# Del 2: Adressedekoding og porter

**Bakgrunnstoff: Læreboka kap. 2, 5 og 6.**

Figur 1 viser den delen av elektronikken som sitter på det VERO PC AT kortet vi skal bruke videre i dette kurset. Den eneste modifikasjonen vi har gjort er å koble adressebit A10 fra ISA-bussen til et ledig buffer i U7 (pinne 6). Bruk datablader for de komponentene som er benyttet og forsøk å forstå hvordan logikken virker.

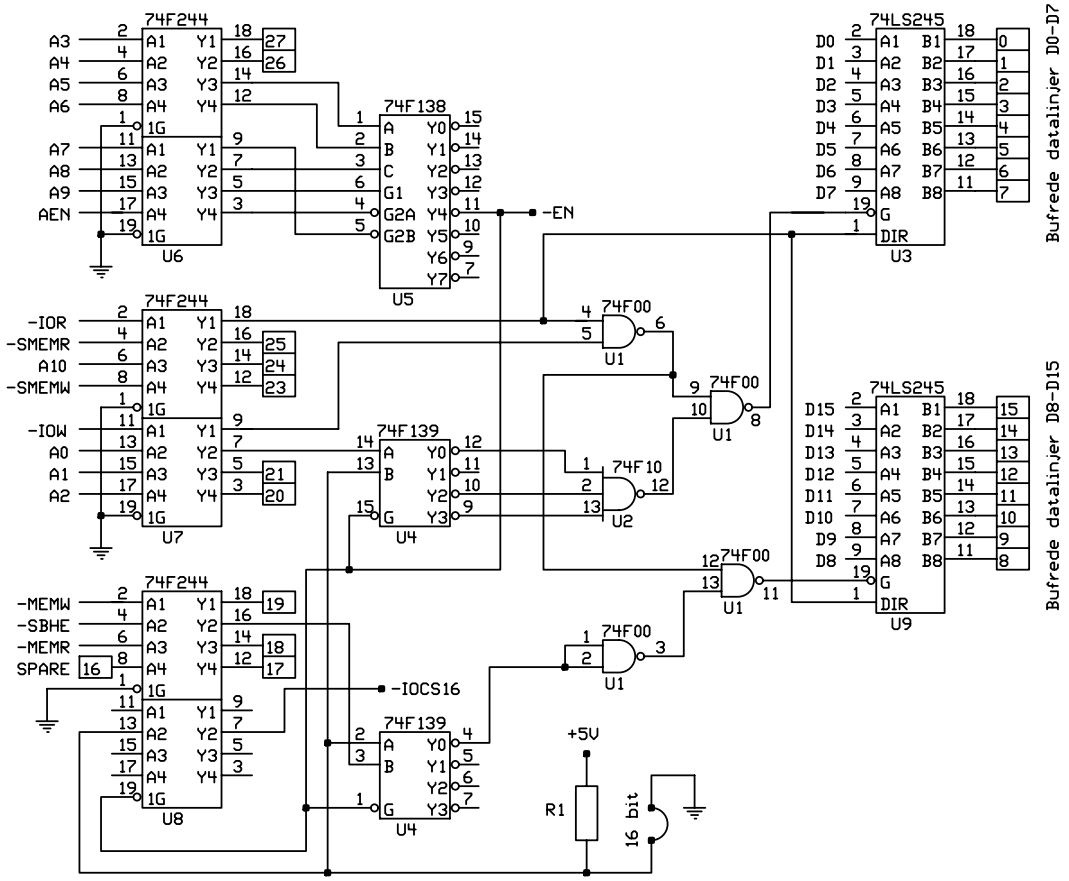


*Fig.1: Ferdigkoblet adressedekoding og bufring på kortet.*

| |
|---|
| Husk følgende:<br>**Når kortet kobles til eller fra ISA-bussen skal PC'en alltid være avslått** |

På kortet er det loddet fast sokler for integrerte kretser som skal brukes i dette kurset. På de fleste kretsene er det dessuten loddet på forbindelser til +5V og 0V. Når du kobler opp kretsene på kortet v.h.a. wire-wrap-metoden, må du derfor undersøke grundig hvilke forbindelser som allerede ligger på kortet, og hvilke forbindelser du må legge.

## Adressedekoding

Hvilket adresseområde (blant grunnadressene dekodet fra A0 - A9) vil gjelde for kortet? Finn dette ut ved å granske koblingene av kretsene U5 og U6 i figur 1 og så fullføre diagrammet under:

| Adressebit: | A11 | A10 | A9 | A8 | A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Binærverdi: | 0 | 0 | 1 | 1 | 0 | | | | | | | |
| Hex-verdi: | | | 3 | | | | | | | | | |

Hvilket adresseområde får vi i tillegg ved å benytte A10 i videre adressedekoding?

Bruk et oscilloskop med to kanaler for å se på enable-signalet -EN som viser at kortet vårt er adressert. Du kan f.eks. bruke følgende program:

Program 1:

```
#include <stdio.h>
#include <dos.h>
void main()
{
  clrscr();
  printf("Trykk en tast for å avbryte programmet...");
  while(!kbhit())
  {
    inportb(0x300);
    inportb(0x350);
  }
}
```

Hvis du trigger på -BIOR skal du få noe som ligner på figur 2. Side 25 i boka sier at 16 bits I/O-operasjoner bruker én wait state, mens åtte bits bruker fire wait state. Dette ser vi i figur 2 ved at adressen 0x350 som ikke gir noen -IOCS16 får flere wait state.

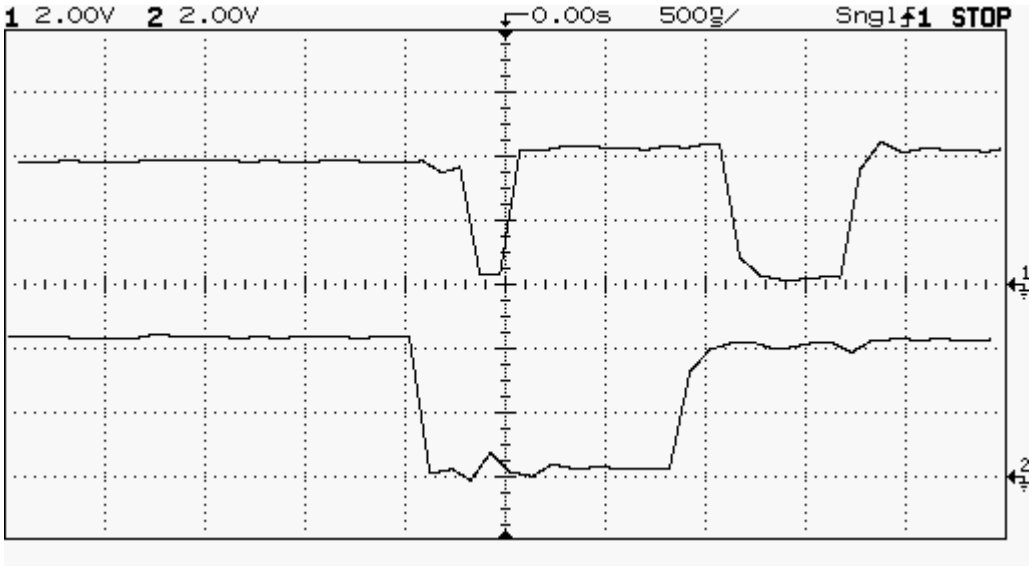


*Fig. 2: Test av adressedekoder. Øvre kurve viser -BIOR og nedre kurve viser -EN når program 1 kjøres. Trigger på -BIOR.*

## Wait state

Vi skal nå koble opp en krets som aktiverer -IOCHRDY ved alle I/O-henvendelser innenfor vårt adresseområde. Koble opp følgende krets på kortet:

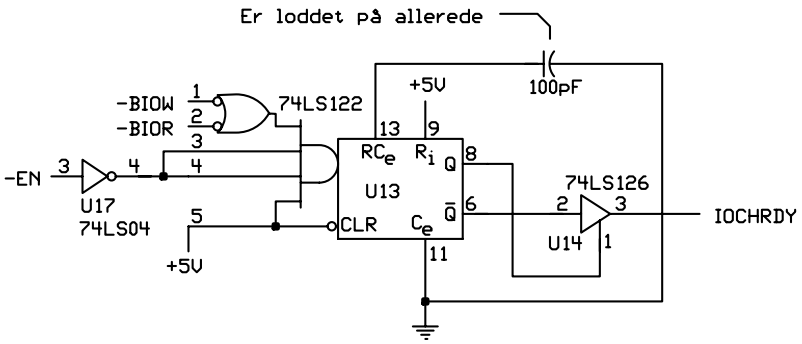


*Fig.3: Generering av wait state*

Bruk datablad for 74LS122 til å bestemme hvor lang IOCHRDY-pulsen vil bli. I læreboka på side 24 vil du se at vi har maksimalt 70 ns på oss fra -IOR eller -IOW går lav til vi må trekke IOCHRDY lav. Bruk datablader for kretsene som er brukt for generering av wait state og sjekk om dette tidskravet blir overholdt ved typiske portforsinkelser. Sjekk også om kravet blir overholdt ved maksimal portforsinkelse i alle kretsene. Hvis du nå kjører program 1 på nytt, vil du få noe som ligner på figur 4 på oscilloscop-skjermen.

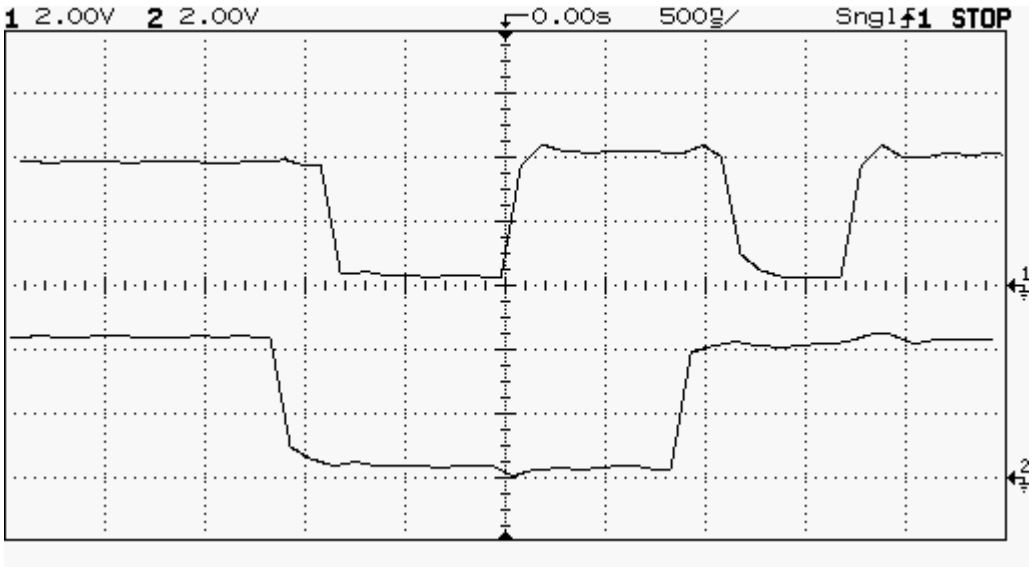


*Fig.4: Ny test av adressedekoder med generering av wait state. Øvre kurve viser -BIOR og nedre kurve viser -EN når program 1 kjøres. Trigger på -BIOR.*

Hvis du bruker oscilloscopet for å se på IOCHRDY-signalet, vil du se noe som ligner på figur 5. Her ser vi at IOCHRDY blir trukket lav i ca. 500 ns når vi adresserer innenfor vårt adresseområde.

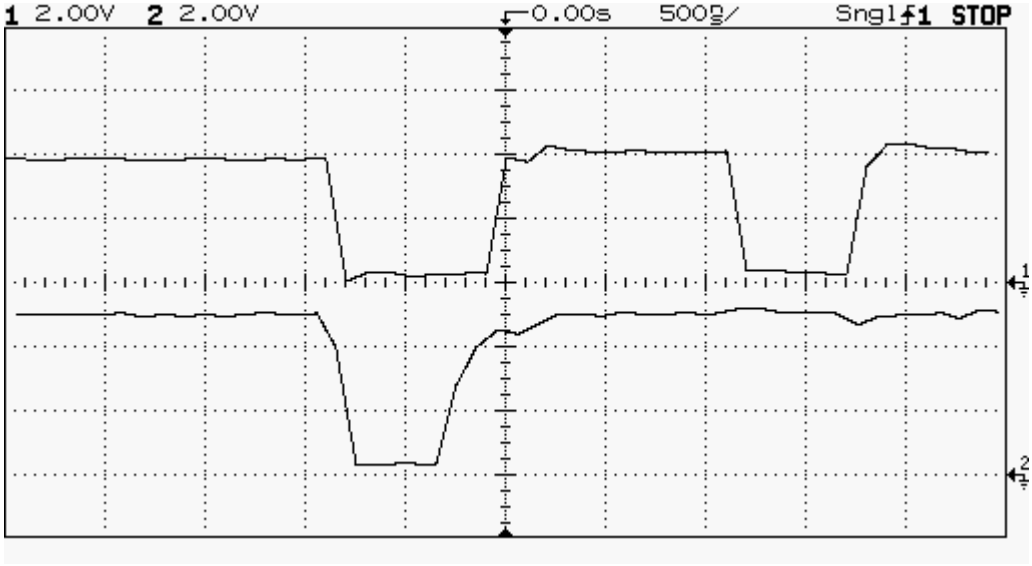


*Fig.5: Generering av wait state ved å trekke IOCHRDY lav innen 70 ns etter at -BIOR gikk lav. Øvre kurve viser -BIOR og nedre kurve viser IOCHRDY når program 1 kjøres. Trigger på -BIOR.*

## Videre adressedekoding

Vi skal nå koble opp en krets som gir oss videre adressedekoding innenfor det adresseområdet som gjelder for kortet. Dette er nødvendig for å kunne gi de forkjellige kretsene som vi etterhvert skal koble opp på kortet, separate adresser. Koble kretsen som er vist i figur 6 opp på kortet.

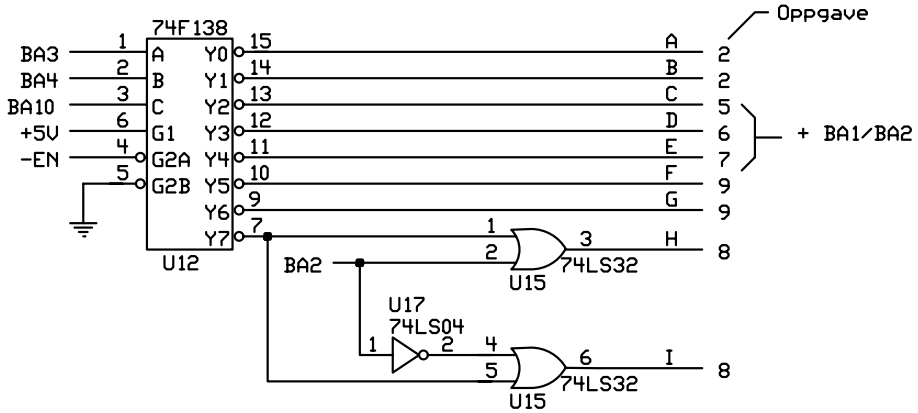


*Fig.6: Videre adressedekoding innenfor adresseområdet som gjelder for kortet.*

I figur 6 er det også angitt i hvilken laboppgave vi skal bruke de forskjellige adressene. Lag en liste som viser hvilke adresser som gjelder for utgang A, B, C, o.s.v.

## 16-bits I/O-port

Vi skal nå konstruere en 16-bits I/O-port slik som beskrevet i læreboka side 74. Koble opp på kortet slik som vist i figur 7.

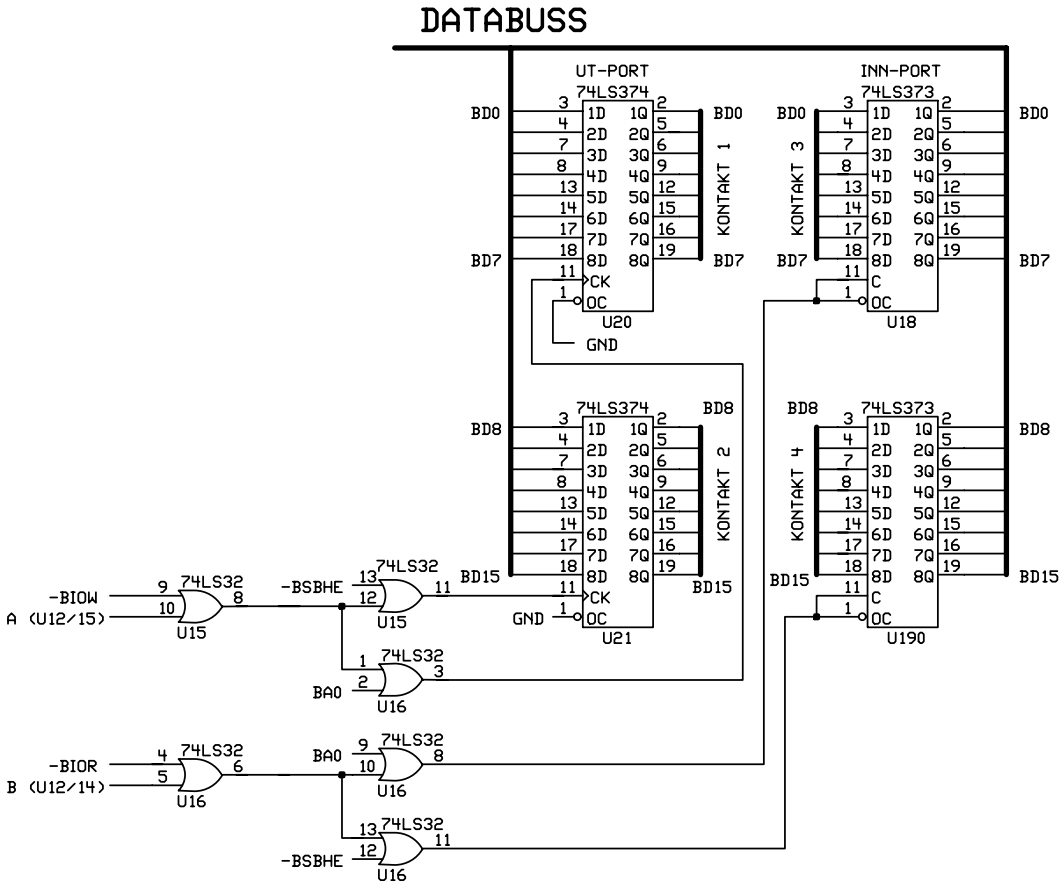


*Fig.7: 16-bits inn-port og ut-port.*

For å bruke disse portene skal vi koble opp 16 brytere som vi kan lese via inn-porten og 16 lysdioder som vi kan styre med ut-porten. Koblingen som er vist i figur 8 for bryterne skulle allerede være koblet opp på kortet. Sjekk at koblingen på kortet er riktig.

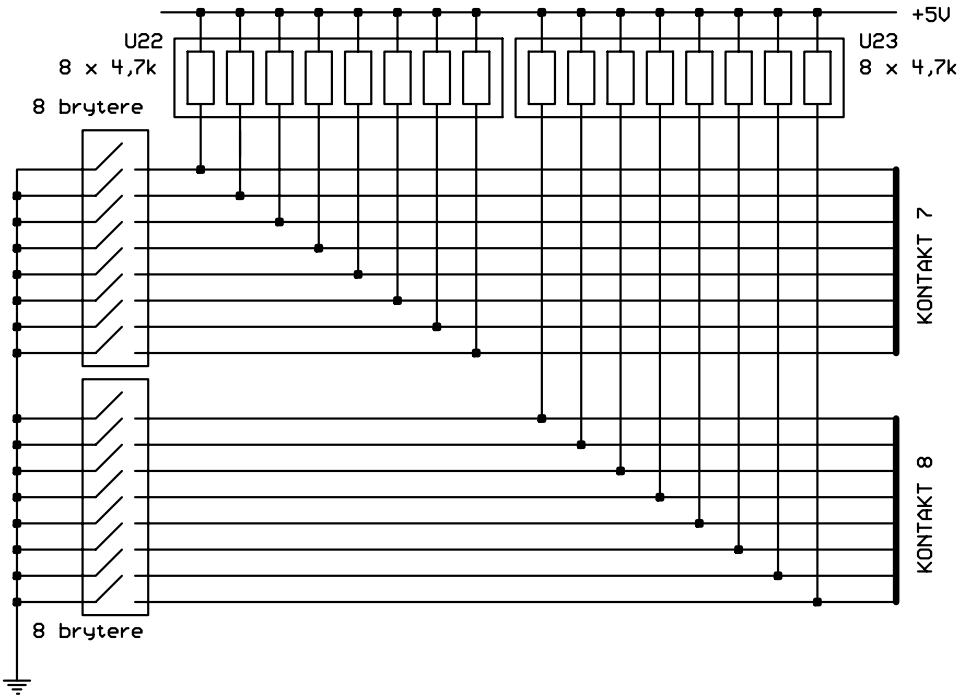


*Fig.8: Brytere som kan leses fra inn-porten.*

Vi skal også koble opp 16 lysdioder som kan styres fra ut-porten. Høyre del av koblingen i figur 9 er allerde loddet på kortet, resten må du koble opp.

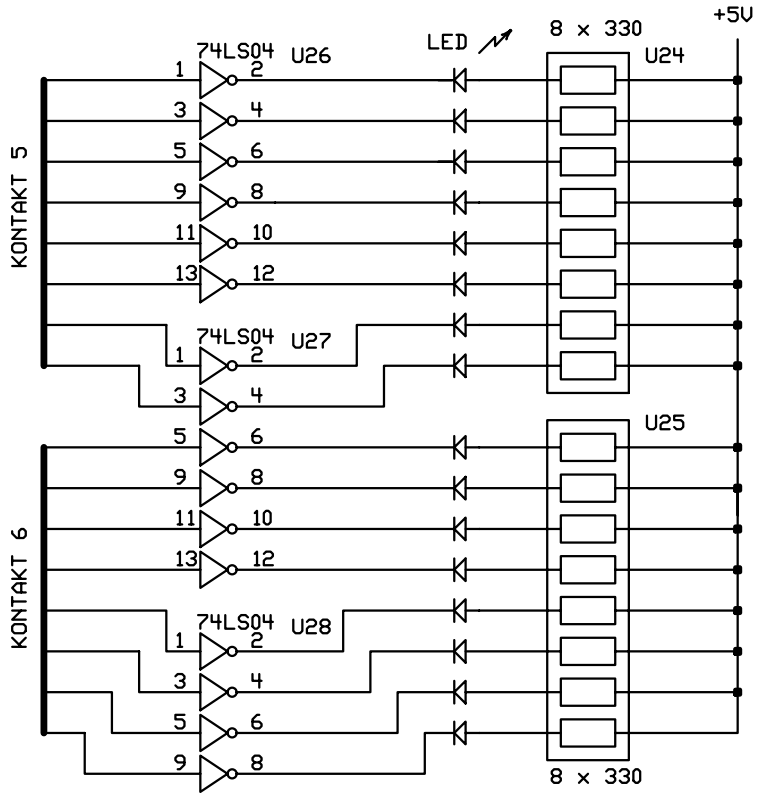


*Fig.9: Lysdioder som kan styres fra ut-porten.*

På lab'en vil du finne noen kabler med 8-pinners molex-kontakt i hver ende. Disse kan trykkes rett inn på wire-wrap-pinner, og kan brukes til å f.eks. forbinde kontakt 1 på ut-porten med kontakt 5 på lysdiode-rekka.

Bruk datablader for kretsene som er brukt i figur 7 og prøv å forstå hvordan portene virker.

Forbind kontakt 1 og 2 med hver sin lysdiode-rekke og skriv et program som tenner lysdiodene i tur og orden. Legg inn en passende venteløkke mellom hver gang du skriver til porten slik at du lett kan se at lysdiodene tenner og slukker. Et trykk på CR skal avslutte programmet.

Forbind så kontakt 3 og 4 med hver sin bryter-rekke og skriv et program som kontinuerlig leser fra bryterne og skriver verdiene fra bryterne ut på lysdiodene. Også her skal et trykk på CR avslutte programmet.

Lag et program som leser de åtte **mest** signifikante bitene fra bryterne og skriver resultatet ut på de åtte **minst** signifikante lampene og på skjermen. Alle de fire åtte-lederne skal være koblet som før, og en forandring av bryter-settingen på de åtte minst signifikante bryterne skal ikke påvirke utskrift eller lamper.

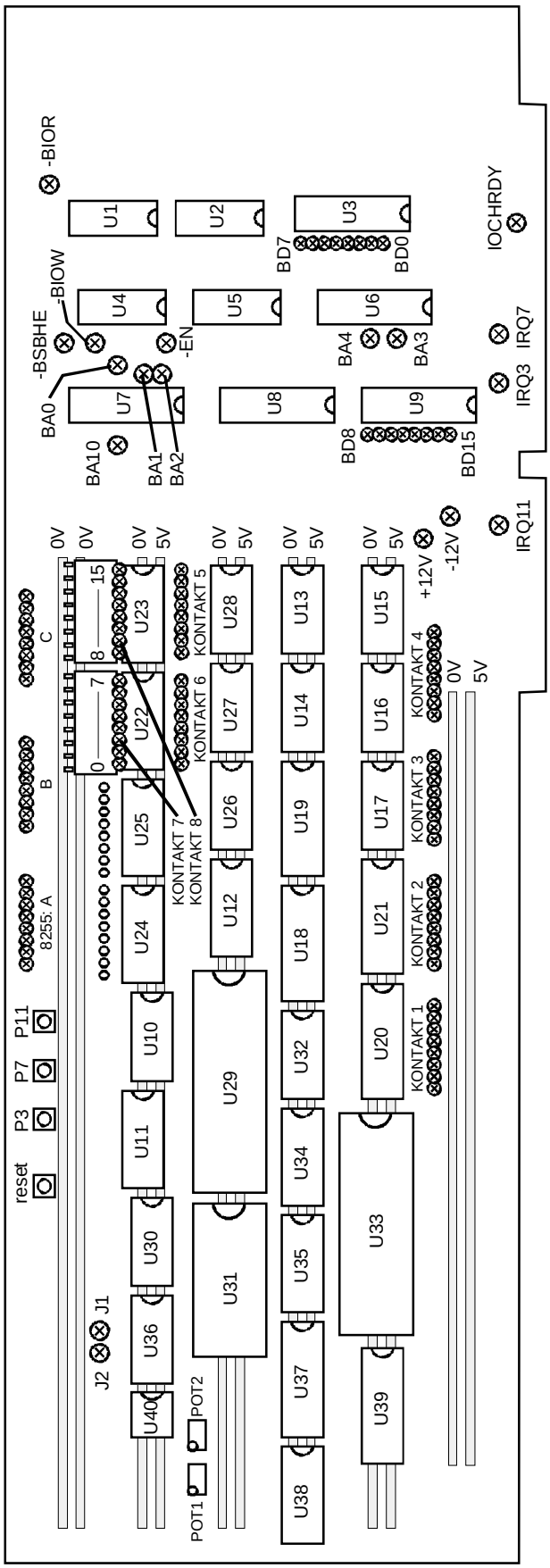


*Fig.10: Kort komponentside.*

Følgende integrerte kretser (IC'er) skal brukes på kortet. Husk at IC'ene ikke er symmetriske – de vil trolig bli ødelagt hvis du setter dem inn feil vei!

| | |
|---|---|
| U1 | 74F00 |
| U2 | 74F10 |
| U3 | 74LS245 |
| U4 | 74F139 |
| U5 | 74F138 |
| U6 | 74F244 |
| U7 | 74F244 |
| U8 | 74F244 |
| U9 | 74LS245 |
| U10 | 74LS14 |
| U11 | 8 x 4,7 kΩ |
| U12 | 74F138 |
| U13 | 74LS122 |
| U14 | 74LS126 |
| U15 | 74LS32 |
| U16 | 74LS32 |
| U17 | 74LS04 |
| U18 | 74LS373 |
| U19 | 74LS373 |
| U20 | 74LS374 |
| U21 | 74LS374 |
| U22 | 8 x 4,7 kΩ |
| U23 | 8 x 4,7 kΩ |
| U24 | 8 x 330 Ω |
| U25 | 8 x 330 Ω |
| U26 | 74LS04 |
| U27 | 74LS04 |
| U28 | 74LS04 |
| U29 | 8255A |
| U30 | 74LS05 |
| U31 | 8254 |
| U32 | 74LS32 |
| U33 | 8274 |
| U34 | AD558 |
| U35 | AD558 |
| U36 | LF347 |
| U37 | AD670 |
| U38 | AD7503 |
| U39 | 74LS273 |
| U40 | REF02 |

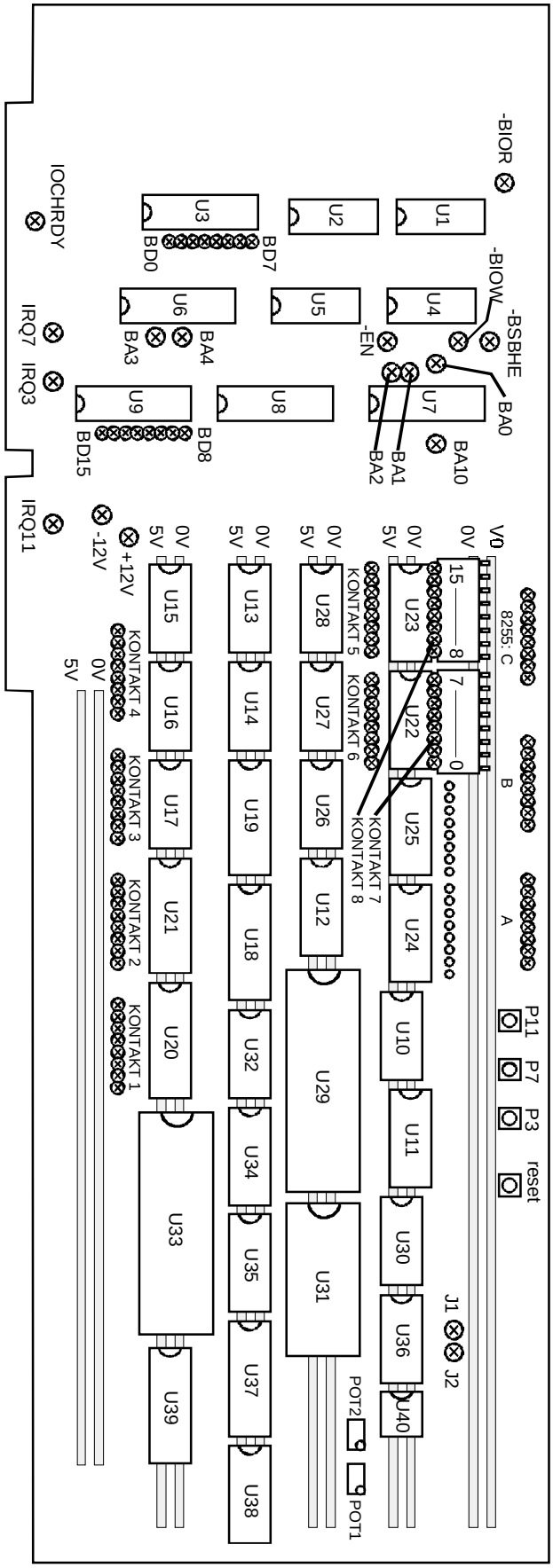


*Fig.11: Kort wire-wrap-side.*