoard 4005

5.1.1.3 Programmering 4005

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen Sätter alla utgångar på kortet till logisk nolla. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H300 (768 decimalt) kommer av att interfacekortet i IBM PC lägger till värdet &H300 till den adress som byglats på I/O-kortet (300 HEX).
INP(&H300)	Avläsning av de 8 bitarna (ingångarna) från kortet. Värdet är ett heltal mellan 0 och 255 beroende på insignalen på de olika ingångarna. Exempel: 110 V=INP(&H300)
OUT &H300, [DATA]	Lägger ut värdet [DATA] på de 8 utgångarna i grupp1 på kortet (0-255). Exempel: 240 OUT &H300,35 lägger ut data enligt följande mall: D0=1,D1=1,D2=0,D3=0,D4=1,D5=0,D6=0,D7=0 1 2 0 0 32 0 0 0 =35
OUT &H302,[DATA]	Samma som ovanstående men avser grupp 2.

5.1.1.4 Installation

Val av kortadress
Bestäm först vilken adress kortet skall ha. Se kapitel 3.2.2 för beskrivning över hu man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
Fig 5.1.1.2 beskriver stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan är numrerade från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker genom att i BASIC skriva OUT &H301,A, där A är kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

5.1.1.5 Programexempel

Se kapitel 5.1.2.5

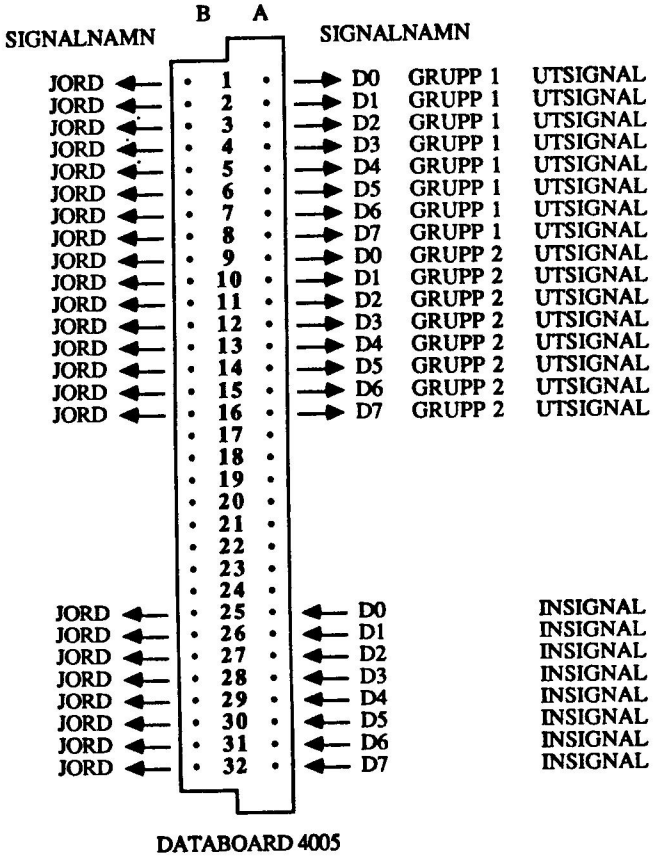


Fig 5.1.1.2 DataBoard 4005, stiftlayout och kontaktnamn

KAPITEL 5.1.2 DataBoard 4006
Digitalt I/O-kort

5.1.2.1 Specifikationer 4006:

Utgångar:	32 TTL Tri-state
Ingångar:	16 TTL

5.1.2.2 Beskrivning 4006

4006 har 32 utgångar, uppdelade i fyra grupper om 8 bitar i vardera. 4006 skiljer sig från 4005 genom att de är av tri-state typ. Det innebär att utgångarna kan anta tre lägen : ettställd, nollställd eller frikopplad, "flytande". Man kan frikoppla en eller alla grupper genom att lägga + 5 V på tri-state kontrollen för gruppen man vill ha frikopplad. De 16 ingångarna fungerar som på 4005, men om man vill använda de åtta mest signifikanta insignalerna väljer man det vid kortval(CS) genom att sända ut kortets adress plus 128, dvs OUT &H301,128+<kortadress>.

4006 kan användas för styrning av yttre digitala enheter och liknande samt avläsning av omkopplare, tumhjul etc. In-och utgångarna är TTL-kompatibla.

Program skrivna för max 16 utgångar och max 8 ingångar kan användas på både 4005 och 4006.

Till skillnad från 4005 har 4006 32 st tri-state TTL utgångar. Detta medför att man med OUT-kommandona kan sätta utgångarna till +5 V (logisk etta) eller 0 V (logisk nolla). Tri-state läget innebär att varje utgång har tre lägen hög, låg och frikopplad. Utgångarna är kopplade i fyra grupper om åtta bitar. Varje grupp styrs av en tri-state signal. Om tri- state funktionen inte skall utnyttjas, kopplas tri-state styrsignalen (TS) till jord. Tri-state funktionen används bl a när två eller fler grupper ligger anslutna parallellt på samma buss. Tri-state styrsignalerna finns i I/O-kontakten och genom att koppla dessa till en utport kan vi styra grupperna programvarumässigt. Efter en reset (RST) signal ligger alla utgångar låga (O V). För att ge utsignaler, skall motsvarande tri-state styringång vara 0 Volt.

Ingångarna finns i två grupper om åtta stycken i varje. Man kan bara använda en grupp i taget, sedan måste en omställning av kortvalet ske för att kunna använda den andra. Om man väljer kort på vanligt vis, t ex: OUT &H301,3 erhålls insignaler från grupp 1. Om man däremot väljer kort och adderar 128 till kortadressen t ex OUT &H301,3+128 får man insignaler från grupp 2. Glöm inte att lägga till värdet &H300 som krävs av kortet i IBM PC.

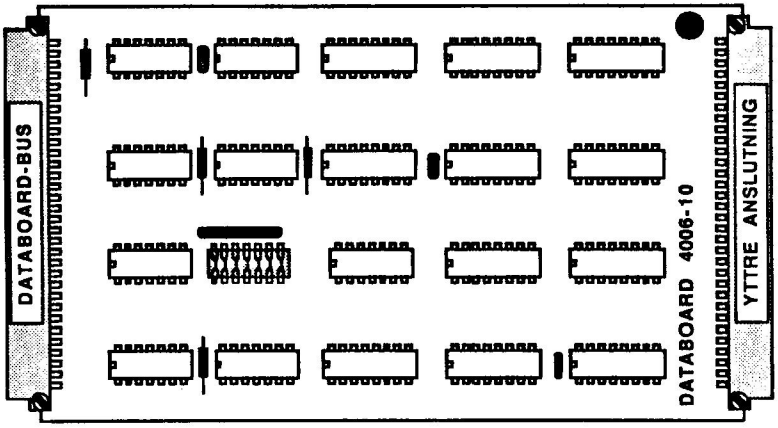


FIG 5.1.2.1 DATABOARD 4006

5.1.2.3 Programmering 4006

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Sätter alla utgångarna på kortet till logisk nolla. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar grupp 1 på kortet med kortadressen [kortadress]. Exempel: OUT &H301,26, väljer ut kortet med kortadressen 26 enligt byglingarna samt grupp1. &H300 kommer av att interfacekortet i IBM PC lägger till värdet 768 decimalt till den adress som byglats på I/O-kortet (300 HEX).
OUT &H301,[KORTADRESS+128]	Aktiverar grupp 2 på kortet med kortadressen [kortadress]. Exempel: OUT 1,26+128, väljer ut kortet med kortadressen 26 enligt byglingarna. &H300 kommer av att interfacekortet i IBM PC lägger till värdet 768 decimalt till den adress som byglats på I/O-kortet (300 HEX).

INP(&H300)	Avläsning av de 8 bitarna (ingångarna) från grupp 1 eller 2 enligt tidigare kortval. Värdet är ett heltal mellan 0 och 255 beroende på insignalen på de olika ingångarna. Exempel: 110 V=INP(&H300) tilldelar variabeln V ett värde mellan 0 och 255.
OUT &H300, [DATA]	Lägger ut värdet [DATA] (0-255) på de 8 utgångarna (D0-D7) i grupp1 på kortet). Exempel: 240 OUT &H300,35 lägger ut data enligt följande mall: D0=1,D1=1,D2=0,D3=0,D4=1,D5=0,D6=0,D7=0 1 2 0 0 32 0 0 0 =35
OUT &H302,[DATA]	Samma som ovanstående men avser D0-D7 i grupp 2.
OUT &H303,[DATA]	Samma som ovanstående men avser D0-D7 i grupp 3
OUT &H304,[DATA]	Samma som ovanstående men avser D0-D7 i grupp 4.

5.1.2.4 Installation

Val av kortadress
Bestäm först vilken adress kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
Fig 5.1.2.3 beskriver stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan är numrerade från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker genom att i BASIC skriva OUT &H301,A, där A är kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas.
Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

5.1.2.5 Programexempel

Styrning av enstaka utgångar i en grupp
Ofta vill man bara styra en utgång i taget medan man vill ha de andra utgångarna opåverkade efter operationen. Detta kan åstadkommas genom maskning. Eftersom maskning för ett- respektive noll-ställning återkommer ganska ofta i program, som använder sig av korten 4005 eller 4006 för digital signalering, är det lämpligt att definiera subrutiner för detta.

```
1000 '
1010 ' SUBRUTIN FÖR NOLLSTÄLLNING AV EN PORT
1020 '
1030 ' In  Port% = Vilken port, Sist% = Vad som sist
skickades
1040 ' Ut  Ut% = Vad som skall skickas nu
1050 '
1060 TAL=65535!-2ÜPORT%: GOSUB 1200  : MASK%=TAL
1070 UT%=SIST% AND MASK%
1080 RETURN
1100 '
1110 ' SUBRUTIN FÖR ETTSTÄLLNING AV EN PORT
1120 '
1130 ' In  Port% = Vilken port, Sist% = Vad som
skickades sist
1140 ' Ut  Ut% = Vad som skall skickas nu
1150 '
1160 TAL=2ÜPORT% : GOSUB 1200 : MASK%=TAL
1170 UT%=SIST% OR MASK%
1180 RETURN
1200 '
1210 ' SUBRUTIN FLYTTAL-->HELTAL
1220 '
1230 ' In  Tal! = Det tal som skall omvandlas
1240 ' Ut  Tal! = Ett tal i intervallet -32768 - 32767
1250 IF TAL>-32769! AND TAL<32768! THEN 1270
1260 TAL=TAL-65536!
1270 RETURN
```

Programexemplet nedan illustrerar, med hjälp av de funktioner vi definierat ovan, hur man kan styra utsignaler med I/O-korten 4005 och 4006. Fig. 5.1.2.2 visar uppkoppling av 4005 respektive 4006 för test av detta program. Vi använder oss av lysdioder för att visa tillståndet hos utgångarna, (strömbrytarna är inte nödvändiga för test av detta program)

Exempel 1

Styrning av utgångarna hos I/O-kort 4006.

```
10 ' Rinn4006.bas  /Datum  P-J H 850118
15 ' Rinnande lysdioder på kortet 4006
17 ' hastigheten bestäms av inställningen på tumhjulen.
20 '
30 DEF FNHELTAL(TAL)=TAL-65536!
40 '
50 Z=INP(&H307) ' Nollställ
70 '
80 ' Programloopen
90 '
93  WHILE -1
100   ' Läs av tumhjulsinställning
101   OUT &H301,4 : A1%=INP(&H300) AND &HF ' Läs tiotal
från tumhjul
102   OUT &H301,&H84 : A1%=((INP(&H300) AND &HF) +
(A1%*10))*10 ' Ental+tiotal
103   OUT &H301,4
105   OUT &H303,&HFD
107   ' Grupp 1 på väg upp
110   FOR I=0 TO 7
120     IF 2ÜI >32767 THEN UT%=FNHELTAL(2ÜI) ELSE UT%=2ÜI
130     OUT &H300,UT% XOR &HFF ' Tänd en port
135     FOR V=1 TO A1% : NEXT V ' Vänta enligt tumhjulet
140   NEXT I
145   OUT &H300,255
147   ' Grupp 2 på väg upp
150   FOR I=0 TO 6
160     IF 2ÜI >32767 THEN UT%=FNHELTAL(2ÜI) ELSE UT%=2ÜI
170     OUT &H302,UT% XOR &HFF
180     FOR V=1 TO A1% : NEXT V
190   NEXT I
200   OUT &H302,255 ' Släck grupp 2
305   OUT &H303,&HFE
307   ' Grupp 2 på väg ner
310   FOR I=6 TO 0 STEP -1
```

```
320     IF 2ÜI >32767 THEN UT%=FNHELTAL(2ÜI) ELSE UT%=2ÜI
330     OUT &H302,UT% XOR &HFF
335     FOR V=1 TO A1% : NEXT V
340   NEXT I
345   OUT &H302,255 ' Släck grupp 2
347 ' Grupp 1 på väg ner
350   FOR I=7 TO 0 STEP -1
360     IF 2ÜI >32767 THEN UT%=FNHELTAL(2ÜI) ELSE UT%=2ÜI
370     OUT &H300,UT% XOR &HFF
380     FOR V=1 TO A1% : NEXT V
390   NEXT I
400   OUT &H300,255   ' Släck grupp 1
410 WEND
```

Exempel 2

Läsning av ingångarna hos I/O-kort 4005 och 4006

```
10 ' Lasin.bas  P-J H 850122
30 '
40 ' LÄSNING AV INGÅNG PÅ KORTEN 4005 OCH 4006
50 ' Testat på johans tumhjul, endast tiotal
100 '
110 ' FUNKTION FÖR KONVERTERING AV FLYTTAL TILL HELTAL
120 '
130 ' In  Tal! = Ett tal som skall omvandlas, OBS
32768<Tal!<65536
140 ' Ut  Ett tal i intervallet -32768 - 0
150 '
160 DEF FNHELTAL%(TAL)=TAL-65536!
1000 '
1010 ' INITIERA
1020 '
1030 Z=INP(&H307) ' NOLLSTÄLL KORTEN
1040 OUT &H301,4 ' KORTVAL 4006 GRUPP 1 Tumhjul tiotal
1050 '
1060 ' HUVUDPROGRAMMET
1070 '
1080 CLS
1085 WHILE -1
1090    INPUT "Vilken ingång vill du läsa (0-3)",PORT%
1100    IF PORT%<0 OR PORT%>3 THEN 1090
1105    '
1110    PRINT :PRINT "På ingången ligger en "; : COLOR 10
1120    GOSUB 2000 ' Läs inporten
```

```
1130   IF STATUS%=1 THEN PRINT "etta." ELSE PRINT
"nolla."
1140   COLOR 2 : PRINT
1150 WEND
1160 END
2000 '
2010 ' SUBRUTIN FÖR INLÄSNING AV EN PORT
2020 '
2030 ' In  Port% kan vara 0-7
2040 ' Ut  Status% Läst värde antingen 1 (sant) eller 0
(falskt)
2050 '
2060 MASK=2ÜPORT%
2070 IF MASK>32767 THEN MASK%=FNHELTAL(MASK) ELSE
MASK%=MASK 'EV KONVERTERA
2080 STATUS%=(INP(&H300) AND MASK%)/MASK%
2090 RETURN
```

Programexempel 3 illustrerar hur man kan läsa insignaler med I/O-korten 4005 och 4006. Fig. 5.1.2.2 visar uppkoppling av 4005 respektive 4006 för test av detta program. Vi använder oss av strömbrytare för att simulera insignaler till kortet, (lysdioderna är inte nödvändiga för detta program).

Exempel 3

Läsning av 2 portar, tumhjul 10-tal och 1-tal

```
10 ' Lastum.bas  P-J H  850122
20 ' Exempel
30 '
40 ' Avläsning av insällning på tumhjul, tiotal och
   ental
50 ' Testat på johans tumhjul
1000 '
1010 ' INITIERA
1020 '
1030 Z=INP(&H307) ' Nollställ korten
1050 '
1060 ' HUVUDPROGRAMMET
1070 '
1080 CLS
1082 PRINT "Stanna programmet med ctrl-break"
1085 WHILE -1
1087   LOCATE 10,1
1090   PRINT "Tumhjulet står nu inställt på";
```

```
1095    '
1100    OUT &H301,4 ' Välj kortet med tumhjul, tiotal
1110    TIOTAL%=INP(&H300) AND &HF ' Läs av tumhjulet,
    tiotal
1120    TIOTAL%=TIOTAL%*10
1130    '
1140    OUT &H301,&H84 ' Välj tumhjul, ental
1150    ENTAL%=INP(&H300) AND &HF ' Läs ental
1160    '
1170    PRINT TIOTAL%+ENTAL%
1180    '
1200 WEND
```

Exempel 4

Styrning av 4005 och 4006

```
10 ' Styrut.bas  P-J H 850110
30 ' Styrning av utsignalerna på kort 4005 och 4006
40 ' Styrningen sker indivinduellt. En utgång påverkas
    per gång
50 '
60 Z=INP(&H307) ' Nollställer alla I/O-korten
70 OUT &H301,4 ' Kortval, 4005=3 och 4006=4
75 WHILE BIT%<2 ' Sluta med en tvåa
80   CLS
90   PRINT "Ange grupp";:INPUT GRUPP%
100  IF GRUPP%<1 OR GRUPP%>4 THEN 90
110  IF GRUPP%=1 THEN GRUPP%=0
120  '
130  PRINT "Ange utgång i gruppen (0-7)";:INPUT PORT%
140  IF PORT%<0 OR PORT%>7 THEN 130
150  '
```

```
160  PRINT "Ange etta eller nolla på porten (0-1,
    2=slut)";:INPUT BIT%
170  IF BIT%<0 OR BIT%>2 THEN 160
180  '
190  IF BIT%=0 THEN GOSUB 1000 ' Nollställ porten
200  IF BIT%=1 THEN GOSUB 1100 ' Ettställ porten
210  OUT &H300+GRUPP%,UT%
220  SIST%=UT% ' Spara sist skickat
230 WEND
300 END
1000 '
1010 ' SUBRUTIN FÖR NOLLSTÄLLNING AV EN PORT
1020 '
1030 ' In  Port% = Vilken port, Sist% = Vad som sist
    skickades
1040 ' Ut  Ut% = Vad som skall skickas nu
1050 '
1060 TAL=65535!-2ÜPORT%: GOSUB 1200  : MASK%=TAL
1070 UT%=SIST% AND MASK%
1080 RETURN
1100 '
1110 ' SUBRUTIN FÖR ETTSTÄLLNING AV EN PORT
1120 '
1130 ' In  Port% = Vilken port, Sist% = Vad som
    skickades sist
1140 ' Ut  Ut% = Vad som skall skickas nu
1150 '
1160 TAL=2ÜPORT% : GOSUB 1200 : MASK%=TAL
1170 UT%=SIST% OR MASK%
1180 RETURN
1200 '
1210 ' SUBRUTIN FLYTTAL-->HELTAL
1220 '
1230 ' In  Tal! = Det tal som skall omvandlas
1240 ' Ut  Tal! = Ett tal i intervallet -32768 - 32767
1250 IF TAL>-32769! AND TAL<32768! THEN 1270
1260 TAL=TAL-65536!
1270 RETURN
```

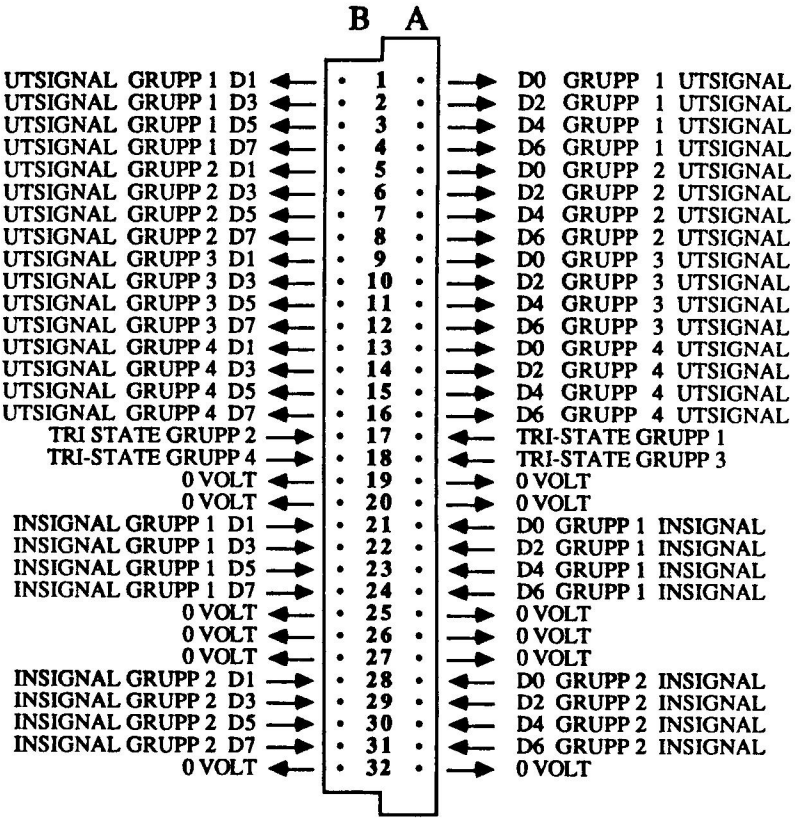


Fig 5.1.2.3 DataBoard 4006, stiftlayout och kontaktnamn

Kapitel 5.2.1 DataBoard 4011/08
Opto I/O-kort

5.2.1.1 Specifikationer

4008
Optoingångar 16 stycken
Insignal: Logisk "0": 10-25 mA alternativt 5-12 Volt
Logisk "1": 0 mA alternativt 0 Volt (öppen)

4011
Optoingångar 16 stycken
Insignal: Logisk "0": 10-50 mA alternativt 5-24 Volt
Logisk "1": 0 mA alternativt 0 Volt (öppen)

5.2.1.2 Beskrivning av 4008 och 4011

4008/4011 har 16 stycken optokopplare på ingångarna. De är av typen CQY 80, (eller motsvarande) med en isolationsspänning på 4 kV. Med hjälp av dessa blir datorn galvaniskt skild från insignalerna. Detta innebär att man slipper ifrån problemet med gemensam jordning vid inkoppling av instrument. Risken för att spänningstransienter, induktionsspänning etc, orsakade av elektriska motorer, åsknedslag eller dylikt skall fortplanta sig in i datorn och störa eller skada den känsliga elektroniken är minimal. Skillnaden mellan korten 4008 och 4011 är att den logiska nollan på 4008 ligger på en spänning mellan 5 och 12 V medan den på 4011 ligger mellan 5 och 24 V. Den logiska ettan ligger i båda fallen på 0 V.

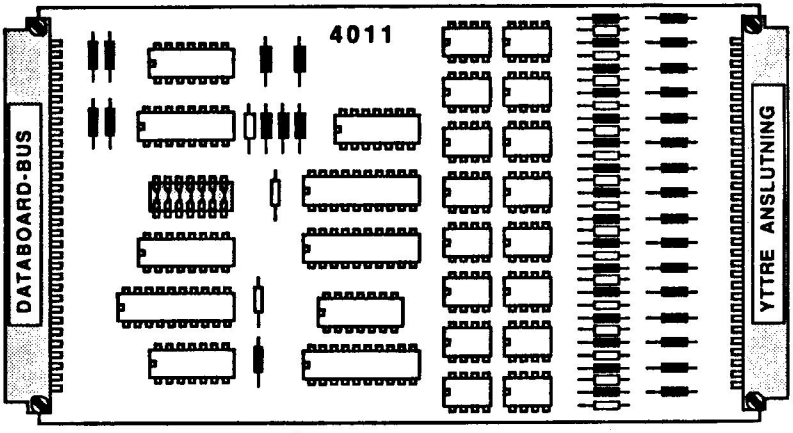


Fig 5.2.1 Databoard optokort 4011

5.2.1.3 Programmering av 4008/4011

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar grupp 1 på 4008/4011-kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H300 kommer av att interfacekortet i IBM PC lägger till värdet 768 decimalt till den adress som byglats på I/O-kortet (300 HEX).
OUT &H301,[KORTADRESS+128]	Aktiverar grupp 2 på 4008/4011-kortet med kortadressen [kortadress]. Exempel: OUT &H301,23+128, väljer ut kortet med kortadressen 23 enligt byglingarna. &H300 kommer av att interfacekortet i IBM PC lägger till värdet 768 decimalt till den adress som byglats på I/O-kortet (300 HEX).
INP(&H300)	Avläsning av de 8 bitarna (=ingångarna på grupp enligt tidigare val) från kortet. Värdet är ett heltal mellan 0 och 255 beroende på insignalen på de olika ingångarna. Exempel: 110 V=INP(&H300)

5.2.1.4 Installation av 4008/4011

Val av kortadress
Bestäm först vilken adress kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
Fig 5.2.3 beskriver stiftlayout och signalnamn på I/O-sidan. ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan är numrerade från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i Expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker genom att i BASIC skriva OUT &H301,A, där A är kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

5.2.1.5 Exempel

Exempel 1

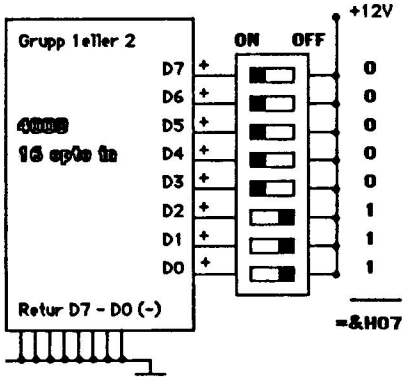


Fig 5.2.2 Koppling till exempel 1

Inläsning av en port och visa resultatet binärt

```
10 ' Läs4011.bas  P-J Högfeldt  850124
30 ' Läser en ingång och visar binärt
```

```
40 '
100 '
110 ' FUNKTION FÖR KONVERTERING AV FLYTTAL TILL HELTAL
120 '
130 ' In  Tal! = Ett tal som skall omvandlas, OBS
     32768<Tal!<65536
140 ' Ut  Ett tal i intervallet -32768 - 0
150 '
160 DEF FNHELTAL%(TAL)=TAL-65536!
200 '
210 ' HUVUDPROGRAMMET
220 '
230 Z=INP(&H307)
240 CLS
250 PRINT "Optoingångarna har följande värde, sluta med
     S." : PRINT
255 PRINT "Bit nr   0 1 2 3 4 5 6 7"
260 '
270 WHILE I$<>"S"
280   '
290   GRUPP1$="" : GRUPP2$=""
300   FOR BIT%=0 TO 7
310     OUT &H301,4
320     GOSUB 700 : GRUPP1$=GRUPP1$+STR$(STATUS%)
330     '
340     OUT &H301,&H84
350     GOSUB 700 : GRUPP2$=GRUPP2$+STR$(STATUS%)
360   NEXT BIT%
370   '
380   PRINT "Grupp 1: " GRUPP1$
390   PRINT "Grupp 2: " GRUPP2$
400   INPUT "Mer? (S=Slut) ",I$ : PRINT
410   '
420 WEND
430 END
700 '
710 ' SUBRUTIN FÖR INLÄSNING AV EN PORT
720 '
730 ' In  Bit% kan vara 0-7
```

```
740 ' Ut  Status% Läst värde antingen 1 (sant) eller 0
    (falskt)
750 '
760 MASK=2ÜBIT%
770 IF MASK>32767 THEN MASK%=FNHELTAL(MASK) ELSE
    MASK%=MASK 'EV KONVERTERA
780 STATUS%=(INP(&H300) AND MASK%)/MASK%
790 RETURN
1000 '
1010 ' HÄR FÖLJER KOMMENTARER TILL PROGRAMMET
1020 '
1030 ' Rad Kommentar
1040 ' 230 Nollställning av all kort.
1050 ' 270 Loopa tills ett S är inmatat.
1060 ' 300 Räkna upp Bit% så att 8 ingångar läses för
    varje grupp.
1070 ' 310 Välj grupp 1 på kort med adress 4.
1080 ' 320 Läsen ingång och spara det binära värdet i
    Grupp1$.
1090 ' 340 Välj grupp 2.
1100 ' 350 Läs och spara ingången.
```

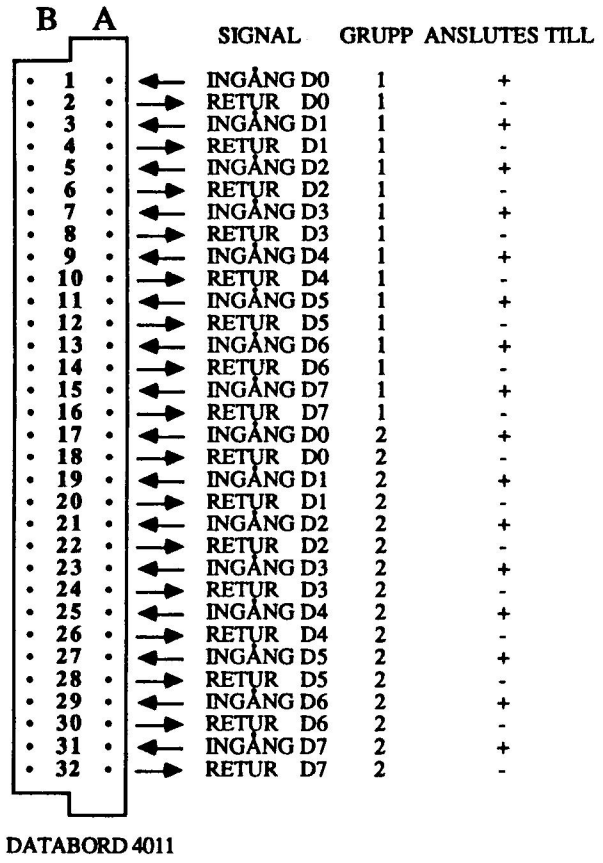


Fig 5.2.1.3 DataBoard 4011, stiftlayout och kontaktnamn

Kapitel 5.2.2 DataBoard 4095

Digitalt 16 utgångars kort 4095 med galvanisk separation.

5.2.2.1 Specifikationer 4095

Optoutgångar: 16 stycken
Utsignal: Maximal spänning 24 Volt och 800 mA, termiskt
Överströmsskydd: Restspänning vid tillslag: 2 Volt.

5.2.2.2 Beskrivning av 4095

Med 4095 kan man styra 16 optoisolerade utgångar. Man slipper således (se även 4008/4011) en gemensam jordning mellan datorn och uppkopplingarna utanför. Optokopplarna på 4095 klarar av att driva maximalt 24V, 0.8 A och är termiskt skyddade mot överström. 4095 är programkompatibel med utgångarna på det digitala I/O-kortet 4005. Med 4103 (16 Reläutgångar) är kortet både program-och kontaktkompatibelt.

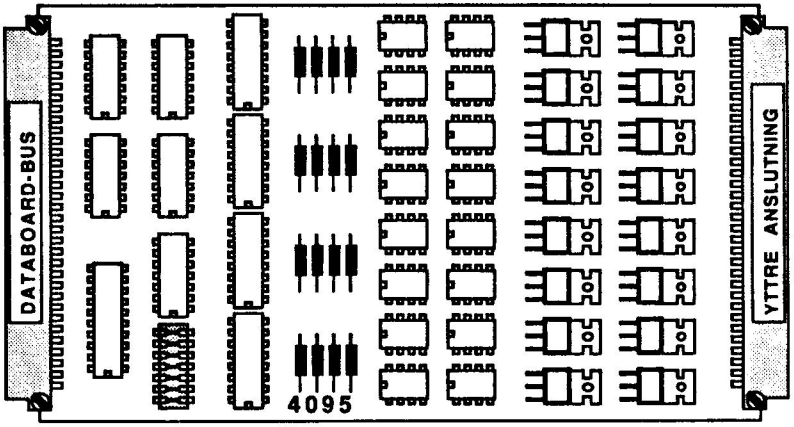


Fig 5.2.2.1 Optokort 4095

5.2.2.3 Programmering av 4095

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar 4095-kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H300 kommer av att interfacekortet i IBM PC lägger till värdet 768 decimalt till den adress som byglats på I/O-kortet (300 HEX).
OUT &H300, [DATA]	Lägger ut värdet [DATA] (0-255) på de 8 utgångarna (D0-D7) i grupp1 på kortet). Exempel: 240 OUT &H300,35 lägger ut data enligt följande mall: D0=1,D1=1,D2=0,D3=0,D4=1,D5=0,D6=0,D7=0 1 2 0 0 32 0 0 (=35 Dvs gör D0, D1 och D4 ledande och resterande utgångar bryts.
OUT &H302,[DATA]	Samma som ovanstående men avser D0-D7 i grupp 2.
OUT &H304,0	Bryter D0-D7 i både grupp 1 och 2.

5.2.2.4 Installation av 4095

Val av kortadress
Bestäm först vilken adress kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
Fig 5.2.2.3 beskriver stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan är numrerade från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker genom att i BASIC skriva OUT &H301,A, där A är kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

5.2.2.5 Programexempel

Exempel 1: Rinnande ljus på 4095

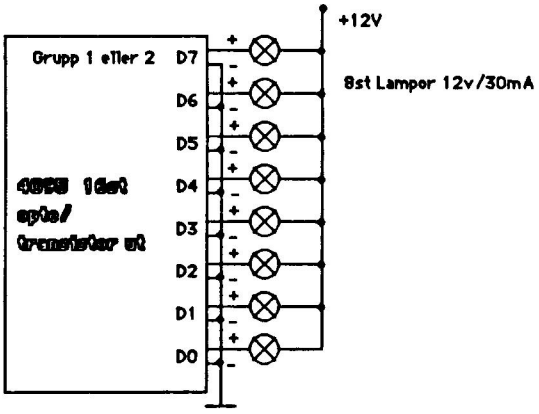


Fig 5.2.2.2 Kopplingsbeskrivning för programexempel "rinnande ljus".

```
10 ' Rinn4095.bas  P-J H  850124
30 ' Rinnande ljus på kort 4095
40 '
50 Z=INP(&H307)
60 OUT &H301,4
70 '
```

```
80 WHILE -1
90   OUT &H300,&H81
100   FOR J=1 TO 100 : NEXT J
110   FOR I=1 TO 7
120     OUT &H300,2ÜI+2Ü(I-1)
130     FOR J=1 TO 100 : NEXT J
140   NEXT I
150 WEND
160 END
1000 '
1010 ' HÄR FÖLJER KOMMENTARER TILL OVANSTÅENDE PROGRAM
1020 '
1030 ' Radnr Kommentar
1040 ' 50    Nollställer alla I/O-kort
1050 ' 60    Väljer kort 4095 som har adress 4
1060 ' 80    Ger en oändlig loop, avbryt med ctrl-break
1070 ' 90    Tänder översta och nedersta dioden
1080 ' 100   Vänteloop
1090 ' 110   Tänder 2 dioder i taget
1100 ' 120   Vänteloop
```

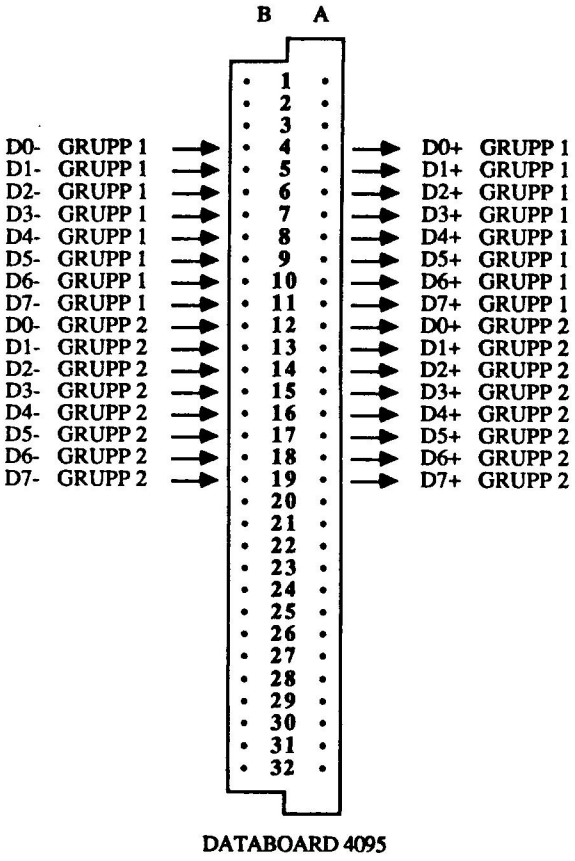


Fig 5.2.2.3 DataBoard 4095, stiftlayout och kontaktnamn

Kapitel 5.3 DataBoard 4085
Digitalt I/O-kort med 32+4 ingångar och 2 utgångar

5.3.1 Specifikationer 4085

Utgångar:	2 TTL
Ingångar:	32+4 TTL

5.3.2 Beskrivning

4085 har 32 ingångar av normal TTL-typ. Ingångarna är ordnade i fyra 8-bitars grupper. Varje grupp har en styringång (STB) för val av lagring i vippor eller direk avläsning av insignalerna. Det finns också fyra interrupt (INT) ingångar som kan användas för att generera avbrott till datorn. Avbrottsrutinen i datorn måste vara skriven i assembler.

De två utgångarna på 4085 är standard TTL-utgångar, som var och en kan driva tic stycken TTL-ingångar.

4085 är speciellt lämpat för att avläsa många digitala insignaler och i system som behöver avbrottshantering. Om de två utgångarna styr en multiplexer som kopplar in 36 insignaler åt gången kan man avläsa 36x4 = 144 stycken insignaler.

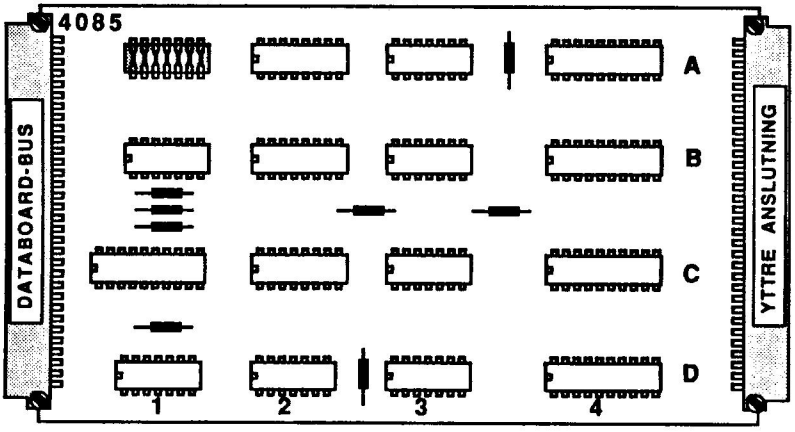


Fig 5.3.1 Digitalt I/O-kort 4085

Styringångarna STB 0-3
4085 har fyra strobe-ingångar (STB), som vardera styr 8 ingångar (en grupp). Med STB-signalen kan t ex en yttre enhet lagra insignaler i en vippa för senare avläsning från IBM PC. Då STB går låg lagras ingångsvärdet i vipporna.
Om STB ligger hög följer vippornas utgångar sina ingångar, dvs de åtta signaler som ligger på ingångarna förpassas direkt in på databussen. Skall STB ej användas för låsning av insignaler kan den bara lämnas öppen så antar den högt läge.

Ingångarna INT FAS 0-3
INT FAS används för att bestämma vilken flank som avbrottet skall ske på. Om man vill ha avbrott på positiv (stigande) flank, dvs när INT-signalen går från låg till hög nivå, skall motsvarande INT FAS-ingång byglas till jord. Vill man däremot ha avbrott på negativ (fallande) flank, dvs när signalen går från hög nivå till låg nivå, skall motsvarande INT FAS-ingång lämnas öppen, eftersom denna ledning är ansluten till +5 V via ett pull-up motstånd på kortet.

5.3.3 Programmering 4085

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H301 kommer av att interfacekortet i IBM PC lägger till värdet &H300 till PORT-adressen på I/O-kortet (300 HEX).
OUT &H300, [Ingrupp+Utg]	Väljer ingrupp för senare läsning med INP-kommandot. [Ingrupp] är ett tal mellan 0 och 3 (bit 0 och bit 1). [Utg] styr utgångarna A och B och är en kombination av värdet för bit 2 och bit 3 (värd 4 för A och 8 för B). Kan anta värdet 0,4,8 eller 12. Exempel: 200 OUT &H300,0+0. Väljer grupp 0 för läsning och nollställer utgång A och B.

	Exempel: 210 OUT &H300,3+8. Väljer grupp 3 för läsning. Nollställer utgång A och ettställer utgång B. Exempel: 230 OUT &H300,2+12. Väljer grupp 2 för läsning. Ettställer utgång A och utgång B.
INP(&H300)	Avläsning av de 8 bitarna (ingångarna) från tidigare vald grupp på kortet. Värdet är ett heltal mellan 0 och 255 beroende på insignalen på de olika ingångarna. Exempel: 110 V=INP(&H300)
OUT &H305, [Interrupt]	Val av ingång för interrupt. Bit 0 styr interrupt 0. Bit 1 styr interrupt 1 o s v. En etta på respektive interruptingång medger interrupt. En nolla stänger av motsvarande interrupt. Exempel: 240 OUT &H305,4. Interrupt bortkopplad för samtliga ingångar utom interrupt 2 (med bitvärdet 4).
INP (&H301)	Avläsning av status hos de fyra interrupt-ingångarna. Bit 0 motsvarar interrupt 0. Bit 1 motsvarar interrupt 1 osv. D4-D7 är ettställda vid läsning och skall ignoreras. Interrupt föreligger om motsvarande inläst värde är en nolla. Exempel 270 IF (INP(&H301) AND 15) = 8 THEN PRINT "INTERRUPT 3 ÄR ETTSTÄLLD"

5.3.4 Installation av 4085

Val av kortadress
Bestäm först vilken adress kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
Fig 5.3.3 beskriver stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan är numrerade från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress

Kontroll av adresspluggens bygling sker enklast genom att i BASIC skriva OUT &H301,A, där A är kortets avsedda adress. Om adresspluggen byglats riktigt, skal lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

5.3.5 Programexempel

När man använder 4085 och skall byta insignaler får man inte glömma att utgångarna ändras med samma kommando. För att undvika den sortens fel bör man ha aktuellt utgångsvärde i en variabel t ex "Utg" och en variabel för val av ingångar t ex "Ingrupp". När det då är dags att välja en ny ingrupp skriver man:

50 OUT &H300,<Utg+Ingrupp>

Där "Utg" får anta värdena 0,4,8,12 och "Ingrupp" får innehålla värdena 0,1,2,3.

När man läser interruptingångarna får man ett värde som motsvarar värdet på interruptingångarna (bit 0-3) plus talet 240. 240 kommer från de övriga bitarna (4-7) i samma byte, som alla är ettställda.

Exempel 1

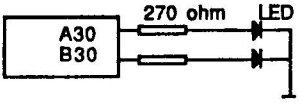


Fig 5.3.2 Koppling till exempel 1

Styr två utsignaler på 4085

```
20 ' Styr4085.bas  P-J H  850212
30 ' STYR TVÅ UTSIGNALER PÅ 4085
40 '
50 Z=INP(&H307) ' Nollställ
60 OUT &H301,8 ' Kortval
70 '
80 CLS
90 PRINT "Detta program kan ställa om två utsignaler, A
   och B.":PRINT
100 WHILE -1
110   PRINT "A står nu som" A%
120   PRINT "B står nu som" B% : PRINT
130   '
140   INPUT "Sätt A till? (1/0) ",A%
150   IF A%<0 OR A%>1 THEN 140
```

```
160    '
170    INPUT "Sätt B till? (1/0) ",B%
180    IF B%<0 OR B%>1 THEN 170
190    '
200    OUT &H300,4*A%+8*B% ' Ställ bit 2 och 3, övriga
   bitar alltid=0.
210 WEND
220 END
1000 '
1010 ' Här följer kommentarer till programmet.
1020 '
1030 ' Rad Kommentar
1040 ' 50  Nollställ korten.
1050 ' 60  Välj kort 4085 som här har adress 8.
1060 ' 100 Oändlig loop börjar.
1070 ' 110-120 Skriv ut hur portarna står nu.
1080 ' 140 Mata in nytt läge för port A.
1090 ' 170 Mata in nytt läge för port B.
1100 ' 200 Sätt bit 2 och 3 till inmatat värde och styr
   portarna.
```

Exempel 2

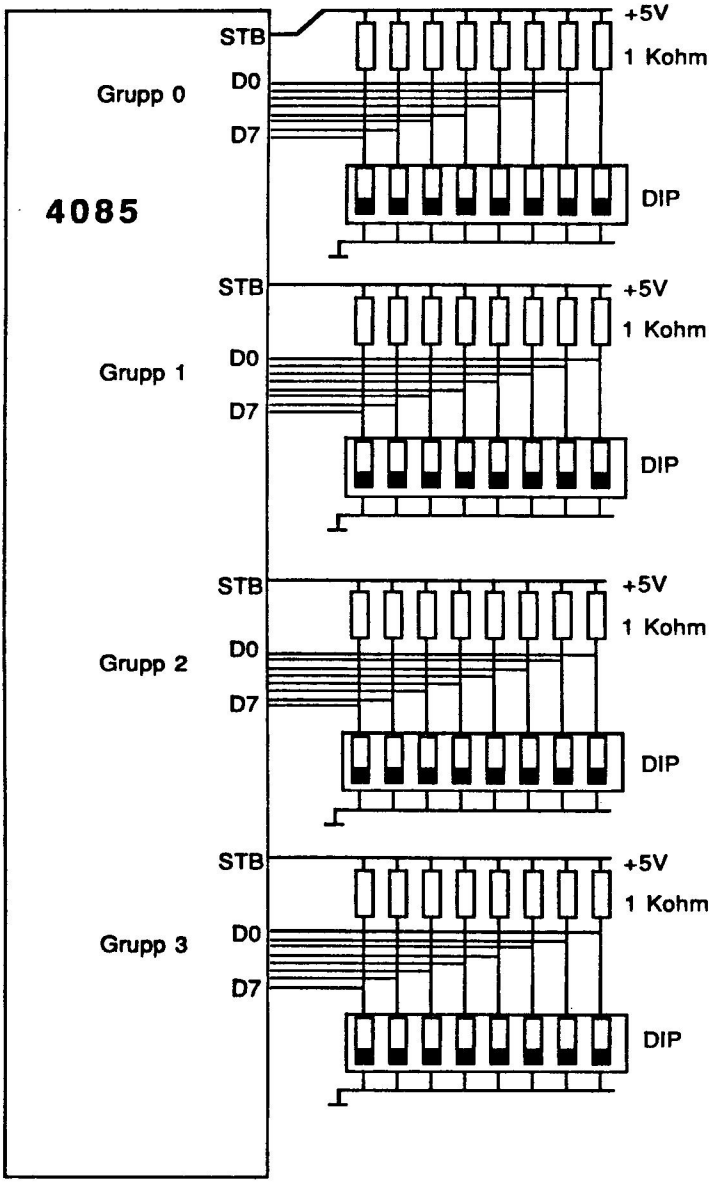

4085
Grupp 0
Grupp 1
Grupp 2
Grupp 3
STB
D0
D7
+5V
1 Kohm
DIP

Fig 5.3.3 Koppling till exempel 2

Avläsning av 32 ingångar med 4085

```
20 ' Las4085.bas  P-J H  850212
30 ' AVLÄSNING AV 32 INGÅNGAR MED 4085
40 '
50 Z=INP(&H307) ' Nollställ
60 OUT &H301,8 ' Kortval
70 '
80 CLS : PRINT "Programmet visar avläst innehåll i
   angiven grupp, binärt, hexadec och decimalt."
90 '
100 WHILE GRUPP%<>9
110   PRINT
120   INPUT "Ange grupp du vill läsa (0-3, 9=Slut)?
   ",GRUPP%
130   IF GRUPP%=9 THEN 260
140   IF GRUPP%>3 OR GRUPP%<0 THEN 110
150   OUT &H300,GRUPP% ' Välj grupp
160   '
170   IN%=INP(&H300) ' Läs indata
180   PRINT "Grupp" GRUPP% "binärt=" TAB(20);
190   FOR L%=0 TO 7 ' Presentera binärt 8 bitar
200     PRINT (IN% AND 2ÜL%)/2ÜL%;
210   NEXT L%
220   '
230   PRINT : PRINT "Hexadecimalt=" TAB(21) HEX$(IN%)
240   PRINT "Decimalt=" TAB(20) IN%
250   '
260 WEND
270 END
1000 '
1010 ' HÄR FÖLJER KOMMENTARER TILL PROGRAMMET
1020 '
1030 ' 50  Nollställ korten.
1040 ' 60  Välj kort 4085 som här har adress 8.
1050 ' 100 Håll på tills inmatad grupp=9.
1060 ' 120 Mata in ett gruppnr.
1070 ' 150 Väljer ut angiven grupp. Bit 0 och 1
   användes.
1080 ' 170 Läs de åtta ingångarna till In%.
1090 ' 180-210 Presentera binärt.
1100 ' 190 Vill du ha bitarna presenterade i omvänd
```

```
   ordning skriv "FOR L%=7 TO 0 STEP -1".
1110 ' 230 Presentera hexadecimalt.
1120 ' 240 Presentera decimalt.
```

Exempel 3

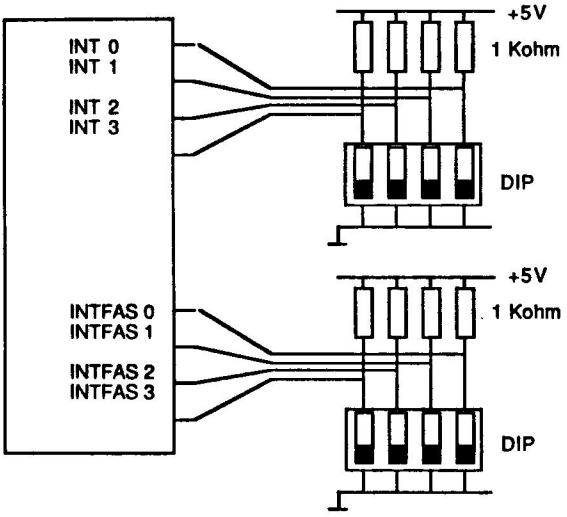


Fig 5.3.4 Koppling till exempel 3

Avläsning av de fyra interruptingångarna med kort 4085

```
10 ' Int4085.bas  P-J H 850212
20 '
30 ' AVLÄSNING AV DE FYRA INTERRUPT-INGÅNGARNA MED KORT
   4085
40 '
50 Z=INP(&H307) ' Nollställ
60 OUT &H301,8 ' Kortval
70 '
80 CLS : PRINT "Programmet visar interrupt-ingångarna
   binärt och decimalt."
90 '
100 WHILE I$<>"S"
110    '
120    IN%=INP(&H301)-&HF0 ' Läs interrupt-ing., dra
   bort de högsta fyra bitarna.
```

```
130    PRINT : PRINT  "Binärt=" TAB(20);
140    FOR L%=3 TO 0 STEP -1
150      PRINT (IN% AND 2ÜL%)/2ÜL%;
160    NEXT L%
170    '
180    PRINT
190    PRINT "Decimalt=" TAB(20) IN%
200    '
210    PRINT
220    INPUT "Return=mer, S=Slut? ",I$
230    '
240 WEND
250 END
1000 '
1010 ' HÄR FÖLJER KOMMENTARER TILL PROGRAMMET
1020 '
1030 ' 50  Nollställ korten.
1040 ' 60  Välj kort 4085 som här har adress 8.
1050 ' 100 Håll på tills ett S är inmatat.
1080 ' 120 Läs de 8 ingångarna till In%, endast de 4
   första är intressanta.
1090 ' 130-160 Presentera binärt.
1120 ' 190 Presentera decimalt.
1130 ' 220 Fråga om vi skall fortsätta.
```

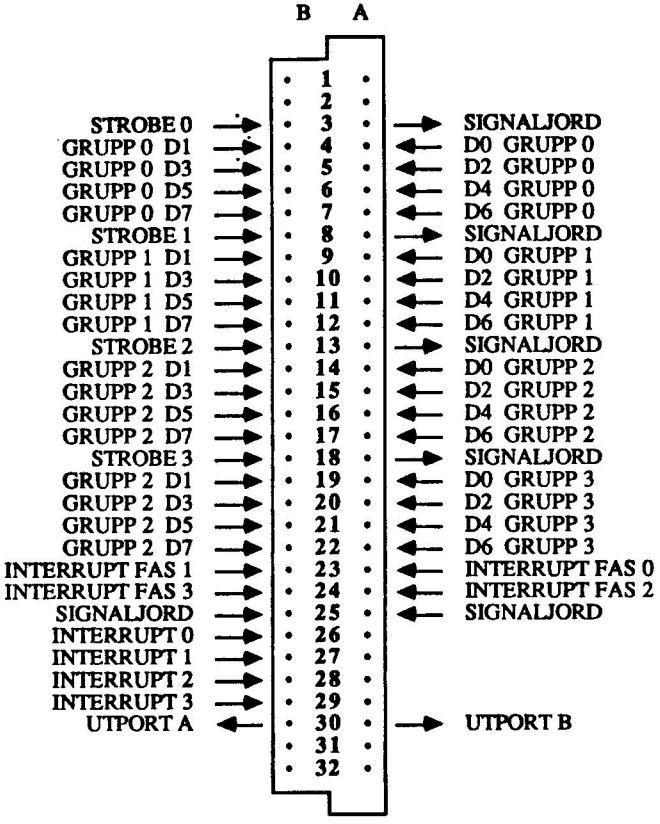


Fig 5.3.3 DataBoard 4085, stiftlayout och kontaktnamn

Kapitel 5.4 DataBoard 4103
Reläkort med 16 växlande kontakter

5.4.1 Specifikationer 4103

Kontaktspänning:	Max 110 Volt
Brytström:	Max 1 A
Bryteffekt:	Max 20 Watt
Drivspänning:	5 Volt
Från/tillslagstid:	1 ms

5.4.2 Beskrivning

Reläerna är arrangerade i två grupper om åtta i varje. Reläerna i de två grupperna kan styras individuellt med två OUT-instruktiner. Om så önskas kan den interna drivspänningen (+5 V) till reläerna lätt ändras till en yttre drivspänning. Detta sker genom att ändra läget på en bygel. 4103 är program och anslutningskompatibelt, med avseende på till- respektive frånslag, med kort 4095. 4103 är program och anslutningskompatibelt med 4005, då med tanke på utgångarna. Reläerna på kortet är av typen RH-5, som klarar 100 V max som kontaktspänning och en maximal kontaktström av 1 A. Reläerna har en bryteffekt på 20 W maximalt med en omslagstid på 1 mS.

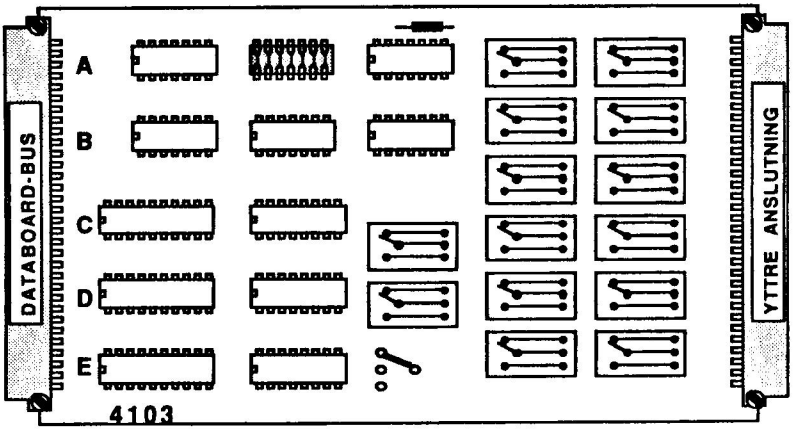


Fig 5.4.1 Reläkort 4103

5.4.3 Programmering av 4103

KOMMANDO	FUNKTION
INP(&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Öppnar alla 16 reläer. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar 4103-kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H301 kommer av att interfacekortet i IBM PC lägger till värdet &H300 till PORT-adressen på I/O-kortet (300 HEX).
OUT &H300, [DATA]	Lägger ut värdet [DATA] (0-255) på de 8 RELÄERNA 0-7 i grupp1 på kortet. Exempel: 240 OUT &H300,35 lägger ut data enligt följande mall: D0=1,D1=1,D2=0,D3=0,D4=1,D5=0,D6=0,D7=0 1 2 0 0 32 0 0 0 =35 dvs gör att RELÄ 0,1 och 4 blir draget och ger anslutning till kontakt 1 på respektive relä. Resterande reläer på grupp 1 bryter.
OUT &H302,[DATA]	Samma som ovanstående men avser relä 0 till 7 i grupp 2.
OUT &H304,0	Samtliga 16 reläer bryter och ger slutning till kontakt 0.

5.4.4 Installation av 4103

Val av kortadress
Bestäm först vilken adress kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
Fig 5.4.3 beskriver stiftlayout och signalnamn på I/O-sidan. ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan är numrerade från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker genom att i BASIC skriva:
OUT &H301,A, där A är kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

5.4.5 Programexempel

Exempel 1

Rinnande ljus på reläkort 4103

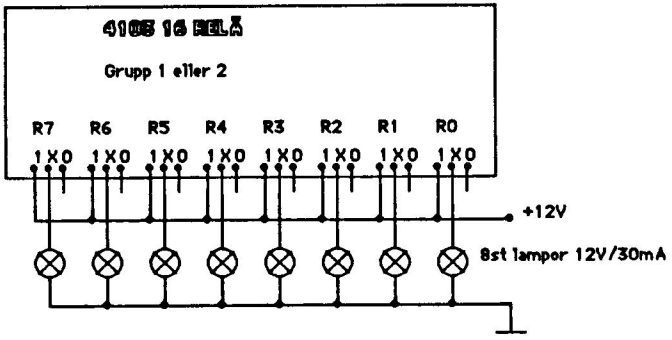


Fig 5.4.2 koppling till exempel 1

```
20 ' Rinn4103.bas  P-J H  850208
30 ' Rinnande ljus på reläkort 4103
40 '
50 Z=INP(&H307) ' Nollställ
60 OUT &H301,4 ' Kortval
70 '
80 WHILE -1
90   OUT &H300,&H81 ' Tänder överst och underst
```

```
100   FOR J=1 TO 100 : NEXT J
110   FOR I=1 TO 7
120     OUT &H300,2ÜI+2Ü(I-1) ' Tänder två i rad
130     FOR J=1 TO 100 : NEXT J
140   NEXT I
150 WEND
160 END
1000 '
1010 ' HÄR FÖLJER KOMMENTARER TILL OVANSTÅENDE PROGRAM
1020 '
1030 ' Radnr Kommentar
1040 ' 50    Nollställer alla I/O-kort
1050 ' 60    Väljer kort 4103 som här har adress 4
1060 ' 80    Ger en oändlig loop, avbryt med ctrl-break
1070 ' 90    Tänder översta och nedersta dioden
1080 ' 100   Vänteloop
1090 ' 110   Tänder 2 dioder i taget
1100 ' 120   Vänteloop
```

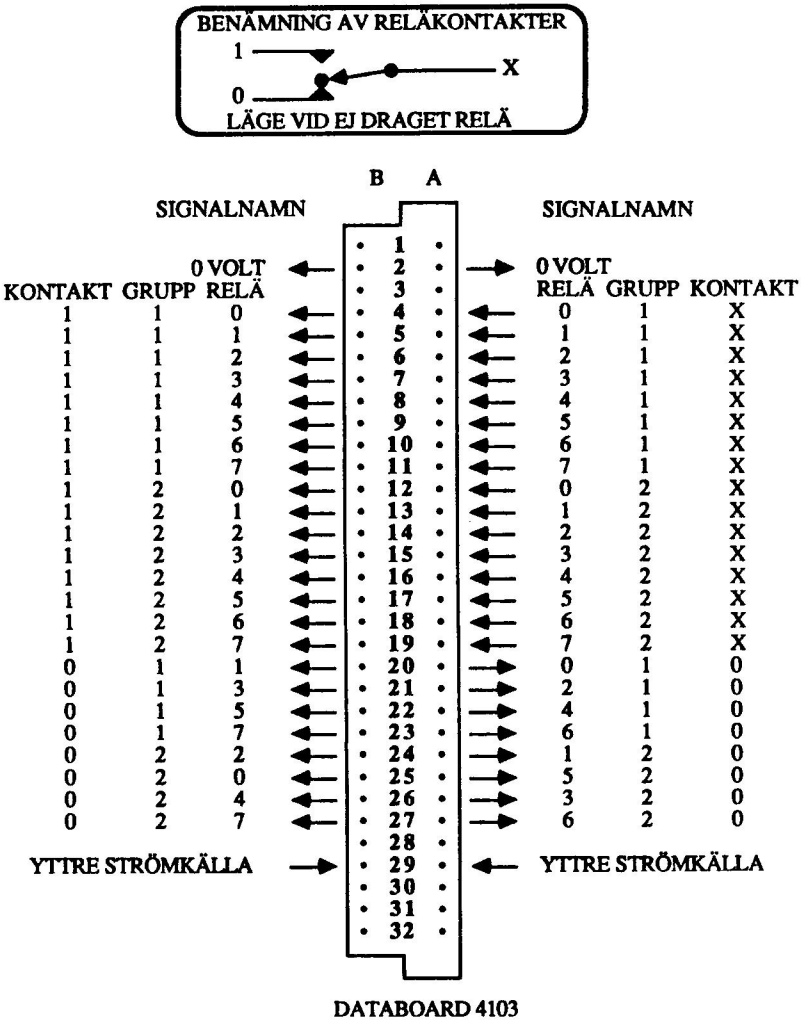


Fig 5.4.3 DataBoard 4103, stiftlayout och kontaktnamn

Kapitel 5.5 DataBoard 4013
Mix I/O, TTL/OPTO

5.5.1 Specifikationer 4013

Utgångar	TTL, 16 stycken
Utgångar (Strobe)	TTL, 2 stycken
Ingångar	TTL, 12 stycken
Ingångar(Strobe)	TTL, 2 stycken
Ingångar	OPTO, 4 stycken

5.5.2 Beskrivning av 4013

De 16 TTL-utgångarna är av tri-state-typ och buffrade. 8 av utgångarna kan riktas till lysdioder på 4013-kortet. 16 alternativt 8 av utgångarna kan aktiveras samtidigt. De 2 TTL strobutgångarna kan generera >300 nanosekunders negativ puls.

Samtliga 12 ingångar kan byglas alternativt till I/O-kontakten eller internt på 4013-kortet.

4 Ingångar vilka även kan väljas individuellt som:
A: Optoisolerade differentiella ingångar +12 volt (eller 24 volt med ett 1-Kohm externt motstånd.
B: TTL-ingångar
C: Från byglingspluggar på 4013.

Interrupt kan väljas separat för de 4 ingångarna. 2 stycken interrupt nivåväljar insignaler. För de två andra interruptingångarna väljs nivå genom ett programkommando.

Alla ingångarna har "pull-up" till logisk nivå "ett", om de inte ansluts. De 16 bygelpluggarna på insignal-linjerna kan även avläsas av externa enheter på TTL-ingångsstiften.

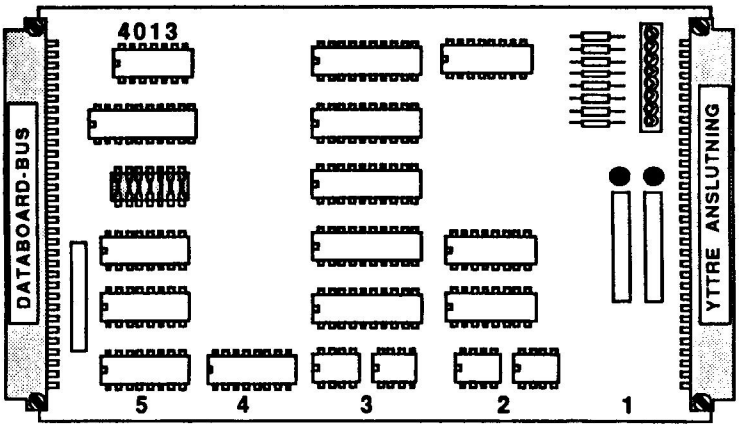


Fig 5.5.1 Mix I/O-kort 4013

5.5.3 Programmering av 4013

KOMMANDO	FUNKTION
INP(&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307). Nollställer kommandoregistret på 4013, vilket stänger av interrupt och gör de 16 utgångarna hög-impediva.
OUT &H301,[KORTADRESS]	Aktiverar 4013-kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H301 kommer av att interfacekortet i IBM PC lägger till värdet &H300 till PORT-adressen på I/O-kortet (300 HEX).

OUT &H305,[DATA]	Lagrar en komando-byte i 4013 kommando-register. Bitarna 0,1 styr tristateutgångarnas läge. Bitarna 2,3 anger "interrupt-nivå" för insignal 7 respektive 6 i grupp D. Bitarna 4,5,6,7 väljer interrupt på eller av för insignalerna 4,5,6,7 i grupp D.
OUT &H304,[DATA]	Sänder ut en negativ strob-puls på I/O-stift 14A. Data ignoreras. Exempel: 130 OUT &H304,0
OUT &H303,[DATA]	Sänder ut en negativ strob-puls på I/O-stift 14B. Data ignoreras. Exempel: 140 OUT &H304,0
OUT &H302,[DATA]	Lagrar [DATA] till buffer för utsignal grupp B. [DATA] läggs på utgångarna om bit 0 i kommandoregistret är logisk 1. (Lysdioderna kan styras av utsignalerna).
OUT &H300,[DATA]	Lagrar [DATA] till buffer för utsignal grupp A. [DATA] läggs på utgångarna om bit 1 i kommandoregistret är logisk 1. (Lysdioderna kan styras av utsignalerna).
INP (&H300)	Läser bit 0-7 från insignal grupp C, vilka antinger är TTL-ingångar eller byglingar Exempel: 170 Variabel=INP (&H300).
INP (&H301)	Läser bit 0-7 från insignal grupp D. Avser antingen Optoingångar, TTLingångar eller intern bygling på 4013-kortet. Databit 4,5,6,7 på port D kan generera avbrott (interrupt). Exempel: 190 Variabel=INP (&H301).

4013 kommandoregister

OUT &H305, [DATA] lagrar 8 bitar i kommandoregistret.
Funktionen för varje bit är:

Bit	Värde	Funktion
0		Styrning av tristate på utgång, grupp B
	0	Utgångarna är högimpediva
	1	Data från buffrarna läggs till utgångarna
1		**Styrning av tristate på utgång, grupp A**
	0	Utgångarna är högimpediva
	1	Data från buffrarna läggs till utgångarna
2		**Interrupt-signalnivå för bit 7, ingrupp D**
	0	Interrupt genereras då insignalen är hög ("1"). Ingång som inte är ansluten ligger alltid hög.
	1	Interrupt genereras då insignalen är låg("0"). vilket inträffar bl a då optokopplaren drivs aktivt.
3		**Interrupt-signalnivå för bit 6, ingrupp D**
	0	Interrupt genereras då insignalen går hög. Ingång som inte är ansluten ligger alltid hög.
	1	Interrupt genereras då insignalen går låg. vilket inträffar bl a då optokopplaren drivs aktivt.
4,5,6,7		**Bitarna 4,5,6,7 i kommandoregistret väljer av eller påslagning av interrupt från grupp D, bitarna 4,5,6,7.**
	1	Slå på interrupt
	0	Stäng av interrupt

5.5.4 Installation

Val av kortadress
Bestäm först vilken adress kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Utgång grupp B och kortets lysdioder
För att styra de 8 lysdioderna på kortet med utsignalerna grupp B, skall motsvarande bygling ske på position 4A på kortet. För signaler byglade till lysdiod, skall ingen yttre anslutning ske. Utgångsnivå "0" tänder lysdioderna.

Bygel 1= bit 0, bygel 2= bit 1 .....bygel 8=bit 7.

Ingång grupp C
Åtta stycken TTL ingångar. Motsvarande bygel i kortposition 4D måste vara öppen då yttre signal skall läsas. Då byglingspluggen på kortet skall läsas, skall yttre signal inte anslutas. En öppen bygel läses som logisk "1".

Bygel 1= bit 0, bygel 2= bit 1 .....bygel 8=bit 7.

Ingång grupp D
Åtta ingångar från individuellt valbara källor, vilka även kan generera interrupt. Då externa signaler skall läsas måste motsvarande bygel vara öppen. Insignaler genom optokopplare ignoreras om motsvarande TTL ingång är låg "0" eller om motsvarande bygel är sluten. Vid läsning från opto-isolerade differentiella ingångarna, läses aktiv signal (+24 volt driver ström) som logiknivå "0".

För bitarna 0,1,2 eller 3, välj insignal från extern TTL eller från byglingsplugg.

För bitarna 4,5,6 eller 7 välj insignal från extern TTL, byglingsplugg eller Optokopplade ingångar. Interruptvillkor sätts genom att sätta motsvarande bit i kommandoregistret på 4013 till "1".

Om interrupt är påkopplat för insignal 4 och 5 i grupp D, väljs den signalnivå som skall generera interrupt, genom att insignalen jämförs med motsvarande "interrupt nivå-signal" (I/O-stift 15A eller 15B). Om dessa signaler är olika, genereras interrupt. Obs att en öppen ingång motsvarar logisk "1" som insignal, vilket betyder att en negativ insignal skall generera interrupt. För bitarna 6 och 7 i grupp D, anges motsvarande "interruptnivå" av bitarna 3 resp 2 i kommandoregistret

Anslutning av yttre enheter
Fig 5.5.3 beskriver stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan är numrerade från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker genom att i BASIC skriva OUT &H301,A, där A är kortets avsedda adress. Om adresspluggen byglats riktigt, skal lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

5.5.5 Programexempel

Exempel 1

Läser byglingarna grupp C och D och presenterar resultatet binärt

Till exemplet finns ingen yttre koppling. Programmet använder de på kortet placerade lysdioderna och byglingarna.

```
10 ' Swir4013.bas  P-J H 850502
20 '
30 ' Läser byglingarna grupp C och D på I/O-kort 4013
   och presenterar binärt.
40 '
100 '
110 ' FUNKTION FÖR KONVERTERING AV FLYTTAL TILL HELTAL
120 '
130 ' In  Tal! = Ett tal som skall omvandlas, OBS
   32768<Tal!<65536
140 ' Ut  Ett tal i intervallet -32768 - 0
150 '
160 DEF FNHELTAL%(TAL)=TAL-65536!
200 '
210 ' HUVUDPROGRAMMET
220 '
230 Z=INP(&H307)
240 OUT &H301,4 ' Kortadress
250 CLS
260 PRINT "Byglingarna är inställda enligt följande,
   sluta med S." : PRINT
270 PRINT "Switch    1 2 3 4 5 6 7 8"
280 '
290 WHILE I$<>"S"
300   '
310   GRUPPC$="" : GRUPPD$=""
320   FOR PORT%=0 TO 7
330     GRUPP%=0 ' Ger inläsning av grupp C
```

```
220 WHILE NOT DOOMSDAY ' Oändlig loop
225   '
230   FOR I%=1 TO R% ' Ett varv för varje utvalt kort.
240     FOR KANAL%=0 TO 31 STEP 2 ' Ett varv per två
   kanaler.
250       FOR K1%=0 TO 1 ' För att fixa uppspaltningen.
260         OUT &H301,KORT%(I%) ' Välj kortadress.
270         OUT &H302,KANAL%+K1%+32+64 ' Väljer
   kanal,-5 till +5V, förstärkning =1.
280         OUT &H303,0 ' 12 bitars upplösning.
290         '
300         IF INP(&H301)>=128 THEN 300 ' Vänta tills
   omvandling är klar.
310         '
320         MEST%=(INP(&H301) AND 15) ' Läs de fyra
   mest signifikanta bitarna.
330         MINST%=INP(&H300) ' Läs de åtta minst
   signifikanta bitarna.
340         IN%=MEST%*256+MINST% ' Lägg ihop bitarna
   till ett tal.
350         VOLT=INT(1000*IN%/409.6)/1000 ' Räkna om
   till volt.
360         '
370         LOCATE KANAL%/2+1,0+40*K1%+1
380         PRINT VOLT " volt på kanal " KANAL%+K1% "
     "
390         LOCATE 20,30 : PRINT "Nu mäts kanal "
   KANAL%+K1%+1 "    "
400         LOCATE 22,30 : PRINT "På kort nr "
   KORT%(I%)
410       NEXT K1%
420     NEXT KANAL%
430   NEXT I%
440 WEND
450 END
```

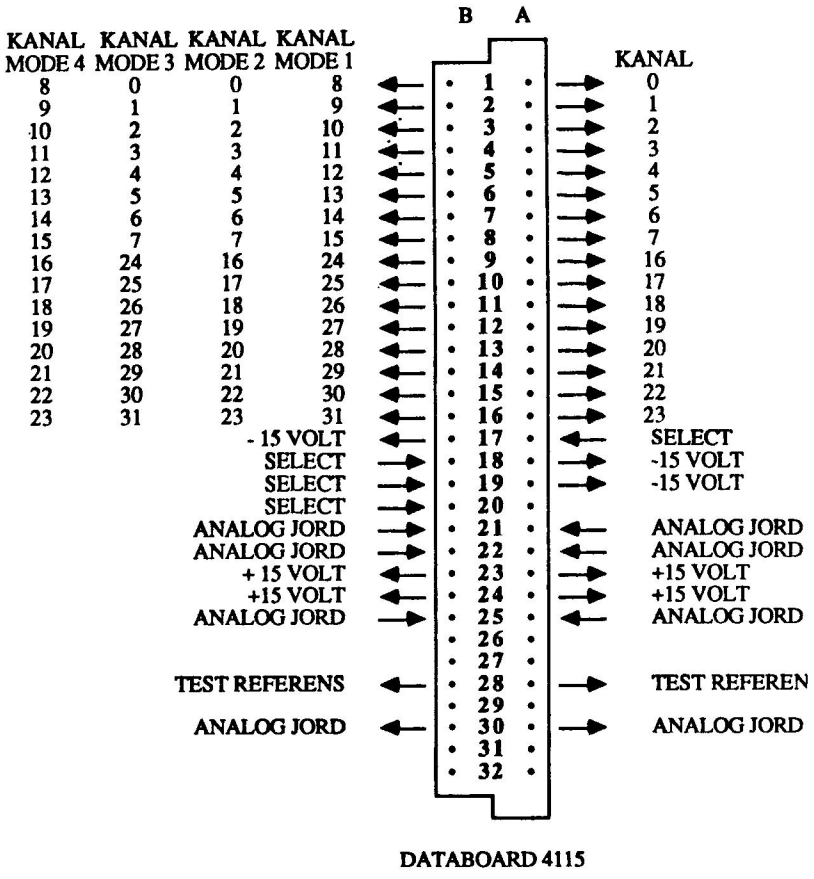


Fig 6.3.3 DataBoard 4115, stiftlayout och kontaktnamn

Kapitel 7 DataBoard för Motorstyrning

7.1 DataBoard 4002, Motorstyrkort

7.2.1 DataBoard 4066, Stegmotorkontrollkort

7.2.2 DataBoard 5085, Drivsteg till stegmotor styrkort

7.3 DataBoard 4014, Drivkort för DC-motorer

Kapitel 7.1 DataBoard 4002
Motorstyrkort

7.1.1 Specifikationer 4002

Matningsspänning:	+5V, 600 mA +12V, 20 mA -12V, 10 mA
Ingångar:	Pulsgivare 5V diff Statusgivare 5V Optoisolerat
Utgångar:	Motorstyrning, 1 kanal, analog +/- 10 mV till +/- 10V, Motorstyrningsfrekvens 140 Hz-10 kHz (Option 25 Hz - 100 kHz), TTL Reläväxling för analogutgång.

7.1.2 Beskrivning 4002

För detaljerad beskrivning av kortet hänvisas originalmanual som medföljer kortet. Motorstyrningskortet 4002 är ett intelligent motorstyrkort (I/O-kort), designat för att arbeta i en industriell miljö och för att uppfylla industriella krav. Maskinvaran på kortet 4002 kan delas in i fem delar, var och en med sin speciella funktion: Ett kommandointerface som sköter den asynkrona kommunikationen mot DataBoard-bussen (IBM PC). En räknardel som beräknar ärvärdet utifrån information från de yttre inkrementpulsgivarna. En analogdel som skickar ut börvärdessignalen dels som en frekvens och dels som en proportionell spänning. En referensdel som synkroniserar mot de externa villkoren. Slutligen en watchdog som övervakar att processorn inte har låst sig.

Kortet innehåller intelligens för styrning och övervakning. Detta implementeras med hjälp av en enchips-processor, INTEL 8748. Med 4002 kan man styra en utrustning via en analog utgång. Regleområde: +/- 10 mV - +/- 10 Volt. Positionsinformation till 4002 erhålls via pulsgivare eller inkrementalgivare.

Riktningsdetektorer och räknare finns på kortet. 4002 kan antingen styra med maxfart eller med ramper för acceleration och retardation. Direkt till kortet ansluts givare för ändstopp och nödstopp vilka kontrolleras direkt av processorn på kortet.

Kortet kan användas i ett brett fält av applikationer. Allt från beräkningskontroll till alla typer av positionering och DC-styrning där en värd-dator, typ IBM PC handhar övervakning av processen. Kortet kontrollerar positionering, acceleration, inbromsning, fastlåsning, ändlägen etc.

Exempel på användningsområden:
Styrning av transfervagnar, traverser och pallastnings-robotar.
Rörliga montage-och testjiggar.

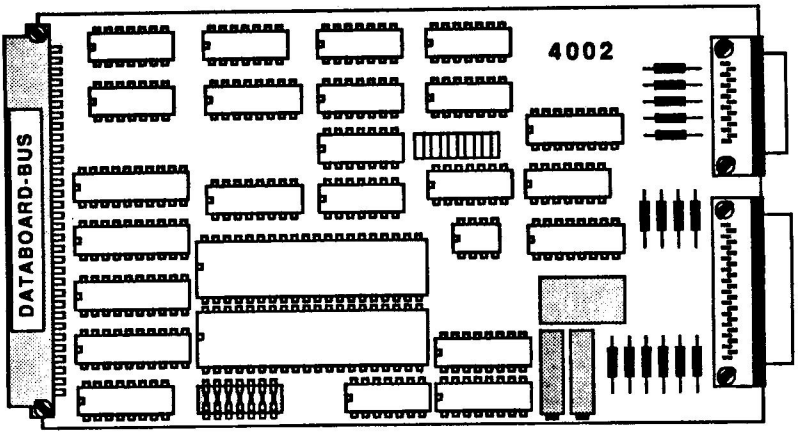


Fig 7.1.1 DataBoard Motorstyrkort 4002

7.1.3 Programmering av 4002

Programfunktioner
Programvaran i processorn på 4002-kortet handhar ett antal funktioner vilka nås vi enkla kommandon från IBM PC. Exempel på dessa kommandon är:

Relativ positionering, absolut positionering, sätt hemmaläge, hastighet, acceleration, retardation, start, stopp, status, hysteres och låsning.
Reglering av hastighet sker i 1024 steg med hastighetskommando. När acceleration och retardationsberäkningar görs på kortet regleras hastigheten i 24+24 intervall.

Kommunikation mellan PC4680 och DataBoard 4002 kan ske via följande standardportar: STAT (nås via INP &H301), C2 (nås via OUT &H303), C3 (nås via OUT &H304), C4 (nås via OUT &H305), UTP (nås via OUT &H300), INP (nås via INP &H300).

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar 4002-kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H301 kommer av att interfacekortet i IBM PC lägger till värdet 300 till portadressen på

I/O-kortet (300 HEX).

INP (&H301)

Läser in data från STAT-porten. Fem av flaggorna i STAT-porten användes för att avkoda den aktuella statusen på ändläges, nödstopp, referens och nollsignalerna. Resterande 3 flaggor ger information från processorn på 4002-kortet. Ready indikerar att 4002 är klar att ta emot data på någon av portarna DATA, C2 ellr C4. Användaren måste alltid vänta på att Ready flaggan är satt, för att vara helt säker på att 4002 reagerar på ett korrekt sätt på de olika portarna. Interrupt-flaggan visar att ett interrupt är aktivt eller "pending". Event enable flaggan visar att Event enable ordet har uppfyllts.

◊◊◊◊◊◊◊√ Interrupt "pending"
◊◊◊◊◊◊√◊ Event enable flagga "pending"
◊◊◊◊√◊◊ Nollpulser
◊◊◊◊√◊◊◊
◊◊◊√◊◊◊◊ Nödstopp
◊◊√◊◊◊◊◊ Ändstopp 1
◊√◊◊◊◊◊◊ Ändstopp 2
√◊◊◊◊◊◊◊ Enheten klar (Ready)
att ta emot kommando

OUT &H303,[KOMMANDO] Sänd KOMMANDO för att styra 4002 till att utföra önskad handling. Argumenten till kommandot sänds över via:
OUT &H300 [DATA]

OUT &H300 [DATA] Förmedla argument till kommandona tex för att sätta upp ny position.

OUT &H304 [DATA] Utför oavsett innehållet i DATA en hårdvaru-reset av 4002-kortet. Både processorn och all logik på kortet återställes.

OUT &H305 [DATA] Gör det möjligt att använda 4002 i interruptmod och att detektera vissa händelser.

Aktivering (Event enabling)
För att de olika händelserna skall kunna aktiveras eller inaktiveras, måste ett flaggmönster sändas med OUT &H305 [DATA]. Endast de händelser kan inträffa vars flagga är aktiv i Event enabling ordet.
◊◊◊◊◊◊◊√ Uppnådd position
◊◊◊◊◊◊√◊ Nollpuls
◊◊◊◊◊√◊◊ Referenspunkt
◊◊◊◊√◊◊◊ Reserverad
◊◊◊√◊◊◊◊ Reserverad
◊◊√◊◊◊◊◊ Ändstopp
√√◊◊◊◊◊◊ Interrupt

EVENT-ordet kan läsas via DATA-porten (INP&H300) efter att OUT&H305 har utfärdats och efter det att den yttre enheten har indikerat på STAT-porten (Bit 1) att den är klar. De sex minst signifikanta flaggorna i EVENT-ordet är reserverade för olika händelser och de har följande flaggmönster.
◊◊◊◊◊◊◊√ Position uppnådd eller passerad
◊◊◊◊◊◊√◊ Nollpuls detekterad
◊◊◊◊◊√◊◊ Referenspunkt detekterad
◊◊◊◊√◊◊◊ Reserverad
◊◊◊√◊◊◊◊ Reserverad
◊◊√◊◊◊◊◊ Ändstopp detekterad
◊√◊◊◊◊◊◊ Reserverad

Den mest signifikanta flaggan används till att detektera om kortet har blivit återställt (reset). Flaggan nollställs med "Watch out"-kommandot.

√◊◊◊◊◊◊◊ "Watch out"-bit, satt vid reset.

Interrupt hantering

I en del applikationer är det för tidskrävande att arbeta med Ready och Eventflaggorna i STAT-porten. Därför erbjuder 4002 användaren en Interrupt facilitet. För att aktivera ett interrupt måste användaren sätta de två mest signifikanta flaggorna i EVENT enable ordet och skriva ut dessa.

Interrupt flaggmönster

◊√◊◊◊◊◊ Interrupt på Ready
√◊◊◊◊◊◊ Interrupt på Event (händelse)

Beskrivning av tillgängliga kommandon
Nedan följer en beskrivning över de kommadon som skickas till 4002 via Kommando-porten. För varje kommando anges koden (ASCII-värdet) och eventuella argument att söndas till DATA-porten. Exakt beskrivning av varje kommando finns i bruksanvisningen till kortet.

Kommando	ASCII	DATA	DATA	DATA (hög byte)
Athome	65	-	-	-
Deviation	68	√	√	√
Go	71	-	-	-
Lock	76	-	-	-
Manual	77	-	-	-
Number	78	√	√	√
On regulation	79	-	-	-
Position	80	√	√	√
Quit	81	-	-	-
Rate	82	√	√	-
Slope	83	√ upp till 240 bytes kan definiera kurvan		
Version	86	-	-	-
Watch Out	87	-	-	-

7.1.4 Installation av 4002

Val av kortadress
Bestäm först vilken adress 4002-kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
För detaljerad beskrivning av inkoppling rekommenderar vi den speciella manual som medföljer 4002-kortet vid leverans. Tänk på att skydda anslutningarna mot magnetiska, elektriska och jordstörningar.

Fig 7.1.3 beskriver stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Se även

kapitel 3.2.4 för anslutning av yttre enheter.

Räkneingångarna
För att räkna yttre händelser finns två principiella arbetsmetoder.

Metod 1: Pulsräkning med två ingångar, en för att räkna upp och en för att räkna ner.

Metod 2: Inkremental räkning med hjälp av inkrementalpulsgivare med faldning (multiplicering) 1, 2, 4.

Räknaren kan nollställas med två olika ingångar; den differentiella ingången ZERO och den OPTO-skilda ingången REF.

Analogutgångarna
Kortet ger förflyttningsinformation både som en analogspänning (ANALOG OUT) och en digital frekvenssignal (FOUT) samt en rörelseriktning (SIGN).

Referensingångar
Det finns fyra statiska ingångar på motorstyrningskortet:

REF - REFerenspunkt för nollställning
EMS -EMergency-Stop, nödstopp
ESL - End Stop Left, vänster ändläge
ESR - End Stop Right, höger ändläge

Watchdog
En övervakninsfunktion som kräver att 8748:an (processorn på 4002-kortet) ingriper minst varje sekund. I annat fall återstartas programmet och analogutgånger faller till noll och enheten blir åter redo för kommandon.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker enklast genom att i BASIC skriva OUT &H301,A, där A är 4002-kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

Exempel på inkoppling av ett servosystem till 4002
För att åskådliggöra ett arbetande servo har nedanstående uppkoppling gjorts.

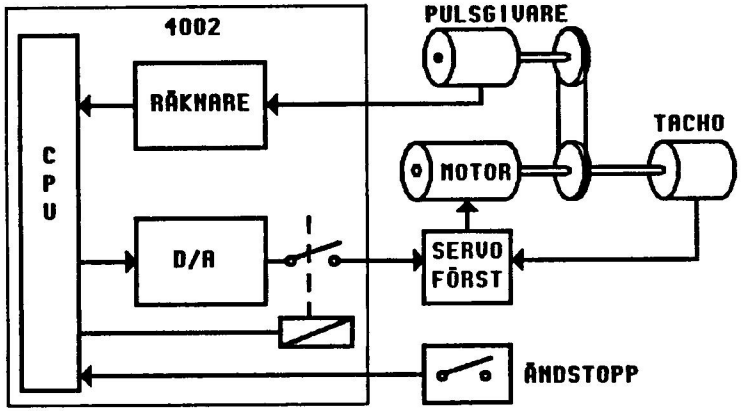
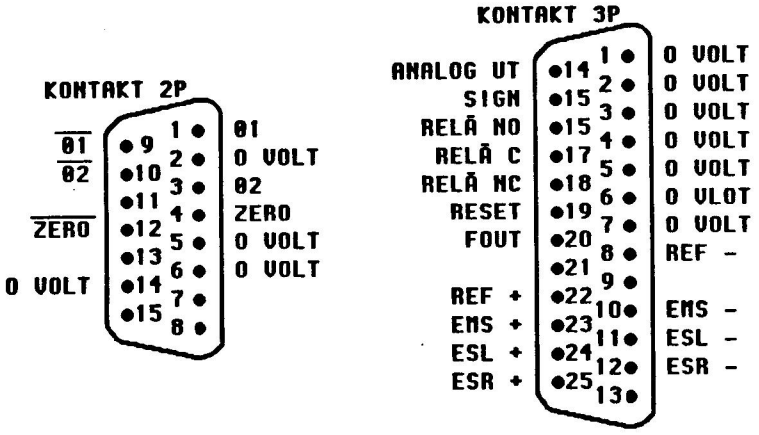


Fig 7.1.2 Uppkoppling av ett servosystem till 4002

Givaren är av fabrikat Haidenhaim, typ RoD 420. Servoförstärkaren är av fabrikat AXODYN typ 05LV. Motorn är av fabrikat BBC typ LB4 3Volt taco.
Kontakten, DB15, till inkrementalgivaren följer ett standardiserat gränssnitt.
Kontakten, DB25, innehåller hastighetsinformation till servot samt information från gränslägesbrytarna.

Signalen FOUT, den digitala hastighetsinformationen till servot, finns utdragen i en BNC-kontakt.



PULSGIVARE

DIGITALA GIVARSIGNALER, FREKVENS OCH ANALOG UT

Fig 7.1.3 stiftlayout för DataBoard 4002

Kapitel 7.2.1 DataBoard 4066
Stegmotorkontrollkort

7.2.1.1 Specifikationer 4066

Strömförsörjning	+5Volt, 400 mA
Ingångar	Gränslägesbrytare 5V TTL
Utgångar	Fasutgångar 5V TTL till 2 st stegmotorer

7.2.1.2 Beskrivning 4066

DataBoard 4066 är ett intelligent stegmotorkontrollkort. Kortet kan hantera 2 st 4-fas stegmotorer helt oberoende av varandra. Det kan användas tillsammans med IBM PC och styrprogrammen kan skrivas i t ex BASIC eller Assembler.

Styrningen av kortet kontrolleras med hjälp av 24 stycken kraftfulla kommandon. Dessa kan ges i direktmode till kontrollern och utföres då fortlöpande eller också kan kommandona lagras i en kommandobuffert på kortet. I det senare fallet är det möjligt att med huvudprogrammet styra start/stopp och rörelseriktning på motorerna samt start/stopp av kommandosekvensen i programbufferten. Med kommandona kan flera olika funktioner utföras. T ex: acceleration, retardation, hastighet, positioneringssätt (absolut, relativ eller direkt stegning), rörelseriktning.

4066 består av 2 stycken identiska positioneringsdelar och en del som känner av statusen från kontrollern samt de externa signalerna. Positioneringsdelarna är uppbyggda kring var sin enchipsprocessor som i sin tur är kopplade till var sin stegmotor. Styrprogrammet hanterar positioneringsdelarna på ett sådant sätt att användaren upplever 4066 som två separata kort med var sin kanalvalsadress.

Kortet medger positionering på tre olika sätt: absolut, relativ eller direkt stegning. Motorerna kan styras med en maxfart på 3359 steg/sekund och med ramper för acceleration och retardation. För att få en finare upplösning är det möjligt att använda halvstegsgång på motorerna.

Direkt till kortet kan givare för ändstopp anslutas och avläsas från IBM PC.

Kortet kan användas i ett brett fält av applikationer. Vid behov kan 4066 kompletteras med DataBoard 5085 som ger signalförstärkning och galvanisk isolering för bägge motorerna.

Några användningsområden är:
◊ Positioneringssystem
◊ Industrirobotar
◊ Doseringsutrustning
◊ Provtagningssystem
◊ Automation

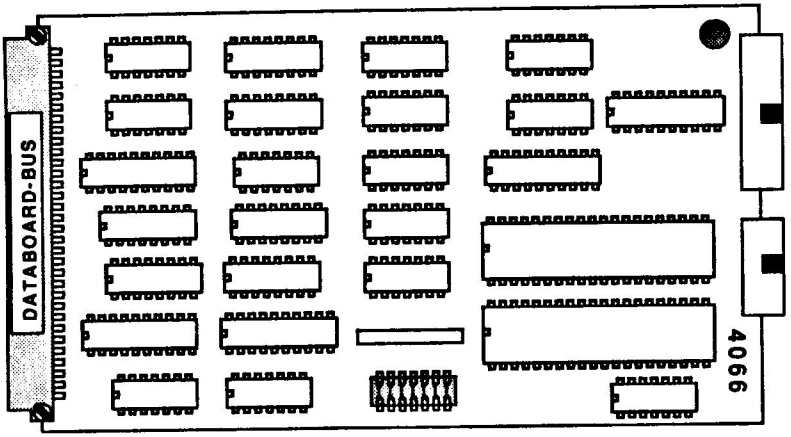


Fig 7.2.1.1 DataBoard 4066

7.2.1.3 Programmering av 4066

Allmänt
4066 programmeras med DataBoard I/O-kommandon. Programkommandona till styrkretsen på 4066 sänds med OUT &H300 kommandot från IBM PC som ASCII-strängar. Andra I/O-kommandon används för att definiera olika moder och styra exekveringen av program-kommandona som lagrats lokalt på kortet.

Ett programkommando består av ett ASCII-tecken (Stor bokstav). Ett eventuellt mellanslag (ASCII 32 decimalt) och en eventuell parameter i form av ett heltal som ASCII-sträng. Alla kommandon utom "Q" avslutas med vagnsretur (ASCII 13 decimalt).

Styrkretsen kan vara i ett av tre moder.

A Kommandomod

Program exekveras omedelbart då de mottages.

Exempel:

R 30	Definiera hastighet
N 347	Definiera antal steg
G	Starta stegning

B Programmeringsmod

Programkommandona lagras i det lokala minnet för senare exekvering.

Exempel:

R 20	Förinställ steghastighet
E	Gå in i programmeringsmod
N 5	Antal steg
G	Ta dessa steg
U	Vänta
T	Repetera från början Till villkor uppfyllts
O	Stoppa programmet
Q	Gå ur programmeringsmod
D	Starta exekveringen

C Exekvering

4066 är då sysselsatt med att utföra ett kommando och inga nya kommandon kan tas emot. Indikeras med bit 0=1 i I/O-kommandot INP(&H300).

Den interna programbufferten rymmer 18 bytes. Varje kommando kräver en byte utom kommandon med parametrar, för vilka storleken ges nedan:

F XXX	2 bytes
N XXXXX	3 bytes
P XXXXX	3 bytes
R XXX	2 bytes
S XXX	2 bytes
X XXXX	3 bytes

Programkommandon

Athome:	**A** Definierar nuvarande position som hemmaläge (origo), vilken används som referens för det absoluta Position-kommandot. Athome får inte föregås av ett riktningsändringskommando ("+" eller "-") och får inte användas två gånger utan att återställning eller initiering görs mellan dem.
Bitset	**B** Ger HOLD* signalen till drivkortet 5085. HOLD* reducerar strömmen till motorn när den skall stå stilla.
Clearbit	**C** Släpper HOLD*-signalen till 5085, varvid full drivström ges till motorn.
Doitnow	**D** Börja exekva det lagrade programmet i styrkretsen på 4066 från början. I programmeringsmod lagras D som ett kommando, för att hoppa till programmets början igen. I kommandomod startas programmet.
Enter	**E** Gå in i programmeringsmod. Alla instruktioner(programkommandon) efter Enter-kommandot läggs i programbufferten.
Factor	**F XXX** (1<= XXX <= 255) Steghastigheten (steg/sekund) divideras med heltalsfaktorn i parametern.
Go	**G** Starta stegning enligt tidigare definierade parametrar.
Halfstep	**H** Sätter halvstegs-läge och stannar där tills reset eller initiering görs.

Initiate	**I** Styrkretsen återställes till det läge den har vid spänningspåslag. Styrkretsen sätts i kommandomod och programmet suddas ut. Riktning sätts medsols men parametrarna Rate och Avstånd nollställs inte. Motorerna stoppas.
Jog	**J** Sätter styrkretsen i en mod så att start/stop av stegning styrs av bit 3 i kommandot OUT &H302,[DATA]. Jog-instruktionen skall normalt vara sista kommandot, men nya programkommandon kan ges så länge bit 3 i [DATA] är "1". Bit 3 i [DATA] = "0" krävs för stegning enligt tidigare givna kommandon och stegning sker så länge som bit 3 är låg ("0"). Då bit 3 är "1" kan ingen stegning ske. OBS Jog- och loopTil-kommandona utesluter varandra.
Left/right	**L** Sätter styrkretsen i en mod så att stegriktningen styrs av bit 4 kommandot OUT &H302,[DATA]. Programkommandona "+" och "-" ignoreras. bit 4="1" ger medurs bit 4="0" ger moturs En ändring av bit 4 avkänns efter varje påbörjat steg och påverkar nästa steg.
Number	**N XXXXX (1<=XXXXX<=65535)** Ger antalet steg som skall tas, relativt nuvarande position. Eventuell tidigare parameter i kommandot "Position" ignoreras.
Onestep	**O** Gör omedelbart ett steg. Enkelsteg kan även triggas av den externa TRIG*-signalen om det tillåts av bit 5(="1") i kommandot OUT &H302,[DATA]
Position	**P XXXXX (0<=XXXXX<=65535)** Sätter moden till "Absolut" och anger position dit stegning skall ske relativt den tidigare definierade "HOME" positionen. Eventuell tidigare given "Number" parameter ignoreras.

Quit

Q (utan vagnsretur ASCII 13 decimalt)
Lämna programmeringsmod och gå över till kommandomod: Observera att Q inte får följas av "CR" !
Kommandot Q skall även användas innan nya parametrar matas in eller innan direkta kommandon ges efter avslutande av ett program, som inte avslutats med "O"-kommandot.

Rate

R XXX
Definierar "Rate"-parametern. Steghastigheten varierar i området 50 till 3339 steg/sekund när "Rate" varierar från 1 till 254 (om hastighetsfaktorn är 1). Den verkliga steghastigheten varierar olinjärt med "Rate"-parametern och faktorn enligt tabellen i slutet av kapitlet.

OBS! Steghastigheten kan istället styras av den externa TRIG*-signalen, varvid ett steg tillåts för varje negativ puls.

Slope

S XXX (1<=XXX<=255)
Ramp-moden Slope används när hög hastighet önskas och startmomentet är stort. Rate måste vara jämnt delbar med Ramp-parametern och totala antalet steg måste vara större än 2*Rate.
Antalet steg att gå är enligt tidigare kommandon ("N" eller "P") och maximala steghastigheten ges av "Rate"-kommandot. Hastighetsfaktorn blir alltid 1 vid stegning med ramper.

Steghastigheten ökas för varje steg (från noll) genom att addera Ramp-parametern "XXX" till "Rate" parametern. När slutpunkten närmar sig, börjar styrkretsen automatiskt retardationen med samma Ramp-parameter som vid starten.

Exempel: Följande kommando tar 500 steg med accelerations och retardations-ramper.

N 500	Antalet steg
R 100	Max "Rate", motsvarar 80 steg/sekund
S 5	Ramp-parametern är 5, dvs "Rate" ändras med 5 steg för varje nytt steg
(5,10,15,...100...,10,5	
G	Starta stegning

loopTil	**T** Detta kommando testar om bit 3 i senaste I/O-kommandot OUT &H302,[DATA] är hög eller låg. Hopp sker till programmets början om bit 3 ="0" i [DATA]. Om bit 3 ="1" fortsätter istället programmet med nästa instruktion. OBS! Kommandona "T" och "J" använder samma bit i kommandot OUT &H302,[DATA] och utesluter därför varandra. Exempel: O Ta ett steg T Gör det igen tills bit 3="1" i [DATA]
waitUntillow	**U** Programmet väntar tills bit 2 i I/O-kommandot OUT &H302,[DATA] är "0" innan exekveringen fortsätter. Jämför kommandot med "W" nedan.
Waituntilhigh	**W** Detta är motsatsen till kommandot "U" ovan. Programmet väntar tills bit 2 i I/O-kommandot OUT &H302,[DATA] är "1" innan exekveringen fortsätter.
eXpend	**X XXXXX (1<=XXXXX<=65535)** Programmet väntar den angivna tiden i millisekunder, dvs upp till ca 1 minut.
+	**+** Sätt riktningen till medurs för alla steg som följer. Ignoreras om "L" -kommandot har givits.
-	**-** Sätt riktningen till moturs för alla steg som följer. Ignoreras om "L" -kommandot har givits.
0	**0** (noll) Stoppar exekveringen av programmet och går in i kommandomod. Notera att kommandot består av siffran noll och inte av en bokstav.

KOMMANDO		FUNKTION
INP (&H307)		Nollställning av samtliga I/O-kort på 4680-bussen. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]		Aktiverar 4066-kortet med kortadressen [kortadress]. Exempel: OUT &H301,24, väljer ut kortet med kortadressen 24 enligt byglingarna. Eftersom 24 är ett jämnt tal väljs X-motorn ut för kommando Med OUT &H301,25 väljs Y-motorn ut. &H301 kommer av att interfacekortet i IBM PC lägger till värdet 300 till portadressen på I/O-kortet (300 HEX).
INP (&H300)		Läser styrkretsens STATUS för den tidigare valda motorn. Aktiva signaler är låga.
	◊◊◊◊◊◊√	READY* Styrkretsen är beredd att ta emot nästa tecken i en kommandosträng som skickats med , kommandot:OUT &H300,[DATA].
	◊◊◊◊◊√◊	MC* Rörelsen klar
	◊◊◊◊√◊◊	PC* Programmet klart, dvs exekveringen har stoppat. Programkommandon kan inte ges när ett program exekverar.
	◊◊◊√◊◊◊	PROG* Styrkretsen är i programmeringsmod. Programkommandon exekveras inte, utan lagras i bufferten.
	◊◊√◊◊◊◊	G1* Signal från externa gränslägesbrytaren 1.
	◊√◊◊◊◊◊	G1* Signal från externa gränslägesbrytaren 2.
	√√◊◊◊◊◊◊	Används inte

OUT &H300,[DATA]	ASCII-strängar sänds som programkommandon till styrkretsen på 4066. Nya kommandon får endast sändas då READY*-biten är låg. Ett CR (vagnsretur = ASCII 13 decimalt) avslutar varje programkommando utom "Q". Bit 7 i [DATA] ignoreras men bör vara "0".
OUT &H302,[DATA]	Direkt styrning av styrkretsen på 4066. Inkluderar Reset och Abort-kommandon samt modbitar.
◊◊◊◊◊◊◊√	RESET* ("0") Nollställ och initiera styrkretsen. OBS! D0 skall vara låg i minst 50 ms för att ge tid för detektering och initiering.
◊◊◊◊◊◊√◊	ABORT* ("0") Stoppar eventuellt pågående stegning och aktiverar status MC* (rörelsen klar). Motorn stoppardirekt även i Ramp-mod (Slope). Om "Abort" ges medan ett program exekverar, fortsätter styrkretsen med nästa programkommando när D1 går hög igen.
◊◊◊◊◊√◊◊	Wait- styrbit. Programkommandona"W" och "U" testar denna bit och programmet väntar tills: D2="0" vid "U" kommandot eller D2="1" vid "W" kommandot
◊◊◊◊√◊◊◊	Start/stopp/loop styrning. Testas av programkommandona "J" och "T". **A:** Om styrkretsen är i extern Start/Stopp mod, styr D3 Start/Stop av stegning. D3="0" Tillåt stegning D3="1" Tillåt inte stegning **B:** Annars kan kommandot "T" användas, vilket Loopar programmet från början eller inte. D3="0" Loopa från början D3="1" Ta nästa steg efter kommado "T"

◊◊◊√◊◊◊◊	EXT.DIR Extern styrning av riktningen. Endast aktiv om kommandot "L" har givits. Bit D4 ger stegriktning. D4="0" Moturs D4="1" Medurs
◊◊√◊◊◊◊◊	Möjliggör/kopplar bort extern stegriktning med yttre signalen TRIG*. D5="0" Stegning kan göras oberoende av TRIG* D5="1" Kräver en extern TRIG* efter varje steg innan nästa steg tillåts.
√√◊◊◊◊◊◊	Används inte

OUT &H305,[DATA] Koppla in/ur interrupt (avbrott) från kretsarna på 4066. Interrupt aktiveras genom en hög bit ("1") i respektive position.

◊◊◊◊◊◊◊√	Avbrott när 4066 är klar att ta emot nästa tecken i programkommandot.
◊◊◊◊◊√◊	Avbrott vid MC, Rörelsen klar.
◊◊◊◊√◊◊	Avbrott vid PC, programexekvering klar.
◊◊◊◊√◊◊◊	Avbrott när gränslägesbrytare aktiveras.
√√√√◊◊◊◊	Används inte.

7.2.1.4 Installation av 4066

Val av kortadress
Bestäm först vilken adress 4066-kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress för att undvika adresskollisioner och för att underlätta programskrivningen. OBS! Bit 0 på kodpluggen används inte till det här kortet. Motor X väljs med jämna CardSelect och motor Y med udda (bit 0=1).

Anslutning av yttre enheter
Fig 7.2.1.3 beskriver stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Anslutningsdon Nycklade kontakter för flatkabel. 10-pol för externa signaler, gränslägesbrytare. 16-pol för stegmotor eller för anslutning till anpassningskort 5085. En kabel finns som tillbehör för anslutning av 5085. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Externa TRIG*- och SYNC*-signalerna

TRIG*-ingången

TRIG*-signalen används för extern styrning av steghastighet och ramp om optimal anpassning krävs till motorns last.

En hög steghastighet väljs med ett programkommando, men den verkliga stegningen styrs (i extern mod) av TRIG* ingången.

Extern triggning väljs med bit 5 (="1") i kommandot OUT &H302,[DATA]. Notera: I extern start/stop mod, aktiverad av kommandot "J" måste även bit 3 ="0" i kommandot OUT &H302,[DATA] för att tillåta stegning.

SYNC*-utgången
En puls ges för varje steg som utförs. Denna signal kan användas för synkronisering av TRIG*-signalen till stegning, varvid de externa kretsarna tillåts ge en ny triggpuls, då SYNC* från det föregående steget har kommit.

SYNC*-utgångarna kan även användas för motorer, som kräver två styrsignaler. En för stegning och en för riktning. Drivkortet 5085 kan inte användas i detta fall.

Som standard används då XH/YH för riktningskontroll och XSYNC*/YSYNC* för stegpulser.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker enklast genom att i BASIC skriva OUT &H301,A, där A är 4066-kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

7.2.1.5 Programexempel

Styr stegmotorer i X- och Y-led

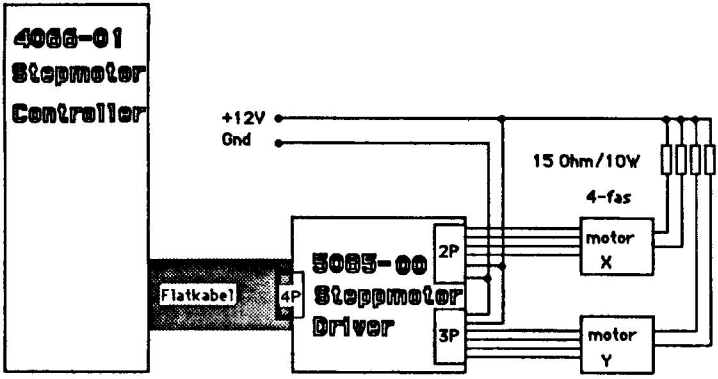


FIG 7.2.1.2 Koppling till programexempel

```
10 ' STEP4066  P-J H  850513
20 ' STYR STEGMOTORER I X- OCH Y-LED MED HASTIGHET,
   STRÄCKA OCH RIKTNING.
25 ' KORT 4066
30 '
40 Z=INP(&H307) ' Nollställ
50 '
60 XKORT%=0 : YKORT%=1 ' Kortadresser
70 '
80 OUT &H301,XKORT%
90  OUT &H302,0 ' Reset
100 FOR I%=0 TO 200 : NEXT I%
110 OUT &H302,3 ' Upphäv reset
120 '
130 OUT &H301,YKORT%
140 OUT &H302,0 ' Reset
150 FOR I%=0 TO 200 : NEXT I%
160 OUT &H302,3 ' Upphäv reset
170 '
182 CTRLBREAK=0
184 WHILE NOT CTRLBREAK ' Snurra till programmet
   bryts
186   '
190   ' Mata in värden för x-motorn.
```

```
195   CLS
200   LOCATE 1,1 : PRINT "X-MOTOR"
210   LOCATE 4,1 : INPUT "Hastighet 100-240 ",XHAST$
220   INPUT "Sträcka 1-64000    ",XSTRACKA$
230   INPUT "Riktning: V/H      ",XRIKTNING$
240   '
250   ' Mata in värden för y-motorn.
260   LOCATE 1,30 : PRINT "Y-MOTOR"
270   LOCATE 4,30 : INPUT "Hastighet 100-240 ",YHAST$
280   LOCATE 5,30 : INPUT "Sträcka 1-64000
   ",YSTRACKA$
290   LOCATE 6,30 : INPUT "Riktning: V/H
   ",YRIKTNING$
300   '
310   ' =============
320   '   X-MOTORN
330   ' =============
340   OUT &H301,XKORT% ' Välj kortadr. för X
345   '
350   KOMMANDO$="S 10" : GOSUB 1000 '
   Accelerationsramp.
360   KOMMANDO$="R "+XHAST$ : GOSUB 1000 ' Max
   hastighet.
370   KOMMANDO$="N "+XSTRACKA$ : GOSUB 1000 ' Antal
   steg.
380   '
390   IF XRIKTNING$="V" OR XRIKTNING$="v" THEN
   KOMMANDO$="-" ELSE KOMMANDO$="+"
400   GOSUB 1000 ' Skicka riktning
405   '
410   KOMMANDO$="C" : GOSUB 1000 ' Full drivström
   till motorn.
420   KOMMANDO$="G" : GOSUB 1000 ' Sätt igång.
430   '
440   GOSUB 1300 ' Vänta tills rörelsen är klar.
450   '
460   KOMMANDO$="B" : GOSUB 1000 ' Ej full
   motorström.
470   '
480   LOCATE 12,1 : PRINT "X-MOTOR KLAR"
490   '
500   ' =============
510   '   Y-MOTORN
520   ' =============
530   OUT &H301,YKORT% ' Välj kortadr. för Y
540   '
550   KOMMANDO$="S 10" : GOSUB 1000 '
```

```
   Accelerationsramp.
560    KOMMANDO$="R "+YHAST$ : GOSUB 1000 ' Max
   hastigheten.
570    KOMMANDO$="N "+YSTRACKA$ : GOSUB 1000 ' Antal
   steg.
580    '
590    IF YRIKTNING$="V" OR YRIKTNING$="v" THEN
   KOMMANDO$="-" ELSE KOMMANDO$="+"
600    GOSUB 1000 ' Skicka riktning vänster eller
   höger.
605    '
610    KOMMANDO$="C" : GOSUB 1000 ' Full drivström
   till motorn.
620    KOMMANDO$="G" : GOSUB 1000 ' Sätt igång.
630    '
640    GOSUB 1300 ' Vänta tills rörelsen är klar.
650    '
660    KOMMANDO$="B" : GOSUB 1000 ' Ej full
   motorström.
670    '
680    LOCATE 12,30 : PRINT "Y-MOTOR KLAR"
690    '
692    INPUT "Tryck return ",X$
694    '
700 WEND
710 END
1000 '
1010 '=============================================
1020 ' Subrutin för att skicka en kommandosträng till
   kort 4066.
1030 '
1040 ' In Kommando$ innehåller de tecken som skall
   skickas.
1050 '
1060 FOR I%=1 TO LEN(KOMMANDO$)
1070    TECKEN$=MID$(KOMMANDO$,I%,1)
1080    '
1090    ' Vänta på Ready-signal, programmet bryts
   efter timeout.
1100    TID%=1000
1110    WHILE TID%
1120      IF (INP(&H300) AND 1)=0 THEN 1170 ' Kolla
   Ready-signal.
1130      TID%=TID%-1
1140    WEND
1150    PRINT "Ingen Ready-signal!":STOP
1160    '
```

```
1170    ' Ok
1180    '
1190    OUT &H300,ASC(TECKEN$) ' Skicka
   kommandotecken.
1200 NEXT I%
1210 '
1220 OUT &H300,13 ' Kommandot avslutas med Carriage
   Return.
1230 RETURN
1300 '
1310 ' ========================================
1320 ' Subrutin som väntar tills rörelsen är klar.
1330 '
1340 WHILE INP(&H300) AND 2 ' Kolla bit 1 i
   statusbyten.
1350 WEND
1360 RETURN
```

Tabeller

Tabell 1 konverterar "Rate" parametern till steghastighet.
Tabell 2 visar effekten av några hastighetsfaktorvärden.
Tabell 3 innehåller användbara värden på "Rate" och "Factor"
Tabell 4 visar sekvenserna vid "halvsteg" och "Fullsteg" på motorlindningarnas signaler.

Tabell 1

Steg/sek	Rate parameter
50	0
54	25
60	50
69	75
80	100
95	125
117	150
152	175
217	200
239	205
264	210
296	215
335	220
387	225
457	230
429	232
560	235
615	237
646	238
682	239
721	240
765	241
815	242
871	243
973	244
1013	245
1102	246
1208	247
1338	248
1498	249
1702	250
1970	251
2339	252
2901	253
3359	254
3359	255

Tabell 2 (Rate=1)

Factor	Steg/sek
1	48.40
2	24.40
3	16.20
4	12.20
5	9.70
10	4.80
20	2.40
24	2.00
48	1.00
100	0.50
150	0.33
200	0.25
250	0.20

Tabell 3 Användbara värden

R=48 och F=50 --> 1 steg/sek
R=49 och F= 1 --> 60 steg/sek

Tabell 4

-Helsteg-

Steg	F1	F2	F3	F4
1	1	0	1	0
2	1	0	0	1
3	0	1	0	1
4	0	1	1	0

-Halvsteg-

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

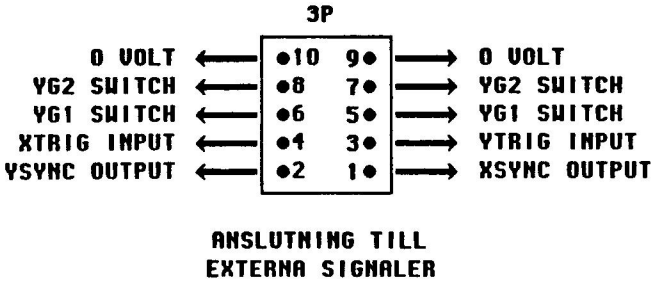


Fig 7.2.1.3 Anslutning av externa signaler, stiftlayout

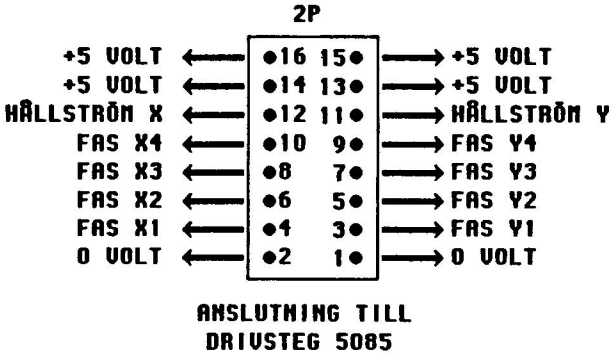


Fig 7.2.1.4 Anslutning till drivsteg 5085, stiftlayout

Kapitel 7.2.2 DataBoard 5085
Stegmotor drivkort 24 Volt, 1A

7.2.2.1 Specifikationer 5085

Anslutning:	16 pol Ansley Flatkabel kontakt
Kabel:	Speciell anslutning till 4066 styrkort finns som tillbehör.
Hold drivström:	Justerbar
Motoreffekt:	Max 24V/1A och minimum 10Volt

7.2.2.2 Beskrivning 5085

5085 är ett anpassningskort för 4066 Step Motor Contoller och stegmotorer. Det är speciellt utvecklat för motorer med 4 motorfaser. Kortet har två kanaler för t ex X/Y-styrning. Opto isolerade ingångar för att minimera störningsrisken och erhållande av galvanisk isolering. Separata ingångar medger hållströmskommendering av X och Y kanalerna. Strömförsörjningen sker från motorspänningen.

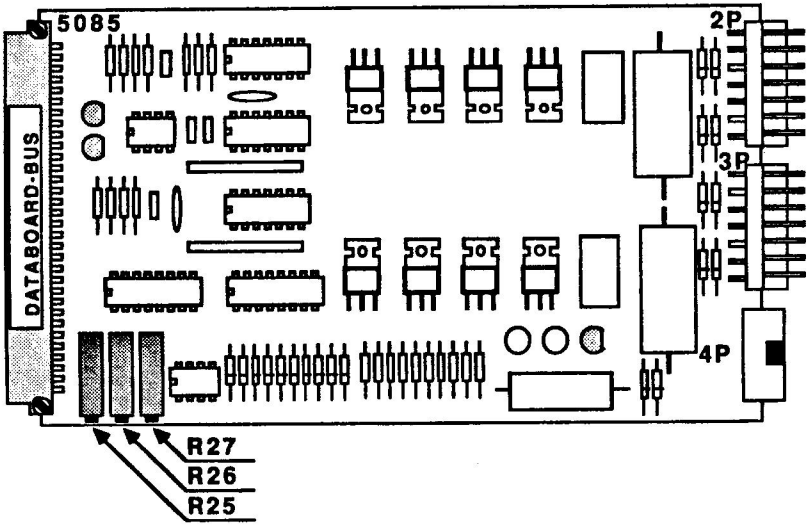


Fig 7.2.2.1 Stegmotor drivkort 5085

7.2.2.3 Programmering av 5085

5085 innehåller inga funktioner som kan programmeras från IBM PC4680.

7.2.2.4 Installation av 5085

Val av kortadress
5085 har ingen kortadress sett från PC4680.

Anslutning av yttre enheter
Anslut flatkabeln från kontakten 4P till styrenhet 4066. Anslut motor X till kontakt 2P och motor Y till 3P. Hållströmsjusering: Anslut motorn och kommendera stegning och hållström via 4066. Mät spänningen över strömbegränsningsmotståndet (som väljs av användaren). För motor X justeras med adj.R25. För motor Y justeras med adj.R26. Begränsningsområde justeras med R27.

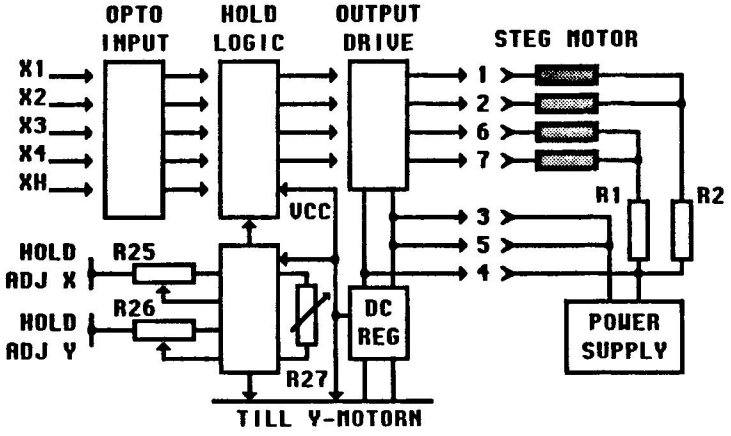


Fig 7.2.2.2 Blockschema för 5085

Montering i expansionslåda.
Slå av spänningen först. Vänd komponentsidan åt höger. Placera kortet i I/O-delen så nära 4066-kortet som möjligt. Anslut flatkabeln mellan 4066 och 5085.

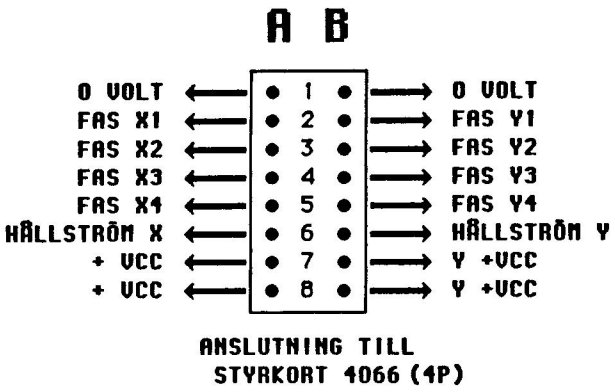


Fig 7.2.2.3 Anslutning av flatkabel till styrkort 4066

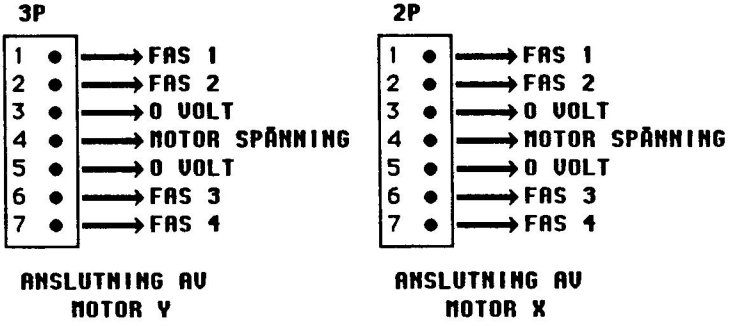


Fig 7.2.2.4 Stiftlayout för DataBoard 5085

Kapitel 7.3 DataBoard 4014
Drivkort för DC-motorer

7.3.1 Specifikationer 4014

Antal motorutgångar:	16
Motorspänning:	9 Volt - 24 Volt
MAX ström 9 Volt:	80 mA
MAX ström 12 Volt:	100 mA

7.3.2 Beskrivning 4014

Varje motor är ansluten till två utgångsstift, vilka är individuellt styrda för "source" eller "sink" drivning. Från programvaran kan man styra riktning, motorbroms eller friläge för respektive motor. Utgångarna styrs av 4 kommandon som vardera hanterar en grupp om 4 motorer. Utsignalerna är CW=medurs, CCW=moturs, Broms=alla utgångar i "sink" och friläge=alla utgångar i "source". Observera att jord (0 Volt) är gemensam för motorspänning och 0 Volt på DataBoards egen I/O-buss.

Tillämpningarna är diverse styr- och positioneringssystem. Drivning av lampor och reläer. "Source-" och "sink"-utgångar.

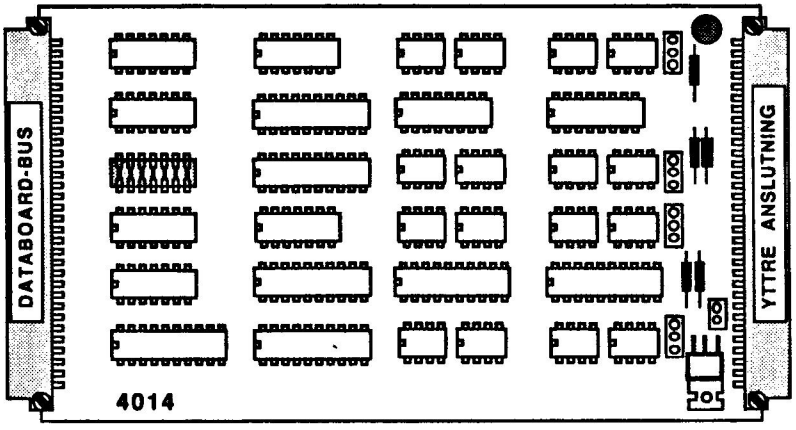


Fig 7.3.1 DataBoard 4014

7.3.3 Programmering av 4014

Följande definitioner gäller för programmering (se även signalnamn enligt stiftlayout).

Friläge +M=Aktiv (1)
-M=Aktiv (1)

Medurs	+M=Aktiv (1) -M=Passiv (0)
Moturs	+M=Passiv (0) -M=Aktiv (1)
Broms	+M=Passiv (0) -M=Passiv (0)

Signalen är aktiv då motsvarande bit i kommandot sätts till "1".

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Alla motorutgångar blir "0". Dvs "sink" och alla motorer bromsas. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar 4014-kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H301 kommer av att interfacekortet i IBM PC lägger till värdet 300 till portadressen på I/O-kortet (300 HEX).
OUT &H300,[DATA]	Styr motorerna 0-3 på på 4014-kortet. Lägger ut värdet [DATA]. Bitvärdet för de 4 signalparen på motorerna kan identifieras med hjälp av Stiftlayout och signalnamn enligt skissen för 4014-kortet. Bitvärden för att ange riktning etc framgår av tablån ovan.

OUT &H302,[DATA]	Styr motorerna 4-7 på på 4014-kortet. Lägger ut värdet [DATA]. Bitvärdet för de 4 signalparen på motorerna kan identifieras med hjälp av Stiftlayout och signalnamn enligt skissen för 4014-kortet. Bitvärden för att ange riktning etc framgår av tablån ovan.
OUT &H303,[DATA]	Styr motorerna 8-11 på på 4014-kortet. Lägger ut värdet [DATA]. Bitvärdet för de 4 signalparen på motorerna kan identifieras med hjälp av Stiflayout och signalnamn enligt skissen för 4014-kortet. Bitvärden för att ange riktning etc framgår av tablån ovan.
OUT &H304,[DATA]	Styr motorerna 12-15 på på 4014-kortet. Lägger ut värdet [DATA]. Bitvärdet för de 4 signalparen på motorerna kan identifieras med hjälp av Stiftlayout och signalnamn enligt skissen för 4014-kortet. Bitvärden för att ange riktning etc framgår av tablån ovan.

7.3.4 Installation av 4014

Val av kortadress
Bestäm först vilken adress 4014-kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
Anslut den externa motorspänningen till stift 3A/B, 4A/B, 29A/B och 30A/B. Jord (0 Volt) anslutes till 5A/B, 6A/B, 27A/B och 28A/B. Spänningsregulatorn kan användas för att reducera den externa spänningen till 9 Volt respektive 12 Volt, beroende på bygel S10. S10 sluten ger 9 Volt. När man väljer att öppna S10 får man en drivspänning på 12 Volt.

För varje motorgrupp väljs spänningen antingen direkt från den externa spänningen eller från spänningsregulatorn genom att ansluta EN bygel för varje grupp. Notera att MAX ström inte får överskridas för spänningsregulatorn.

Fig 7.3.3 beskriver stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan är numrerade från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker enklast genom att i BASIC skriva OUT &H301,A, där A är 4014-kortets avsedda adress. Om adresspluggen byglats riktigt,

skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

7.3.5 Programexmpel

Styr en DC-motor i taget via kort 4014

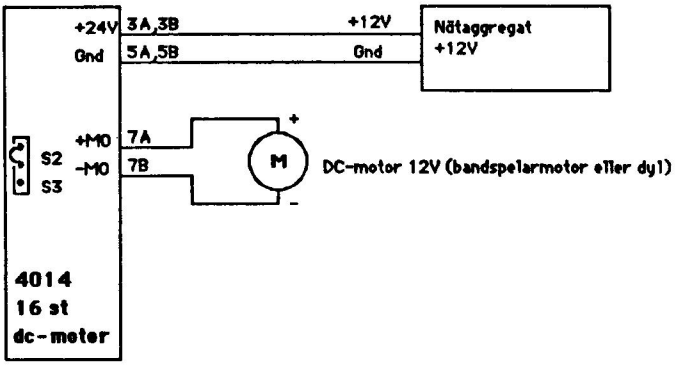


Fig 7.2.2 koppling för programexempel

```
10 ' Mot4014.bas P-J H  850510
20 ' Styr en DC-motor i taget via kort 4014.
30 '
40 Z=INP(&H307)
50 OUT &H301,1 ' Välj kort
60 '
70 CLS
80 PRINT "Vilken motor vill du köra (0-15)"; ' Välj
   motornummer.
90 INPUT MOT%
100 IF MOT%<0 OR MOT%>15 THEN 80
110 IF MOT%<4 THEN GRUPP%=0 ELSE GRUPP%=MOT%\4+1 ' Räkna
   ut grupptillhörighet
120 UT%=(MOT%-(MOT%\4)*4)*2 ' Denna motors plats i byten
   som skickas
130 '
140 PRINT "Styr motorn med följande piltangenter:
   vänster-moturs, höger-medurs,"
150 PRINT "uppåt-friläge, nedåt-stopp. Sluta med
   escape."
160 '
```

```
170 WHILE A$<>CHR$(27)
180  LOCATE 5,1 : A$=INKEY$
190  IF A$=CHR$(27) THEN PRINT : GOTO 480 ' Slut
200  IF LEN(A$)<=1 THEN 180 ' Vänta på piltangent
210  '
220  IF RIGHT$(A$,1)<>"K" THEN 280 ' Testa om tryckt
   tangent ej är pil vänster.
230  '
240  ' Moturs
250    OUT &H300+GRUPP%,2ÜUT%*2 ' Skicka en tvåa till
   vald motor.
260    PRINT "Moturs "
270  GOTO 480
280  '
290  IF RIGHT$(A$,1)<>"M" THEN 350 ' Testa om tryckt
   tangent ej är pil höger.
300  '
310  ' Medurs
320    OUT &H300+GRUPP%,2ÜUT%*1 ' Skicka en etta till
   vald motor.
330    PRINT "Medurs "
340  GOTO 480
350  '
360  IF RIGHT$(A$,1)<>"H" THEN 420
370  '
380  ' Friläge
390    OUT &H300+GRUPP%,2ÜUT%*3 ' Skicka en trea till
   vald motor.
400    PRINT "Friläge"
410  GOTO 480
420  '
430  IF RIGHT$(A$,1)<>"P" THEN 480
440  '
450  ' Stopp
460    OUT &H300+GRUPP%,2ÜUT%*0 ' Skicka noll.
470    PRINT "Stopp  "
480  '
490 WEND
500 END
```

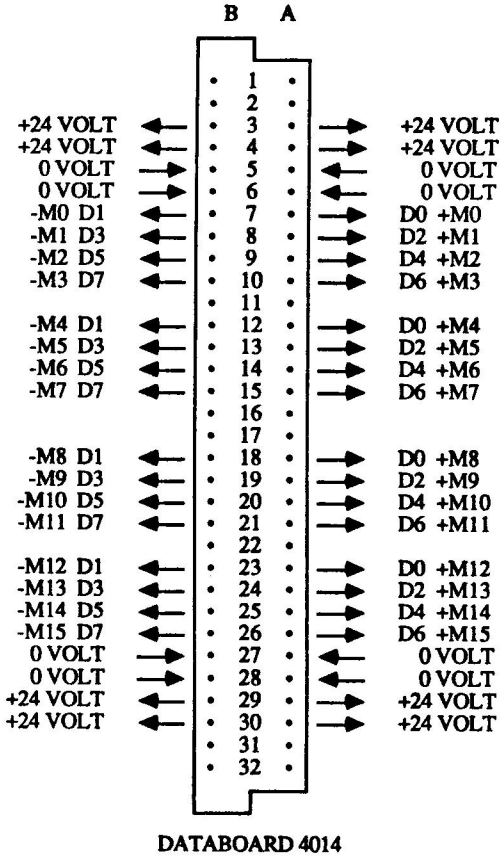


Fig 7.3.3 DataBoard 4014, stiftlayout och kontaktnamn

Kapitel 8.1 DataBoard 4117

Asynkront serie-interface

8.1.1 Specifikationer 4117

Antal kanaler	1 st
Signalnivå	V24 (RS232C) alt Current loop
Baudrate, intern	75-19.200
Split speed	Ja
Baudrate extern	Omkopplingsbar
Paritet	Udda, jämn eller ingen
Ordlängd	5-8 bitar
Antal stoppbitar	1 eller 2 (1.5 vid 5 bitars ordlängd)

8.1.2 Beskrivning 4117

4117 är ett seriellt I/O-kort för asynkron anslutning av skrivare, bildskärmar, mätinstrument eller annan utrustning som arbetar med serieöverföring. 4117 arbetar med sk V24-snitt (RS 232C) och kan anslutas till enheter som använder buffer full signalering via hårdvaran.

Överföringshatigheten är valbar mellan 75 och 19.200 baud och är normalt ställd till 1200 Baud vid leverans. Om annan hastighet skall användas krävs bygling enligt installations- anvisningen nedan. Ordlängd, antal stoppbitar och paritet kan även byglas på kortet. Signalsnittet omfattar följande signaler:

Insignaler:	
RxD (Received Data)	Inkommande data
REC SPEC 3	
CTS/DCD (Clear To Send)	Statussignal från exvis modem
Ring signal	Kan generera avbrott
Utgående signaler:	
TxD (Transmitted Data)	Utgående data
RTS (Request To Send)	Data kommer
DTR (Data Terminal Ready)	
SRTS	
TTL OUT 2, TTL OUT 4	

Exempel på användningsområden: Anslutning av skrivare, plotter, bildskärms-terminaler, modem eller andra datorer.

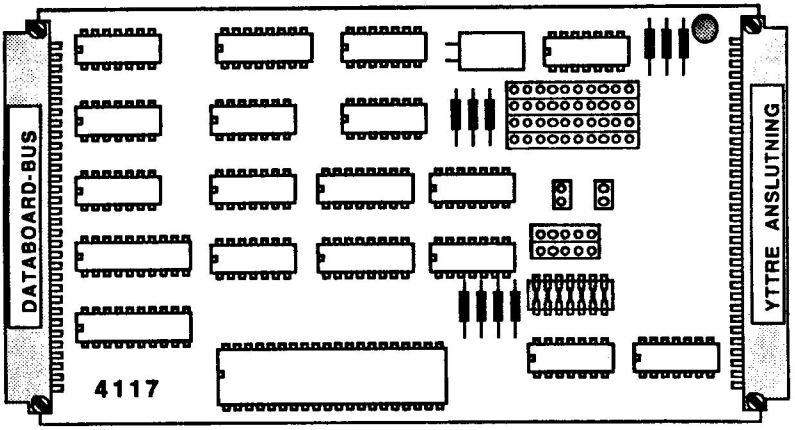


Fig 8.1.1 DataBoard 4117

8.1.3 Programmering av 4117

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Väljer 4117-kort med adress= [KORTADRESS] (=0-63) som I/O-kort. Exempel i BASIC: OUT &H301, 60. Väljer ett kort med adressbygeln satt till 60.
INP(&H300)	Läser ett mottaget tecken (på 8 bitar) från 4117-kortet. Bör göras först när bit 7 på STATUS-porten är = 0. Exempel i BASIC: Variabel= INP(&H301) Läs in ett tecken från serieporten.

INP(&H301)

Läser statusbitarna från 4117-kortet. Återställer även ring-signal vippan.

Databit	Databitens betydelse
◊◊◊◊◊◊◊√	DR READY interrupt. 0 aktiv.
◊◊◊◊◊◊√◊	TBR EMPTY. 0 = tecken kan sändas
◊◊◊◊◊√◊◊	TBR EMPTY interrupt. 0 aktiv
◊◊◊◊√◊◊◊	CTS/DCD. aktiv 0 eller 1 (bygel J7)
◊◊◊√◊◊◊◊	REC SPEC 3. (Pin 21B)
◊◊√◊◊◊◊◊	Ring signal. 0 aktiv (Pin 13B)
◊√◊◊◊◊◊◊	ERROR. 0=paritet, frame, overrun
√◊◊◊◊◊◊◊	DR READY. 0 då tecken finns att hämta

OUT &H300,[DATA]

Sänd [DATA] till serieporten. Får göras endast då bit 1 på statusporten är 0. Med DATA menas det ASCII-värde som skall sändas ut. Exempelvis är ASCII-värdet för A=65 och för B=66.

Exempel i BASIC: OUT &300,67 Sänd bokstaven C till serieporten.

OUT &H303,[DATA]

Aktivering respektive deaktivering av signaler på anslutningssidan.

◊◊◊◊◊◊◊√	Används inte
◊◊◊◊◊◊√◊	Används inte
◊◊◊◊◊√◊◊	Används inte
◊◊◊◊√◊◊◊	Används inte
◊◊◊√◊◊◊◊	= 16, Aktivering av TTL OUT 2
◊◊√◊◊◊◊◊	= 32, Aktivering av RTS
◊√◊◊◊◊◊◊	= 64, Aktivering av SRTS
√◊◊◊◊◊◊◊	=128, Aktivering av DTR

Exempel i BASIC: 560 OUT &H303,128 Aktiverar signalen DTR.

OUT &H304,[DATA]

Nollställer kortet. Motsvarar INP (&H307). Obs detta är en special för detta kort och gäller inte som nollställning för andra kort.

Exempel i BASIC: 780 OUT &H304,0 nollställer det här kortet men inte andra kort på DataBoard-bussen.

OUT &H305,[DATA]

Aktivering respektive deaktivering av signaler på anslutningssidan.

	Aktivering av
◊◊◊◊◊◊◊√	Används inte
◊◊◊◊◊◊√◊	Används inte
◊◊◊◊◊√◊◊	Används inte
◊◊◊◊√◊◊◊	Används inte
◊◊◊√◊◊◊◊	16=TTL OUT 4
◊◊√◊◊◊◊◊	32=RING SIGNAL interrupt enable
◊√◊◊◊◊◊◊	64=Recieve interrupt enable
√◊◊◊◊◊◊◊	128=Send interrupt enable

Exempel i BASIC: 230 OUT &H305,192 Sätt 4117 i sändmode och slå på interrupt för inkommande tecken.

8.1.4 Installation av 4117

Val av kortadress

Bestäm först vilken adress 4117-kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Byglingar
Överföringshastighet väljs med byglingar i position 5B på kortet.
Sändningshastighet väljs med byglingar mellan rad A och B. Mottagningshastighet väljs med byglingar mellan rad C och D.

	RAD A B C D	BAUD
1	••••	19.200
2	••••	9.600
3	••••	4.800
4	••••	2.400
5	••••	1.200
6	••••	600
7	••••	300
8	••••	150
9	••••	75
10	••••	EXTERN

Kolumn 1 till 9 väljer intern klocka 19.200 Baud till 75 Baud. Kolumn 10 väljer extern klocka. Anslut eventuell extern klocka till pin 2P:20B eller 3P:17 (på den 25 poliga Cannon-kontakten).

Välj paritet och antal stoppbitar med byglingar J1 till J3 i position 5D.

Bygel	Byglad	Ej byglad
J1	Paritet gäller	Ingen paritet
J2	Udda paritet	Jämn paritet
J3	En stoppbit	två stoppbitar

Välj ordlängd med bygling J4 och J5 i position 5D

Ordlängd	J4	J5
5	Byglad	Byglad
6	Ingen bygling	Byglad
7	Byglad	Ingen bygling
8	Ingen bygling	Ingen bygling

Välj CTS-fas och sändstopp ON/OFF vid signal CTS med byglingar J6 och J7 i position 5C. OBS! CTS används normalt för avkänning av DTR (exempelvis avkänning av printer Busy) Kan stoppa sändning från 4117 varvid status bit 1 hålls hög tills CTS frisläppes av den externa utrustningen.

Bygel	Byglad	Ej byglad
J6	CTS ger sändstopp	Ej sändstopp
J7	CTS hög ger sändstopp	CTS låg ger sändstopp

Ring Signal kan sätta en vippa för att generera avbrott. Ring signal selektiv interrupt enable måste vara valt via kommando OUT &H305 . Anslut bygling J8 position 3B för denna funktion.

Anslutning av yttre enheter
Fig 8.1.3 beskriver stiftlayout och signalnamn på I/O-sidan (V24 /RS232C). Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift.

Montering expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker enklast genom att i BASIC skriva OUT &H301,A, där A är 4117-kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.

8.1.5 Programexempel

Skickar text mellan 4117 och IBM's seriekort (COM1:)

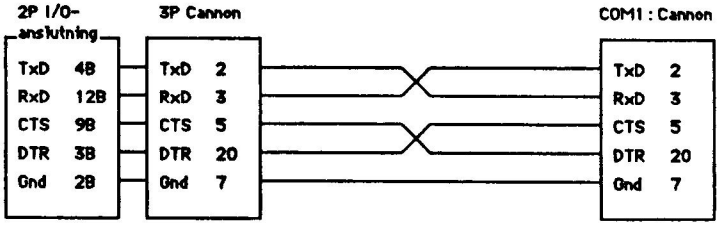


Fig 8.1.2 Kopplingsbeskrivning för programexempel

```
10 ' Text4117  P-J H 850513
20 ' Skickar text mellan 4117 och IBM's seriekort .
30 ' Följande byglingar är på: J2,J7,A5-B5,C5-D5.
    Resten är ej byglade.
50 Z=INP(&H307) ' Nollställ
60 OUT &H301,1 ' Kortval
70 OUT &H303,128 ' Aktiverar DTR
```

```
80 '
90 OPEN "com1:1200,N,8,2,RS,CS10000,DS0,CD0" AS #2 '
    öppna seriefilen.
100 '
110 CLS
120 WHILE MOTTAG$<>"SLUT" ' Sluta med SLUT
130   '
140   MOTTAG$=""
150   INPUT "Text som skall skickas? ",TEXT$
160   PRINT "Mottagen text:          ";
170   '
180   FOR I%=1 TO LEN(TEXT$) ' Gå igenom strängen
    tecken för tecken.
190     '
200     ' Läs status-byten.
210     IF (INP(&H301) AND 2)<>0 THEN 210 ' Vänta tills
    D1 blir 0.
220     IF (INP(&H301) AND 8)<>0 THEN 220 'Kolla CTS
    "handskaket",D3 ska bli 0
230     OUT &H300,ASC(MID$(TEXT$,I%,1)) ' Skicka ett
    tecken från 4117.
240     '
250     ' Ta emot tecknet på seriekortet.
260     MELLAN$=INPUT$(1,#2) ' Läs ett tecken.
270     '
280     ' Skicka tillbaka tecknet till 4117.
290     PRINT #2,MELLAN$; ' Endast ett tecken skickas.
300     '
310     ' Ta emot texten igen på kort 4117.
320     ' Läs status-byte.
330     IF (INP(&H301) AND 128)<>0 THEN 330 ' Vänta
    till status D7=0.
340     OUT &H303,0 ' DTR låg, så att inget mer tecken
    hinner komma.
350     I$=CHR$(INP(&H300)) ' Läs tecken.
360     OUT &H303,128 ' DTR hög, nu nästa tecken.
365     '
370     MOTTAG$=MOTTAG$+I$
380     PRINT I$;
390   NEXT I%
400   PRINT
410 WEND
420 END
```

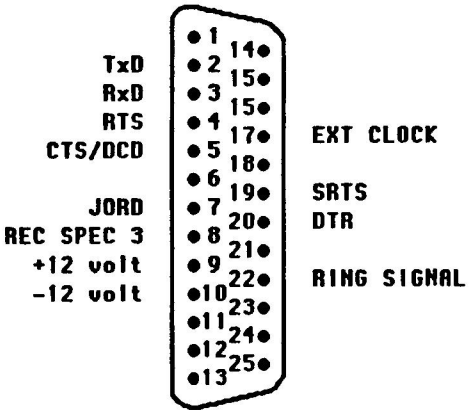


Fig 8.1.3 stiftlayout för DataBoard 4117

Kapitel 9.1 DataBoard 5070
Prototypkort för egna konstruktioner

9.1.1 Specifikationer 5070

I/O-kort med färdig avkodningslogik.

9.1.2 Beskrivning 5070

Kortet 5070 innehåller bussanpassning av samtliga DataBoard 4680-signaler. Kretslogik och komponentplacering framgår av krets-schema , komponentlista och komponentplacerings-schema som medföljer kortet vid leverans. Bussignalerna beskrivs i systembeskrivningen som även anger erforderlig drivning och belastning på bussen.
Kortet har standardformat för användning i 4680-systemet och är utrustad med standard kontaktdon för anslutning till bakplan och I/O. Användaren disponerar virstift för anslutning av egen kretslogik till disponibla stift i respektive kontaktdon (3P och 4P) samt för anslutning, inne på kortet, till samtliga bussignaler (5P, 6P och 7P). De vanliga IC-socklarna, 8 till 40-poliga kan användas. Plats finns för (12+4) 16 poliga IC-socklar. Fyra av dessa kan utnyttjas för flexibelt pol-antal.

Strömförsörjning +12 Volt, +5 Volt, -12 Volt och 0 Volt är framdragen till plaserna för 16 poliga socklarna.

9.1.3 Programmering av 5070

KOMMANDO	FUNKTION
INP (&H307)	Nollställning av samtliga I/O-kort på 4680-bussen. Bör inleda alla program för att säkerställa utgångsläget hos I/O-korten. Exempel i BASIC: 10 Variabel=INP(&H307).
OUT &H301,[KORTADRESS]	Aktiverar 5070-kortet med kortadressen [kortadress]. Exempel: OUT &H301,23, väljer ut kortet med kortadressen 23 enligt byglingarna. &H301 kommer av att interfacekortet i IBM PC lägger till värdet 300 till portadressen på I/O-kortet (300 HEX).

Nedan listade signaler har anpassats mot IN/OUT-bussen. Användaren fyller i för respektive kommando som tillämpas.

INP &H300

INP &H301

OUT &H300,[DATA]

OUT &H302,[DATA]

OUT &H303,[DATA]

OUT &H304,[DATA]

OUT &H305,[DATA]

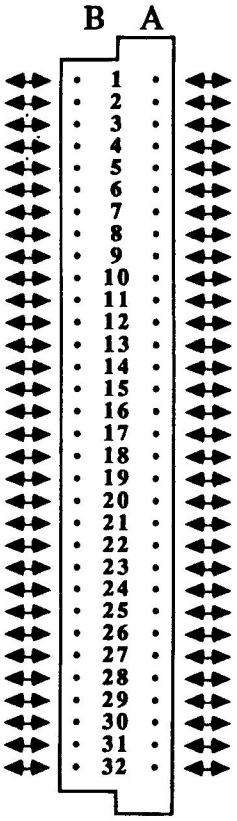
9.1.4 Installation av 5070

Val av kortadress
Bestäm först vilken adress 5070-kortet skall ha. Se kapitel 3.2.2 för beskrivning över hur man byglar kortadressen. Gör en förteckning av vilket kort som har vilken adress, för att undvika adresskollisioner och för att underlätta programskrivningen.

Anslutning av yttre enheter
Gör en egen beskrivning över stiftlayout och signalnamn på I/O-sidan. Ta en fotostatkopia på stiftlayouten och dokumentera vad som är anslutet på respektive stift. Signalnamn på datasidan bör numreras från 0 till 7 för att stämma med krav på programsidan. Se även kapitel 3.2.4 för anslutning av yttre enheter.

Montering i expansionslåda.
Slå av spänningen först. Vänd kortet så att I/O-kontakten (vid lysdioden) kommer utåt. Se kapitel 3.2.1 för ytterligare beskrivning av monteringen i expansionslådan.

Kontroll av byglad kortadress
Kontroll av adresspluggens bygling sker enklast genom att i BASIC skriva OUT &H301,A, där A är 5070-kortets avsedda adress. Om adresspluggen byglats riktigt, skall lysdioden vid I/O-kontakten tändas. Vid nytt adressval med annan adress, t ex OUT &H301,0 skall lysdioden släckas.



DATABOARD 5070

Fig 9.1.1 DataBoard 5070, stiftlayout

APPENDIX A

Klocka / kalender-krets HD146818

Detta appendix tar endast upp den information som gäller för programmering av HD146818. För utförligare information om tekniska data och hårdvaran hänvisas till Hitachis datablad och databöcker.

HD146818 RTC
(real time clock plus RAM)

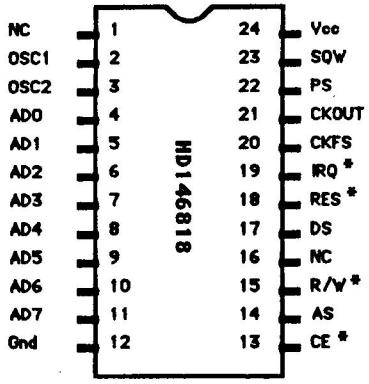
HD146818 är en periferikrets som kombinerar tre olika funktioner i en krets: klocka/kalender med alarmtid, periodisk avbrottssignal och fyrkantsvåg, och 50 byte med RAM av lågeffekttyp.
RTCn är gjord för i första hand två olika uppgifter. Den första är att hålla reda på tid och datum i ett microprocessor baserat system, den andra är att tillhandahålla avbrottsfri tid genom sin låga effektförbrukning och därigenom möjlighet till batteriuppbackning.

MÖJLIGHETER:

*TID timmar, minuter och sekunder
*DATUM veckodag, dag, månad och år
*Binär eller BCD presentation av tid/datum
*12- el 24-timmars mode med AM o PM i 12tim
*Automatisk kompensation för skottår
*Ses från mjukvara som 64 minnesceller
 14 byte klocka och kontrollregister
 50 byte RAM
*Tre olika avbrottssignaler som är separat
 mask- och testbara
 tid på dagen alarm, en gång per sekund/minut/
  timme
 periodiskt avbrott från 30,5uS till 500mS
 slut på uppdateringscykeln
*Programerbar fyrkantsvåg
*Tre tidbaser
 4.194304 MHz
 1.048576 MHz
 32.768 KHz
*Lågeffekt, höghastighets CMOS teknik
*Färdig för batteriupbackning

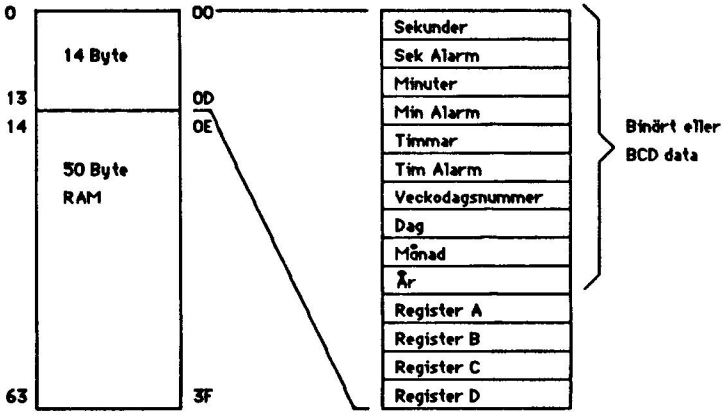
Figur 1

pinlayout



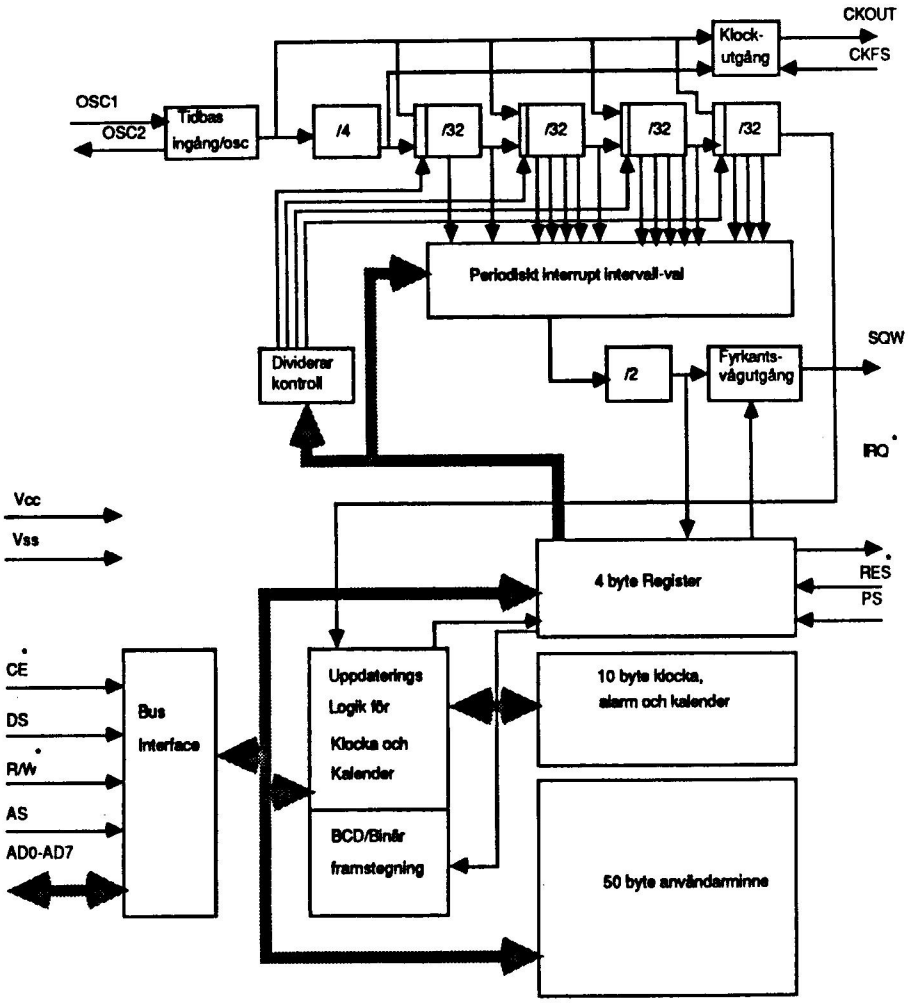
Figur 2

Adresskarta



Figur 3

Blockschema HD146818



ADRESSKARTA

Figur 2 visar adresskartan för RTCn. Minnet innhåller 50 byte användarRAM, 10 byte som normalt innhåller tid, kalender och alarmdata, och 4 byte för kontroll och statusord
Alla 64 byten är direkt läs- och skrivbara från program utom register C och D som bara kan läsas. Bit 7 i register A och "sekund"-byten kan också bara läsas. Bit 7 i "sekund"-byten kommer alltid att läsas som "0". Innehållet i de fyra kontrollregistren beskrivs senare i "REGISTER" sektionen.

Tid, Kalender, och Alarm adresser

Ett program kan ta in information om tid och datum genom att läsa från respektive adress. Programmet kan också ställa tiden, datumet eller alarmet genom att skriva till dessa adresser. Innehållet i de 10 byten för tid etc kan vara i binär- eller BCD-form.
Innan initiering av tid eller register sker skall "SET"biten i register B sättas till "1" för att förhindra att den interna uppdateringscykeln inträffar Programmet initierar sedan de 10 byten för tid på valt format (binär el BCD) och indikerar sedan formatet i "DM"biten i reg B. Alla 10 byten måste vara på samma format. Nu kan "SET"biten sättas till "0" så att uppdatering kan ske. Väl initierad kommer RTCn att uppdatera tiden på valt format. Formatet kan inte ändras utan att all 10 byten ställs in igen. Tabel 4 visar binär- och BCD-formatet för de 10 tid-, kalender-, och alarmbyten. 24/12 biten i register B anger om timmarna är 1-12 eller 1-24 och den kan inte ändras utan att timmarna ändras samtidigt. I 12-timmarsformat visar bit 7 i tim-byten AM("0") eller PM("1").
Det går inte alltid att läsa de 10 tidbyten från ett programm. En gång per sekund kopplas dessa in till uppdateringslogiken för att räknas upp en sekund och koll av om alarmvilkoret är uppfyllt. Om läsning av tid sker under denna period så kommer de 10 byten att vara odefinierade. Med tidbasen 4.19 MHz och 1.04 MHz är uppdateringstiden 248 µs och med 32.7 KHz är den 1948 µs. Under kapitel "Uppdateringscykel" sägs hur den ska hanteras i ett program.
De tre alarmbyten kan användas på två olika sätt. När alarmtiden är inskriven i respektive byte, kommer ett alarm-interrupt att genereras en gång per dag på utsatt tid, förutsatt att alarm enable biten (AIE) i reg B är satt till "1". Det andra sättet att använda alarm är att sätta in ett "gäller ej"-värde i en eller flera av alarmbyten. Detta värde kan vara mellan hex C0 och hex FF. Ett alarminterrupt varje timme genereras sålunda om hex FF skrivs i tim-alarmbyten. Ett interupt varje minut kräver hex FF i både tim- och minut-alarmbyten och för interrupt varje sekund ska alla byten vara satta till hex FF.

INTERRUPT

I RTCn finns tre separata helt automatiska källor för interrupt. Alarminterruptet kan genereras med tim-, minut-, eller sekundintervall. Det periodiska interruptet kan ställas in så att det genereras från var 500 ms ner till 30.517 µs (4MHz). Det tredje interruptet genereras i slutet av varje uppdateringscykel och kan användas för att tala om för ett program att nu är cykeln klar. Varje interrupt kommer att beskrivas nogrannare senare i texten.

Programmet väljer om inget, ett eller flera av interrupten skall vara aktiva Tre bitar i reg B talar om vilka som är aktiva resp inaktiva. En bit satt till "1" ger att interrupt är aktiv vilket medför att signalen IRQ* går låg när vilkoret uppfylls.Om biten är satt till "0" är interruptet inaktivt.

När ett interrupt inträffar pga att ett vilkor uppfyllts sätts en flagga i reg C för resp interrupt, även om interruptet är inaktivt. Dessa flaggor kan alltid läsas från ett program.

Om en flagga är satt när interruptet görs aktivt kommer IRQ* att gå "låg" (aktiv) direkt även om händelsen inträffade (flaggan sattes) långt tidigare. Därför finns det fall när programmet måste nollställa flaggan innan interruptet sätts aktivt.

När programmet bara läser flaggorna används dessa enbart som statusbitar och ingen aktivering av interrupt förekommer. Om en flagga är satt innebär det att ett interrupt har inträffat sedan flaggan lästes förra gången.

Det finns en sak man bör tänka på och det är att när läsning av en eller flera flaggor i reg C har gjorts så nollställs alla flaggorna efter läsningen. Läs därför av alla flaggorna på samma gång så att inget interrupt "försvinner" och maska sedan i programmet den eller de som är aktuella.

Ett annat sätt att använda flaggorna på är när alla interrupt är aktiva. När både en interruptbit är aktiv och en flagga är satt går IRQ* "låg" och förblir i detta läge så länge minst en bit och flagga är satta. IRQF biten i reg C är satt "1" så länge IRQ* är "låg".

Ett program kan se om det var RTCn som genererade interruptet genom att läsa reg C. Om bit 7 är "1" (IRQF) indikerar detta att ett eller flera interrupt har genererats i och av kretsen. Läsningen av reg C medför att alla aktiva flaggor inkl IRQF nollställs. När ett program hittar att IRQF är satt, ska det testa alla flaggor och hantera de interrupt som visas på flaggorna. Än en gång, mer än en flagga kan vara satt på samma gång.

Tabell 1

Tidbasfrekvens	Delningsbitar Register A			Aktiv mode	Delningssteg. Reset	Koppla ur N-stegbitar
	DV2	DV1	DV0			
4.194304 MHz	0	0	0	Ja	-	N = 0
1.048576 MHz	0	0	1	Ja	-	N = 2
32.768 kHz	0	1	0	Ja	-	N = 7
Alla	1	1	0	Nej	Ja	-
Alla	1	1	1	Nej	Ja	-

Tabell 2

Intervall-val Kontroll Register 1				4.194304 or 1.048576 MHz Tid Bas		32.768 kHz Tid Bas	
				Periodisk interrupt tpi	SQW Frekvens	Periodisk interrupt tpi	SQW Frekvens
RS3	RS2	RS1	RS0				
0	0	0	0	inget	ingen	inget	ingen
0	0	0	1	30.517 µs	32.768 kHz	3.90625 ms	256 Hz
0	0	1	0	61.035 µs	16.384 kHz	7.8125 ms	128 Hz
0	0	1	1	122.070 µs	8.192 kHz	122.070 µs	8.192 kHz
0	1	0	0	244.141 µs	4.096 kHz	244.141 µs	4.096 kHz
0	1	0	1	488.281 µs	2.048 kHz	488.281 µs	2.048 kHz
0	1	1	0	976.562 µs	1.024 kHz	976.562 µs	1.024 kHz
0	1	1	1	1.953125 ms	512 Hz	1.953125 ms	512 Hz
1	0	0	0	3.90625 ms	256 Hz	3.90625 ms	256 Hz
1	0	0	1	7.8125 ms	128 Hz	7.8125 ms	128 Hz
1	0	1	0	15.625 ms	64 Hz	15.625 ms	64 Hz
1	0	1	1	31.25 ms	32 Hz	31.25 ms	32 Hz
1	1	0	0	62.5 ms	16 Hz	62.5 ms	16 Hz
1	1	0	1	125 ms	8 Hz	125 ms	8 Hz
1	1	1	0	250 ms	4 Hz	250 ms	4 Hz
1	1	1	1	500 ms	2 Hz	500 ms	2 Hz

FREKVENSDELARSTEGEN

RTCn har internt en 22-bitars frekvensdelarstege som visas i figur 3. Från denna stege erhålls en 1 Hz signal som styr logiken för uppdateringscykeln. Stegen kan styras med tre bitar i reg A (DV2, DV1, DV0).

Styrning av stegen

Styrbitarna för stegen har tre funktioner (tabell 1). Tre olika tidbaser kan väljas: 4.194 MHz, 1.048 MHz, 32.768 KHz. Stegen kan hållas i resetläge så att tiden kan ställas med hög precision. När stegen övergår från resetläge till vald tidbas, kommer första uppdateringscykeln 1 sek senare. Styrbitarna används också för att underlätta test av RTCn.

Val av fyrkantsvåg

Femton av de 22 bitarna i stegen finns för åtkomst i en 1-av-15 valslogik (fig 3). Den första anledningen att välja i denna logik är för att ta ut en fyrkantsvåg på SQW-pinnen. Fyra bitar i reg A sätter vilken frekvens som ska erhållas på SQW (tabell 2). Dessa bitar styr också det periodiska interruptet.
När en frekvens är vald kan SQW-utgången slås till och från under programmets gång med hjälp av SQWE-biten i reg B. Med hjälp av ändring av styrbitarna i reg A, frekvensvalsbitarna i reg A, eller SQWE kan en asymetrisk våg erhållas på SQW-pinnen. Denna våg kan användas för många olika ändamål tex: tidbasgenerator, frekvenssyntes, eller tongenerering.

Periodiska interrupt (PI)

Det periodiska interruptet (PI) kan aktivera IRQ* från 1/0.5 sek till 1/30.517 µs och det är ett av de separata interrupten i RTCn. I tabell 2 visas hur PI väljs med samma bitar som styr fyrkantvågen. Om PI ändras så ändras även fyrkantvågen men båda kan väljas att vara aktiva separat så ett program kan använda båda med olika inställning av periodisitet. PI aktiveras genom att "1"-ställa PIE-biten i reg B.
PI är användbart i snart sagt alla realtidssystem. Den kan tex användas till att generera en intern tidbas, klocka in seriell data, multiplexa displayer etc.

Inställning av tid och startsekvens

Den första uppdateringscykeln kommer ca 500ms efter det att SET-biten i reg B har nollställts. Håll därför följande i minne när du ställer klockan:

(1) Ställ SET-biten till "1"
(2) Sätt DV0,1,2 till "1"
(3) Ställ tid, datum och alarmtid
(4) Ställ DV0,1,2 för rätt tidbas
(5) Nollställ SET-biten

Restriktioner för inställning av tid och datum

I RTCn finns ett fall när inte uppdatering av tid och datum sker på rätt sätt. På grund av detta bör man inte initiera tid och datum till värden som visas i tabell 3.

UPPDATERINGSCYKEL

RTCn genomgår en uppdateringscykel varje sekund förutsatt att rätt tidbas är vald, att SET-biten i reg B är "0", och att stegen inte är i reset-läge. Om SET-biten är "1" så tillåter den programmet att initiera klockan och hindrar samtidigt uppdatering att ske.
Den huvudsakliga uppgiften för uppdateringscykeln är att stega fram klockan en sekund åt gången och testa om ny minut, timme, dag, månad eller år ska sättas. Cykeln testar också om något alarmvilkor är uppfyllt.
Om 4.194 MHz eller 1.048 MHz tidbas används tar cykeln 248µs när däremot 32.768 KHz tidbas ger 1984 µs cykeltid. Under denna tid kan inte tid och kalender läsas från programmet. RTCn skyddar dessa byte från läsning under uppdateringen genom att koppla bort dom från databussen under hela cykeltiden. Samtidigt sätts UIP-biten (Uppdate In Progress) under cykeltiden. Om en läsning av tid utförs under uppdateringscykeln kommer läst data att vara "ogiltigt".
Det finns tre sätt att undvika att läs "ogiltigt" data. Den första metoden innebär att man använder "slut på updateringscykel-interruptet" för att starta inläsningen. När ett sådant interrupt inträffar har man mer än 998 ms på sig att läsa tiden innan nästa cykel inträffar. Efter det att interruptet har blivit omhändertaget ska man nollställa IRQF-biten i reg C.

Nästa metod för att unvika "ogiltig" tid är att läsa UIP-biten i reg A för att se om uppdatering pågår. Denna bit kommer att pulsas en gång per sekund. När UIP går "1" kommer cykeln att starta 244 µs senare. Om UIP läses "0" så har man alltså minst 244 µs på sig att läsa tiden, men man bör undvika läsrutiner som tar längre tid på sig än detta för att läsa data.
Dem tredje metoden använder periodiskt interrupt (PI) för att avgöra om läsning får ske. PI som kommer med högre hastighet än 244+248µs eller 244+1984µs (32KHz) medger tid för läsning vid varje interrupt. Denna läsning ska vara avslutad innom en tid av PI-intervall/2 (ex 488µs/2) för att inte inträffa under uppdatering.

REGISTER

RTCn har fyra interna register som programmet har full åtkomst till, även under uppdateringscykeln.

Tabell 5

	b7	b6	b5	b4	b3	b2	b1	b0
Reg A	UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0
Reg B	SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE
Reg C	IRQF	PF	AF	UF	"0"	"0"	"0"	"0"
Reg D	VRT	"0"	"0"	"0"	"0"	"0"	"0"	"0"

Register A (hex 0A)

UIP - "Uppdate In Progress" är en statusflagga som läsas från ett program. När UIP är satt "1" är uppdatering igång eller kommer att börja snart. Om UIP är "0" kommer inte uppdatering att tidigare än om 244µs. För detaljer se tabel 6. Tid, kalender och alarm är fullt läsbara när UIP är "0". UIP-biten går endast att läsa, och påverkas inte av reset. Om SET-biten i reg B sätts till "1" stoppas alla uppdateringar och UIP sätts till "0".

DV2, DV1, DV0 - Dessa tre bitar väljer hur frekvensdelarstegen skall ställas in dvs vilken tidbas som ska användas. Tabel 1 visar de olika kombinationerna och deras resultat. De tre bitarna påverkas aldrig av reset eller av RTCn i sig.

RS3, RS2, RS1, RS0 - Dessa intervallvalsbitar (Rate Selection bits) väljer 1 av 15 frekvenser på stegen eller slår av intervallgenereringen. Intervallet kan användas för att skicka ut en fyrkantsvåg på SQW-pinnen och/eller generera ett periodiskt interrupt. Tabell 2 visar de intervall som kan ställas in med hjälp av RS-bitarna. RS-bitarna är både läs- och skrivbara och påverkas inte av reset eller RTCn.

Tabell 6

UIP Bit	Tidbas	Uppdaterings-cykeltid	Min tid före uppdatering
1	4.194304 MHz	248 µs	-
1	1.048576 MHz	248 µs	-
1	32.768 KHz	1984 µs	-
0	4.194304 MHz	-	244 µs
0	1.048576 MHz	-	244 µs
0	32.768 KHz	-	244 µs

Register B (hex 0B)

SET - När SET-biten är "0" sker en uppdatering av klockan en gång per sekund. När den sätts "1" avbryts ev pågående uppdatering och vidare uppdateringar stoppas tills den sätts "0" igen. Detta sker för att initiering av tid och datum skall kunna ske. SET är en läs/skriv-bit och påverkas inte av reset eller RTCn.

PIE - Periodiskt Interrupt aktiv biten är en läs/skriv-bit som medger att PF (Periodic Interrupt Flag) kan generera ett interrupt och sätta IRQ*-pinnen "Låg". Programmet ska sätta PIE till "1" för att periodiska interrupt ska genereras med det intervall som har valts med RS-bitarna i reg A. Om PIE sätts till "0" blockeras PF från att generera interrupt. PIE påverkas inte av RTCn men nollställs av reset.

AIE - Alarm Interrupt aktiv är läs/skriv-bit som medger att AF (Alarm Flag) genererar ett interrupt. När AIE är satt till "1" inträffar interruptet när alarmvillkoret uppfylls. Satt till "0" blockerar AIE AF från att ge interrupt. Reset nollställer AIE men AIE påverkas inte av RTCn.

UIE - Uppdaterings Interrupt aktiv är också den en läs/skriv-bit. Den medger att UF (Update interrupt Flag) genererar interrupt om UIE är satt "1". Om reset går "Låg" eller SET-biten sätts "1" så nollställs UIE.

SQWE - Fyrkantsvåg aktiv (Square wave enable) ger, om satt till "1", en fyrkantsvåg på SQW-pinnen med ett intervall som bestäms av RS-bitarna. Sätts SQWE till "0" kommer SQW-pinnen att hållas "Låg" konstant. SQWE är en läs/skriv-bit som nollställs av reset.

DM - Data Mode biten talar om i vilket format tid och datum ligger och uppdateras i. Med DM satt till "1" väljs binärt format och "0" väljer BCD format. DM är en läs/skriv-bit som inte påverkas av reset eller RTCn.

24/12 - 24/12-biten anger om klockan skall gå i 24-timmars mode ("1") eller i 12-timmars mode ("0"). 24/12 är en läs/skriv-bit som bara kan ändras av mjukvara.

DSE - Satt till "1" hanterar denna bit "sommartid". Detta innebär att sista söndagen i april slår klockan om från 01:59:59 till 03:00:00 och sista söndagen i oktober slår den om från 01:59:59 till 01:00:00. Denna funktion träder inte i kraft om DSE är satt "0". OBS! Funktionen finns INTE implementerad på RTC-kretsar med beteckningen HD146818P (X- el RX-typ).

Register C (hex 0C)

IRQF - Interrupt Request Flag sätts till "1" om något eller flera av nedanstående vilkor uppfylls:
PF=PIE="1"
AF=AIE="1"
UF=UIE="1"
Så fort IRQF sätts "1" kommer IRQ* att gå "Låg". Reg C är ett läs-endast register där alla flaggor nollställs efter en läsning eller vid reset.

PF - Periodiskt interrupt Flaggan sätts till "1" varje gång en puls från valt intervall uppträder. PF sätts till "1" oavsett hur PIE är satt. PF sätter IRQF till "1" och genererar ett interrupt om PIE är "1". PF nollställs vid läsning av reg C eller vid reset.

AF - Alarm interrupt Flaggan sätts till "1" när alarmvillkoret uppfylls. AF sätter IRQF till "1" och genererar ett interrupt om AIE är "1". Läsning av reg C eller reset nollställer AF.

UF - Uppdateringsslut Flaggan sätts till "1" varje gång en uppdateringscykel är avslutad, dvs normalt en gång i sekunden. När UIE är "1" sätter UF IRQF till "1" och ger ett interrupt. UF nollställs av reset eller läsning av reg C.

b3 till b0 - Dessa bitar är oanvända och läses som "0"

Register D (hex 0D)

VRT - Giltig tid och minne (Valid RAM and Time). Denna bit sätts till "0" om PS (Power Sense)-pinnen har gått "Låg". VRT kan bara sättas till "1" genom läsning av reg D eftersom reg D är ett läs-endast register (förutsatt att PS är "Hög"). VRT påverkas ej av reset.

b6 till b0 - Dessa bitar är oanvända och läses som "0".

APPENDIX B

Programmen på PC4680-disketten

Detta appendix innehåller en lista över de program som normalt följer med på disketten. I listan finns också hänvisningar till de kapitel i manualen där programmen förekommer. Datum System förbehåller sig rätten att ändra innehållet på disketten utan att ändra i listan. Detta kan innebära vissa skillnader mellan listan och disketten.

Programmen på medföljande PC 4680-disketten

Programnamn Kort beskrivning	Sida i manualen	Kort
Funk.bas Användbara funktioner och subrutiner.	3.5	
Decbin.bas Omvandling av decimaltal till binärtal och tvärtom.	4.6	
Rinn4006.bas Rinnande ljus på kort 4006.	5.1.2.7	4006
Lasin.bas Läsning av ingångarna på korten 4005 och 4006.	5.1.2.8	4005, 4006
Lastum .bas Läsning av två portar, tumhjul tiotal och ental.	5.1.2.9	4005, 4006
Styrut.bas Styrning av utsignalerna på kort 4005 och 4006.	5.1.2.10	4005, 4006
Las4008.bas Inläsning av en port och visa resultatet binärt.	5.2.1.5	4008, 4011
Rinn4095.bas Rinnande ljus på kort 4095.	5.2.2.4	4095
Styr4085.bas Styr två utsignaler på 4085.	5.3.4	4085
Las4085.bas Avläsning av 32 ingångar med 4085.	5.3.7	4085
Int4085.bas Avläsning av de fyra interruptingångarna.	5.3.8	4085
Rinn4103.bas Rinnande ljus på reläkort 4103.	5.4.4	4103
Swir4013.bas Läser byglingarna grupp C och D och presenterar resultatet binärt.	5.5.6	4013
Rinn4013.bas Rinnande ljus på I/O-kort 4013.	5.5.8	4013

Programnamn Kort beskrivning	Sida i manualen	Kort
Amp4083.bas Styr strömmen på vald kanal på D/A-kortet 4083.	6.1.6	4083
Volt4083.bas Styr spänningen på vald kanal på D/A-kortet 4083.	6.1.8	4083
Sin4083.bas Variera spänningen enligt sinuskurva på D/A-kortet 4083.	6.1.9	4083
Amp4084.bas Styr strömmen på vald kanal med D/A-kortet 4084.	6.2.5	4084
Volt4084.bas Styr spänningen på vald kanal med D/A-kortet 4084.	6.2.6	4084
Volt4115.bas Läs av spänningen på A/D-kort 4115 med 32 kanaler.	6.3.5	4115
Step4066.bas Styr stegmotorer i X- och Y-led.	7.2.1.12	4066
Mot4014.bas Styr en DC-motor i taget via kort 4014.	7.3.4	4014
Text4117.bas Skickar text mellan 4117 och IBM's seriekort (COM1:).	8.1.7	4117
PCTime.bas Läser av klockan från PC 4680 och lägger tiden i TIME$ och DATE$.		PC 4680
DBTime.bas Ställer klockan på PC 4680 och initierar interrupt.		PC 4680

APPENDIX C

Teknisk Specifikation av PC4680

Informationen i detta appendix är till för avancerad programmering av PC4680 och DataBoard-kort.

Switchar och byglingar

På PC4680-kortet finns en switchgrupp SW1 och tre byglingsfält S1,S2 och S3.

Switcharna 1-5 (SW1) ställer vilken grundadress kortet skall ha. Med hjälp av dessa kan kortet placeras inom hela det adressområde för I/O-enheter som IBM PC hanterar dvs &H000 - &H3FF. Kortet i sig tar upp 16 adresser medan däremot adressvalslogiken inte medger tätare adressering än i steg om 32 adresser (&H20). Switch 1-5 har följande bitvärden och refererar till signaler enligt tabellen nedan:

1	&H20	A5
2	&H40	A6
3	&H80	A7
4	&H100	A8
5	&H200	A9

Observera att respektive switch får motsvarande bitvärde när den sätts i läge OFF.

Switch 6 kopplar i och ur Non Maskeble Interrupt (NMI) från DataBoard-bussen till IBM PC. Om sw 6 ställs i läge ON kommer NMI att kopplas till signalen IOCHCK på IBM-bussen. Denna signal genererar normalt ett PARITY CHECK 1 meddelande och därför krävs en assemblerrutin i IBM för att ta hand om sinalen, om NMI kopplas in. Ytterligare information om detta finns i Technical Reference Manual och DOS Technical Reference Manual till IBM.

Switch 7 styr signalen IOCHRDY på IBM-bussen. På DataBoard-bussen ligger signalen IORDY, som atktiveras av långsamma enheter, kopplad till en monovippa på PC4680-kortet som ger en puls på ca 8µs vid negativ flank. Denna puls fördröjer CPUn, om sw 7 = ON, med samma tid så att enheten hinner bli klar.

Bygling S1 kopplar in interrupt från PC4680 till signalen IRQ2 på IBM-bussen. För att hantera denna signal måste man se till att initiera interrupthanteringskretsen, lägga upp en interruptvektor, lägga in en interruptservicerutin m.m. Mer information om hur detta går till finns i DOS-manualen, Technical Reference Manual och DOS Technical Reference Manual till IBM och i INTELs databöcker för periferikretsar.

Bygling S2 ger, satt i läge mot kortets fästplåt, en klockfrekvens till DataBoard-bussen på 2.38 MHz och i läge från plåten en frekvens på 4.77 MHz.

Bygling S3 påverkar pinne 27 på PROM-sockeln. I övre läget är p27 = +5V, och i undre läget kopplas p27 till port B bit 6 på parallellkretsen typ 8255A-5. Denna krets ligger kopplad till och styr alla

signaler till PROMhållaren.

Portadressering

Till nedanstående portnummer skall läggas den grundadress kortet är inställt på. Tabellen visar till vilken funktion varje port på kortet går.

Port nr	In-funktion	Ut-funktion	Anmärkning
0	DB data	DB data	
1	DB stat	DB card select	
2	DB option	DB control 1	
3	-	DB control 2	
4	-	DB control 3	
5	PC4680 interrupt stat	DB control 4	Se avsnitt Interrupt
6	-	-	
7	DB reset	-	
8	Port A IC15 (8255)	Port A IC15	Styr promhållaren
9	Port B IC15	Port B IC15	Styr promhållaren
A	Port C IC15	Port C IC15	Styr promhållaren
B	-	Controlreg IC15	
C	Port A IC14 (8255)	Port A IC14	Styr klockchippet
D	Port B IC14	Port B IC14	Styr promhållaren
E	Port C IC14	Port C IC14	Styr klockchippet
F	-	Controlreg IC14	

Interrupt

PC4680-kortet kan internt generera sex olika interrupt, 3 från DataBoard-bussen och 3 från klockan. DB kan generera interrupt med signalerna NMI (se switchar och byglingar), INT och RESIN. INT-signalen är den vanligaste interruptorsaken från DB då RESIN oftast bara förekomme i en- och tvåkortsdatorsystem där den är en hårdvarureset istället för interrup Interrupten från klockan är ett för alarmtid, ett periodiskt interrupt och ett son talar om att uppdateringscykeln är slut (1ggr/sek). Hur dessa interrupt initier och hanteras går att läsa om i appendix A.

För att IBM ska uppfatta att ett interrupt har inträffat måste någon av dess interruptsignaler på bussen aktiveras (IRQ2-IRQ7). Den enda signal av dessa som inte har specifiserats att användas av något IBM-kort är IRQ2. Denna signal finns indragen på PC4680 och aktiveras vid interrupt om bygel S1 är monterad. När sedan interruptrutinen ska ta reda på vilken enhet på PC4680 som gav interrupt kan den detektera detta genom att läsa status på port 5. En bit satt till "0" indikerar att ett interrupt har skett, där bit 7 är RESIN, bit 6 är INT och bit 0 är klockan. Om det var ett interrupt från klockan ska rutinen gå ut i klockretsen och läsa vilket eller vilka av det tre interrupt som klockan kan ge som var aktiva (se appendix A). Läsning och skrivning av klockan ska alltid ske genom drivrutinerna för densamma. Om bit 6 är satt till "0" ska rutinen söka av (polla) de kort på DB som kan generera interrupt och beroende på vilket det var utföra rätt operation. Att RESIN är kopplat som ett interrupt beror på att det ska finnas en möjligh att ta hand om signalen när IBM och PC4680 körs i någon form av utvecklingssytem.

Alla ovan nämda interrupt utom RESIN sätts inaktiva efter läsning av interruptande enhet.
NMI går inte att läsa på port 5 utan går direkt via switch 6 till IOCHCKsignalen på IBM-bussen.
För mer information om hur interruptrutiner, initiering av interrupt och uppläggning av vektorer går till hävisas till DOS manualen, DOS Technical Reference Manual, Technical Reference Manual för IBM och INTELs databöcker över 8088 CPU och periferikretsar.

Klocka / PROMhållare

För normal användning av klockan används de program som lämnas med till kortet (DBTIME och PCTIME, se appendix B). PROMhållaren kan läsas med de stödrutiner som medföljer (READROM, app B), runt vilka användaren själv bygger upp ett program för att hämta information från PROMmet lägga i minnet på IBM.
Den som vill utnyttja klocka / PROMhållare och de parallellkretsar dessa ligger kopplade till på ett mer avancerat sätt kan dock ha nytta av nedanstående information. Den innehåller en beskrivning över hur klocka / PROMhållare ligger kopplade till respektive parallellkrets.

IC nr	Port	Bit	Chip	Signal	Pinne	
15	A	0	PROM	A0	10	
		1		A1	9	
		2		A2	8	
		3		A3	7	
		4		A4	6	
		5		A5	5	
		6		A6	4	
		7		A7	3	
	B	0		A8	25	
		1		A9	24	
		2		A10	21	
		3		A11	23	
		4		A12	2	
		5		A13	26	
		6			27	se bygling S3
		7		fri		
	C	0		CE	20	
		1		OE	22	
		2-7		fri		
14	A	0	Klocka	AD0	4	
		1		AD1	5	
		2		AD2	6	

	3		AD3	7
	4		AD4	8
	5		AD5	9
	6		AD6	10
	7		AD7	11
B	0	PROM	Q0	11
	1		Q1	12
	2		Q2	13
	3		Q3	15
	4		Q4	16
	5		Q5	17
	6		Q6	18
	7		Q7	19
C	0-3		fri	
	4	Klocka	CE	13
	5		AS	14
	6		R/W	15
	7		DS	17

De signaler som är märkta "fri" finns utdragna och kan användas valfritt.

För klockretsen gäller dessutom att följande signaler / pinnar finns utdragna:

Pinne	Signal	Komentar
1	NC	används ej
3	OSC2	oscillatorutgång (32.768 KHz)
16	NC	används ej
21	CKOUT	tidbasutgång (CKFS=1)
23	SQW	fyrkantsvågsutgång 2-8192 Hz

Mer information om klockan finns i appendix A och i HITACHIs databöcker över processorer och periferikretsar.

För den som vill använda sig av de "fria" signalerna på parallellkretsarna, så finns instruktioner för hur dessa kretsar (av typ 8255) programmeras och styrs bla i Technical Reference Manual till IBM eller i INTELs databöcker.

Rent allmänt är alla signaler på kortet av typ LS TTL, även de ut på DataBoard-bussen. En signal är av Open Collector-typ och det är IOCHRDY ut på IBM-bussen.
De signaler som används på DB och IBM-bussen finns listade nedan:

DataBoard 4680

D0 - D7, DATA, CS, C1 - C4, DATA, STAT, OPS, RST, RESIN, INT, NMI, IORDY, Ø, Gnd

IBM-bussen

D0 - D7, A0 - A9, IOR, IOW, AEN, IOCHCK, IOCHRDY, IRQ2, CLK, RESET, Gnd, +5V, +12V, -12V

APPENDIX D

Interruptprogrammering

Informationen i detta appendix är till för avancerad programmering av IBM PC, PC4680 och DataBoard-kort.

Användare som vill utnyttja denna information bör ha god kännedom om assembler och programmering av kretsar.

Datum System AB tar inget ansvar för de följder som kan bli fallet vid användning av den information som finns i appendixet.

Vad är "interrupt" ?

Interrupt är precis som det engelska ordet säger, ett "avbrott". Med avbrott menar man i datorsammanhang offtast att CPU'n avbryts i sin exekvering av kommandon av en yttre signal som påkallar uppmärksamhet. När detta inträffar börjar CPU'n att köra ett speciellt program avsett för interrupt och när den är klar med detta återgår den till samma ställe i sitt vanliga program där den blev avbruten.

Varför interrupt ?

En anledning till att man använder interrupt är att man vill avlasta CPU'n med arbetet att kolla om någon yttre enhet, tex ett tangentbord, vill ha uppmärksamhet. Istället för att med jämna intervall kolla om någon tangen är nedtryckt så låter man tangenbordet skicka en signal till CPU'n när en tangen trycks ned. Denna signal är då kopplad till CPU'ns interruptingång.

Vidare används interrupt av enheter som måste få företräde framför andra till CPU'n. Ett exempel är en seriell port kopplad till ett modem där det kommer en ström av data. För att inte porten ska behöva vänta på att CPU'n frågar om det finns något tecken att hämta, och därmed riskera att tappa tecknet om det kommer ett nytt, så skickar porten en avbrottssignal till CPU'n.

En tredje användning är intervallstyrda "bakrundsjobb". Detta innebär att någon krets som kan skicka en signal med jämna intervall, tex en gång i sekunden, styr interruptingången. När avbrottet sker kan CPU'n tex göra en mätning på en A/D eller kolla status på någon ingång och därefter återgå till sitt ordinarie jobb. På detta sätt kan mätningar göras i exakt rätt tid och utan att man behöver låsa upp datorn för enbart denna uppgift.

Hantering av interrupt, allmänt

När en CPU får ett avbrott så spar den undan positionen eller prgramräknaren (PC) för programmet den håller på med på sin stack. Därefter detekterar den, på något olika sätt för olika processorer, vilken enhet som gav interrupt. När detta är gjort går den till en tabell i minnet och läser en vektor eller adressinformation från en plats i tabellen som motsvarar den interruptande enheten. På detta sätt kan CPU'n särskilja olika enheter. Den inlästa vektorn pekar på den position i minnet där programmet för interrupt från en speciell enhet finns och programräknaren får nu detta värde, dvs man "hoppar" till detta ställe. I denna "interruptrutin" utförs de instruktioner som behövs och när det är klart görs ett "return from interrupt (RTI)" eller återhopp från avbrott. Då hämtas den gamla programräknaren från stacken och CPU'n fortsätter från det ställe där den blev avbruten.

Interrupt på IBM PC

IBM PC har en processor som heter 8088 och tillverkas av INTEL. Till denna CPU finns en krets med beteckningen 8259, som är en sk "interrupt controller". Denna krets har 8 st ingångar som ansluts till enheter som kan skicka avbrottssignaler. Om någon ingång aktiveras skickar kretsen en signal till CPU'ns interruptingång. Två av 8259'ans ingångar (IRQ0 o 1) används internt på moderkortet och de övriga sex finns utdragna på bussen (IRQ2 - IRQ7). 8259 måste initieras på rätt sätt innan den kan hantera avbrott och detta görs normalt av BIOS vid uppstart.

CPU'n arbetar med en interruptvektortabell längst ner i minnet. Tabellens längd kan vara max 1 Kbyte. Eftersom varje vektor tar 4 byte, segment plus offset, kan alltså 256 vektorer (enheter) finnas i tabellen. Varje vektor har ett nummer, ett interruptnummer som identifierar vilken enhet vektorn hör till, och det kan vara mellan 0 och 255.

En interruptrutin på IBM PC med DOS operativsystem bör skrivas enligt ett visst mönster. Den består av i princip två delar, en initieringsdel och själva interruptrutinen. I initieringen sker följande saker:

1. Ta reda på effektiva adressen (EA) till interruptrutinens startpunkt.
2. Lägg segmentvärdet i DS registret och offset i DX reg.
3. Lägg i AL vilket interruptnummer vektorn har och AH 25 hex.
4. Gör ett funtionsanrop till DOS med INT 21 hex för att sätta vektorn.
5. Läs nuvarande interruptmask från 8259'ans port 1
6. Sätt aktuell bit för ingången till "0" för att tillåta interrupt.
7. Lägg tillbaka nya masken till 8259'ans port 1.
8. Ta reda på effektiva adressen för slutet på rutinen.
9. Justera den så att den är på jämna 16 byte över slutet.
10. Lägg 3100 hex i AX och EA i DX, och gör ett INT 21 hex igen.

Med punkterna 1-4 sätter man med hjälp av DOS upp interruptvektorn till sin rutin på rätt plats i tabellen. Punkt 5-7 talar om för 8259 att en interruptsignal på en speciell ingång får generera interrupt, och punkt 8-10 talar om för DOS hur stor rutinen är och att den skall ligga kvar i minnet samt att man återgår till DOS.

Då man är tillbaka i DOS efter denna initiering så är rutinen klar att hantera interrupt. När sedan ett interrupt är hanterat måste en del saker göras av rutinen innan återhoppet görs.

1. Ladda AL med 60 hex + bit satt "1" motsvarande interruptande ingång
2. Lägg ut det till 8259'ans port 0

Efter denna sekvens är kretsen redo att taemot ett nytt interrupt. För att närmare se vad som händer så finns listan på rutinen "INT_XT" medlagd

Interrupt på IBM AT

Skillnaden mellan IBM PC och AT är att AT'n har två stycken 8259 interrupt controller kretsar där den ena är kopplad som slav till den andra via ingång IRQ2 på masterkretsen. Det innebär att när ett interrupt detekteras på någon av slavens ingångar kommer denna att signalera till ingången på mastern, som i sin tur signalerar processorn att interrupt har inträffat. Denna koppling gör att det blir några små tillägg till initieringen och när man går ur interruptrutinen.

1. Ny interrupt vektor behöver bara läggas upp för slaven eftersom BIOS har initierat mastern och dom för slaven aktuella ingångarna.
2. Innan återhopp från interruptrutinen skall kvittens först skickas till mastern, och sen till slaven.

I övrigt är förfarandet detsamma som för IBM PC. Listan på rutinen "INT_ATCO" visar hur detta går till.

För att få en mer detaljerad bild av hela systemet bör man ha tillgång till följande böcker:

1. Technical Reference PC/XT eller PC/AT
2. DOS Technical Reference (2.0, 2.1, 3.0, 3.1, 3.2)
3. INTEL Microsystem Component Handbook vol 1 och 2
4. INTEL iAPX 86/88 186/188 Programmer's Reference Manual eller iAPX 286 Programmer's Reference Manual .

Dessa böcker ger ett totalt underlag för programmering på så här maskinnära nivå.

I detta appendix finns medlagt listorna till 5 stycken olika interruptrutiner varav två ej är helt uttestade. Inte heller för de andra tar Datum System något ansvar för utan dessa 5 rutiner ska ses som exempel på hur man kan göra. Rutinerna har följande funktion:

1. "INT_XT" hanterar interrupt från PC4680 på IBM PC och PC/XT.
2. "INT_ATCO" hanterar interrupt från PC4680 på IBM AT med färg.
3. "INT_ATMO" som ovan fast för monochrom skärm.
4. "INT_ATC" hanterar interrupt från IBM AT's klockrets (färg).
5. "INT_ATM" som ovan fast för momoskärm.

De ej uttestade rutinerna är "INT_ATCO" och "INT_ATMO".

INT_XT

```
                                ;
                                ;-------- STACK ---------------------
                                ;
0000                            stack   segment para stack 'stack'
0000 0100 [                             db 256 dup ('stack   ')
         73 74 61 63
         6B 20 20 20
               ]

0800                            stack   ends
                                ;
                                ;-------------------------------------
                                ;
                                ;
                                ;-------- CODE ----------------------
                                ;
0000                            cseg            segment para public 'code'
                                                assume cs:cseg,ds:cseg,es:cseg,ss:stack
                                ;
                                ;-------- INTTINT -------------------
                                ;
0000                            intxt           proc far
0000 FA                                         cli
0001 1E                                         push ds
0002 06                                         push es
0003 8C CA                                      mov dx,cs
0005 8E C2                                      mov es,dx
0007 8E DA                                      mov ds,dx
0009 8D 16 0052 R                               lea dx,int
000D B8 250A                                    mov ax,250Ah
0010 CD 21                                      int 21h
0012 E4 21                                      in al,21h
0014 24 FB                                      and al,0FBh
0016 A3 02AF R                                  mov intmask,ax
0019 E6 21                                      out 21h,al
001B BA 0300                                    mov dx,0300h
001E B4 0C                                      mov ah,0Ch
0020 E8 027A R                                  call get
0023 E4 61                                      in al,61h
0025 0C 08                                      or al,08h
0027 E6 61                                      out 61h,al
0029 E4 62                                      in al,62h
002B A8 01                                      test al,01h
002D 75 09                                      jnz monodisp
002F C7 06 02B1 R B800                          mov disptype,0B800h
0035 EB 07 90                                   jmp last
0038 C7 06 02B1 R B000          monodisp:       mov disptype,0B000h
003E 8D 06 02B3 R               last:           lea ax,last_adr
0042 25 FFF0                                    and ax,0FFF0h
```

```
0045 05 0010                        add ax,0010h
0048 8B D0                          mov dx,ax
004A B8 3100                        mov ax,3100h
004D 07                             pop es
004E 1F                             pop ds
004F FB                             sti
0050 CD 21                          int 21h
0052                        intxt   endp
                            ;
                            ;------ INT --------------------------------
                            ;
0052                        int     proc far
0052 FA                             cli
0053 50                             push ax
0054 52                             push dx
0055 51                     more:   push cx
0056 53                             push bx
0057 57                             push di
0058 56                             push si
0059 55                             push bp
005A 1E                             push ds
005B 06                             push es
005C 8C C8                          mov ax,cs
005E 8E D8                          mov ds,ax
0060 8E C0                          mov es,ax
0062 BD 0000                        mov bp,0
0065 BA 0300                        mov dx,0300h
0068 E8 0082 R                      call intvekt
006B B0 62                          mov al,62h
006D E6 20                          out 20h,al
006F 07                             pop es
0070 1F                             pop ds
0071 5D                             pop bp
0072 5E                             pop si
0073 5F                             pop di
0074 5B                             pop bx
0075 59                             pop cx
0076 BA 0305                        mov dx,0305h
0079 EC                             in al,(dx)
007A 34 FF                          xor al,0FFh
007C 75 D7                          jnz more
007E 5A                             pop dx
007F 58                             pop ax
0080 FB                             sti
0081 CF                             iret
0082                        int     endp
                            ;
C                                   include intvekt.asm
C                           ;------ INTVEKT -----------------------------
C                           ;
C                           ; DX=PC4680 base adress
C                           ;
```

```
                C   ; AX,BX,CX changed
                C   ;
                C   ;--------------------------------------------------
                C   ;
0082              C         intvekt         proc near
0082  52          C                         push dx
0083  83 C2 05      C                       add dx,05h
0086  EC          C                         in al,(dx)
0087  5A          C                         pop dx
0088  8A C8        C                       mov cl,al
008A  B5 05        C                       mov ch,05h
008C  B8 0008       C                      mov ax,0008h
008F  8D 1E 00AF R     C                   lea bx,inttable
0093  50          C         test_bit:       push ax
0094  53          C                         push bx
0095  51          C                         push cx
0096  52          C                         push dx
0097  F6 C1 01      C                       test cl,01h
009A  75 08        C                       jnz nextint
009C  F6 ED         C                      imul ch
009E  2C 05         C                      sub al,05h
00A0  03 D8         C                      add bx,ax
00A2  FF E3         C                      jmp bx
00A4  5A            C       nextint:       pop dx
00A5  59          C                         pop cx
00A6  5B          C                         pop bx
00A7  58          C                         pop ax
00A8  D0 E9         C                      shr cl,1
00AA  FE C8          C                     dec al
00AC  75 E5         C                      jnz test_bit
00AE  C3            C                      ret
00AF              C         intvekt         endp
                C                   include inttable.asm
                C   ;------ INTTABLE ------------------------------
                C   ;
                C   ; Interrupt routin call table for 'intvekt'
                C   ;
                C   ; No registers changed
                C   ;
                C   ;--------------------------------------------------
                C   ;
00AF              C         inttable        proc near
                C   ;
00AF  E8 01B0 R     C       bit_7:          call resin
00B2  EB F0       C                         jmp nextint
00B4  E8 01E2 R     C       bit_6:          call dbint
00B7  EB EB       C                         jmp nextint
00B9  E8 0224 R     C       bit_5:          call nolegint
00BC  EB E6       C                         jmp nextint
00BE  E8 0224 R     C       bit_4:          call nolegint
00C1  EB E1       C                         jmp nextint
```

```
00C3 E8 0224 R          C   bit_3:          call nolegint
00C6 EB DC              C                   jmp nextint
00C8 E8 0224 R          C   bit_2:          call nolegint
00CB EB D7              C                   jmp nextint
00CD E8 0224 R          C   bit_1:          call nolegint
00D0 EB D2              C                   jmp nextint
00D2 E8 00D7 R          C   bit_0:          call clivekt
00D5 EB CD              C                   jmp nextint
                        C   ;
00D7                    C   inttable        endp
                        C                   include clivekt.asm
                        C   ;------ CLIVEKT ------------------------------
                        C   ;
                        C   ; DX=PC4680 base adress
                        C   ;
                        C   ; Interrupt routine call select for clock interrupts
                        C   ;
                        C   ;-------------------------------------------
                        C   ;
00D7                    C   clivekt         proc near
00D7 B4 0B              C                   mov ah,0Bh
00D9 E8 027A R          C                   call get
00DC 8A C1              C                   mov al,cl
00DE 0C 8F              C                   or al,8Fh
00E0 50                 C                   push ax
00E1 B4 0C              C                   mov ah,0Ch
00E3 E8 027A R          C                   call get
00E6 58                 C                   pop ax
00E7 22 C8              C                   and cl,al
00E9 8B D9              C                   mov bx,cx
00EB F6 C1 80           C   IRQF:           test cl,80h
00EE 74 1C              C                   jz exit
00F0 F6 C1 40           C   PF:             test cl,40h
00F3 74 05              C                   jz AF
00F5 E8 013F R          C                   call period
00F8 8B CB              C                   mov cx,bx
00FA F6 C1 20           C   AF:             test cl,20h
00FD 74 05              C                   jz UF
00FF E8 010F R          C                   call alarm
0102 8B CB              C                   mov cx,bx
0104 F6 C1 10           C   UF:             test cl,10h
0107 74 03              C                   jz exit
0109 E8 0158 R          C                   call clockint
010C 8B CB              C   exit:           mov cx,bx
010E C3                 C                   ret
010F                    C   clivekt         endp
                        C                   include alarm.asm
                        C   ;------ ALARM -------------------------------
                        C   ;
                        C   ; Routine for alarminterrupt from clock
```

```
                        C   ;
                        C   ; CX,DI,SI changed
                        C   ;
                        C   ;--------------------------------------------
                        C   ;
010F                      C         alarm           proc near
010F 8D 36 0135 R           C                       lea si,ala_str
0113 BF 0136              C                         mov di,310
0116 B9 000A              C                         mov cx,10
0119 E8 026C R              C                       call dispint
011C 50                   C                         push ax
011D E4 61                C                         in al,61h
011F 0C 03                C                         or al,03h
0121 E6 61                C                         out 61h,al
0123 50                   C                         push ax
0124 B8 2000                C                       mov ax,2000h
0127 50                   C         loopa:          push ax
0128 58                   C                         pop ax
0129 50                   C                         push ax
012A 58                   C                         pop ax
012B 48                   C                         dec ax
012C 75 F9                  C                       jnz loopa
012E 58                   C                         pop ax
012F 24 FC                  C                       and al,0FCh
0131 E6 61                C                         out 61h,al
0133 58                   C                         pop ax
0134 C3                   C                         ret
                        C   ;
0135 41 0F 4C 0F 41 0F    C   ala_str:              db  'A L A R M '
     52 0F 4D 0F            C
                        C   ;
013F                      C         alarm           endp
                        C                   include period.asm
                        C   ;------- PERIOD ---------------------------
                        C   ;
                        C   ; Interrupt routine for periodic interrupt from clock
                        C   ;
                        C   ; No registers changed
                        C   ;
                        C   ;--------------------------------------------
                        C   ;
013F                      C         period          proc near
013F 50                   C                         push ax
0140 E4 61                C                         in al,61h
0142 0C 03                C                         or al,03h
0144 E6 61                C                         out 61h,al
0146 50                   C                         push ax
0147 B8 0010                C                       mov ax,0010h
014A 50                   C         loop:           push ax
```

```
014B 58              C                      pop ax
014C 50              C                      push ax
014D 58              C                      pop ax
014E 48              C                      dec ax
014F 75 F9           C                      jnz loop
0151 58              C                      pop ax
0152 24 FC           C                      and al,0FCh
0154 E6 61           C                      out 61h,al
0156 58              C                      pop ax
0157 C3              C                      ret
0158                 C      period          endp
                     C                  include clockint.asm
                     C      ;-------- CLOCKINT ----------------------------
                     C      ;
                     C      ; DX=PC4680 base adress
                     C      ;
                     C      ; AX,CX,DI,SI changed
                     C      ;
                     C      ;-------------------------------------------
0158                 C  clockint proc near
0158 8D 36 01A0 R    C                      lea si,cli_str
015C BF 0090         C                      mov di,90h
015F B9 0010         C                      mov cx,10h
0162 56              C                      push si
0163 51              C                      push cx
0164 83 C6 0E        C                      add si,0Eh
0167 B4 00           C                      mov ah,00h
0169 E8 027A R       C      next:           call get
016C 8A E9           C                      mov ch,cl
016E 81 E1 F00F      C                      and cx,0F00Fh
0172 D0 CD           C                      ror ch,1
0174 D0 CD           C                      ror ch,1
0176 D0 CD           C                      ror ch,1
0178 D0 CD           C                      ror ch,1
017A 81 C1 3030      C                      add cx,3030h
017E 2E: 88 0C       C                      mov cs:[si],cl
0181 4E              C                      dec si
0182 4E              C                      dec si
0183 2E: 88 2C       C                      mov cs:[si],ch
0186 4E              C                      dec si
0187 4E              C                      dec si
0188 F6 C4 04        C                      test ah,4
018B 75 0D           C                      jnz display
018D FE C4           C                      inc ah
018F FE C4           C                      inc ah
0191 B0 3A           C                      mov al,3Ah
0193 2E: 88 04       C                      mov cs:[si],al
0196 4E              C                      dec si
0197 4E              C                      dec si
0198 EB CF           C                      jmp next
019A 59              C      display:        pop cx
019B 5E              C                      pop si
```

```
019C E8 026C R          C                       call dispint
019F C3                 C                       ret
                        C ;
01A0    08 [            C       cli_str:        dw 8 dup (' ')
          0F20          C
              ]         C
                        C
                        C ;
01B0                    C       clockint        endp
                        C                       include resin.asm
                        C ;---------- RESIN ----------------------------------
                        C ;
                        C ; DX=PC4680 base adress
                        C ;
                        C ; CX,DI,SI changed
                        C ;
                        C ;--------------------------------------------------
01B0                    C       resin           proc near
01B0 8D 36 01BE R       C                       lea si,res_str
01B4 BF 0000            C                       mov di,0000
01B7 B9 0024            C                       mov cx,36
01BA E8 026C R          C                       call dispint
01BD C3                 C                       ret
                        C ;
01BE 52 0F 65 0F 73 0F  C       res_str:        db 'R e s e t   o n   D a t a B
                                                 o a r d '
     65 0F 74 0F 20 0F  C
     6F 0F 6E 0F 20 0F  C
     44 0F 61 0F 74 0F  C
     61 0F 42 0F 6F 0F  C
     61 0F 72 0F 64 0F  C
                        C ;
01E2                    C       resin           endp
                        C                       include dbint.asm
                        C ;---------- DBINT ----------------------------------
                        C ;
                        C ; DX=PC4680 base adress
                        C ;
                        C ; CX,DI,SI changed
                        C ;
                        C ;--------------------------------------------------
01E2                    C       dbint           proc near
01E2 8D 36 01F8 R       C                       lea si,dbi_str
01E6 BF 0026            C                       mov di,0038
01E9 B9 002C            C                       mov cx,44
01EC E8 026C R          C                       call dispint
01EF 50                 C                       push ax
01F0 52                 C                       push dx
01F1 BA 0307            C                       mov dx,0307h
01F4 EC                 C                       in al,(dx)
```

```
01F5 5A                 C                       pop dx
01F6 58                 C                       pop ax
01F7 C3                 C                       ret
                        C ;
01F8 49 0F 6E 0F 74 0F  C  dbi_str:             db 'Interrupt on D
                                                ataBoard'
     65 0F 72 0F 72 0F  C
     75 0F 70 0F 74 0F  C
     20 0F 6F 0F 6E 0F  C
     20 0F 44 0F 61 0F  C
     74 0F 61 0F 42 0F  C
     6F 0F 61 0F 72 0F  C
     64 0F              C
                        C ;
0224                    C       dbint           endp
                        C                       include nolegint.asm
                        C ;---------- NOLEGINT ----------------------------------
                        C ;
                        C ; DX=PC4680 base adress
                        C ;
                        C ; CX,DI,SI changed
                        C ;
                        C ;-------------------------------------------------
0224                    C       nolegint        proc near
0224 8D 36 0232 R       C                       lea si,nli_str
0228 BF 0054            C                       mov di,0084
022B B9 003A            C                       mov cx,58
022E E8 026C R          C                       call dispint
0231 C3                 C                       ret
                        C ;
0232 49 0F 6C 0F 6C 0F  C  nli_str:             db 'Illegal interr
                                                upt has occured'
     65 0F 67 0F 61 0F  C
     6C 0F 20 0F 69 0F  C
     6E 0F 74 0F 65 0F  C
     72 0F 72 0F 75 0F  C
     70 0F 74 0F 20 0F  C
     68 0F 61 0F 73 0F  C
     20 0F 6F 0F 63 0F  C
     63 0F 75 0F 72 0F  C
     65 0F 64 0F        C
                        C ;
026C                    C       nolegint        endp
                        C                       include dispint.asm
                        C ;---------- DISPINT ----------------------------------
                        C ;
                        C ; SI=Startadress of string to display
                        C ; DI=Destinationadress for string
                        C ; CX=String length
                        C ;
                        C ;-------------------------------------------------
```

```
026C                    C       dispint         proc near
026C 06                  C                      push es
026D 52                  C                      push dx
026E 8B 16 02B1 R          C                    mov dx,disptype
0272 8E C2                C                     mov es,dx
0274 FC                   C                     cld
0275 F3/ A4               C                     rep movsb
0277 5A                   C                     pop dx
0278 07                   C                     pop es
0279 C3                   C                     ret
027A                    C       dispint         endp
                        C               include get.asm
                        C       ;-------- Läs data från klockan i PC4680 ----------
                        C       ;
                        C       ;  DX=basadress för kortet
                        C       ;  AH=adress i klockan
                        C       ;  CL=data från klockan
                        C       ;
                        C       ;  AX,CX changed
                        C       ;
                        C       ;-------------------------------------------------
027A                    C       get             proc near
027A 52                  C                      push dx
027B B0 83                C                    mov al,83H
027D 83 C2 0F               C                  add dx,0Fh
0280 EE                   C                     out (DX),AL
0281 4A                   C                     dec dx
0282 B0 D0                C                     mov al,0DH*10H
0284 EE                   C                     out (DX),al
0285 B0 50                C                     mov al,5*10H
0287 EE                   C                     out (DX),al
0288 B0 70                C                     mov al,7*10H
028A EE                   C                     out (DX),al
028B 8A C4                C                     mov al,ah
028D 4A                   C                     dec dx
028E 4A                   C                     dec dx
028F EE                   C                     out (DX),al
0290 B0 60                C                     mov al,6*10H
0292 42                   C                     inc dx
0293 42                   C                     inc dx
0294 EE                   C                     out (DX),al
0295 B0 40                C                     mov al,4*10H
0297 EE                   C                     out (DX),al
0298 B0 C0                C                     mov al,0CH*10H
029A EE                   C                     out (DX),al
029B B0 93                C                     mov al,93H
029D 42                   C                     inc dx
029E EE                   C                     out (DX),al
029F 4A                   C                     dec dx
02A0 4A                   C                     dec dx
02A1 4A                   C                     dec dx
```

```
02A2 EC                 C                       in al,(DX)
02A3 8A C8               C                      mov cl,al
02A5 B0 50               C                      mov al,5*10H
02A7 42                 C                       inc dx
02A8 42                 C                       inc dx
02A9 EE                 C                       out (DX),al
02AA B0 D0               C                      mov al,0DH*10H
02AC EE                 C                       out (DX),al
02AD 5A                 C                       pop dx
02AE C3                 C                       ret
02AF                    C               get     endp
                                        ;
02AF    01 [                            intmask dw 1 dup(?)
         ????
            ]

                                        ;
02B1    01 [                            disptype dw 1 dup (?)
         ????
            ]

                                        ;
02B3    01 [                            last_adr dw 1 dup(?)
         ????
            ]

                                        ;
02B5                                    cseg    ends
                                                end
```

Segments and Groups:

Name	Size	Align	Combine	Class
CSEG	02B5	PARA	PUBLIC	'CODE'
STACK	0800	PARA	STACK	'STACK'

Symbols:

Name	Type	Value	Attr	
AF	L NEAR	00FA	CSEG	
ALARM	N PROC	010F	CSEG	Length =0030
ALA_STR	L NEAR	0135	CSEG	
BIT_0	L NEAR	00D2	CSEG	
BIT_1	L NEAR	00CD	CSEG	
BIT_2	L NEAR	00C8	CSEG	
BIT_3	L NEAR	00C3	CSEG	
BIT_4	L NEAR	00BE	CSEG	
BIT_5	L NEAR	00B9	CSEG	
BIT_6	L NEAR	00B4	CSEG	
BIT_7	L NEAR	00AF	CSEG	
CLIVEKT	N PROC	00D7	CSEG	Length =0038
CLI_STR	L NEAR	01A0	CSEG	
CLOCKINT	N PROC	0158	CSEG	Length =0058
DBINT	N PROC	01E2	CSEG	Length =0042
DBI_STR	L NEAR	01F8	CSEG	
DISPINT	N PROC	026C	CSEG	Length =000E
DISPLAY	L NEAR	019A	CSEG	
DISPTYPE	L WORD	02B1	CSEG	
EXIT	L NEAR	010C	CSEG	
GET	N PROC	027A	CSEG	Length =0035
INT	F PROC	0052	CSEG	Length =0030
INTMASK	L WORD	02AF	CSEG	
INTTABLE	N PROC	00AF	CSEG	Length =0028
INTVEKT	N PROC	0082	CSEG	Length =002D
INTXT	F PROC	0000	CSEG	Length =0052
IRQF	L NEAR	00EB	CSEG	
LAST	L NEAR	003E	CSEG	
LAST_ADR	L WORD	02B3	CSEG	
LOOP	L NEAR	014A	CSEG	
LOOPA	L NEAR	0127	CSEG	
MONODISP	L NEAR	0038	CSEG	
MORE	L NEAR	0055	CSEG	
NEXT	L NEAR	0169	CSEG	
NEXTINT	L NEAR	00A4	CSEG	
NLI_STR	L NEAR	0232	CSEG	
NOLEGINT	N PROC	0224	CSEG	Length =0048
PERIOD	N PROC	013F	CSEG	Length =0019
PF	L NEAR	00F0	CSEG	
RESIN	N PROC	01B0	CSEG	Length =0032
RES_STR	L NEAR	01BE	CSEG	
TEST_BIT	L NEAR	0093	CSEG	

UP . . . . . L NEAR 0104 CSEG

48650 Bytes free

Warning Severe
Errors Errors
0 0

INT_ATCO

```
                        ;
                        ;--------- STACK ----------
                        ;
0000                    stack   segment para stack 'stack'
0000 0100 [                             db 256 dup ('stack   ')
         73 74 61 63
         6B 20 20 20
               ]

0800                    stack   ends
                        ;
                        ;---------------------------------
                        ;
                        ;
                        ;--------- CODE ----------
                        ;
0000                    cseg                segment para public 'code'
                                            assume cs:cseg,ds:cseg,es:cseg,
                        ss:stack
                        ;
                        ;--------- INITINT ----------
                        ;
0000                    intatco             proc far
0000 FA                                     cli
0001 1E                                     push ds
0002 06                                     push es
0003 8C CA                                  mov dx,cs
0005 8E C2                                  mov es,dx
0007 8E DA                                  mov ds,dx
0009 8D 16 003A R                           lea dx,int
000D B8 2571                                mov ax,2571h
0010 CD 21                                  int 21h
0012 E4 A1                                  in al,0A1h
0014 24 FD                                  and al,0FDh
0016 E6 A1                                  out 0A1h,al
0018 BA 0300                                mov dx,0300h
001B B4 0C                                  mov ah,0Ch
001D E8 0266 R                              call get
0020 C7 06 02D3 R B800                      mov disptype,0B800h
0026 8D 06 02D5 R                           lea ax,last_adr
002A 25 FFF0                                and ax,0FFF0h
002D 05 0010                                add ax,0010h
0030 8B D0                                  mov dx,ax
0032 B8 3100                                mov ax,3100h
0035 07                                     pop es
0036 1F                                     pop ds
0037 FB                                     sti
0038 CD 21                                  int 21h
003A                    intatco             endp
                        ;
```

```
                        ;-------- INT ---------------------------
                        ;
003A                    int             proc far
003A FA                                 cli
003B 50                                 push ax
003C 52                                 push dx
003D 51                 more:           push cx
003E 53                                 push bx
003F 57                                 push di
0040 56                                 push si
0041 55                                 push bp
0042 1E                                 push ds
0043 06                                 push es
0044 8C C8                              mov ax,cs
0046 8E D8                              mov ds,ax
0048 8E C0                              mov es,ax
004A BD 0000                            mov bp,0
004D BA 0300                            mov dx,0300h
0050 E8 006E R                          call intvekt
0053 B0 62                              mov al,62h
0055 E6 20                              out 20h,al
0057 B0 61                              mov al,61h
0059 E6 A0                              out 0A0h,al
005B 07                                 pop es
005C 1F                                 pop ds
005D 5D                                 pop bp
005E 5E                                 pop si
005F 5F                                 pop di
0060 5B                                 pop bx
0061 59                                 pop cx
0062 BA 0305                            mov dx,0305h
0065 EC                                 in al,(dx)
0066 34 FF                              xor al,0FFh
0068 75 D3                              jnz more
006A 5A                                 pop dx
006B 58                                 pop ax
006C FB                                 sti
006D CF                                 iret
006E                    int             endp
                        ;
                C               include intvekt.asm
                C       ;-------- INTVEKT ---------------------------
                C       ;
                C       ; DX=PC4680 base adress
                C       ;
                C       ; AX,BX,CX changed
                C       ;
                C       ;---------------------------------------
                C       ;
006E                      C     intvekt         proc near
006E 52                   C                     push dx
```

```
006F 83 C2 05          C                       add dx,05h
0072 EC              C                         in al,(dx)
0073 5A              C                         pop dx
0074 8A C8             C                       mov cl,al
0076 B5 05             C                       mov ch,05h
0078 B8 0008           C                       mov ax,0008h
007B 8D 1E 009B R        C                     lea bx,inttable
007F 50              C           test_bit:     push ax
0080 53              C                         push bx
0081 51              C                         push cx
0082 52              C                         push dx
0083 F6 C1 01          C                       test cl,01h
0086 75 08             C                       jnz nextint
0088 F6 ED             C                       imul ch
008A 2C 05             C                       sub al,05h
008C 03 D8             C                       add bx,ax
008E FF E3             C                       jmp bx
0090 5A              C           nextint:      pop dx
0091 59              C                         pop cx
0092 5B              C                         pop bx
0093 58              C                         pop ax
0094 D0 E9             C                       shr cl,1
0096 FE C8             C                       dec al
0098 75 E5             C                       jnz test_bit
009A C3                C                       ret
009B                 C           intvekt       endp
                   C                       include inttable.asm
                   C           ;-------- INTTABLE ------------------------------
                   C           ;
                   C           ; Interrupt routin call table for 'intvekt'
                   C           ;
                   C           ; No registers changed
                   C           ;
                   C           ;-----------------------------------------------
                   C           ;
009B                 C           inttable      proc near
                   C           ;
009B E8 019C R         C       bit_7:          call resin
009E EB F0             C                       jmp nextint
00A0 E8 01CE R           C     bit_6:          call dbint
00A3 EB EB             C                       jmp nextint
00A5 E8 0210 R           C     bit_5:          call nolegint
00A8 EB E6             C                       jmp nextint
00AA E8 0210 R           C     bit_4:          call nolegint
00AD EB E1             C                       jmp nextint
00AF E8 0210 R           C     bit_3:          call nolegint
00B2 EB DC             C                       jmp nextint
00B4 E8 0210 R           C     bit_2:          call nolegint
00B7 EB D7             C                       jmp nextint
00B9 E8 0210 R           C     bit_1:          call nolegint
00BC EB D2             C                       jmp nextint
00BE E8 00C3 R           C     bit_0:          call clivekt
```

```
00C1 EB CD                C                     jmp nextint
                        C ;
00C3                      C       inttable      endp
                        C                 include clivekt.asm
                        C ;------ CLIVEKT ------------------------------
                        C ;
                        C ; DX=PC4680 base adress
                        C ;
                        C ; Interrupt routine call select for clock inte
                                  rrupts
                        C ;
                        C ;-------------------------------------------
                        C ;
00C3                      C       clivekt       proc near
00C3 B4 0B                C                     mov ah,0Bh
00C5 E8 0266 R            C                     call get
00C8 8A C1                C                     mov al,cl
00CA 0C 8F                C                     or al,8Fh
00CC 50                   C                     push ax
00CD B4 0C                C                     mov ah,0Ch
00CF E8 0266 R            C                     call get
00D2 58                   C                     pop ax
00D3 22 C8                C                     and cl,al
00D5 8B D9                C                     mov bx,cx
00D7 F6 C1 80             C       IRQF:         test cl,80h
00DA 74 1C                C                     jz exit
00DC F6 C1 40             C       PF:           test cl,40h
00DF 74 05                C                     jz AF
00E1 E8 012B R            C                     call period
00E4 8B CB                C                     mov cx,bx
00E6 F6 C1 20             C       AF:           test cl,20h
00E9 74 05                C                     jz UF
00EB E8 00FB R            C                     call alarm
00EE 8B CB                C                     mov cx,bx
00F0 F6 C1 10             C       UF:           test cl,10h
00F3 74 03                C                     jz exit
00F5 E8 0144 R            C                     call clockint
00F8 8B CB                C       exit:         mov cx,bx
00FA C3                   C                     ret
00FB                      C       clivekt       endp
                        C                 include alarm.asm
                        C ;------ ALARM ------------------------------
                        C ;
                        C ; Routine for alarminterrupt from clock
                        C ;
                        C ; CX,DI,SI changed
                        C ;
                        C ;-------------------------------------------
                        C ;
00FB                      C       alarm         proc near
```

```
00FB  8D 36 0121 R      C               lea si,ala_str
00FF  BF 0136           C               mov di,310
0102  B9 000A           C               mov cx,10
0105  E8 0258 R         C               call dispint
0108  50                C               push ax
0109  E4 61             C               in al,61h
010B  0C 03             C               or al,03h
010D  E6 61             C               out 61h,al
010F  50                C               push ax
0110  B8 2000           C               mov ax,2000h
0113  50                C       loopa:  push ax
0114  58                C               pop ax
0115  50                C               push ax
0116  58                C               pop ax
0117  48                C               dec ax
0118  75 F9             C               jnz loopa
011A  58                C               pop ax
011B  24 FC             C               and al,0FCh
011D  E6 61             C               out 61h,al
011F  58                C               pop ax
0120  C3                C               ret
                        C       ;
0121  41 0F 4C 0F 41 0F C      ala_str:  db  'A L A R M'
      52 0F 4D 0F       C
                        C       ;
012B                    C       alarm   endp
                        C               include period.asm
                        C       ;-------- PERIOD ----------------------------
                        C       ;
                        C       ; Interrupt routine for periodic interrupt from clock
                        C       ;
                        C       ; No registers changed
                        C       ;
                        C       ;--------------------------------------------
                        C       ;
012B                    C       period  proc near
012B  50                C               push ax
012C  E4 61             C               in al,61h
012E  0C 03             C               or al,03h
0130  E6 61             C               out 61h,al
0132  50                C               push ax
0133  B8 0010           C               mov ax,0010h
0136  50                C       loop:   push ax
0137  58                C               pop ax
0138  50                C               push ax
0139  58                C               pop ax
013A  48                C               dec ax
013B  75 F9             C               jnz loop
013D  58                C               pop ax
013E  24 FC             C               and al,0FCh
0140  E6 61             C               out 61h,al
```

```
0142 58                 C                       pop ax
0143 C3                 C                       ret
0144                    C       period          endp
                        C                   include clockint.asm
                        C   ;---------- CLOCKINT ----------------------------
                        C   ;
                        C   ; DX=PC4680 base adress
                        C   ;
                        C   ; AX,CX,DI,SI changed
                        C   ;
                        C   ;-----------------------------------------------
0144                    C       clockint        proc near
0144 8D 36 018C R       C                       lea si,cli_str
0148 BF 0090            C                       mov di,90h
014B B9 0010            C                       mov cx,10h
014E 56                 C                       push si
014F 51                 C                       push cx
0150 83 C6 0E           C                       add si,0Eh
0153 B4 00              C                       mov ah,00h
0155 E8 0266 R          C       next:           call get
0158 8A E9              C                       mov ch,cl
015A 81 E1 F00F         C                       and cx,0F00Fh
015E D0 CD              C                       ror ch,1
0160 D0 CD              C                       ror ch,1
0162 D0 CD              C                       ror ch,1
0164 D0 CD              C                       ror ch,1
0166 81 C1 3030         C                       add cx,3030h
016A 2E: 88 0C          C                       mov cs:[si],cl
016D 4E                 C                       dec si
016E 4E                 C                       dec si
016F 2E: 88 2C          C                       mov cs:[si],ch
0172 4E                 C                       dec si
0173 4E                 C                       dec si
0174 F6 C4 04           C                       test ah,4
0177 75 0D              C                       jnz display
0179 FE C4              C                       inc ah
017B FE C4              C                       inc ah
017D B0 3A              C                       mov al,3Ah
017F 2E: 88 04          C                       mov cs:[si],al
0182 4E                 C                       dec si
0183 4E                 C                       dec si
0184 EB CF              C                       jmp next
0186 59                 C       display:        pop cx
0187 5E                 C                       pop si
0188 E8 0258 R          C                       call dispint
018B C3                 C                       ret
                        C   ;
018C    08 [            C       cli_str:        dw 8 dup (' ')
           0F20         C
             ]          C
                        C
                        C   ;
```

```
019C                    C       clockint        endp
                        C                       include resin.asm
                        C       ;---------- RESIN ------------------------------
                        C       ;
                        C       ; DX=PC4680 base adress
                        C       ;
                        C       ; CX,DI,SI changed
                        C       ;
                        C       ;-----------------------------------------------
019C                    C       resin           proc near
019C 8D 36 01AA R       C                       lea si,res_str
01A0 BF 0000            C                       mov di,0000
01A3 B9 0024            C                       mov cx,36
01A6 E8 0258 R          C                       call dispint
01A9 C3                 C                       ret
                        C       ;
01AA 52 0F 65 0F 73 0F  C       res_str:        db 'Reset on DataB
                                                oard'
     65 0F 74 0F 20 0F  C
     6F 0F 6E 0F 20 0F  C
     44 0F 61 0F 74 0F  C
     61 0F 42 0F 6F 0F  C
     61 0F 72 0F 64 0F  C
                        C       ;
01CE                    C       resin           endp
                        C                       include dbint.asm
                        C       ;---------- DBINT ------------------------------
                        C       ;
                        C       ; DX=PC4680 base adress
                        C       ;
                        C       ; CX,DI,SI changed
                        C       ;
                        C       ;-----------------------------------------------
01CE                    C       dbint           proc near
01CE 8D 36 01E4 R       C                       lea si,dbi_str
01D2 BF 0026            C                       mov di,0038
01D5 B9 002C            C                       mov cx,44
01D8 E8 0258 R          C                       call dispint
01DB 50                 C                       push ax
01DC 52                 C                       push dx
01DD BA 0307            C                       mov dx,0307h
01E0 EC                 C                       in al,(dx)
01E1 5A                 C                       pop dx
01E2 58                 C                       pop ax
01E3 C3                 C                       ret
                        C       ;
01E4 49 0F 6E 0F 74 0F  C       dbi_str:        db 'Interrupt on D
                                                ataBoard'
     65 0F 72 0F 72 0F  C
     75 0F 70 0F 74 0F  C
```

```
    20 0F 6F 0F 6E 0F    C
    20 0F 44 0F 61 0F    C
    74 0F 61 0F 42 0F    C
    6F 0F 61 0F 72 0F    C
    64 0F                C
                         C ;
0210                     C          dbint           endp
                         C                  include nolegint.asm
                         C ;--------- NOLEGINT ----------------------------------
                         C ;
                         C ; DX=PC4680 base adress
                         C ;
                         C ; CX,DI,SI changed
                         C ;
                         C ;-------------------------------------------
0210                     C          nolegint        proc near
0210 8D 36 021E R        C                          lea si,nli_str
0214 BF 0054             C                          mov di,0084
0217 B9 003A             C                          mov cx,58
021A E8 0258 R           C                          call dispint
021D C3                  C                          ret
                         C ;
021E 49 0F 6C 0F 6C 0F   C  nli_str:                db 'Illegal   interr
                                                     upt   has   occured'
    65 0F 67 0F 61 0F    C
    6C 0F 20 0F 69 0F    C
    6E 0F 74 0F 65 0F    C
    72 0F 72 0F 75 0F    C
    70 0F 74 0F 20 0F    C
    68 0F 61 0F 73 0F    C
    20 0F 6F 0F 63 0F    C
    63 0F 75 0F 72 0F    C
    65 0F 64 0F          C
                         C ;
0258                     C          nolegint        endp
                         C                  include dispint.asm
                         C ;---------- DISPINT ----------------------------------
                         C ;
                         C ; SI=Startadress of string to display
                         C ; DI=Destinationadress for string
                         C ; CX=String length
                         C ;
                         C ;-------------------------------------------
0258                     C          dispint         proc near
0258 06                  C                          push es
0259 52                  C                          push dx
025A 8B 16 02D3 R        C                          mov dx,disptype
025E 8E C2               C                          mov es,dx
0260 FC                  C                          cld
0261 F3/ A4              C                          rep movsb
```

```
0263 5A              C                      pop dx
0264 07              C                      pop es
0265 C3              C                      ret
0266                 C          dispint     endp
                     C                  include get.asm
                     C          ;-------- Läs data från klockan i PC4680 ----------
                     C          ;
                     C          ; DX=basadress för kortet
                     C          ; AH=adress i klockan
                     C          ; CL=data från klockan
                     C          ;
                     C          ; AX,CX changed
                     C          ;
                     C          ;-----------------------------------------------
0266                 C          get         proc near
0266 52              C                      push dx
0267 B0 83           C                      mov al,83H
0269 83 C2 0F        C                      add dx,0Fh
026C EE              C                      out (DX),AL
026D 4A              C                      dec dx
026E B0 D0           C                      mov al,0DH*10H
0270 EE              C                      out (DX),al
0271 B0 50           C                      mov al,5*10H
0273 EE              C                      out (DX),al
0274 B0 70           C                      mov al,7*10H
0276 EE              C                      out (DX),al
0277 8A C4           C                      mov al,ah
0279 4A              C                      dec dx
027A 4A              C                      dec dx
027B EE              C                      out (DX),al
027C B0 60           C                      mov al,6*10H
027E 42              C                      inc dx
027F 42              C                      inc dx
0280 EE              C                      out (DX),al
0281 B0 40           C                      mov al,4*10H
0283 EE              C                      out (DX),al
0284 B0 C0           C                      mov al,0CH*10H
0286 EE              C                      out (DX),al
0287 B0 93           C                      mov al,93H
0289 42              C                      inc dx
028A EE              C                      out (DX),al
028B 4A              C                      dec dx
028C 4A              C                      dec dx
028D 4A              C                      dec dx
028E EC              C                      in al,(DX)
028F 8A C8           C                      mov cl,al
0291 B0 50           C                      mov al,5*10H
0293 42              C                      inc dx
0294 42              C                      inc dx
0295 EE              C                      out (DX),al
0296 B0 D0           C                      mov al,0DH*10H
0298 EE              C                      out (DX),al
```

```
0299 5A                 C                       pop dx
029A C3                 C                       ret
029B                    C       get             endp
                        C                       include set.asm
                        C       ;-------- Skriv data till klockan i PC4680 ----------
                        C       ;
                        C       ; DX=basadress för kortet
                        C       ; AH=adress i klockan
                        C       ; CL=data till klockan
                        C       ;
                        C       ;-----------------------------------------
                        C       ;
029B                    C       set             proc near
029B 52                 C                       push dx
029C B0 83              C                       mov AL,83H
029E 83 C2 0F           C                       add dx,0Fh
02A1 EE                 C                       out (DX),AL
02A2 4A                 C                       dec dx
02A3 B0 90              C                       mov AL,9*10H
02A5 EE                 C                       out (DX),AL
02A6 B0 10              C                       mov AL,1*10H
02A8 EE                 C                       out (DX),AL
02A9 B0 30              C                       mov AL,3*10H
02AB EE                 C                       out (DX),AL
02AC 8A C4              C                       mov AL,AH
02AE 4A                 C                       dec dx
02AF 4A                 C                       dec dx
02B0 EE                 C                       out (DX),AL
02B1 B0 20              C                       mov AL,2*10H
02B3 42                 C                       inc dx
02B4 42                 C                       inc dx
02B5 EE                 C                       out (DX),AL
02B6 B0 00              C                       mov AL,0*10H
02B8 EE                 C                       out (DX),AL
02B9 B0 80              C                       mov AL,8*10H
02BB EE                 C                       out (DX),AL
02BC 8A C1              C                       mov AL,CL
02BE 4A                 C                       dec dx
02BF 4A                 C                       dec dx
02C0 EE                 C                       out (DX),AL
02C1 B0 10              C                       mov AL,1*10H
02C3 42                 C                       inc dx
02C4 42                 C                       inc dx
02C5 EE                 C                       out (DX),AL
02C6 B0 90              C                       mov AL,9*10H
02C8 EE                 C                       out (DX),AL
02C9 B0 93              C                       mov AL,93H
02CB 42                 C                       inc dx
02CC EE                 C                       out (DX),AL
02CD 5A                 C                       pop dx
02CE C3                 C                       ret
02CF                    C       set             endp
```

```
                                ;
02CF  01 [                      im_master       dw 1 dup (?)
         ????
            ]

                                ;
02D1  01 [                      im_slave        dw 1 dup (?)
         ????
            ]

                                ;
02D3  01 [                      disptype dw 1 dup (?)
         ????
            ]

                                ;
02D5  01 [                      last_adr  dw 1 dup (?)
         ????
            ]

                                ;
02D7                            cseg            ends
                                                end
```

Segments and Groups:

Name	Size	Align	Combine	Class
CSEG	02D7	PARA	PUBLIC	'CODE'
STACK	0800	PARA	STACK	'STACK'

Symbols:

Name	Type	Value	Attr		
AF	L NEAR		00E6	CSEG	
ALARM	N PROC		00FB	CSEG	Length =0030
ALA_STR	L NEAR		0121	CSEG	
BIT_0	L NEAR		00BE	CSEG	
BIT_1	L NEAR		00B9	CSEG	
BIT_2	L NEAR		00B4	CSEG	
BIT_3	L NEAR		00AF	CSEG	
BIT_4	L NEAR		00AA	CSEG	
BIT_5	L NEAR		00A5	CSEG	
BIT_6	L NEAR		00A0	CSEG	
BIT_7	L NEAR		009B	CSEG	
CLIVEKT	N PROC		00C3	CSEG	Length =0038
CLI_STR	L NEAR		018C	CSEG	
CLOCKINT	N PROC		0144	CSEG	Length =0058
DBINT	N PROC		01CE	CSEG	Length =0042
DBI_STR	L NEAR		01E4	CSEG	
DISPINT	N PROC		0258	CSEG	Length =000E
DISPLAY	L NEAR		0186	CSEG	
DISPTYPE	L WORD		02D3	CSEG	
EXIT	L NEAR		00F8	CSEG	
GET	N PROC		0266	CSEG	Length =0035
IM_MASTER	L WORD		02CF	CSEG	
IM_SLAVE	L WORD		02D1	CSEG	
INT	F PROC		003A	CSEG	Length =0034
INTATCO	F PROC		0000	CSEG	Length =003A
INTTABLE	N PROC		009B	CSEG	Length =0028
INTVEKT	N PROC		006E	CSEG	Length =002D
IRQF	L NEAR		00D7	CSEG	
LAST_ADR	L WORD		02D5	CSEG	
LOOP	L NEAR		0136	CSEG	
LOOPA	L NEAR		0113	CSEG	
MORE	L NEAR		003D	CSEG	
NEXT	L NEAR		0155	CSEG	
NEXTINT	L NEAR		0090	CSEG	
NLI_STR	L NEAR		021E	CSEG	
NOLEGINT	N PROC		0210	CSEG	Length =0048
PERIOD	N PROC		012B	CSEG	Length =0019
PF	L NEAR		00DC	CSEG	
RESIN	N PROC		019C	CSEG	Length =0032
RES_STR	L NEAR		01AA	CSEG	
SET	N PROC		029B	CSEG	Length =0034
TEST_BIT	L NEAR		007F	CSEG	

UF . . . . . L NEAR 00F0 CSEG

48642 Bytes free

Warning Severe
Errors Errors
0 0

INT_ATMO

```
                                ;
                                ;-------- STACK ----------------------
                                ;
0000                            stack   segment para stack 'stack'
0000 0100 [                             db 256 dup ('stack   ')
         73 74 61 63
         6B 20 20 20
             ]

0800                            stack   ends
                                ;
                                ;-------------------------------------
                                ;
                                ;
                                ;-------- CODE ----------------------
                                ;
0000                            cseg            segment para public 'code'
                                                assume cs:cseg,ds:cseg,es:cseg,ss:stack
                                ;
                                ;-------- INITINT -------------------
                                ;
0000                            intatmo         proc far
0000 FA                                         cli
0001 1E                                         push ds
0002 06                                         push es
0003 8C CA                                      mov dx,cs
0005 8E C2                                      mov es,dx
0007 8E DA                                      mov ds,dx
0009 8D 16 003A R                               lea dx,int
000D B8 2571                                    mov ax,2571h
0010 CD 21                                      int 21h
0012 E4 A1                                      in al,0A1h
0014 24 FD                                      and al,0FDh
0016 E6 A1                                      out 0A1h,al
0018 BA 0300                                    mov dx,0300h
001B B4 0C                                      mov ah,0Ch
001D E8 0266 R                                  call get
0020 C7 06 029F R B000                          mov disptype,0B000h
0026 8D 06 02A1 R                               lea ax,last_adr
002A 25 FFF0                                    and ax,0FFF0h
002D 05 0010                                    add ax,0010h
0030 8B D0                                      mov dx,ax
0032 B8 3100                                    mov ax,3100h
0035 07                                         pop es
0036 1F                                         pop ds
0037 FB                                         sti
0038 CD 21                                      int 21h
003A                            intatmo         endp
                                ;
```

```
                        ;------ INT ---------------------------
                        ;
003A                    int             proc far
003A FA                                 cli
003B 50                                 push ax
003C 52                                 push dx
003D 51                 more:           push cx
003E 53                                 push bx
003F 57                                 push di
0040 56                                 push si
0041 55                                 push bp
0042 1E                                 push ds
0043 06                                 push es
0044 8C C8                              mov ax,cs
0046 8E D8                              mov ds,ax
0048 8E C0                              mov es,ax
004A BD 0000                            mov bp,0
004D BA 0300                            mov dx,0300h
0050 E8 006E R                          call intvekt
0053 B0 62                              mov al,62h
0055 E6 20                              out 20h,al
0057 B0 61                              mov al,61h
0059 E6 A0                              out 0A0h,al
005B 07                                 pop es
005C 1F                                 pop ds
005D 5D                                 pop bp
005E 5E                                 pop si
005F 5F                                 pop di
0060 5B                                 pop bx
0061 59                                 pop cx
0062 BA 0305                            mov dx,0305h
0065 EC                                 in al,(dx)
0066 34 FF                              xor al,0FFh
0068 75 D3                              jnz more
006A 5A                                 pop dx
006B 58                                 pop ax
006C FB                                 sti
006D CF                                 iret
006E                    int             endp
                        ;
                C               include intvekt.asm
                C       ;------ INTVEKT ---------------------------
                C       ;
                C       ; DX=PC4680 base adress
                C       ;
                C       ; AX,BX,CX changed
                C       ;
                C       ;-----------------------------------------
                C       ;
006E                      C     intvekt         proc near
006E 52                   C                     push dx
```

```
006F 83 C2 05          C                add dx,05h
0072 EC                C                in al,(dx)
0073 5A                C                pop dx
0074 8A C8             C                mov cl,al
0076 B5 05             C                mov ch,05h
0078 B8 0008           C                mov ax,0008h
007B 8D 1E 009B R      C                lea bx,inttable
007F 50                C    test_bit:   push ax
0080 53                C                push bx
0081 51                C                push cx
0082 52                C                push dx
0083 F6 C1 01          C                test cl,01h
0086 75 08             C                jnz nextint
0088 F6 ED             C                imul ch
008A 2C 05             C                sub al,05h
008C 03 D8             C                add bx,ax
008E FF E3             C                jmp bx
0090 5A                C    nextint:    pop dx
0091 59                C                pop cx
0092 5B                C                pop bx
0093 58                C                pop ax
0094 D0 E9             C                shr cl,1
0096 FE C8             C                dec al
0098 75 E5             C                jnz test_bit
009A C3                C                ret
009B                   C    intvekt     endp
                       C                include inttable.asm
                       C    ;---------- INTTABLE ----------------------------------
                       C    ;
                       C    ; Interrupt routin call table for 'intvekt'
                       C    ;
                       C    ; No registers changed
                       C    ;
                       C    ;-----------------------------------------------------
                       C    ;
009B                   C    inttable    proc near
                       C    ;
009B E8 019C R         C    bit_7:      call resin
009E EB F0             C                jmp nextint
00A0 E8 01CE R         C    bit_6:      call dbint
00A3 EB EB             C                jmp nextint
00A5 E8 0210 R         C    bit_5:      call nolegint
00A8 EB E6             C                jmp nextint
00AA E8 0210 R         C    bit_4:      call nolegint
00AD EB E1             C                jmp nextint
00AF E8 0210 R         C    bit_3:      call nolegint
00B2 EB DC             C                jmp nextint
00B4 E8 0210 R         C    bit_2:      call nolegint
00B7 EB D7             C                jmp nextint
00B9 E8 0210 R         C    bit_1:      call nolegint
00BC EB D2             C                jmp nextint
00BE E8 00C3 R         C    bit_0:      call clivekt
```

```
00C1 EB CD              C                       jmp nextint
                        C   ;
00C3                    C   inttable            endp
                        C                   include clivekt.asm
                        C   ;------ CLIVEKT ----------------------------
                        C   ;
                        C   ; DX=PC4680 base adress
                        C   ;
                        C   ; Interrupt routine call select for clock inte
                                rrupts
                        C   ;
                        C   ;-----------------------------------------
                        C   ;
00C3                      C clivekt             proc near
00C3 B4 0B                C                     mov ah,0Bh
00C5 E8 0266 R            C                     call get
00C8 8A C1                C                     mov al,cl
00CA 0C 8F                C                     or al,8Fh
00CC 50                   C                     push ax
00CD B4 0C                C                     mov ah,0Ch
00CF E8 0266 R            C                     call get
00D2 58                   C                     pop ax
00D3 22 C8                C                     and cl,al
00D5 8B D9                C                     mov bx,cx
00D7 F6 C1 80             C   IRQF:             test cl,80h
00DA 74 1C                C                     jz exit
00DC F6 C1 40             C   PF:               test cl,40h
00DF 74 05                C                     jz AF
00E1 E8 012B R            C                     call period
00E4 8B CB                C                     mov cx,bx
00E6 F6 C1 20             C   AF:               test cl,20h
00E9 74 05                C                     jz UF
00EB E8 00FB R            C                     call alarm
00EE 8B CB                C                     mov cx,bx
00F0 F6 C1 10             C   UF:               test cl,10h
00F3 74 03                C                     jz exit
00F5 E8 0144 R            C                     call clockint
00F8 8B CB                C   exit:             mov cx,bx
00FA C3                   C                     ret
00FB                      C   clivekt           endp
                        C                   include alarm.asm
                        C   ;------ ALARM ----------------------------------
                        C   ;
                        C   ; Routine for alarminterrupt from clock
                        C   ;
                        C   ; CX,DI,SI changed
                        C   ;
                        C   ;-----------------------------------------
                        C   ;
00FB                      C alarm               proc near
```

```
00FB 8D 36 0121 R       C               lea si,ala_str
00FF BF 0136            C               mov di,310
0102 B9 000A            C               mov cx,10
0105 E8 0258 R          C               call dispint
0108 50                 C               push ax
0109 E4 61              C               in al,61h
010B 0C 03              C               or al,03h
010D E6 61              C               out 61h,al
010F 50                 C               push ax
0110 B8 2000            C               mov ax,2000h
0113 50                 C       loopa:  push ax
0114 58                 C               pop ax
0115 50                 C               push ax
0116 58                 C               pop ax
0117 48                 C               dec ax
0118 75 F9              C               jnz loopa
011A 58                 C               pop ax
011B 24 FC              C               and al,0FCh
011D E6 61              C               out 61h,al
011F 58                 C               pop ax
0120 C3                 C               ret
                        C ;
0121 41 0F 4C 0F 41 0F  C   ala_str:    db 'A L A R M '
     52 0F 4D 0F        C
                        C ;
012B                    C       alarm   endp
                        C               include period.asm
                        C ;-------- PERIOD --------------------------------
                        C ;
                        C ; Interrupt routine for periodic interrupt fro
                                m clock
                        C ;
                        C ; No registers changed
                        C ;
                        C ;-------------------------------------------
                        C ;
012B                    C       period  proc near
012B 50                 C               push ax
012C E4 61              C               in al,61h
012E 0C 03              C               or al,03h
0130 E6 61              C               out 61h,al
0132 50                 C               push ax
0133 B8 0010            C               mov ax,0010h
0136 50                 C       loop:   push ax
0137 58                 C               pop ax
0138 50                 C               push ax
0139 58                 C               pop ax
013A 48                 C               dec ax
013B 75 F9              C               jnz loop
013D 58                 C               pop ax
013E 24 FC              C               and al,0FCh
```

```
0140 E6 61              C                       out 61h,al
0142 58                 C                       pop ax
0143 C3                 C                       ret
0144                    C       period          endp
                        C                       include clockint.asm
                        C       ;---------- CLOCKINT ----------------------------
                        C       ;
                        C       ; DX=PC4680 base adress
                        C       ;
                        C       ; AX,CX,DI,SI changed
                        C       ;
                        C       ;-----------------------------------------------
0144                    C       clockint        proc near
0144 8D 36 018C R       C                       lea si,cli_str
0148 BF 0090            C                       mov di,90h
014B B9 0010            C                       mov cx,10h
014E 56                 C                       push si
014F 51                 C                       push cx
0150 83 C6 0E           C                       add si,0Eh
0153 B4 00              C                       mov ah,00h
0155 E8 0266 R          C       next:           call get
0158 8A E9              C                       mov ch,cl
015A 81 E1 F00F         C                       and cx,0F00Fh
015E D0 CD              C                       ror ch,1
0160 D0 CD              C                       ror ch,1
0162 D0 CD              C                       ror ch,1
0164 D0 CD              C                       ror ch,1
0166 81 C1 3030         C                       add cx,3030h
016A 2E: 88 0C          C                       mov cs:[si],cl
016D 4E                 C                       dec si
016E 4E                 C                       dec si
016F 2E: 88 2C          C                       mov cs:[si],ch
0172 4E                 C                       dec si
0173 4E                 C                       dec si
0174 F6 C4 04           C                       test ah,4
0177 75 0D              C                       jnz display
0179 FE C4              C                       inc ah
017B FE C4              C                       inc ah
017D B0 3A              C                       mov al,3Ah
017F 2E: 88 04          C                       mov cs:[si],al
0182 4E                 C                       dec si
0183 4E                 C                       dec si
0184 EB CF              C                       jmp next
0186 59                 C       display:        pop cx
0187 5E                 C                       pop si
0188 E8 0258 R          C                       call dispint
018B C3                 C                       ret
                        C       ;
018C    08 [            C       cli_str:        dw 8 dup (' ')
           0F20         C
                ]       C
                        C
```

```
                        C ;
019C                    C       clockint        endp
                        C                       include resin.asm
                        C ;-------- RESIN ------------------------------
                        C ;
                        C ; DX=PC4680 base adress
                        C ;
                        C ; CX,DI,SI changed
                        C ;
                        C ;-------------------------------------------
019C                    C       resin           proc near
019C 8D 36 01AA R       C                       lea si,res_str
01A0 BF 0000            C                       mov di,0000
01A3 B9 0024            C                       mov cx,36
01A6 E8 0258 R          C                       call dispint
01A9 C3                 C                       ret
                        C ;
01AA 52 0F 65 0F 73 0F  C   res_str:            db 'R e s e t  o n  D a t a B
                                                o a r d '
     65 0F 74 0F 20 0F  C
     6F 0F 6E 0F 20 0F  C
     44 0F 61 0F 74 0F  C
     61 0F 42 0F 6F 0F  C
     61 0F 72 0F 64 0F  C
                        C ;
01CE                    C       resin           endp
                        C                       include dbint.asm
                        C ;-------- DBINT ------------------------------
                        C ;
                        C ; DX=PC4680 base adress
                        C ;
                        C ; CX,DI,SI changed
                        C ;
                        C ;-------------------------------------------
01CE                    C       dbint           proc near
01CE 8D 36 01E4 R       C                       lea si,dbi_str
01D2 BF 0026            C                       mov di,0038
01D5 B9 002C            C                       mov cx,44
01D8 E8 0258 R          C                       call dispint
01DB 50                 C                       push ax
01DC 52                 C                       push dx
01DD BA 0307            C                       mov dx,0307h
01E0 EC                 C                       in al,(dx)
01E1 5A                 C                       pop dx
01E2 58                 C                       pop ax
01E3 C3                 C                       ret
                        C ;
01E4 49 0F 6E 0F 74 0F  C   dbi_str:            db 'I n t e r r u p t  o n  D
                                                a t a B o a r d '
     65 0F 72 0F 72 0F  C
```

```
   75 0F 70 0F 74 0F    C
   20 0F 6F 0F 6E 0F    C
   20 0F 44 0F 61 0F    C
   74 0F 61 0F 42 0F    C
   6F 0F 61 0F 72 0F    C
   64 0F                C
                        C ;
0210                    C       dbint           endp
                        C                   include nolegint.asm
                        C ;-------- NOLEGINT ----------------------------------
                        C ;
                        C ; DX=PC4680 base adress
                        C ;
                        C ; CX,DI,SI changed
                        C ;
                        C ;-------------------------------------------------
0210                    C       nolegint        proc near
0210 8D 36 021E R       C                       lea si,nli_str
0214 BF 0054            C                       mov di,0084
0217 B9 003A            C                       mov cx,58
021A E8 0258 R          C                       call dispint
021D C3                 C                       ret
                        C ;
021E 49 0F 6C 0F 6C 0F  C   nli_str:            db 'Illegal interr
                                                upt has occured'
   65 0F 67 0F 61 0F    C
   6C 0F 20 0F 69 0F    C
   6E 0F 74 0F 65 0F    C
   72 0F 72 0F 75 0F    C
   70 0F 74 0F 20 0F    C
   68 0F 61 0F 73 0F    C
   20 0F 6F 0F 63 0F    C
   63 0F 75 0F 72 0F    C
   65 0F 64 0F          C
                        C ;
0258                    C       nolegint        endp
                        C                   include dispint.asm
                        C ;-------- DISPINT ----------------------------------
                        C ;
                        C ; SI=Startadress of string to display
                        C ; DI=Destinationadress for string
                        C ; CX=String length
                        C ;
                        C ;-------------------------------------------------
0258                    C       dispint         proc near
0258 06                 C                       push es
0259 52                 C                       push dx
025A 8B 16 029F R       C                       mov dx,disptype
025E 8E C2              C                       mov es,dx
0260 FC                 C                       cld
```

```
0261 F3/ A4          C                  rep movsb
0263 5A              C                  pop dx
0264 07              C                  pop es
0265 C3              C                  ret
0266                 C      dispint     endp
                     C                  include get.asm
                     C      ;------ Lås data från klockan i PC4680 ----------
                     C      ;
                     C      ; DX=basadress för kortet
                     C      ; AH=adress i klockan
                     C      ; CL=data från klockan
                     C      ;
                     C      ; AX,CX changed
                     C      ;
                     C      ;-------------------------------------------------
0266                 C      get         proc near
0266 52              C                  push dx
0267 B0 83           C                  mov al,83H
0269 83 C2 0F        C                  add dx,0Fh
026C EE              C                  out (DX),AL
026D 4A              C                  dec dx
026E B0 D0           C                  mov al,0DH*10H
0270 EE              C                  out (DX),al
0271 B0 50           C                  mov al,5*10H
0273 EE              C                  out (DX),al
0274 B0 70           C                  mov al,7*10H
0276 EE              C                  out (DX),al
0277 8A C4           C                  mov al,ah
0279 4A              C                  dec dx
027A 4A              C                  dec dx
027B EE              C                  out (DX),al
027C B0 60           C                  mov al,6*10H
027E 42              C                  inc dx
027F 42              C                  inc dx
0280 EE              C                  out (DX),al
0281 B0 40           C                  mov al,4*10H
0283 EE              C                  out (DX),al
0284 B0 C0           C                  mov al,0CH*10H
0286 EE              C                  out (DX),al
0287 B0 93           C                  mov al,93H
0289 42              C                  inc dx
028A EE              C                  out (DX),al
028B 4A              C                  dec dx
028C 4A              C                  dec dx
028D 4A              C                  dec dx
028E EC              C                  in al,(DX)
028F 8A C8           C                  mov cl,al
0291 B0 50           C                  mov al,5*10H
0293 42              C                  inc dx
0294 42              C                  inc dx
0295 EE              C                  out (DX),al
0296 B0 D0           C                  mov al,0DH*10H
```

```
0298 EE                 C                       out (DX),al
0299 5A                 C                       pop dx
029A C3                 C                       ret
029B                    C       get             endp
                                ;
029B    01 [                    im_master       dw 1 dup (?)
            ????
          ]

                                ;
029D    01 [                    im_slave        dw 1 dup (?)
            ????
          ]

                                ;
029F    01 [                    disptype        dw 1 dup (?)
            ????
          ]

                                ;
02A1    01 [                    last_adr        dw 1 dup (?)
            ????
          ]

                                ;
02A3                            cseg            ends
                                                end
```

Segments and Groups:

Name	Size	Align	Combine	Class
CSEG	02A3	PARA	PUBLIC	'CODE'
STACK	0800	PARA	STACK	'STACK'

Symbols:

Name	Type	Value	Attr	
AF	L NEAR	00E6	CSEG	
ALARM	N PROC	00FB	CSEG	Length =0030
ALA_STR	L NEAR	0121	CSEG	
BIT_0	L NEAR	00BE	CSEG	
BIT_1	L NEAR	00B9	CSEG	
BIT_2	L NEAR	00B4	CSEG	
BIT_3	L NEAR	00AF	CSEG	
BIT_4	L NEAR	00AA	CSEG	
BIT_5	L NEAR	00A5	CSEG	
BIT_6	L NEAR	00A0	CSEG	
BIT_7	L NEAR	009B	CSEG	
CLIVEKT	N PROC	00C3	CSEG	Length =0038
CLI_STR	L NEAR	018C	CSEG	
CLOCKINT	N PROC	0144	CSEG	Length =0058
DBINT	N PROC	01CE	CSEG	Length =0042
DBI_STR	L NEAR	01E4	CSEG	
DISPINT	N PROC	0258	CSEG	Length =000E
DISPLAY	L NEAR	0186	CSEG	
DISPTYPE	L WORD	029F	CSEG	
EXIT	L NEAR	00F8	CSEG	
GET	N PROC	0266	CSEG	Length =0035
IM_MASTER	L WORD	029B	CSEG	
IM_SLAVE	L WORD	029D	CSEG	
INT	F PROC	003A	CSEG	Length =0034
INTATMO	F PROC	0000	CSEG	Length =003A
INTTABLE	N PROC	009B	CSEG	Length =0028
INTVEKT	N PROC	006E	CSEG	Length =002D
IRQF	L NEAR	00D7	CSEG	
LAST_ADR	L WORD	02A1	CSEG	
LOOP	L NEAR	0136	CSEG	
LOOPA	L NEAR	0113	CSEG	
MORE	L NEAR	003D	CSEG	
NEXT	L NEAR	0155	CSEG	
NEXTINT	L NEAR	0090	CSEG	
NLI_STR	L NEAR	021E	CSEG	
NOLEGINT	N PROC	0210	CSEG	Length =0048
PERIOD	N PROC	012B	CSEG	Length =0019
PF	L NEAR	00DC	CSEG	
RESIN	N PROC	019C	CSEG	Length =0032
RES_STR	L NEAR	01AA	CSEG	
TEST_BIT	L NEAR	007F	CSEG	
UF	L NEAR	00F0	CSEG	

48684 Bytes free

Warning Errors	Severe Errors
0	0

INT_ATC

```
                        ;
                        ;--------- STACK ---------------------
                        ;
0000                    stack   segment para stack 'stack'
0000 0200 [                             db 512 dup ('stack   ')
         73 74 61 63
         6B 20 20 20
               ]

1000                    stack   ends
                        ;
                        ;-------------------------------------
                        ;
                        ;
                        ;--------- CODE ----------------------
                        ;
0000                    cseg            segment para public 'code'
                                        assume cs:cseg,ds:cseg,es:cseg,
                        ss:stack
                        ;
                        ;--------- INITINT -------------------
                        ;
0000                    intatco         proc far
0000 FA                                 cli
0001 1E                                 push ds
0002 06                                 push es
0003 8C CA                              mov dx,cs
0005 8E C2                              mov es,dx
0007 8E DA                              mov ds,dx
                                        mov ax,3571h
                                        int 21h
                                        mov ax,es
                                        mov old_vekt+2,ax
                                        mov old_vekt,bx
0009 8D 16 004E R                       lea dx,int
000D B8 2570                            mov ax,2570h
0010 CD 21                              int 21h
0012 E4 A1                              in al,0A1h
0014 24 FE                              and al,0FEh
0016 8A C8                              mov cl,al
0018 B0 15                              mov al,15h
001A E6 A0                              out 0A0h,al  ;ICW1
001C B0 70                              mov al,70h
001E E6 A1                              out 0A1h,al  ;ICW2
0020 B0 02                              mov al,02h
0022 E6 A1                              out 0A1h,al  ;ICW3
0024 B0 0B                              mov al,0Bh
0026 E6 A1                              out 0A1h,al  ;ICW4
```

```
0028 8A C1                              mov al,cl
002A E6 A1                              out 0A1h,al ;OCW1
002C BA 0300                            mov dx,0300h
002F B4 0C                              mov ah,0Ch
0031 E8 0161 R                          call get
0034 C7 06 0177 R B800                  mov disptype,0B800h
003A 8D 06 0179 R                       lea ax,last_adr
003E 25 FFF0                            and ax,0FFF0h
0041 05 0010                            add ax,0010h
0044 8B D0                              mov dx,ax
0046 B8 3100                            mov ax,3100h
0049 07                                 pop es
004A 1F                                 pop ds
004B FB                                 sti
004C CD 21                              int 21h
004E                    intatco endp
                        ;
                        ;-------- INT ----------------------
                        ---------
                        ;
004E                    int             proc far
004E FA                                 cli
004F 50                                 push ax
0050 52                                 push dx
0051 51                 more:           push cx
0052 53                                 push bx
0053 57                                 push di
0054 56                                 push si
0055 55                                 push bp
0056 1E                                 push ds
0057 06                                 push es
0058 8C C8                              mov ax,cs
005A 8E D8                              mov ds,ax
005C 8E C0                              mov es,ax
005E BD 0000                            mov bp,0
0061 BA 0300                            mov dx,0300h
0064 E8 007A R                          call clivekt
0067 B0 62                              mov al,62h
0069 E6 20                              out 20h,al
006B B0 60                              mov al,60h
006D E6 A0                              out 0A0h,al
006F 07                                 pop es
0070 1F                                 pop ds
0071 5D                                 pop bp
0072 5E                                 pop si
0073 5F                                 pop di
0074 5B                                 pop bx
0075 59                                 pop cx
0076 5A                                 pop dx
0077 58                                 pop ax
                                ;       mov ax,old_vekt+2
                                ;       mov es,ax
                                ;       mov bx,old_vekt
0078 FB                                 sti
```

```
0079 CF                                 iret
007A                            int     endp
                                ;
                        C               include clivekt.asm
                        C       ;------ CLIVEKT ------------------------------
                        C       ;
                        C       ; DX=PC4680 base adress
                        C       ;
                        C       ; Interrupt routine call select for clock inte
                                rrupts
                        C       ;
                        C       ;-------------------------------------------
                        C       ;
007A                      C             clivekt         proc near
007A B4 0B                  C                           mov ah,0Bh
007C E8 0161 R                C                         call get
007F 8A C1                  C                           mov al,cl
0081 0C 8F                  C                           or al,8Fh
0083 50                   C                             push ax
0084 B4 0C                  C                           mov ah,0Ch
0086 E8 0161 R                C                         call get
0089 58                   C                             pop ax
008A 22 C8                  C                           and cl,al
008C 8B D9                  C                           mov bx,cx
008E F6 C1 80               C             IRQF:         test cl,80h
0091 74 1C                C                             jz exit
0093 F6 C1 40               C             PF:           test cl,40h
0096 74 05                C                             jz AF
0098 E8 00E2 R                C                         call period
009B 8B CB                  C                           mov cx,bx
009D F6 C1 20               C             AF:           test cl,20h
00A0 74 05                C                             jz UF
00A2 E8 00B2 R                C                         call alarm
00A5 8B CB                  C                           mov cx,bx
00A7 F6 C1 10               C             UF:           test cl,10h
00AA 74 03                C                             jz exit
00AC E8 00FB R                C                         call clockint
00AF 8B CB                  C             exit:         mov cx,bx
00B1 C3                   C                             ret
00B2                      C             clivekt         endp
                        C               include alarm.asm
                        C       ;------ ALARM ------------------------------
                        C       ;
                        C       ; Routine for alarminterrupt from clock
                        C       ;
                        C       ; CX,DI,SI changed
                        C       ;
                        C       ;-------------------------------------------
                        C       ;
00B2                      C             alarm           proc near
```

```
00B2  8D 36 00D8 R      C               lea si,ala_str
00B6  BF 0136           C               mov di,310
00B9  B9 000A           C               mov cx,10
00BC  E8 0153 R         C               call dispint
00BF  50                C               push ax
00C0  E4 61             C               in al,61h
00C2  0C 03             C               or al,03h
00C4  E6 61             C               out 61h,al
00C6  50                C               push ax
00C7  B8 2000           C               mov ax,2000h
00CA  50                C       loopa:  push ax
00CB  58                C               pop ax
00CC  50                C               push ax
00CD  58                C               pop ax
00CE  48                C               dec ax
00CF  75 F9             C               jnz loopa
00D1  58                C               pop ax
00D2  24 FC             C               and al,0FCh
00D4  E6 61             C               out 61h,al
00D6  58                C               pop ax
00D7  C3                C               ret
                        C   ;
00D8  41 0F 4C 0F 41 0F C   ala_str:    db  'A L A R M '
      52 0F 4D 0F       C
                        C   ;
00E2                    C       alarm   endp
                        C               include period.asm
                        C   ;-------- PERIOD ------------------------------
                        C   ;
                        C   ; Interrupt routine for periodic interrupt from clock
                        C   ;
                        C   ; No registers changed
                        C   ;
                        C   ;-----------------------------------------------
                        C   ;
00E2                    C       period  proc near
00E2  50                C               push ax
00E3  E4 61             C               in al,61h
00E5  0C 03             C               or al,03h
00E7  E6 61             C               out 61h,al
00E9  50                C               push ax
00EA  B8 0010           C               mov ax,0010h
00ED  50                C       loop:   push ax
00EE  58                C               pop ax
00EF  50                C               push ax
00F0  58                C               pop ax
00F1  48                C               dec ax
00F2  75 F9             C               jnz loop
00F4  58                C               pop ax
00F5  24 FC             C               and al,0FCh
```

```
00F7 E6 61              C                       out 61h,al
00F9 58                 C                       pop ax
00FA C3                 C                       ret
00FB                    C       period          endp
                        C                       include clockint.asm
                        C       ;---------- CLOCKINT ----------------------------
                        C       ;
                        C       ; DX=PC4680 base adress
                        C       ;
                        C       ; AX,CX,DI,SI changed
                        C       ;
                        C       ;-----------------------------------------------
00FB                    C       clockint        proc near
00FB 8D 36 0143 R       C                       lea si,cli_str
00FF BF 0090            C                       mov di,90h
0102 B9 0010            C                       mov cx,10h
0105 56                 C                       push si
0106 51                 C                       push cx
0107 83 C6 0E           C                       add si,0Eh
010A B4 00              C                       mov ah,00h
010C E8 0161 R          C       next:           call get
010F 8A E9              C                       mov ch,cl
0111 81 E1 F00F         C                       and cx,0F00Fh
0115 D0 CD              C                       ror ch,1
0117 D0 CD              C                       ror ch,1
0119 D0 CD              C                       ror ch,1
011B D0 CD              C                       ror ch,1
011D 81 C1 3030         C                       add cx,3030h
0121 2E: 88 0C          C                       mov cs:[si],cl
0124 4E                 C                       dec si
0125 4E                 C                       dec si
0126 2E: 88 2C          C                       mov cs:[si],ch
0129 4E                 C                       dec si
012A 4E                 C                       dec si
012B F6 C4 04           C                       test ah,4
012E 75 0D              C                       jnz display
0130 FE C4              C                       inc ah
0132 FE C4              C                       inc ah
0134 B0 3A              C                       mov al,3Ah
0136 2E: 88 04          C                       mov cs:[si],al
0139 4E                 C                       dec si
013A 4E                 C                       dec si
013B EB CF              C                       jmp next
013D 59                 C       display:        pop cx
013E 5E                 C                       pop si
013F E8 0153 R          C                       call dispint
0142 C3                 C                       ret
                        C       ;
0143    08 [            C       cli_str:        dw 8 dup (' ')
           0F20         C
             ]          C
                        C
```

```
                C   ;
0153            C   clockint        endp
                C                   include dispint.asm
                C   ;-------- DISPINT ----------------------------------
                C   ;
                C   ; SI=Startadress of string to display
                C   ; DI=Destinationadress for string
                C   ; CX=String length
                C   ;
                C   ;-------------------------------------------------
0153            C   dispint         proc near
0153 06         C                   push es
0154 52         C                   push dx
0155 8B 16 0177 R   C               mov dx,disptype
0159 8E C2      C                   mov es,dx
015B FC         C                   cld
015C F3/ A4     C                   rep movsb
015E 5A         C                   pop dx
015F 07         C                   pop es
0160 C3         C                   ret
0161            C   dispint         endp
                C                   include get.asm
                C   ;-------- Läs data från klockan ---------------
                C   ;
                C   ;   CH = data från klockan
                C   ;   AH = adress till klockan
                C   ;
                C   ;-------------------------------------------
                C   ;
0161            C   get             proc near
0161 8A C4      C                   mov al,ah
0163 E6 70      C                   out 70h,al
0165 E4 71      C                   in al,71h
0167 8A C8      C                   mov cl,al
0169 C3         C                   ret
016A            C   get             endp
                C                   include set.asm
                C   ;-------- Skriv data till klockan ---------------
                C   ;
                C   ;   CH = data till klockan
                C   ;   AH = adress till klockan
                C   ;
                C   ;-------------------------------------------
                C   ;
016A            C   set             proc near
016A 8A C4      C                   mov al,ah
016C E6 70      C                   out 70h,al
016E 8A C1      C                   mov al,cl
0170 E6 71      C                   out 71h,al
0172 C3         C                   ret
```

```
0173                    C       set             endp
                                ;
0173    02 [                    old_vekt        dw 2 dup (?)
            ????
               ]

                                ;
0177    01 [                    disptype dw 1 dup (?)
            ????
               ]

                                ;
0179    01 [                    last_adr  dw 1 dup (?)
            ????
               ]

                                ;
017B                            cseg            ends
                                                end
```

Segments and Groups:

Name	Size	Align	Combine	Class
CSEG	017B	PARA	PUBLIC	'CODE'
STACK	1000	PARA	STACK	'STACK'

Symbols:

Name	Type	Value	Attr	
AF	L NEAR	009D	CSEG	
ALARM	N PROC	00B2	CSEG	Length =0030
ALA_STR	L NEAR	00D8	CSEG	
CLIVEKT	N PROC	007A	CSEG	Length =0038
CLI_STR	L NEAR	0143	CSEG	
CLOCKINT	N PROC	00FB	CSEG	Length =0058
DISPINT	N PROC	0153	CSEG	Length =000E
DISPLAY	L NEAR	013D	CSEG	
DISPTYPE	L WORD	0177	CSEG	
EXIT	L NEAR	00AF	CSEG	
GET	N PROC	0161	CSEG	Length =0009
INT	F PROC	004E	CSEG	Length =002C
INTATCO	F PROC	0000	CSEG	Length =004E
IRQF	L NEAR	008E	CSEG	
LAST_ADR	L WORD	0179	CSEG	
LOOP	L NEAR	00ED	CSEG	
LOOPA	L NEAR	00CA	CSEG	
MORE	L NEAR	0051	CSEG	
NEXT	L NEAR	010C	CSEG	
OLD_VEKT	L WORD	0173	CSEG	Length =0002
PERIOD	N PROC	00E2	CSEG	Length =0019
PF	L NEAR	0093	CSEG	
SET	N PROC	016A	CSEG	Length =0009
UF	L NEAR	00A7	CSEG	

49432 Bytes free

Warning Errors	Severe Errors
0	0

INT_ATM

```
                        ;
                        ;-------- STACK ----------------------
                        ;
0000                    stack   segment para stack 'stack'
0000 0200 [                             db 512 dup ('stack   ')
         73 74 61 63
         6B 20 20 20
               ]

1000                    stack   ends
                        ;
                        ;-------------------------------------
                        ;
                        ;
                        ;-------- CODE -----------------------
                        ;
0000                    cseg            segment para public 'code'
                                        assume cs:cseg,ds:cseg,es:cseg,ss:stack
                        ;
                        ;-------- INITINT --------------------
                        ;
0000                    intatco         proc far
0000 FA                                 cli
0001 1E                                 push ds
0002 06                                 push es
0003 8C CA                              mov dx,cs
0005 8E C2                              mov es,dx
0007 8E DA                              mov ds,dx
                        ;               mov ax,3571h
                        ;               int 21h
                        ;               mov ax,es
                        ;               mov old_vekt+2,ax
                        ;               mov old_vekt,bx
0009 8D 16 004E R                       lea dx,int
000D B8 2570                            mov ax,2570h
0010 CD 21                              int 21h
0012 E4 A1                              in al,0A1h
0014 24 FE                              and al,0FEh
0016 8A C8                              mov cl,al
0018 B0 15                              mov al,15h
001A E6 A0                              out 0A0h,al  ;ICW1
001C B0 70                              mov al,70h
001E E6 A1                              out 0A1h,al  ;ICW2
0020 B0 02                              mov al,02h
0022 E6 A1                              out 0A1h,al  ;ICW3
0024 B0 0B                              mov al,0Bh
0026 E6 A1                              out 0A1h,al  ;ICW4
```

```
0028 8A C1                      mov al,cl
002A E6 A1                      out 0A1h,al ;OCW1
002C BA 0300                    mov dx,0300h
002F B4 0C                      mov ah,0Ch
0031 E8 0161 R                  call get
0034 C7 06 0177 R B000          mov disptype,0B000h
003A 8D 06 0179 R               lea ax,last_adr
003E 25 FFF0                    and ax,0FFF0h
0041 05 0010                    add ax,0010h
0044 8B D0                      mov dx,ax
0046 B8 3100                    mov ax,3100h
0049 07                         pop es
004A 1F                         pop ds
004B FB                         sti
004C CD 21                      int 21h
004E                    intatco endp
                        ;
                        ;-------- INT ----------------------------
                        ;
004E                    int     proc far
004E FA                         cli
004F 50                         push ax
0050 52                         push dx
0051 51                 more:   push cx
0052 53                         push bx
0053 57                         push di
0054 56                         push si
0055 55                         push bp
0056 1E                         push ds
0057 06                         push es
0058 8C C8                      mov ax,cs
005A 8E D8                      mov ds,ax
005C 8E C0                      mov es,ax
005E BD 0000                    mov bp,0
0061 BA 0300                    mov dx,0300h
0064 E8 007A R                  call clivekt
0067 B0 62                      mov al,62h
0069 E6 20                      out 20h,al
006B B0 60                      mov al,60h
006D E6 A0                      out 0A0h,al
006F 07                         pop es
0070 1F                         pop ds
0071 5D                         pop bp
0072 5E                         pop si
0073 5F                         pop di
0074 5B                         pop bx
0075 59                         pop cx
0076 5A                         pop dx
0077 58                         pop ax
                        ;               mov ax,old_vekt+2
                        ;               mov es,ax
                        ;               mov bx,old_vekt
0078 FB                         sti
```

```
0079 CF                                 iret
007A                            int     endp
                                ;
                        C               include clivekt.asm
                        C       ;------ CLIVEKT ------------------------------
                        C       ;
                        C       ; DX=PC4680 base adress
                        C       ;
                        C       ; Interrupt routine call select for clock inte
                                rrupts
                        C       ;
                        C       ;-------------------------------------------
                        C       ;
007A                      C     clivekt         proc near
007A  B4 0B                 C                   mov ah,0Bh
007C  E8 0161 R               C                 call get
007F  8A C1                 C                   mov al,cl
0081  0C 8F                 C                   or al,8Fh
0083  50                  C                     push ax
0084  B4 0C                 C                   mov ah,0Ch
0086  E8 0161 R               C                 call get
0089  58                  C                     pop ax
008A  22 C8                 C                   and cl,al
008C  8B D9                 C                   mov bx,cx
008E  F6 C1 80                C     IRQF:       test cl,80h
0091  74 1C                 C                   jz exit
0093  F6 C1 40                C     PF:         test cl,40h
0096  74 05                 C                   jz AF
0098  E8 00E2 R               C                 call period
009B  8B CB                 C                   mov cx,bx
009D  F6 C1 20                C     AF:         test cl,20h
00A0  74 05                 C                   jz UF
00A2  E8 00B2 R                C                call alarm
00A5  8B CB                 C                   mov cx,bx
00A7  F6 C1 10                C     UF:         test cl,10h
00AA  74 03                 C                   jz exit
00AC  E8 00FB R                C                call clockint
00AF  8B CB                   C     exit:       mov cx,bx
00B1  C3                    C                   ret
00B2                      C     clivekt         endp
                        C               include alarm.asm
                        C       ;------ ALARM -------------------------------
                        C       ;
                        C       ; Routine for alarminterrupt from clock
                        C       ;
                        C       ; CX,DI,SI changed
                        C       ;
                        C       ;-------------------------------------------
                        C       ;
00B2                      C     alarm           proc near
```

```
00B2 8D 36 00D8 R      C                lea si,ala_str
00B6 BF 0136           C                mov di,310
00B9 B9 000A           C                mov cx,10
00BC E8 0153 R         C                call dispint
00BF 50                C                push ax
00C0 E4 61             C                in al,61h
00C2 0C 03             C                or al,03h
00C4 E6 61             C                out 61h,al
00C6 50                C                push ax
00C7 B8 2000           C                mov ax,2000h
00CA 50                C        loopa:  push ax
00CB 58                C                pop ax
00CC 50                C                push ax
00CD 58                C                pop ax
00CE 48                C                dec ax
00CF 75 F9             C                jnz loopa
00D1 58                C                pop ax
00D2 24 FC             C                and al,0FCh
00D4 E6 61             C                out 61h,al
00D6 58                C                pop ax
00D7 C3                C                ret
                       C ;
00D8 41 0F 4C 0F 41 0F C  ala_str:      db 'A L A R M'
    52 0F 4D 0F        C
                       C ;
00E2                   C        alarm   endp
                       C                include period.asm
                       C ;-------- PERIOD ----------------------------
                       C ;
                       C ; Interrupt routine for periodic interrupt from clock
                       C ;
                       C ; No registers changed
                       C ;
                       C ;-------------------------------------------
                       C ;
00E2                   C        period  proc near
00E2 50                C                push ax
00E3 E4 61             C                in al,61h
00E5 0C 03             C                or al,03h
00E7 E6 61             C                out 61h,al
00E9 50                C                push ax
00EA B8 0010           C                mov ax,0010h
00ED 50                C        loop:   push ax
00EE 58                C                pop ax
00EF 50                C                push ax
00F0 58                C                pop ax
00F1 48                C                dec ax
00F2 75 F9             C                jnz loop
00F4 58                C                pop ax
00F5 24 FC             C                and al,0FCh
```

```
00F7 E6 61              C                       out 61h,al
00F9 58                 C                       pop ax
00FA C3                 C                       ret
00FB                    C       period          endp
                        C                       include clockint.asm
                        C       ;--------- CLOCKINT ----------------------------
                        C       ;
                        C       ;DX=PC4680 base adress
                        C       ;
                        C       ; AX,CX,DI,SI changed
                        C       ;
                        C       ;-------------------------------------------------
00FB                    C       clockint        proc near
00FB 8D 36 0143 R       C                       lea si,cli_str
00FF BF 0090            C                       mov di,90h
0102 B9 0010            C                       mov cx,10h
0105 56                 C                       push si
0106 51                 C                       push cx
0107 83 C6 0E           C                       add si,0Eh
010A B4 00              C                       mov ah,00h
010C E8 0161 R          C       next:           call get
010F 8A E9              C                       mov ch,cl
0111 81 E1 F00F         C                       and cx,0F00Fh
0115 D0 CD              C                       ror ch,1
0117 D0 CD              C                       ror ch,1
0119 D0 CD              C                       ror ch,1
011B D0 CD              C                       ror ch,1
011D 81 C1 3030         C                       add cx,3030h
0121 2E: 88 0C          C                       mov cs:[si],cl
0124 4E                 C                       dec si
0125 4E                 C                       dec si
0126 2E: 88 2C          C                       mov cs:[si],ch
0129 4E                 C                       dec si
012A 4E                 C                       dec si
012B F6 C4 04           C                       test ah,4
012E 75 0D              C                       jnz display
0130 FE C4              C                       inc ah
0132 FE C4              C                       inc ah
0134 B0 3A              C                       mov al,3Ah
0136 2E: 88 04          C                       mov cs:[si],al
0139 4E                 C                       dec si
013A 4E                 C                       dec si
013B EB CF              C                       jmp next
013D 59                 C       display:        pop cx
013E 5E                 C                       pop si
013F E8 0153 R          C                       call dispint
0142 C3                 C                       ret
                        C       ;
0143    08 [            C       cli_str:        dw 8 dup (' ')
           0F20         C
             ]          C
                        C
```

```
                C   ;
0153             C          clockint        endp
                C                   include dispint.asm
                C   ;-------- DISPINT ----------------------------------
                C   ;
                C   ; SI=Startadress of string to display
                C   ; DI=Destinationadress for string
                C   ; CX=String length
                C   ;
                C   ;-----------------------------------------------------
0153             C          dispint         proc near
0153 06           C                         push es
0154 52           C                         push dx
0155 8B 16 0177 R    C                      mov dx,disptype
0159 8E C2        C                         mov es,dx
015B FC           C                         cld
015C F3/ A4        C                        rep movsb
015E 5A           C                         pop dx
015F 07           C                         pop es
0160 C3           C                         ret
0161             C          dispint         endp
                C                   include get.asm
                C   ;-------- Läs data från klockan ---------------------
                C   ;
                C   ;       CH = data från klockan
                C   ;       AH = adress till klockan
                C   ;
                C   ;-----------------------------------
                C   ;
0161              C          get             proc near
0161 8A C4           C                      mov al,ah
0163 E6 70        C                         out 70h,al
0165 E4 71        C                         in al,71h
0167 8A C8        C                         mov cl,al
0169 C3           C                         ret
016A             C          get             endp
                C                   include set.asm
                C   ;-------- Skriv data till klockan ---------------
                C   ;
                C   ;       CH = data till klockan
                C   ;       AH = adress till klockan
                C   ;
                C   ;-----------------------------------
                C   ;
016A              C          set             proc near
016A 8A C4           C                      mov al,ah
016C E6 70        C                         out 70h,al
016E 8A C1        C                         mov al,cl
0170 E6 71        C                         out 71h,al
0172 C3           C                         ret
```

```
0173                    C       set             endp
                                ;
0173    02 [                    old_vekt        dw 2 dup (?)
             ????
               ]

                                ;
0177    01 [                    disptype        dw 1 dup (?)
             ????
               ]

                                ;
0179    01 [                    last_adr        dw 1 dup (?)
             ????
               ]

                                ;
017B                            cseg            ends
                                                end
```

Segments and Groups:

Name	Size	Align	Combine	Class
CSEG	017B	PARA	PUBLIC	'CODE'
STACK	1000	PARA	STACK	'STACK'

Symbols:

Name	Type	Value	Attr	
AF	L NEAR	009D	CSEG	
ALARM	N PROC	00B2	CSEG	Length =0030
ALA_STR	L NEAR	00D8	CSEG	
CLIVEKT	N PROC	007A	CSEG	Length =0038
CLI_STR	L NEAR	0143	CSEG	
CLOCKINT	N PROC	00FB	CSEG	Length =0058
DISPINT	N PROC	0153	CSEG	Length =000E
DISPLAY	L NEAR	013D	CSEG	
DISPTYPE	L WORD	0177	CSEG	
EXIT	L NEAR	00AF	CSEG	
GET	N PROC	0161	CSEG	Length =0009
INT	F PROC	004E	CSEG	Length =002C
INTATCO	F PROC	0000	CSEG	Length =004E
IRQF	L NEAR	008E	CSEG	
LAST_ADR	L WORD	0179	CSEG	
LOOP	L NEAR	00ED	CSEG	
LOOPA	L NEAR	00CA	CSEG	
MORE	L NEAR	0051	CSEG	
NEXT	L NEAR	010C	CSEG	
OLD_VEKT	L WORD	0173	CSEG	Length =0002
PERIOD	N PROC	00E2	CSEG	Length =0019
PF	L NEAR	0093	CSEG	
SET	N PROC	016A	CSEG	Length =0009
UF	L NEAR	00A7	CSEG	

49432 Bytes free

Warning Errors	Severe Errors
0	0

APPENDIX E

Kända problem

Detta appendix tar upp de problem som kan uppkomma vid installation av PC4680 i olika system. Om Du stöter på något annat problem än de som finns i appendixet är vi på Datum System tacksamma om du meddelar oss om detta.

Kända problem

1. **PC4680 - EGA.** Om PC4680 skall installeras tillsammans med ett EGAkort kan vissa problem uppstå om man vill använda interrupt. Detta beror på att både EGA och PC4680 använder signalen IRQ2. Den åtgärd som kan vidtagas är att antingen koppla ur interrupt från PC4680 eller EGA (om möjligt).

2. **PC4680 - IBM PC Network.** Här gäller samma problem som i förra fallet. Dock kan man bygla om nätverket till IRQ3. Man får då se upp så att man inte kolliderar med serieport nr 2 (com2), som också använder IRQ3.

3. **Vissa "kompatibla" maskiner.** Vid RESET på 4680-bussen, kommandot Q%=INP(&H307), bromsar PC4680 datorns CPU i 16 klockcykler, vilket är tillåtet enligt IBM's tekniska specifikation. En del "kompatibla" maskiner kan inte hantera detta utan "tappar minnet". Kolla med maskinens leverantör att detta fungerar.

4. **Klockinterrupt på IBM AT.** Om man vill använda sig av tidbasinterrupt från PC4680's klockrets kan man i vissa fall få problem om PC'n är av AT modell. I dessa fall rekomenderar vi att man utnyttjar AT'ns klockkrets istället, som dessutom är av samma typ som på PC4680.